シミュレーション実習
中間レポート

[redacted]

2022 年 5 月 10 日

1　Buffon の針

1.1　解析的解答

y_s と θ の確率密度関数を、それぞれ、$\Phi_1(y_s)$ と $\Phi_2(\theta)$ とすると、それらの確率密度関数は一様分布であるため、

$$\Phi_1(y_s) = \begin{cases} \dfrac{2}{d} : 0 \leq y_s \leq \frac{d}{2} \\ 0 : \text{otherwise} \end{cases} \tag{1}$$

$$\Phi_2(\theta) = \begin{cases} \dfrac{2}{\pi} : 0 \leq \theta \leq \frac{\pi}{2} \\ 0 : \text{otherwise} \end{cases} \tag{2}$$

となる。$\Phi_1(y_s)$ と $\Phi_2(\theta)$ は互いに独立であるため、同時確率密度関数 $\Phi(y_s, \theta)$ は、単純に積を取ればよく、

$$\Phi(y_s, \theta) = \begin{cases} \dfrac{4}{\pi d} : 0 \leq y_s \leq \frac{d}{2}, 0 \leq \theta \leq \frac{\pi}{2} \\ 0 : \text{otherwise} \end{cases} \tag{3}$$

となる。今、$l < d$ であるため、針が線と交わる条件は、$\frac{l}{2} cos\theta \geq y_s$ である。よってその確率は、

$$\int_{\theta=0}^{\theta=\frac{\pi}{2}} \int_{y_s=0}^{\frac{l}{2}cos\theta} \Phi(y_s, \theta) dy_s d\theta = \frac{2l}{\pi d} \tag{4}$$

である。

1.2　(2) と (3) のシミュレーションプロット

シミュレーションに用いた、ソースコード (buffon.cpp) と、それをプロットするためのソースコード (buffon.py) を以下に示す。

ソースコード 1　buffon.cpp

```
#include <stdio.h>
```

```
#include <math.h>
#include <iostream>
#include <iomanip>
#include <fstream>
#include <time.h>
#include "MT.h"

double calc_buffon(int n, double d, double l);

int main(){
    const double d = 2.0;
    const double l = 1.0;
    int n_lim = 100;
    std::ofstream outfile;
    std::string filename = "calc_buffone.csv";
    outfile.open(filename);
    outfile << "n_lim,pi,pi_error" << std::endl;
    double answer, pi, pi_error;
    while (n_lim <= (int)1e8)
    {
        answer = calc_buffon(n_lim, d, l);
        pi = 1 / answer;
        pi_error = fabs(pi - M_PI);
        outfile << n_lim << "," << std::setprecision(7) << pi << "," << std::setprecision(7) << pi_error << std::endl;
        std::cout << n_lim << "," << std::setprecision(7) << pi << "," << std::setprecision(7) << pi_error << std::endl;
        n_lim *= 1.2;
    }
    n_lim = (int)1e8;
    answer = calc_buffon(n_lim, d, l);
    pi = (2.0 * l) / (answer * d);
    pi_error = fabs(pi - M_PI);
    outfile << n_lim << "," << std::setprecision(7) << pi << "," << std::setprecision(7) << pi_error << std::endl;
    std::cout << n_lim << "," << std::setprecision(7) << pi << "," << std::setprecision(7) << pi_error << std::endl;
    outfile.close();
    return 0;
}

double calc_buffon(int n, double d, double l){
    int count = 0;
    init_genrand(time(NULL));
    for (int i = 0; i < n + 1; i++)
    {
        double y_s = d / 2 * genrand_res53();
        double theta = genrand_res53() / 2 * M_PI;
        if (l / 2 * cos(theta) > y_s)
```

```
        {
            count++;
        }
    }
    double p = count / (double)n;
    return p;
}
```

ソースコード 2　buffon.py

```python
import matplotlib.pyplot as plt
import numpy as np
import pandas as pn
import math

plt.rcParams['text.usetex'] = True
xrange = (1e2, 1e8)
yrange_1 = (2.5, 4)
yrange_2 = (1e-6, 1)
xtitle = 'Number of trials'

fig = plt.figure(figsize=(8.27, 11.69))
plt.subplots_adjust(hspace=0.3)
csv_path = 'calc_buffone.csv'
csv_input = pn.read_csv(csv_path, header=0)
n = csv_input['n_lim'].to_numpy(dtype=int)
pi = csv_input['pi'].to_numpy(dtype=float)
pi_error = csv_input['pi_error'].to_numpy(dtype=float)

ax1 = fig.add_subplot(211, xscale='log', xlim=xrange, ylim=yrange_1)
ax1.plot(n, pi, 'x-', ms=5, color="r", label=r'Calculated $\pi$')
ax1.plot(xrange, [math.pi, math.pi], '--', color='k', label='$\pi$')
ax1.legend(loc=0, borderaxespad=0, fontsize=15)
ax1.set_title(r'Calculate $\pi$ by buffon', fontsize=20)
ax1.set_xlabel(xtitle, fontsize=15)
ax1.set_ylabel('Calculated data', fontsize=15)
ax1.spines['top'].set_linewidth(1.5)
ax1.spines['bottom'].set_linewidth(1.5)
ax1.spines['left'].set_linewidth(1.5)
ax1.spines['right'].set_linewidth(1.5)

ax2 = fig.add_subplot(212, xscale='log', yscale='log', ylim=yrange_2)
ax2.plot(n, pi_error, '+-', ms=5, color='b', label='error')
x = np.linspace(100, int(1e8), int(1e3))
y = 1 / x ** 0.5
ax2.plot(x, y, '--', color='k', label=r'$\propto \frac{1}{\sqrt{n}}$')
ax2.legend(loc=0, borderaxespad=0, fontsize=15)
ax2.set_title(r'Error between calculate and $\pi$', fontsize=20)
ax2.set_xlabel(xtitle, fontsize=15)
```

```
ax2.set_ylabel('Error', fontsize=15)
ax2.spines['top'].set_linewidth(1.5)
ax2.spines['bottom'].set_linewidth(1.5)
ax2.spines['left'].set_linewidth(1.5)
ax2.spines['right'].set_linewidth(1.5)
plt.savefig('calc_buffon_plot.png')
```

また、シミュレーションした時の、円周率を計算した時のプロットと、理論値 π とのずれ量をを図 1 に示す。その際 $d = 2.0, l = 1.0$ とし、試行回数に対する計算結果と理論値との誤差について、参考として $1/\sqrt{n}$ を図示した。

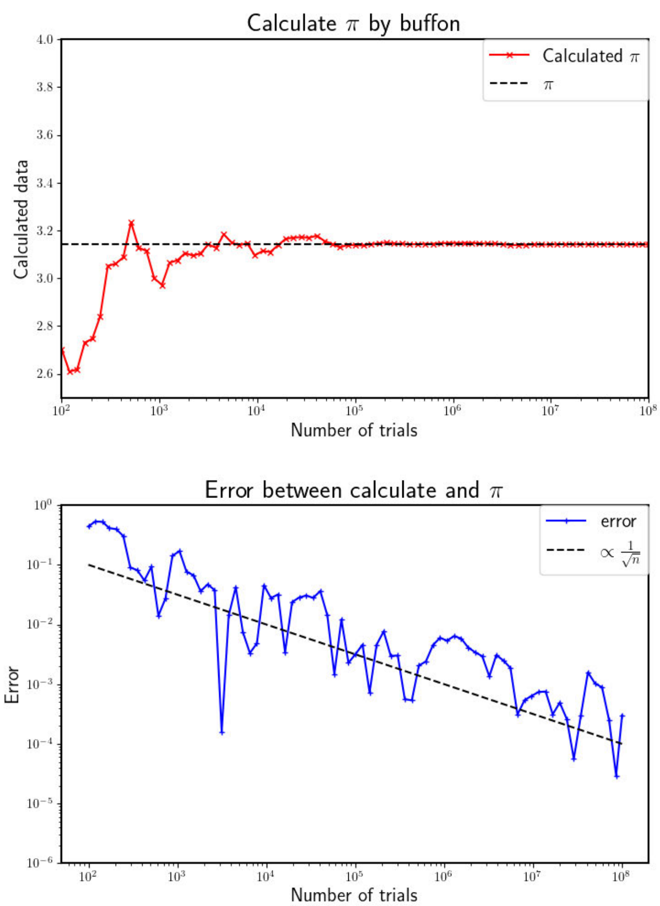

図 1
上図が試行回数に対する計算結果の分布、下図が理論値 π とのずれ量をプロットしたものである。

2　溶媒中におけるばねで繋がれた物体の運動

2.1　小球の運動の条件について

$x(t)$ の特殊解を $\mathrm{e}^{\lambda t}$ とし、運動方程式に、代入して整理すると、

$$\lambda^2 + \frac{\zeta}{m}\lambda + \frac{k}{m} = 0 \tag{5}$$

となる。(5) の判別式を D とすると、$D < 0$ の時減衰振動であり、$D > 0$ の時過減衰となり、$D = 0$ の時臨界減衰となる。よって、

$$\begin{cases} D < 0 \Leftrightarrow \left(\dfrac{\zeta}{m}\right)^2 < \dfrac{4k}{m} : \text{減衰振動} \\ D > 0 \Leftrightarrow \left(\dfrac{\zeta}{m}\right)^2 > \dfrac{4k}{m} : \text{過減衰} \\ D = 0 \Leftrightarrow \left(\dfrac{\zeta}{m}\right)^2 = \dfrac{4k}{m} : \text{臨界減衰} \end{cases} \tag{6}$$

である。

2.2　運動方程式の離散化

まず、問題文で与えられた運動方程式を、$\frac{dv}{dt}, v(t), x(t)$ を用いて表すと、

$$m\frac{dv}{dt} = -\zeta v(t) - kv(t) \tag{7}$$

となり、これを離散化すると、

$$\begin{aligned} m\frac{v(t+\Delta t) - v(t)}{\Delta t} &= -\zeta v(t) - kv(t) \\ \Leftrightarrow \quad v(t+\Delta t) &= \left(1 - \frac{\zeta}{m}\Delta t\right) v(t) - \frac{k}{m}x(t)\Delta t \end{aligned} \tag{8}$$

となる。さらに、無次元化するために、$x = a\tilde{x}, \quad t = t_0\tilde{t}, \quad \dot{x} = v = (a/t_0)\tilde{v}$ とすると $\tilde{v}(\tilde{t})$ の二項間漸化式は、

$$\begin{aligned} \frac{a}{t_0}\tilde{v}(\tilde{t} + \Delta\tilde{t}) &= \left(1 - \frac{\zeta}{m}t_0\Delta\tilde{t}\right)\frac{a}{t_0}\tilde{v}(\tilde{t}) - \frac{k}{m}a\tilde{x}(\tilde{t})t_0\Delta\tilde{t} \\ \Leftrightarrow \quad \tilde{v}(\tilde{t} + \Delta\tilde{t}) &= \left(1 - \frac{\zeta}{m}t_0\Delta\tilde{t}\right)\tilde{v}(\tilde{t}) - \frac{k}{m}{t_0}^2\tilde{x}(\tilde{t})\Delta\tilde{t} \end{aligned} \tag{9}$$

となる。また、$x(t)$ に関しては、

$$x(t+\Delta t) = x(t) + v(t+\Delta t)\Delta t \tag{10}$$

$$= x(t) + \left(1 - \frac{\zeta}{m}\Delta t\right) v(t)\Delta t - \frac{k}{m}x(t)\Delta t^2 \tag{11}$$

であるため、v と同じように無次元化すると、

$$\tilde{x}(\tilde{t}+\Delta\tilde{t}) = \tilde{x}(\tilde{t}) + \tilde{v}(\tilde{t}+\Delta\tilde{t})\Delta\tilde{t} \tag{12}$$

$$= \left(1 - \frac{\zeta}{m}t_0\Delta\tilde{t}\right)\tilde{v}(\tilde{t})\Delta\tilde{t} + \left(1 - \frac{k}{m}{t_0}^2\right)\tilde{x}(\tilde{t})\Delta t^2 \tag{13}$$

となる。

2.3 パラメータの導入

$t_d = \frac{m}{\zeta}$, $t_s = \sqrt{\frac{m}{k}}$ を用いて、式 (9),(13) を書き換えると、

$$\tilde{v}(\tilde{t}+\Delta\tilde{t}) = \left(1 - \frac{t_0}{t_d}\Delta\tilde{t}\right)\tilde{v}(\tilde{t}) - \left(\frac{t_0}{t_s}\right)^2\tilde{x}(\tilde{t})\Delta\tilde{t} \tag{14}$$

$$\tilde{x}(\tilde{t}+\Delta\tilde{t}) = \tilde{x}(\tilde{t}) + \left(1 - \frac{t_0}{t_d}\Delta\tilde{t}\right)\tilde{v}(\tilde{t})\Delta\tilde{t} - \left(1 - \frac{{t_0}^2}{{t_s}^2}\right)\tilde{x}(\tilde{t})\Delta t^2 \tag{15}$$

となる。

2.4 シミュレーション結果

以上のことを踏まえて、物体の抵抗がある運動方程式について数値的に解析した。そのソースコード (dampingEoM.cpp) とその結果をプロットするソースコード (dampingEoM.py) を以下に示す。ただし、$a = 10$ とし、振動のパラメータは減衰振動、過減衰、臨界減衰の 3 つの条件が出るように t_d の値を定めた。

ソースコード 3 dampingEoM.cpp

```cpp
#include <stdio.h>
#include <math.h>
#include <iostream>
#include <iomanip>
#include <fstream>
#include <cfloat>

void damping(double a, double dt, double t_d, double t_0);

int main(){
    const double a = 10.;
    double dt = 1e-3, t_d = 0.25, t_0 = 1.0;
    damping(a, dt, t_d, t_0);
    t_d = 0.1;
    damping(a, dt, t_d, t_0);
    t_d = 4;
    damping(a, dt, t_d, t_0);
    return 0;
}

void damping(double a, double dt, double t_d, double t_0){
```

```cpp
    char filename[128];
    std::ofstream file;
    int i = 0, div = (int)1/dt / 10;
    double v = a / t_0, x = 0.;
    sprintf(filename,"pos_%.2f_%.4f_%.2f_%.2f.csv", a, dt, t_d, t_0);
    file.open(filename);
    file << "t,v,x" << std::endl;
    file << (double)i*dt << "," << v << "," << x <<std::endl;
    std::cout << (double)i*dt << "," << v << "," << x <<std::endl;
    while(fabs(v) > 1e-4){
        v = v - (t_0/t_d)*dt*v - x*dt;
        x = x + v*dt;
        i++;
        if(i % div == 0){
            file << (double)i*dt << "," << v << "," << x <<std::endl;
            std::cout << (double)i*dt << "," << v << "," << x <<std::endl;
        }
  }
  file.close();
}
```

ソースコード 4　dampingEoM.py

```python
import matplotlib.pyplot as plt
import pandas as pn
import matplotlib.cm as cm


def draw(pos, a, dt, t_d, t_0):
    csv_path = 'pos_{:.2f}_{:.4f}_{:.2f}_{:.2f}.csv'.format(a, dt, t_d, t_0)
    position = int('22{}'.format(pos))
    plt.subplot(position)
    csv_input = pn.read_csv(csv_path, header=0)
    t = csv_input['t'].to_numpy(dtype=float)
    v = csv_input['v'].to_numpy(dtype=float)
    x = csv_input['x'].to_numpy(dtype=float)
    plt.plot(t, x, 'x-', ms=3, color=cm.jet(t_d), label=r'$t_s$␣=␣{:.2},␣$t_0$␣=␣{:.2}'.format(t_d, t_0))
    plt.legend(loc=0, borderaxespad=0, fontsize=15)
    plt.xlabel(r'$t/t_0(=\tilde{t})$', fontsize=15)
    plt.ylabel(r'$x(t)/a(=\tilde{x}(t))$', fontsize=15)
    if t_d < t_0 / 4:
        plt.title('over␣damping', fontsize=20)
    elif t_d > t_0 /4:
        plt.title('under␣damping', fontsize=20)
    else:
        plt.title('critical␣damping', fontsize=20)


```

```python
plt.rcParams['text.usetex'] = True
a = 10.
fig = plt.figure(figsize=(11.69, 8.27), tight_layout=True)
fig.suptitle('simulated position (a = {:.2f})'.format(a), fontsize=30)
dt = 1e-3
t_d = 0.1
t_0 = 1.
draw(1, a, dt, t_d, t_0)
t_d = 0.25
draw(2, a, dt, t_d, t_0)
t_d = 4.
draw(3, a, dt, t_d, t_0)

plt.show()
plt.savefig('plot_dampingEoM.png')
```

また、シミュレーション結果をプロットしたものを図 2 に示す。

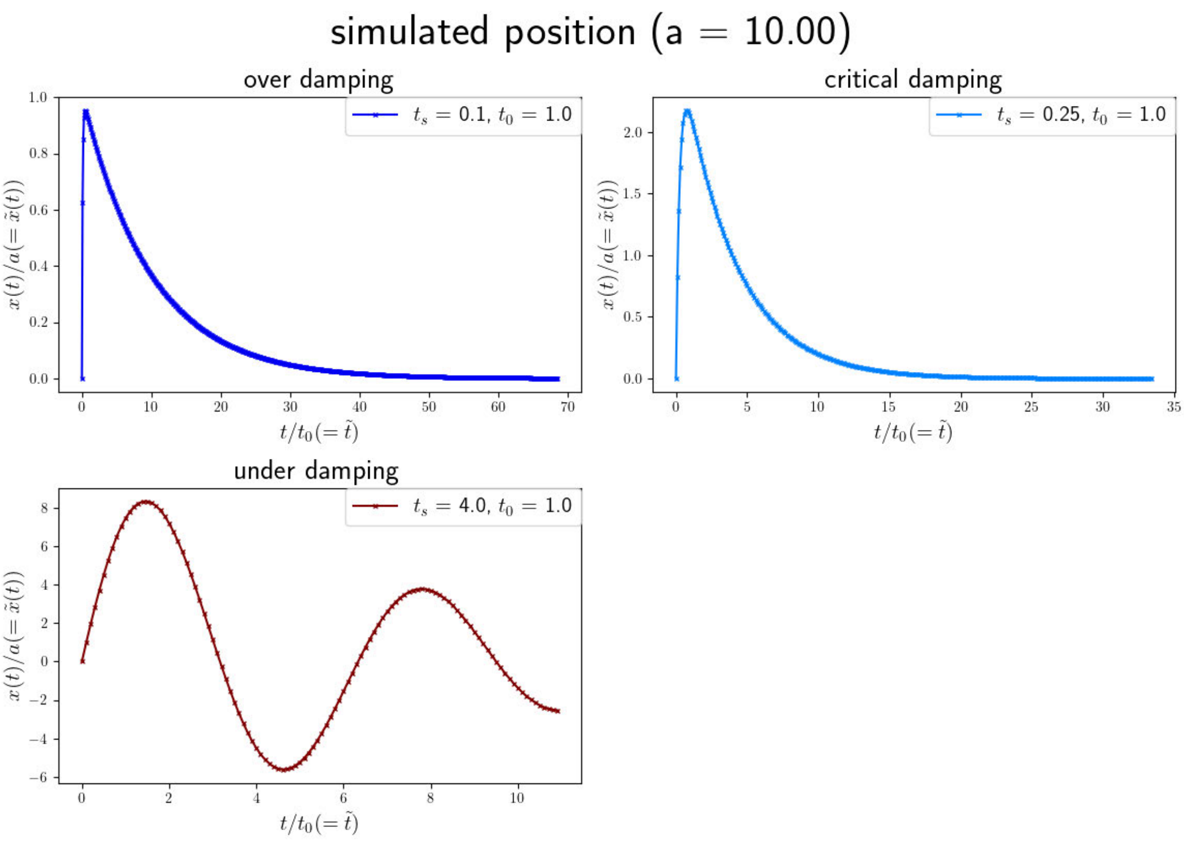

図 2
それぞれの t_d に対する、位置の時間発展