ビリヤードのシミュレーション

2019/02/23

明治大学　総合数理学部　現象数理学科

卒業研究レポート

2610150019　近藤 拓馬

目次

第 1 章　はじめに

私の卒業論文の目的はビリヤードの玉の軌道をプログラムし、シミュレーションすることである。この卒業論文を選定した理由としては私自身ビリヤードをしており、打った球の動きを予測することはできないのか疑問に思い注目してみた。こうして力学を勉強することに決め、数値計算により明らかにすることを目標として本研究を開始した。

第 2 章　ニュートンの運動の第二法則

物体に外力 f が作用すると、物体には比例した f と同じ向きの加速度 a が生じる。すなわち次の式が成り立つ。

$$f=ma$$

(m:質量)

第 3 章　摩擦力

この章では摩擦力について考える。ビリヤードの球の摩擦を考える時２パターンが考えられる。ビリヤードの球が転がる時と滑る時である。結論から先に言ってしまえば、ボールが転がる時は静止摩擦力で、ボールが滑る時は動摩擦力が働く。

静止摩擦力

物体が面に対して静止している時、面が物体に及ぼす力のうち、面に平行な成分の力を静止摩擦力と言う。転がる球に働く力は静止摩擦である。ビリヤードの場合について考える。ビリヤードにおいてボールが転がる条件は次の式が成り立つ時である。

$$F \leq \mu N$$

第 4 章　モーメント

4.1 力のモーメント

モーメントとは定点に対する物理量の回転の能力で示すもので、定点からの方向性を考

慮して位置ベクトル$\vec{r}$との積で表される。
力のモーメント($\vec{N}=\vec{r}\times\vec{F}$)は、力と力に垂直な定点からの距離の積で回転の強さを表す。
また、角運動量（$\vec{L}=\vec{r}\times\vec{p}$）も同様に、運動量ベクトル$\vec{p}$の物体の回転する勢いである。

4.2 慣性モーメント

慣性モーメントは質量と回転軸に垂直な距離の二乗の積であり、物体の回転に対して示す慣性（動きにくさ）の程度を示す。
ここでビリヤードの球同様、密度が一様な球の慣性モーメントを考える。
半径 a、密度 ρ の球の質量を M と置くと、

$$M=\frac{4}{3}\pi a^3 \rho$$

となる。
この球の重心 G を通る軸の周りの慣性モーメント I を M と a で表す。軸から r と r+dr の間にある微笑部分の体積を dV、質量を dm とおくと

$$dm=\rho\, dV$$

$$=\rho\ \cdot 2\pi r\cdot 2\sqrt{a^2-r^2}dr$$

$$=4\pi\ \rho\, r\sqrt{a^2-r^2}dr$$

となる。よってこれより求められる球の慣性モーメントは

$$I=\int_0^a r^2 dm$$

$$=4\pi\ \rho\int_0^a r^3\sqrt{a^2-r^2}dr$$

ここで、$a^2-r^2=u$ とおくと

r：0→a　のとき　u：$a^2\rightarrow 0$

-2rdr=du、つまり

$rdr=-\frac{1}{2}du$

$I=4\pi\ \rho\int_{a^2}^{0}(a^2-u)\sqrt{u}\left(-\frac{1}{2}\right)du$

$=2\pi\ \rho\int_{0}^{a^2}\left(a^2u^{\frac{1}{2}}-u^{\frac{3}{2}}\right)du$

$I=2\pi\ \rho\int_{0}^{a^2}\left(a^2u^{\frac{1}{2}}-u^{\frac{2}{2}}\right)du$

$=2\pi\ \rho\left[\frac{2}{3}a^2u^{\frac{3}{2}}-\frac{5}{5}\right]_{0}^{a^2}$

$=2\pi\ \rho\left(\frac{2}{3}a^5-\frac{2}{5}a^5\right)$

$=2\pi\ \rho\ \cdot\frac{4}{15}a^5$

$=\frac{2}{5}\ \cdot\frac{4}{3}\pi a^3\rho\ \cdot a^2$

$M=\frac{4}{3}\pi a^3\ \rho$ より

$I=\frac{2}{5}Ma^2$

第 5 章　Runge-kutta 法

ルンゲ・クッタ（runge-kutta）法は、コンピュータを使った微分方程式解法である。しかしコンピューターでは、連続的な情報を扱うことが非常に困難である。 そのため離散化（連続した値をとびとびの値に置き換えて、コンピューターで扱えるようにする）する必要がある。

4 次のルンゲクッタ法

一階の常微分方程式

dy/dx=f(x, y)

において、x が x0 から h だけ増加した点 x1=x0+h における値 y1 を以下の計算で求めます。

k1=hf(x0, y0)

k2=hf(x0+h/2, y0+k1/2)

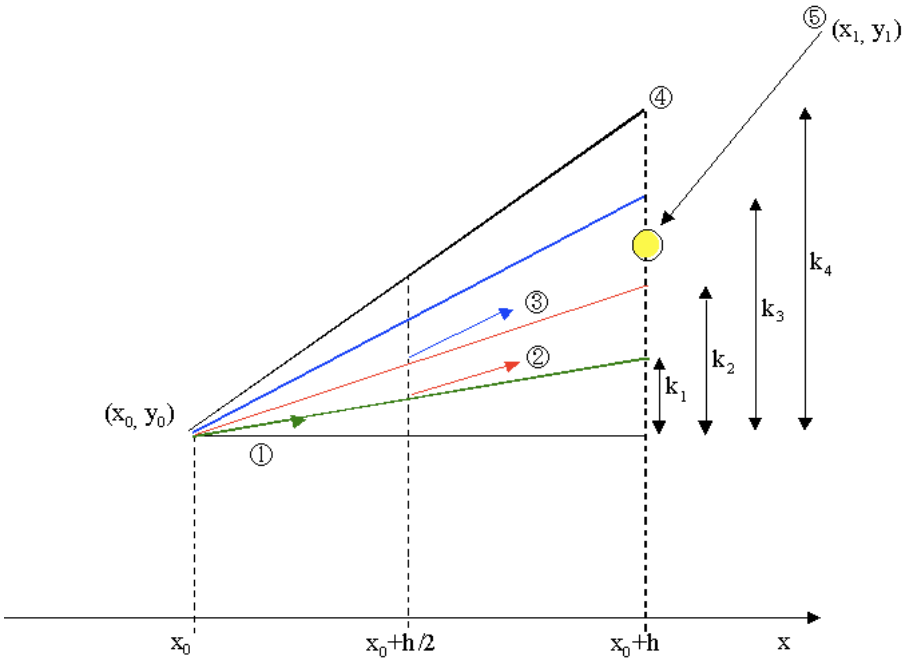
k3=hf(x0+h/2, y0+k2/2)

k4=hf(x0+h, y0+k3)

k=(k1+2k2+2k3+k4)/6

y1=y0+k

これらの式の意味は図 1 に示す通りです。



図を捕捉すると、

①点(x0, y0)で傾き f(x0, y0)で h だけ進むと y は k1 だけ変化することになる。

②この①の線上の点(x0+h/2, y0+k1/2)で傾きを計算し、(x0, y0)から h だけ進むと y は k2 だけ変化することになる。

③同じように(x0, y0)から傾き(x0+h/2, y0+k2/2)で h だけ進むと k3 の変化となる。

④(x0, y0)から傾き(x0+h, y0+k3)で h だけ進むと k4 の変化となる。
⑤以上の計算結果の加重平均として点(x1, y1)を得る。

第 6 章　斜面を滑る物体の運動方程式

6.1 摩擦なし

$$\begin{cases} m\dfrac{d^2x}{dt^2} = mgsin\theta \\ m\dfrac{d^2y}{dt^2} = 0 \end{cases}$$

この二階微分の方程式を一階の微分方程式に直すと

$\begin{cases} x_1 = x \\ x_2 = y \\ x_3 = x' \\ x_4 = y' \end{cases}$ と置くと

$$\frac{d}{dt}\begin{Bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{Bmatrix} = \begin{Bmatrix} x_1{}' \\ x_2{}' \\ x_3{}' \\ x_4{}' \end{Bmatrix} = \begin{Bmatrix} x_3 \\ x_4 \\ gsin\theta \\ 0 \end{Bmatrix}$$ となる

6.2 摩擦あり

$$\begin{cases} \dfrac{d^2x}{dt^2} = mgsin\theta - f \\ m\dfrac{d^2y}{dt^2} = O \end{cases}$$

この二階微分の方程式を一階の微分方程式に直すと

$$\begin{cases} x_1 = x \\ x_2 = y \\ x_3 = x' \\ x_4 = y' \end{cases}$$ と置くと

$$\frac{d}{dt}\begin{Bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{Bmatrix} = \begin{Bmatrix} x_1{}' \\ x_2{}' \\ x_3{}' \\ x_4{}' \end{Bmatrix} = \begin{Bmatrix} x_3 \\ x_4 \\ gsin\theta - f \\ 0 \end{Bmatrix}$$

6.3 回転する物体

$$M\frac{d^2x}{dt^2} = Mgsin\theta - f$$

重心まわりの回転の方程式より

$$I\frac{d\omega}{dt} = fa$$

この式を回転する角度 β に関する式にすると

$$I\frac{d^2\beta}{dt^2} = fg$$

$$\begin{cases} x_1 = x \\ x_2 = \beta \\ x_3 = \frac{dx}{dt} \\ x_4 = \frac{d\beta}{dt} \end{cases}$$ と置くと

第 7 章　プログラミング

摩擦なし

```
/*
 * masatsunasi1.cpp --- 斜面をすべる質点の運動（摩擦あり）
 *
 *   c++ masatsunasi1.cpp
 *     もしかしたら -I/usr/local/include のような指定が必要かも。
 *   ./a.out > masatsunasi1.dat
 *   gnuplot
 *   gnuplot> plot "masatsunasi1.dat" using 1:2 with l, "masatsunasi1.dat" using
1:6 with l
 */

#include <iostream>
#include <cmath>
#include <Eigen/Dense>
using namespace Eigen;

double m, g, Gamma, e,N,mu, theta;

VectorXd f(double t, VectorXd x)
{
  VectorXd y(4);
  y(0) = x(2);
  y(1) = x(3);
  y(2) = g*sin(theta);
  y(3) = 0.0;
  return y;
}

int main(void)
```

```
{
  int n, N;
  double tau, Tmax, t,pi;
  VectorXd x(4),k1(4),k2(4),k3(4),k4(4);
  double X;

  pi = 4 * atan(1.0);
  theta = 30 * (pi / 180);
  mu = 0.3;
  m = 100;
  g = 9.8;
  Gamma = 1.0;
  e = 1.0;

  Tmax = 20;
  N = 1000;
  tau = Tmax / N;
  x << 0,0,0,0;
  for (n = 0; n < N; n++) {
    t = n * tau;
    k1 = tau * f(t, x);
    k2 = tau * f(t+tau/2, x+k1/2);
    k3 = tau * f(t+tau/2, x+k2/2);
    k4 = tau * f(t+tau, x+k3);
    x = x + (k1 + 2 * k2 + 2 * k3 + k4) / 6;
    if (x(1)<0) {
      x(1) = - x(1);
      x(3) = - x(3);
    }
    X = 0.5 * g * (sin(theta) - mu * cos(theta)) * t * t;
    std::cout << t+tau << " " << x(0) << " " << x(1) << " " << x(2) << " " << x(3)
<< " " << X << std::endl;
  }
```

```
  return 0;
}
```

摩擦あり

```
/*
 * masatsunasi1.cpp --- 斜面をすべる質点の運動（摩擦あり）
 *
 *   c++ masatsunasi1.cpp
 *     もしかしたら -I/usr/local/include のような指定が必要かも。
 *   ./a.out > masatsunasi1.dat
 *   gnuplot
 *   gnuplot> plot "masatsunasi1.dat" using 1:2 with l, "masatsunasi1.dat" using
1:6 with l
 */

#include <iostream>
#include <cmath>
#include <Eigen/Dense>
using namespace Eigen;

double m, g, Gamma, e,N,mu, theta;

VectorXd f(double t, VectorXd x)
{
  VectorXd y(4);
  y(0) = x(2);
  y(1) = x(3);
  y(2) = g*sin(theta)-mu*g*cos(theta);
  y(3) = 0.0;
  return y;
}
```

```
int main(void)
{
  int n, N;
  double tau, Tmax, t,pi;
  VectorXd x(4),k1(4),k2(4),k3(4),k4(4);
  double X;

  pi = 4 * atan(1.0);
  theta = 30 * (pi / 180);
  mu = 0.3;
  m = 100;
  g = 9.8;
  Gamma = 1.0;
  e = 1.0;

  Tmax = 20;
  N = 1000;
  tau = Tmax / N;
  x << 0,0,0,0;
  for (n = 0; n < N; n++) {
    t = n * tau;
    k1 = tau * f(t, x);
    k2 = tau * f(t+tau/2, x+k1/2);
    k3 = tau * f(t+tau/2, x+k2/2);
    k4 = tau * f(t+tau, x+k3);
    x = x + (k1 + 2 * k2 + 2 * k3 + k4) / 6;
    if (x(1)<0) {
      x(1) = - x(1);
      x(3) = - x(3);
    }
    X = 0.5 * g * (sin(theta) - mu * cos(theta)) * t * t;
    std::cout << t+tau << " " << x(0) << " " << x(1) << " " << x(2) << " " << x(3)
```

```
<< " " << X << std::endl;
  }
  return 0;
}
```

回転する物体

```
* ball-bound.cpp --- Eigen を使ってはずむボールのシミュレーション
 *
 *   c++ ball-bound.cpp
 *     もしかしたら -I/usr/local/include のような指定が必要かも。
 *   ./a.out > ball-bound.dat
 *   gnuplot
 *   gnuplot> plot "ball-bound.dat"
 */

#include <iostream>
#include <cmath>
#include <Eigen/Dense>
using namespace Eigen;

double m, g, Gamma, e,N,mu, theta,a,beta,I;

VectorXd f(double t, VectorXd x)
{
  VectorXd y(4);
```

```
  y(0) = x(2);
  y(1) = x(3);
  y(2) = 5/7*g*sin(theta);
  y(3) = (2/7*I)*m*g*sin(theta);
  return y;
}

int main(void)
{
  int n, N;
  double tau, Tmax, t,pi;
  VectorXd x(4),k1(4),k2(4),k3(4),k4(4);

  pi = 4 * atan(1.0);
  theta = 30 * (pi / 180);
  mu = 0.3;
  m = 100;
  g = 9.8;
  Gamma = 1.0;
  e = 1.0;

  Tmax = 20;
  N = 1000;
  tau = Tmax / N;
  x << 0,0,0,0;
  for (n = 0; n < N; n++) {
    t = n * tau;
    k1 = tau * f(t, x);
    k2 = tau * f(t+tau/2, x+k1/2);
    k3 = tau * f(t+tau/2, x+k2/2);
    k4 = tau * f(t+tau, x+k3);
    x = x + (k1 + 2 * k2 + 2 * k3 + k4) / 6;
    if (x(1)<0) {
```

```
      x(1) = - x(1);
      x(3) = - x(3);
    }
    std::cout << x(0) << " " << x(1) << std::endl;
  }
  return 0;
}
```

第９章　まとめ

今回ビリヤードの研究について進めてみたものの、具体的にシミュレーションするところまで進むことはできなかった。今後この研究をするにあたって実際のビリヤードは回転軸が変化していくので、回転軸が変化する場合も考える必要がある。また、球が壁にぶつかった場合にどう変化するのか、そして球同士がぶつかった場合の変化も考える必要がある。まだまだ研究としては多くの課題が残ることとなってしまった。

参考文献

[1]ルンゲクッタ法

http://www.me.tokushima-u.ac.jp/~miw/wadai/IP3/06/10/10.html

[2]馬場敬之　スバラシク実力がつくと評判の力学　キャンパス・ゼミ　改訂４

マセマ出版社　（２０１７）