2023-知能情報基礎演習
4-1 微分方程式の数値解法

提出者氏名: 城間 颯
提出者学籍番号: 225719A
担当教員名: 宮里 智樹

提出日: 2023 年 12 月 7 日
提出期限日: 2023 年 12 月 5 日
実験日: 2023 年 11 月 28 日

目次

1 **問題 1** 2

2 **問題 2** 3

3 **実験: 野球ボールの軌道計算** 3

4 **問題 3** 6

5 **問題 4（実験結果）** 7

6 **問題 5** 11

6.1 ホイン法 11

6.2 ホイン法の実装 12

6.3 ホイン法の結果 15

7 **考察** 19

1　問題 1

常微分方程式に初期値を与えることで、積分定数などの未知数を含まない形で、元の関数を求めることができる。このような問題を初期値問題と呼ぶ。オイラー法は、初期値問題を数値的に解く方法である。例として、以下のような常微分方程式の初期値問題を考える。

$$\frac{dy}{dx} = f(x, y), y(x_0) = y_0$$

この常微分方程式の解 $y = y(x)$ が図 1 のような曲線になっているとする。

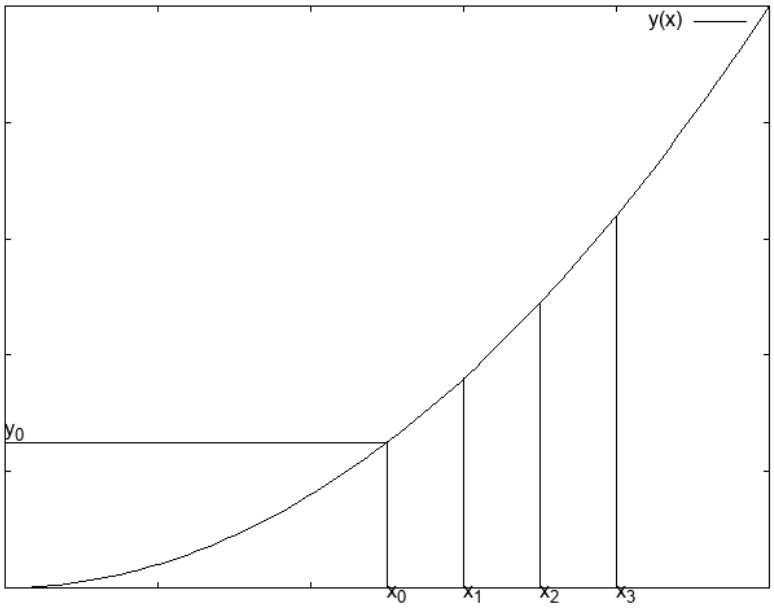

図 1　y=y(x) のグラフ

微分方程式を数値的に解くとは、初期値 x_0 から始めて、$x_1 = x_0 + h, x_2 = x_1 + h, \ldots$ における $y(x)$ の値を順次求めていくことに対応する。このときの h を刻み幅と呼ぶ。

最も単純に考えると、今 y の x に関する微分 $\frac{dy}{dx}$、すなわち曲線の傾きが f(x,y) なので、

$$y(x_1) = y(x_0) + hf(x_0, y_0)$$

のように近似すれば、十分に小さい h に対しては、それなりに良い解を得られると考えられる。$y(x_1)$ をオイラー法で求めたときの様子を図 2 に示す。

これを一般化して、

$$y(x_i) = y(x_{i-1}) + hf(x_{i-1}, y_{i-1})$$

という漸化式にしたがって、順次 $y(x_1), y(x2), \ldots$ を求めていく方法を、オイラー法または前進オイラー法と呼ぶ。

オイラー法は刻み幅 h を十分に小さくすることで、高い精度を得られるはずだが、実際には、

- h を小さくすると、計算回数が増え、計算量が増加する
- h を小さくすると、丸め誤差が大きくなる
- 計算回数が増えると、誤差の累積も増加する

といった問題があり、それほど良い精度は得られない。

2 問題 2

初期値問題

$$\frac{dx}{dt} = -x, x(0) = 1$$

を解く。

$$\begin{aligned}
\frac{dx}{dt} &= -x \\
dx &= -xdt \\
\int \frac{1}{x} dx &= -\int dt \\
\ln x &= -t + C \\
x &= e^{-t+c} = e^c e^{-t} = C' e^{-t} \\
x(0) &= C' e^{-0} = C' = 1
\end{aligned}$$

よって、$x = e^{-t}$。

3 実験: 野球ボールの軌道計算

Listing1 に、実験で使用したオイラー法による野球ボールの軌道計算のプログラムを示す。
このプログラムは、オイラー法により、野球ボールの軌道計算を行うプログラムで、

$$\begin{aligned}
v_{i+1} &= v_i + \tau \cdot (\frac{1}{m} F_a(v_i) - g\hat{y}) \\
r_{i+1} &= r_i + \tau \cdot v_i
\end{aligned}$$

$$F_a(v) = -\frac{1}{2} C_d \rho A |v| v$$

の漸化式にしたがって、各時刻の、速度と位置の値を計算している。

Listing 1　オイラー法による野球ボールの軌道計算のプログラム

```
1 // baseball.cpp: オイラー法を用いて野球ボールの軌道を計算するプログラム
2 #include "NumMeth.h"
```

```
3
4  using namespace std;
5  int main() {
6  //* ボールの初期位置及び初期速度を設定する.
7  double y1, speed, theta;
8  double r1[2+1], v1[2+1], r[2+1], v[2+1], accel[2+1]; cout << "高さの初期値メートル
       ()␣:␣"; cin >> y1;
9  r1[1] = 0;
10 r1[2] = y1; // 初期位置ベクトル
11 cout << "初期速度(m/s)␣:␣"; cin >> speed;
12 cout << "初期角度度()␣:␣"; cin >> theta;
13 const double pi = 3.141592654;
14 v1[1] = speed*cos(theta*pi/180); // 初期速度 (x)
15 v1[2] = speed*sin(theta*pi/180); // 初期速度 (y)
16 r[1] = r1[1];
17 r[2] = r1[2]; // 初期位置および初期速度を設定
18 v[1] = v1[1];
19 v[2] = v1[2];
20 //* 物理パラメータを設定質量(, Cd 値など)
21 double Cd = 0.35;
22 double area = 4.3e-3;
23 double grav = 9.81;
24 double mass = 0.145;
25 double airFlag, rho;
26 cout << "空気抵抗あり(:1,␣なし:0)␣:␣";
27 cin >> airFlag; if( airFlag == 0 )
28 rho = 0; // 空気抵抗なし else
29 rho = 1.2; // 空気の密度(kg/m^3)
30 double air_const = -0.5*Cd*rho*area/mass; // 空気抵抗定数
31 //* ボールが地面に着くまで, あるいは最大の刻み数になるまでループ
32 double tau;
33 cout << "時間刻み␣τ秒()␣:␣"; cin >> tau;
34 int iStep, maxStep = 1000; // 最大の刻み数
35 double *xplot, *yplot, *xNoAir, *yNoAir;
36 xplot = new double [maxStep + 1];
37 yplot = new double [maxStep + 1];
38 xNoAir = new double [maxStep + 1];
39 yNoAir = new double [maxStep + 1];
40
41 double max_y = 0;
42
43 for( iStep=1; iStep<=maxStep; iStep++ ) {
44     //* プロット用に位置計算値および理論値()を記録する
45     xplot[iStep] = r[1]; // プロット用に軌道を記録
46     yplot[iStep] = r[2];
47     double t = ( iStep-1 )*tau; // 現在時刻
48     xNoAir[iStep] = r1[1] + v1[1]*t; // 位置(x)
49     yNoAir[iStep] = r1[2] + v1[2]*t - (0.5*grav*t*t); // 位置(y) ここに位置を求める式
           を書く
50     //* ボールの加速度を計算する
51     // 空気抵抗無次元()
52     // 投射物の横断面積(m^2)
53     // 重力加速度(m/s^2)
54     // 投射物の質量(kg)
```

```
55      double normV = sqrt( v[1]*v[1] + v[2]*v[2] );
56      accel[1] = air_const*normV*v[1]; // 空気抵抗
57      accel[2] = air_const*normV*v[2]; // 空気抵抗
58      accel[2] -= grav; // 重力
59      //* オイラー法を用いて，新しい位置および速度を計算する
60
61      r[1] = r[1] + tau*v[1];
62      r[2] = r[2] + tau*v[2];
63
64      v[1] = v[1] + tau*accel[1];
65      v[2] = v[2] + tau*accel[2];
66
67      //* ボールが地面に着いたら(y < 0)ループを抜ける
68      /*
69      if 文を使って書く
70      */
71      if(yplot[iStep]<0) {
72          break;
73      }
74
75      if(r[2]>max_y) {
76          max_y = r[2];
77      }
78  }
79  //* 最大到達高さと滞空時間を表示するこのままだと正しい結果ではない()
80  cout << "最大到達高さは" << max_y << "メートル" << endl;
81  // 滞空時間の表示をここに書く
82  cout << "滞空時間は" << tau*iStep << "秒" << endl;
83  //* プロットする変数を出力する
84  // xplot, yplot xNoAir, yNoAir
85  ofstream xplotOut("xplot.txt"), yplotOut("yplot.txt"),
86  xNoAirOut("xNoAir.txt"), yNoAirOut("yNoAir.txt");
87  int i;
88
89  for( i=1; i<=iStep; i++ ) {
90      xplotOut << xplot[i] << endl;
91      yplotOut << yplot[i] << endl;
92  }
93
94  for( i=1; i<=iStep; i++ ) {
95      xNoAirOut << xNoAir[i] << endl;
96      yNoAirOut << yNoAir[i] << endl;
97  }
98
99  ofstream xyplotOut("xyplot.txt"), xyNoAirOut("xyNoAir.txt");
100 xyplotOut << "#X␣Y" << endl;
101 xyNoAirOut << "#X␣Y" << endl;
102
103 for( i=1; i<=iStep; i++ ) {
104     if (i>0 && yplot[i] == 0){
105         continue;
106     }
107     xyplotOut << xplot[i] << "␣" << yplot[i] << endl;
```

```
108 }
109
110 for( i=1; i<=iStep; i++ ) {
111     if (i>0 && yplot[i] == 0){
112         continue;
113     }
114     xyNoAirOut << xNoAir[i] << "␣" << yNoAir[i] << endl;
115 }
116
117 delete [] xplot;
118 delete [] yplot;
119 delete [] xNoAir;
120 delete [] yNoAir; // メモリを開放
121 }
```

4 問題 3

$$\frac{dy}{dx} = d(x, y), y(x_0) = y_0$$

の初期値問題を考える。

オイラー法では、初期値問題の解 y(x) はわからず、現在わかっている関数 y(x) 上の点 (x_i, y_i) を接点とする y(x) の接線を求め、その接線が、点 (x_i, y_i) の近くでは、関数 y(x) を近似的に表しているとみなして、x_{i+1} における接線の高さ y'_{i+1} を、真の値である $y(x_{i+1})$ の近似値であると考える。この時のイメージを図 2 に示す。

τ は、x_{i+1}と x_i の差を表す。したがって、τ を大きくするほど、x_{i+1}と x_i は離れるため、接線から求める点 (x_{i+1}, y_{i+1}) も、接点 (x_i, y_i) から離れていく。ゆえに、τ を大きくするほど、接線が、真の関数 y(x) を近似的に表すという仮定は成り立ちにくくなるため、真の値と、オイラー法で求めた近似値は、誤差が大きくなることが考えられる。

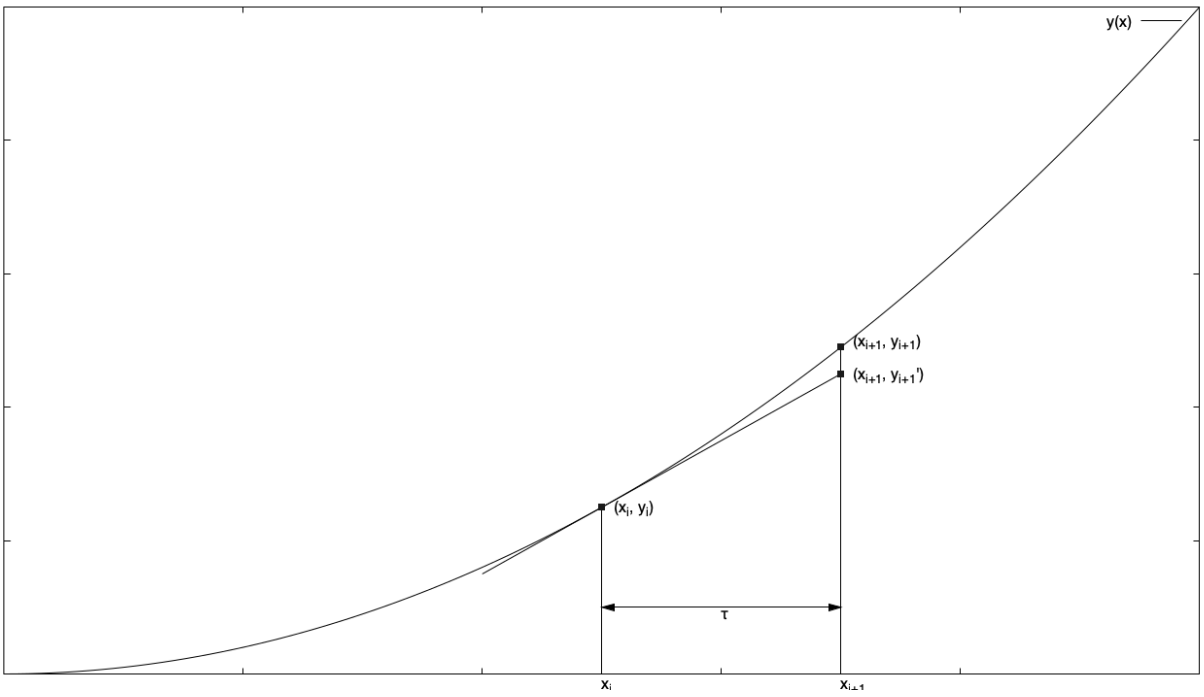

図 2　y=y(x) のオイラー法による近似

5　問題 4（実験結果）

時間刻み τ は、0.2、0.15、0.1、0.05、0.01、0.005、0.001 の 7 つを選んだ。

初期値は、高さ 0m、初期速度 40m/s、投射角度 45 度、空気抵抗ありとした。

$\tau = 0.2$ の時のグラフを、図 3 に示す。

$\tau = 0.15$ の時のグラフを、図 4 に示す。

$\tau = 0.1$ の時のグラフを、図 5 に示す。

$\tau = 0.05$ の時のグラフを、図 6 に示す。

$\tau = 0.01$ の時のグラフを、図 7 に示す。

$\tau = 0.005$ の時のグラフを、図 8 に示す。

$\tau = 0.001$ の時のグラフを、図 9 に示す。

x 軸は、ボールの水平方向への移動距離 (m) を、y 軸は、ボールの軌道の高さ (m) を表す。

理論値というラベルがついた曲線は、空気抵抗なしの場合の、解析的に計算したボールの軌道である。

実測値というラベルがついた曲線は、オイラー法により求めた、ボールの軌道である。

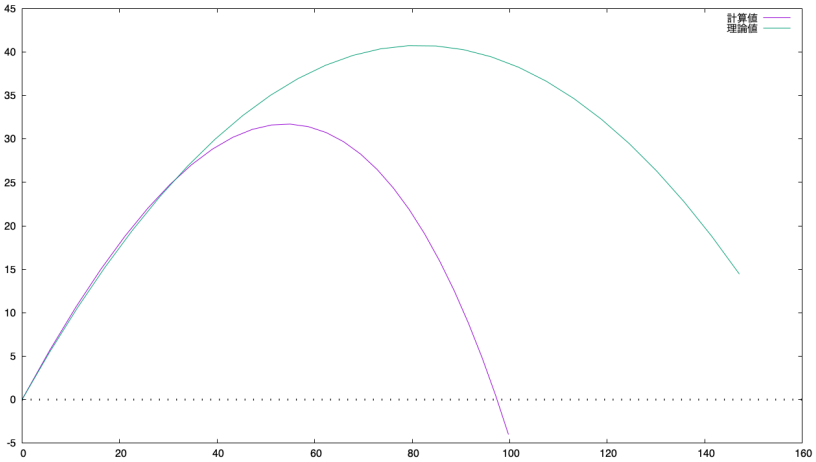

図 3　$\tau = 0.2$ の時のボールの軌道

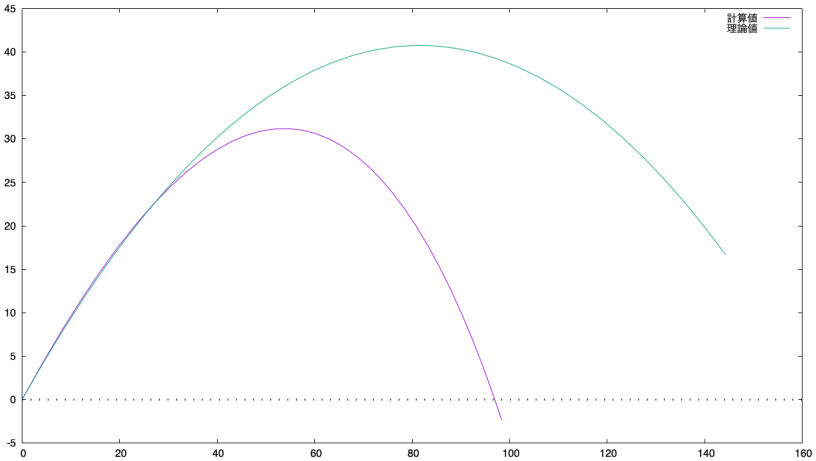

図 4　$\tau = 0.15$ の時のボールの軌道

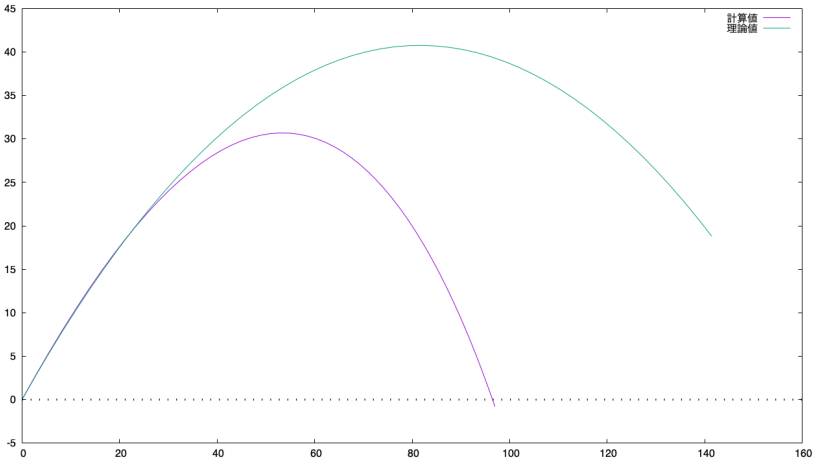

図 5　$\tau = 0.1$ の時のボールの軌道

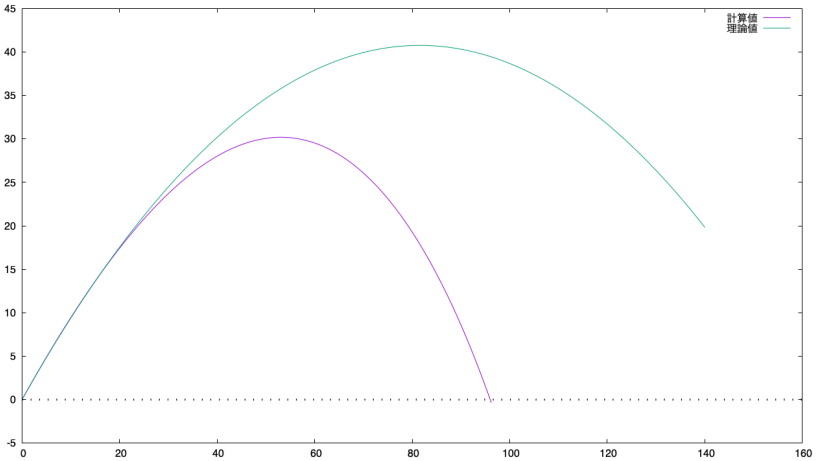

図 6　$\tau = 0.05$ の時のボールの軌道

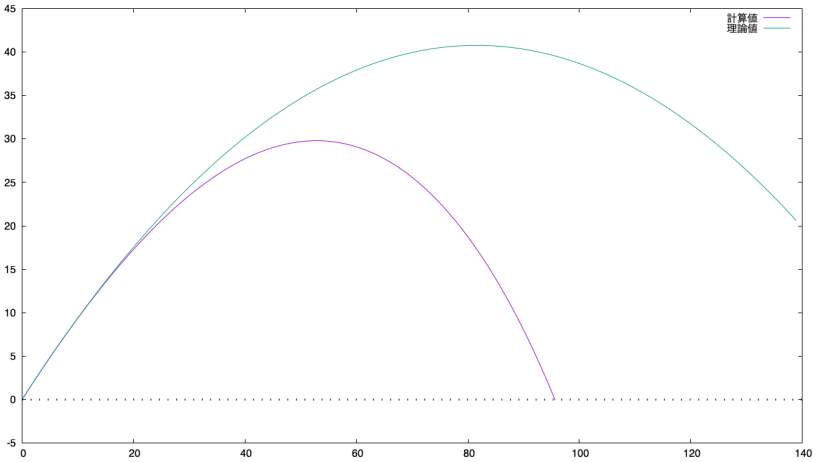

図 7　$\tau = 0.01$ の時のボールの軌道

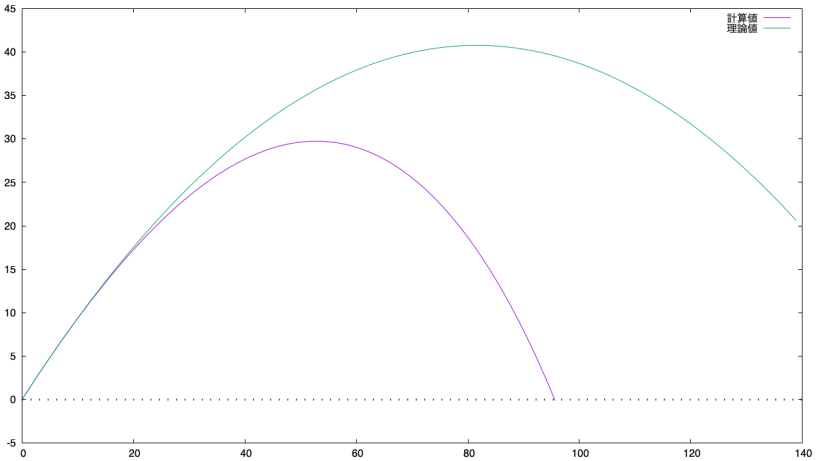

図 8　$\tau = 0.005$ の時のボールの軌道

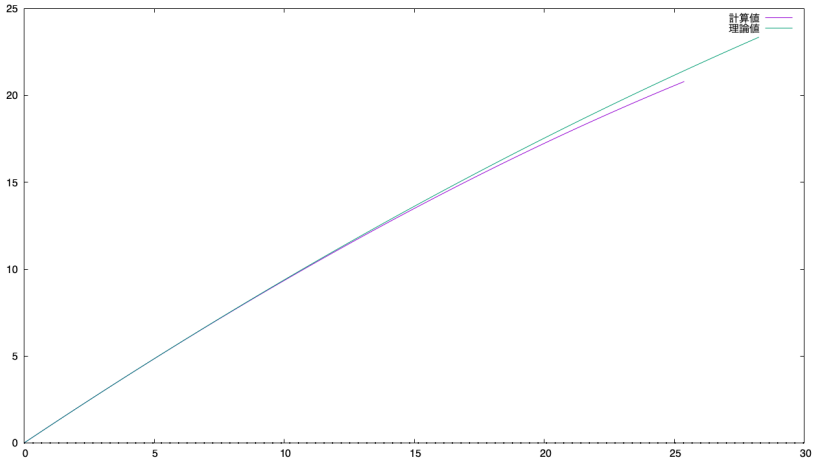

図 9　$\tau = 0.001$ の時のボールの軌道

6　問題 5

6.1　ホイン法

$$\frac{dy}{dx} = d(x, y), y(x_0) = y_0$$

の初期値問題を考える。y(x+h) の値は、関数 y の点 x の周りのテイラー展開で、

$$\begin{aligned} y(x+h) &= y(x) + \frac{y'(x)}{1!}((x+h)-x) + \frac{y''(x)}{2!}((x+h)-x)^2 + \cdots + \frac{y^{(n)}(x)}{n!}((x+h)-x)^n + \ldots \\ &= y(x) + \frac{h}{1!}y'(x) + \frac{h^2}{2!}y''(x) + \cdots + \frac{h^n}{n!}y^{(n)}(x) + \ldots \end{aligned}$$

と表せる。これを 1 階微分の項まで用いた時、

$$y(x+h) = y(x) + \frac{h}{1!}y'(x) = y(x) + hf(x, y)$$

となる。

すなわち、オイラー法は微分方程式の解を h の一次式で近似しているといえる。

したがって、精度を上げるには、2 次以上の項を用いて、テイラー級数近似を行えば良い。

テイラー展開を二次項まで利用した場合を考える。

y(x+h) の近似は、

$$y(x+h) \approx y(x) + \frac{h}{1!}y'(x) + \frac{h^2}{2!}y''(x)$$

と表せる。

ここで、f(x,y) の x による微分を、適当な刻み幅 kh で差分法により計算すると、

$$\begin{aligned}\frac{df(x,y)}{dx} &\approx \frac{f(x+kh, y(x+kh)) - f(x,y)}{kh} \\ &\approx \frac{f(x+kh, y(x)+khf(x,y)) - f(x,y)}{kh}\end{aligned}$$

と書けるので、これを元の式に代入すると、

$$y(x+h) = y(x) + (1 - \frac{1}{2k})hf(x,y) + \frac{h}{2k}f(x+kh, y(x)+khf(x,y))$$

という公式が得られる。

この公式において、k=1 と置いた時に得られる

$$y(x+h) = y(x) + \frac{h}{2}f(x,y) + \frac{h}{2}f(x+kh, y(x)+hf(x,y))$$

という公式をホイン法または、改良オイラー法と呼ぶ。

6.2 ホイン法の実装

今回の実験の、初期値問題は、

$$\begin{aligned}&\frac{dv}{dt} = f(v) = \frac{1}{m}F_a(v) - g\hat{y}, F_a(v) = -\frac{1}{2}C_d\rho A|v|v \\ &\frac{dv}{dt} = v\end{aligned}$$

である。

ホイン法で、$v(t+\tau)$ の値を求める式は、

$$\begin{aligned}v(t+\tau) &= v(t) + \frac{\tau}{2}(k_1 + k_2) \\ k_1 &= \frac{dv}{dt} = f(v) \\ k_2 &= \frac{d^2}{dt^2}v = f(v(t) + \tau f(v)) = f(v + tk_1)\end{aligned}$$

である。

$r(t+\tau)$ を求める式は、

$$r(t+\tau) = r(t) + \tau v(t)$$

である。

ホイン法を実装したプログラムを、Listing2 に示す。

このプログラムは、上で求めた、ホイン法による漸化式にしたがって、各時刻の、速度と位置の値を計算している。

Listing 2　ホイン法による野球ボールの軌道計算のプログラム

```
1  // baseball.cpp: オイラー法を用いて野球ボールの軌道を計算するプログラム
2  #include "NumMeth.h"
3
4  using namespace std;
5  int main() {
6  //* ボールの初期位置及び初期速度を設定する.
7  double y1, speed, theta;
8  double r1[2+1], v1[2+1], r[2+1], v[2+1], accel[2+1]; cout << "高さの初期値メートル
       ()␣:␣"; cin >> y1;
9  r1[1] = 0;
10 r1[2] = y1; // 初期位置ベクトル
11 cout << "初期速度(m/s)␣:␣"; cin >> speed;
12 cout << "初期角度度()␣:␣"; cin >> theta;
13 const double pi = 3.141592654;
14 v1[1] = speed*cos(theta*pi/180); // 初期速度 (x)
15 v1[2] = speed*sin(theta*pi/180); // 初期速度 (y)
16 r[1] = r1[1];
17 r[2] = r1[2]; // 初期位置および初期速度を設定
18 v[1] = v1[1];
19 v[2] = v1[2];
20 //* 物理パラメータを設定質量(, Cd 値など)
21 double Cd = 0.35;
22 double area = 4.3e-3;
23 double grav = 9.81;
24 double mass = 0.145;
25 double airFlag, rho;
26 cout << "空気抵抗あり(:1,␣なし:0)␣:␣";
27 cin >> airFlag; if( airFlag == 0 )
28 rho = 0; // 空気抵抗なし else
29 rho = 1.2; // 空気の密度(kg/m^3)
30 double air_const = -0.5*Cd*rho*area/mass; // 空気抵抗定数
31 //* ボールが地面に着くまで, あるいは最大の刻み数になるまでループ
32 double tau;
33 cout << "時間刻み␣τ秒()␣:␣"; cin >> tau;
34 int iStep, maxStep = 1000; // 最大の刻み数
35 double *xplot, *yplot, *xNoAir, *yNoAir;
36 xplot = new double [maxStep + 1];
37 yplot = new double [maxStep + 1];
38 xNoAir = new double [maxStep + 1];
39 yNoAir = new double [maxStep + 1];
40
41 double max_y = 0;
42 double k_1[2+1];
43 double k_2[2+1];
44
45 for( iStep=1; iStep<=maxStep; iStep++ ) {
46     //* プロット用に位置計算値および理論値()を記録する
47     xplot[iStep] = r[1]; // プロット用に軌道を記録
48     yplot[iStep] = r[2];
49     double t = ( iStep-1 )*tau; // 現在時刻
50     xNoAir[iStep] = r1[1] + v1[1]*t; // 位置(x)
51     yNoAir[iStep] = r1[2] + v1[2]*t - (0.5*grav*t*t); // 位置(y) ここに位置を求める式
```

```
        を書く
52      //* ボールの加速度を計算する
53      // 空気抵抗無次元() air_const
54      // 投射物の横断面積(m^2) area
55      // 重力加速度(m/s^2) g
56      // 投射物の質量(kg) mass
57      //* ホイン法を用いて，新しい位置および速度を計算する
58
59      double normV = sqrt( v[1]*v[1] + v[2]*v[2] );
60      accel[1] = air_const*normV*v[1]; // 空気抵抗
61      accel[2] = air_const*normV*v[2]; // 空気抵抗
62      accel[2] -= grav; // 重力
63
64      k_1[1] = accel[1];
65      k_1[2] = accel[2];
66
67      normV = sqrt( (v[1]+tau*k_1[1])*(v[1]+tau*k_1[1]) + (v[2]+tau*k_1[2])*(v
            [2]+tau*k_1[2]) );
68      accel[1] = air_const*normV*(v[1]+tau*k_1[1]); // 空気抵抗
69      accel[2] = air_const*normV*(v[2]+tau*k_1[2]); // 空気抵抗
70      accel[2] -= grav; // 重力
71
72      k_2[1] = accel[1];
73      k_2[2] = accel[2];
74
75      // は、の一次式なので、の数値計算は一次のテイラー展開（オイラー法）で十分。rtr
76      r[1] = r[1] + tau*v[1];
77      r[2] = r[2] + tau*v[2];
78
79      v[1] = v[1] + 0.5*tau*k_1[1] + 0.5*tau*k_2[1];
80      v[2] = v[2] + 0.5*tau*k_1[2] + 0.5*tau*k_2[2];
81
82      //* ボールが地面に着いたら(y < 0)ループを抜ける
83      /*
84      if 文を使って書く
85      */
86      if(yplot[iStep]<0) {
87          break;
88      }
89
90      if(r[2]>max_y) {
91          max_y = r[2];
92      }
93  }
94  //* 最大到達高さと滞空時間を表示するこのままだと正しい結果ではない()
95  cout << "最大到達高さは" << max_y << "メートル" << endl;
96  // 滞空時間の表示をここに書く
97  cout << "滞空時間は" << tau*iStep << "秒" << endl;
98  //* プロットする変数を出力する
99  // xplot, yplot xNoAir, yNoAir
100 ofstream xplotOut("xplot.txt"), yplotOut("yplot.txt"),
101 xNoAirOut("xNoAir.txt"), yNoAirOut("yNoAir.txt");
102 int i;
```

```
103
104 for( i=1; i<=iStep; i++ ) {
105     xplotOut << xplot[i] << endl;
106     yplotOut << yplot[i] << endl;
107 }
108
109 for( i=1; i<=iStep; i++ ) {
110     xNoAirOut << xNoAir[i] << endl;
111     yNoAirOut << yNoAir[i] << endl;
112 }
113
114 ofstream xyplotOut("xyplot.txt"), xyNoAirOut("xyNoAir.txt");
115 xyplotOut << "#X␣Y" << endl;
116 xyNoAirOut << "#X␣Y" << endl;
117
118 for( i=1; i<=iStep; i++ ) {
119     if (i>0 && yplot[i] == 0){
120         continue;
121     }
122     xyplotOut << xplot[i] << "␣" << yplot[i] << endl;
123 }
124
125 for( i=1; i<=iStep; i++ ) {
126     if (i>0 && yplot[i] == 0){
127         continue;
128     }
129     xyNoAirOut << xNoAir[i] << "␣" << yNoAir[i] << endl;
130 }
131
132 delete [] xplot;
133 delete [] yplot;
134 delete [] xNoAir;
135 delete [] yNoAir; // メモリを開放
136 }
```

6.3 ホイン法の結果

時間刻み τ は、0.2、0.15、0.1、0.05、0.01、0.005、0.001 の 7 つを選んだ。

初期値は、高さ 0m、初期速度 40m/s、投射角度 45 度、空気抵抗ありとした。

$\tau = 0.2$ の時のグラフを、図 10 に示す。

$\tau = 0.15$ の時のグラフを、図 11 に示す。

$\tau = 0.1$ の時のグラフを、図 12 に示す。

$\tau = 0.05$ の時のグラフを、図 13 に示す。

$\tau = 0.01$ の時のグラフを、図 14 に示す。

$\tau = 0.005$ の時のグラフを、図 15 に示す。

$\tau = 0.001$ の時のグラフを、図 16 に示す。

x 軸は、ボールの水平方向への移動距離 (m) を、y 軸は、ボールの軌道の高さ (m) を表す。

理論値というラベルがついた曲線は、空気抵抗なしの場合の、解析的に計算したボールの軌道である。

実測値というラベルがついた曲線は、オイラー法により求めた、ボールの軌道である。

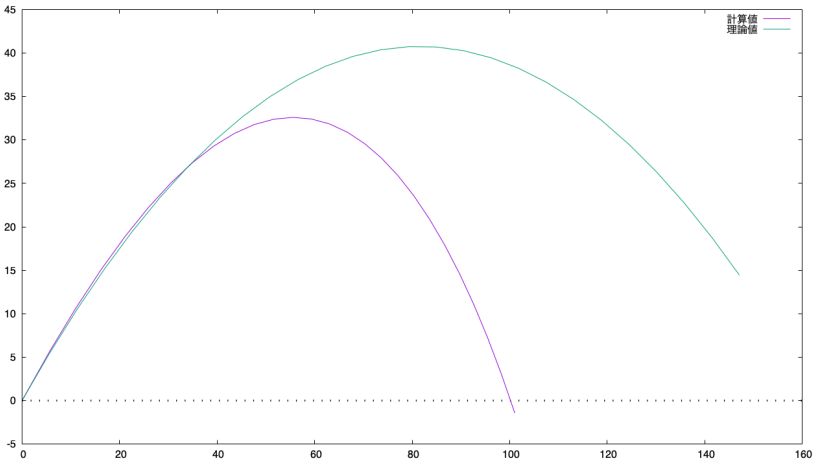

図 10　ホイン法で求めた $\tau = 0.2$ の時のボールの軌道

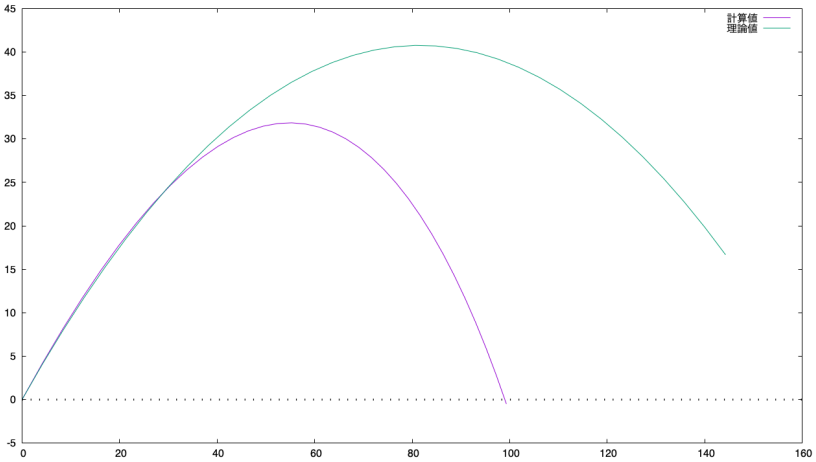

図 11　ホイン法で求めた $\tau = 0.15$ の時のボールの軌道

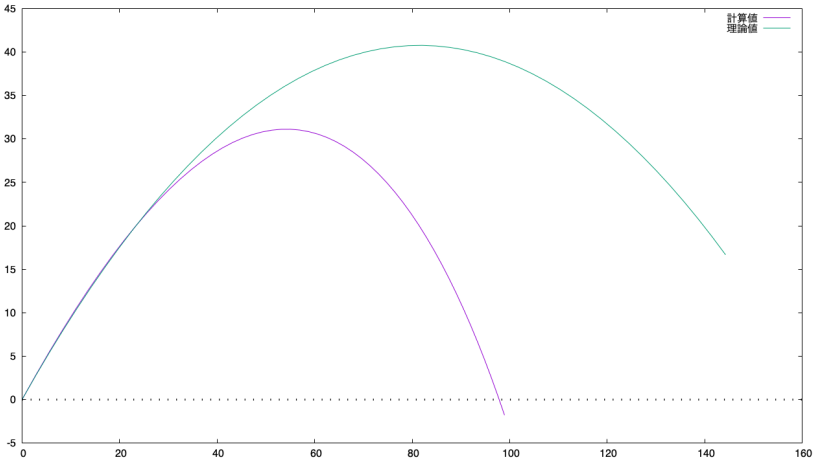

図 12　ホイン法で求めた $\tau = 0.1$ の時のボールの軌道

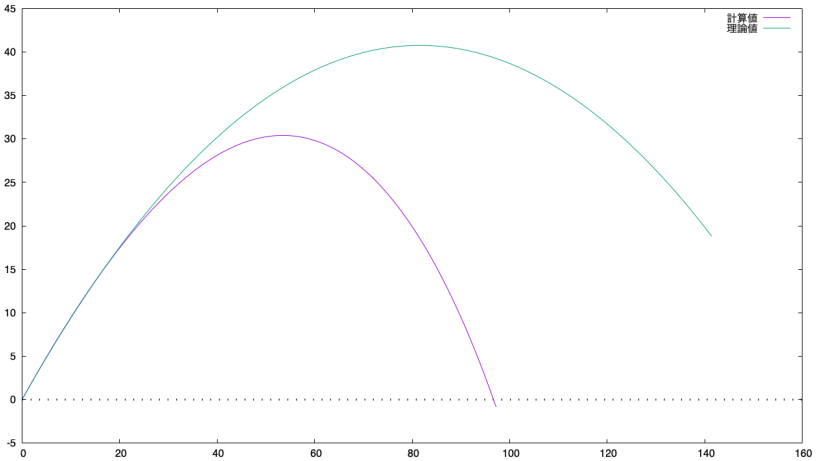

図 13　ホイン法で求めた $\tau = 0.05$ の時のボールの軌道

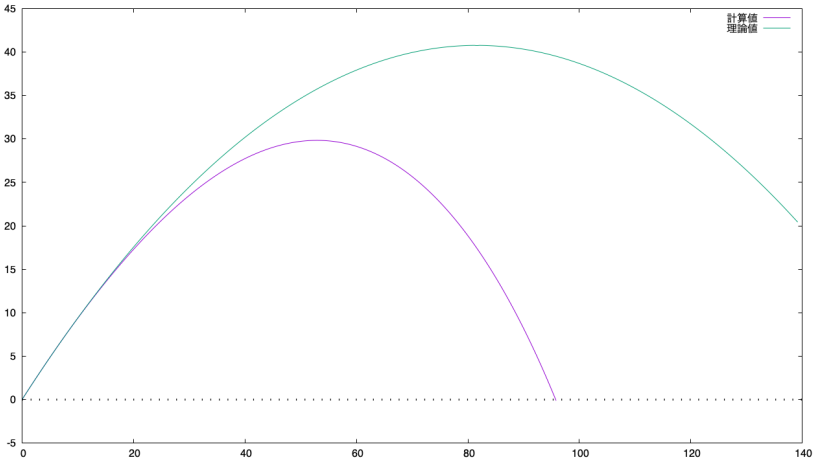

図 14　ホイン法で求めた $\tau = 0.01$ の時のボールの軌道

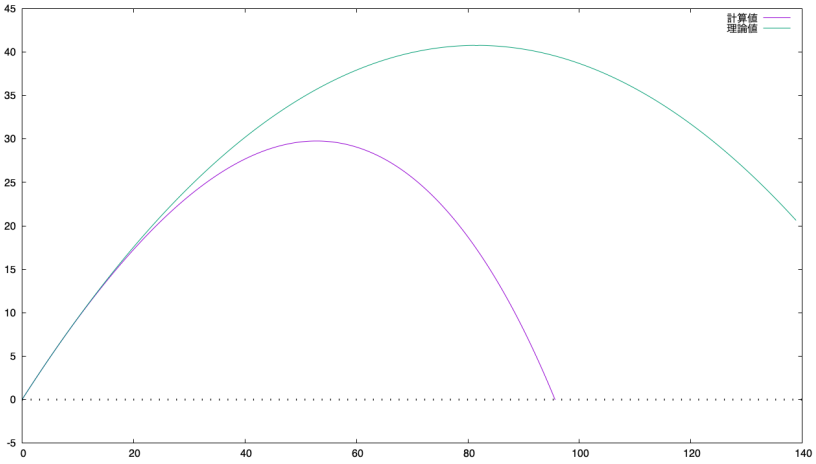

図 15　ホイン法で求めた $\tau = 0.005$ の時のボールの軌道

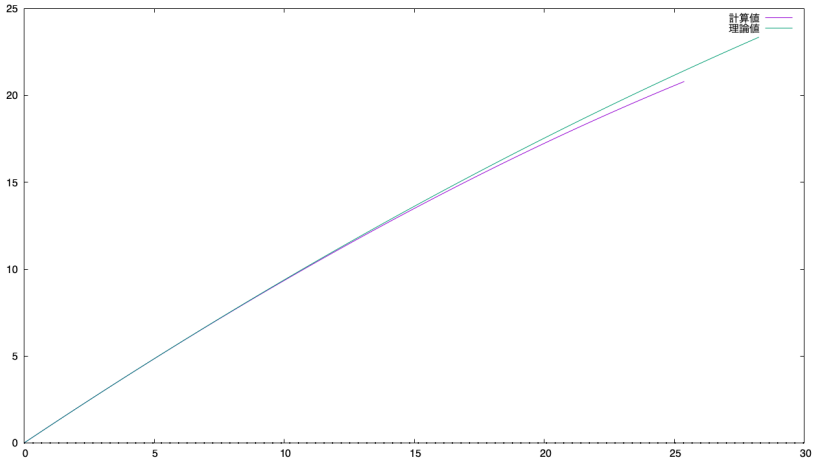

図 16　ホイン法で求めた $\tau = 0.001$ の時のボールの軌道

7　考察

グラフに理論値として示した曲線は、空気抵抗なしの場合のボールの軌道である。一方、実測値として示した曲線は、空気抵抗ありの場合のボールの軌道である。二つの曲線を比較すると、随分飛距離に差があることがわかる。2 つの大きな違いは、空気抵抗があるか、ないかであるから、この結果に影響を及ぼしたのは、空気抵抗であろうと考えられる。

果たして、空気抵抗の影響とは、こんなに大きいものだろうか。空気抵抗の影響の大きさを、計算式から考えてみる。

ここで、オイラー法による、速度の計算式は、

$$v_{i+1} = v_i + \tau \cdot (\frac{1}{m} F_a(v_i) - g\hat{y})$$
$$F_a(v) = -\frac{1}{2} C_d \rho A |v| v$$

である。

この式の大まかな意味を考えると、F_a は負の値であるから、v_{i+1} は、v_i から、F_a の値に比例した大きさの値を引いた値、であることがわかる。

ここで、F_a の式は、変数の $|v|v$ に係数をかけた式と見ることができるが、$|v|$ は、プログラム中、L2 ノルムで定義されているから、直角三角形の斜辺が常に、3 辺のうち、長さが最大であることを考えると、$|v| > v$ が成り立つ。したがって、$|v|v > v^2$ が成り立つ。

よって、単純化して考えると、F_a は、v の 2 乗に比例した値よりも大きい値を出力する関数である。すなわち、空気抵抗は、v の 2 乗に比例した値よりも大きい値であると考えることができる。

ゆえに、このような大きな値が空気抵抗として、ボールの速度を減衰させるから、空気抵抗がない場合と、空気抵抗がある場合の飛距離に、大きな差が出たのではないだろうか。

参考文献・引用文献

参考文献

[1] 宮里 智樹 . “2023-知能情報基礎演習 4-1 微分方程式の数値解法” . 2023 知能情報基礎演習 4 授業資料 .

[2] 大竹 豊, AN Qi . “微分方程式の数値解法” . 東京大学工学部精密工学科プログラミング応用 I・II 授業資料 . http://www.den.t.u-tokyo.ac.jp/ad_prog/ode/),2023 年 12 月 7 日) .