情報工学実験 I

C++/Octave による微分方程式の数値解法

055702B 池野谷克俊

提出日:2006 年 7 月 7 日 金曜日

1　解答及び考察

問題 1

実際にオイラー法を適用しようとすると, 幾つかの問題のために精度が悪く, 使われることはほとんどない. オイラー法の問題点を説明せよ.

　オイラー法を用いた時に, 刻み幅 (τ) の値でかなりの違いがでてくる.
τ の値を大きくすると未知関数のグラフの精度がかなり落ちることになり, τ の値を小さくすると計算量が増えすぎることになる.
オイラー法では計算点が空間に固定されているため, 多相流れや自由表面を持つ流れ, 移動境界問題など, 複雑な問題を自然に扱うことが出来ない.
以上がオイラー法の問題点と言える.

問題 2

微分方程式

$$\begin{aligned} \frac{d}{dt}x &= -x \\ x_{(0)} &= 1 \end{aligned}$$

の解を導出して確かめよ.

$$\begin{aligned} \frac{d}{dt}x &= -x \\ -\frac{1}{x}dx &= 1dt \end{aligned}$$

ここで両辺を積分する

$$\begin{aligned} \int -\frac{1}{x}dx &= \int 1dt \\ -log|x| + C_1 &= t + C_2 \\ log|x| &= -(t + C_2 - C_1) \end{aligned}$$

$C_2 - C_1$を C_3とおく

$$\begin{aligned} log|x| &= -(t + C_3) \\ x_{(t)} &= e^{-(t+C_3)} \end{aligned}$$

$x_{(0)} = 1$ であるから

$$\begin{aligned} e^{-C_3} &= 1 \\ C_3 &= 0 \end{aligned}$$

ゆえに

$$x_{(t)} \quad = \quad e^{-t}$$

問題 3 で用いるプログラムのソースコードは以下のものである.

ソースコード:baseball.cpp

```
// 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
#include "NumMeth.h"

int main() {

  //* ボールの初期位置及び初期速度を設定する.
  double y1, speed, theta;
  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;
  r1[1] = 0; r1[2] = y1;    // 初期位置ベクトル
  cout << "初期速度 (m/s) : "; cin >> speed;
  cout << "初期角度 (度) : "; cin >> theta;
  const double pi = 3.141592654;
  v1[1] = speed*cos(theta*pi/180);   // 初期速度 (x)
  v1[2] = speed*sin(theta*pi/180);   // 初期速度 (y)
  r[1] = r1[1]; r[2] = r1[2];   // 初期位置および初期速度を設定
  v[1] = v1[1]; v[2] = v1[2];

  //* 物理パラメータを設定 (質量, Cd 値など)
  double Cd = 0.35;        // 空気抵抗 (無次元)
  double area = 4.3e-3;    // 投射物の横断面積 (m^2)
  double grav = 9.81;      // 重力加速度 (m/s^2)
  double mass = 0.145;     // 投射物の質量 (kg)
  double airFlag, rho;
  cout << "空気抵抗 (あり:1, なし:0) : "; cin >> airFlag;
  if( airFlag == 0 )
    rho = 0;   // 空気抵抗なし
  else
    rho = 1.2;   // 空気の密度 (kg/m^3)
  double air_const = -0.5*Cd*rho*area/mass;   // 空気抵抗定数

  //* 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
  double tau;
  cout << "時間刻み τ (秒) : "; cin >> tau;
  int iStep, maxStep = 1000;   // 最大の刻み数
  double *xplot, *yplot, *xNoAir, *yNoAir;
  xplot = new double [maxStep + 1];
  yplot = new double [maxStep + 1];
  xNoAir = new double [maxStep + 1];
  yNoAir = new double [maxStep + 1];
  for( iStep=1; iStep<=maxStep; iStep++ ) {

    //* プロット用に位置 (計算値および理論値) を記録する
    xplot[iStep] = r[1];   // プロット用に軌道を記録
    yplot[iStep] = r[2];
    double t = ( iStep-1 )*tau;   // 現在時刻
    xNoAir[iStep] = r1[1] + v1[1]*t; // 位置 (x)
    yNoAir[iStep] = r1[2] + v1[2]*t - 0.5*grav*t*t; // 位置 (y)

    //* ボールの加速度を計算する
```

```
    double normV = sqrt( v[1]*v[1] + v[2]*v[2] );
    accel[1] = air_const*normV*v[1];   // 空気抵抗
    accel[2] = air_const*normV*v[2];   // 空気抵抗
    accel[2] -= grav;                  // 重力

    //* オイラー法を用いて，新しい位置および速度を計算する

    v[1] = v[1] + accel[1]*tau;
    v[2] = v[2] + accel[2]*tau;
    r[1] = r[1] + v[1]*tau;
    r[2] = r[2] + v[2]*tau;


    //* ボールが地面に着いたら (y < 0) ループを抜ける

    if(r[2] < 0){
      xplot[iStep] = r[1];
      yplot[iStep] = 0;
      break;
    }
  }

  //* 最大到達距離と滞空時間を表示する
  cout << "最大到達距離は" << r[1] << "メートル" << endl;
  cout << "滞空時間は" << (iStep-1)*tau << "秒" << endl;

  //* プロットする変数を出力する
  //   xplot, yplot xNoAir, yNoAir
  ofstream xplotOut("xyplot.txt"), yplotOut("xyplot.txt"),
    xNoAirOut("xNoAir.txt"), yNoAirOut("yNoAir.txt");

  int i;
    for( i=1; i<=iStep; i++ ) {
    //GnuPlot で出力しやすいように,"xplot[i] yplot[i]"
    //の形でデータが書き込まれるように設定
    xplotOut << xplot[i] << " " << yplot[i] << endl;

  }
  for( i=1; i<=iStep+1; i++ ) {
    xNoAirOut << xNoAir[i] << endl;
    yNoAirOut << yNoAir[i] << endl;
  }

  delete [] xplot, yplot, xNoAir, yNoAir;  // メモリを開放

}
```

問題 3

ボールの軌道のグラフを gnuplot で出力せよ. なお, 時間の刻み τ は $0 < t <= 2$ の間で 5 個選ぶこと. 補助的に Octave を用いてもよい.

高さの初期値を 1m, 初期速度を 30m, 初期角度を 45 度, 空気抵抗は無しに固定して τ の値が 0.01, 0.1, 0.8, 1.4, 2 の場合を出力する.

1. $\tau = 0.01$ の時

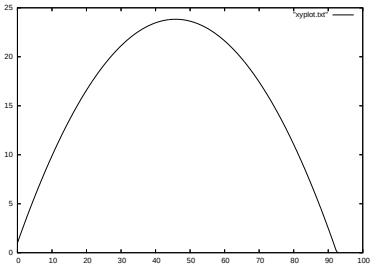

図 1: $\tau = 0.01$

2. $\tau = 0.1$ の時

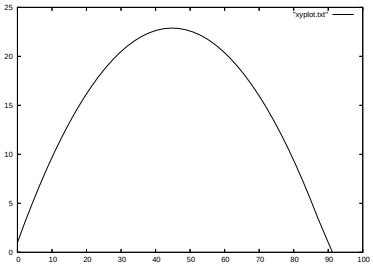

図 2: $\tau = 0.1$

3. $\tau = 0.8$ の時

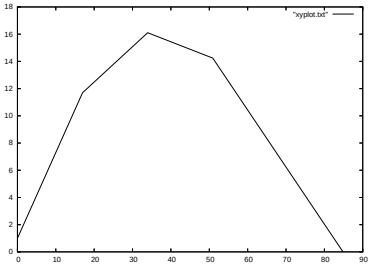

図 3: $\tau = 0.8$

4. $\tau = 1.4$ の時

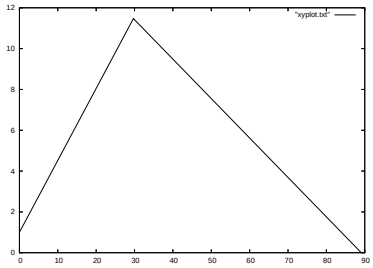

図 4: $\tau = 1.4$

5. $\tau = 2$ の時

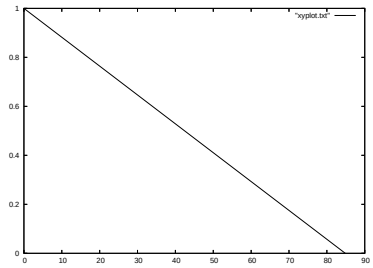

図 5: $\tau = 2$

問題 4

時間刻み τ の設定によって, 計算結果が大きく異なることが確認できるが, その理由を考察せよ.

時間刻みが変わるということは観測する地点間の差が大きくなるということである. 観測地点を直線で結んでことによってグラフを描いているので, 時間刻みを大きくすると, 観測地点の間にある観測されない点が増え, それだけ誤差が大きくなる. 例えば滞空時間が 4 秒の場合で時間刻みを 3 秒にすると, ボールが上昇している範囲は観測されないことになる.

問題 5

オイラー法よりも高精度な数値計算アルゴリズムについて調べよ. 余力のある人は, アルゴリズムを実装しその結果をオイラー法と比較してみよ.

オイラー法よりも高精度な数値計算アルゴリズムとして, ルンゲ・クッタ法について調べた.

- ルンゲ・クッタ法

導関数 $\frac{dy}{dx} = f_{(x,y)}$ が与えられているとき, 関数 y を求めることを目的とする.

ルンゲ・クッタ法の基本的な, 大まかな手順はオイラー法と同じであるが, 傾きの求め方を調整・工夫する.

例えば,y_3 の値は, 位置 x_2 における y_2 の値が前回計算で求められており, 位置 x_2 における傾きは与えられた導関数 $\frac{dy}{dx}$ から調整した傾き k を算出することと定めるので, 刻み幅 h を調整した傾き k に乗算することにより,y_2 にこれを加えて,y_3 の値とする. 以下同様の手順を限界まで繰り返す.

y_3 の値 = 位置 x_2 における y_2 の値 + 位置 x_2 における調整した傾き k * 刻み幅 h

この手順において,x_0 における y_0 の値を初期値として与える必要がある. この初期値を与えれば, 全ての x に対して y の値が算出可能である. なお, 初期条件の個数としては, 一階の微分方程式では一つ, 二階の微分方程式では二つ必要.

- 計算手順

1. $x_1 = x_0 + h$,$y_1 = y_0 + k_0 \cdot h$
2. $x_2 = x_1 + h$,$y_2 = y_1 + k_1 \cdot h$
3. $x_3 = x_2 + h$,$y_1 = y_2 + k_2 \cdot h$
   - .
   - .
   - .
   - .

ただし, 調整する傾き k_n は以下のようにして求める.
$k_n = (k_1 + 2k_2 + 2k_3 + k_4)/6$

ここで,

$k_1 = f_{(x_n+0/1, y_n+000/1)}$
$k_2 = f_{(x_n+h/2, y_n+k_{n1}/2)}$
$k_3 = f_{(x_n+h/2, y_n+k_{n2}/2)}$
$k_4 = f_{(x_n+h/1, y_n+k_{n3}/1)}$
の計算を付加的に行う必要がある.

必要回数にて, 計算を終了する.

2 感想

今回の課題はオイラー法を用いて放物運動を計算するという内容であったが, C++で書かれたプログラムを解析するのがけっこう難しかった.
まだ一回目のレポートでこれから難易度が上がっていくと思うが, しっかりこなしていきたい.

参考文献

[1] 数値計算の説明ルンゲ・クッタ法
http://www.ny.airnet.ne.jp/satoh/aznprungekuttamethod.htm