情報工学実験
+C+/Octaveによる微分方程式の数値解法

担当教員名:宮里智樹
氏名:澤岨千明
学籍番号:0757230G

実験日:2008/06/13
提出日 2008/06/20

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
使われる事はほとんどない。オイラー法の問題点を説明せよ。

解答

オイラー法では、時間刻み τ の値によってその解が異なる。
時間刻み τ の値によって、その時点での速度を求めるのでどうしても誤差ができてしまう。
この値が細かい程、誤差は少なくなるがその分計算量が増え、
この値を大きめに取ると計算量は減るが誤差が大きくなるという問題がある。

問題 2

微分方程式（ ）の解を導出して確かめよ。これをオイラー法で計算すると
$x_{n+1} = x_n - x_n\tau$　となる。

解答

$$\frac{d}{dt}x = -x, \quad x(0) = 1 \ (t = 0, x = 1)$$

$$\frac{1}{x}dx = -1 \cdot dt$$

ここで両辺を積分すると

$$\int \frac{1}{x}dx = -1 \int dt$$

$$log_e x + C_1 = -1 \cdot t + C_2$$

$$log_e x = -t + A \ , \ A = C_1 + C_2$$

$$x = e^{(-t+A)} = e^{-t} \cdot e^A$$

$$1 = e^0 \cdot e^A = 1 \cdot e^A = e^A$$

よって

$$x(t) = e^{-t}$$

問題 3

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

baseball.cpp のソース

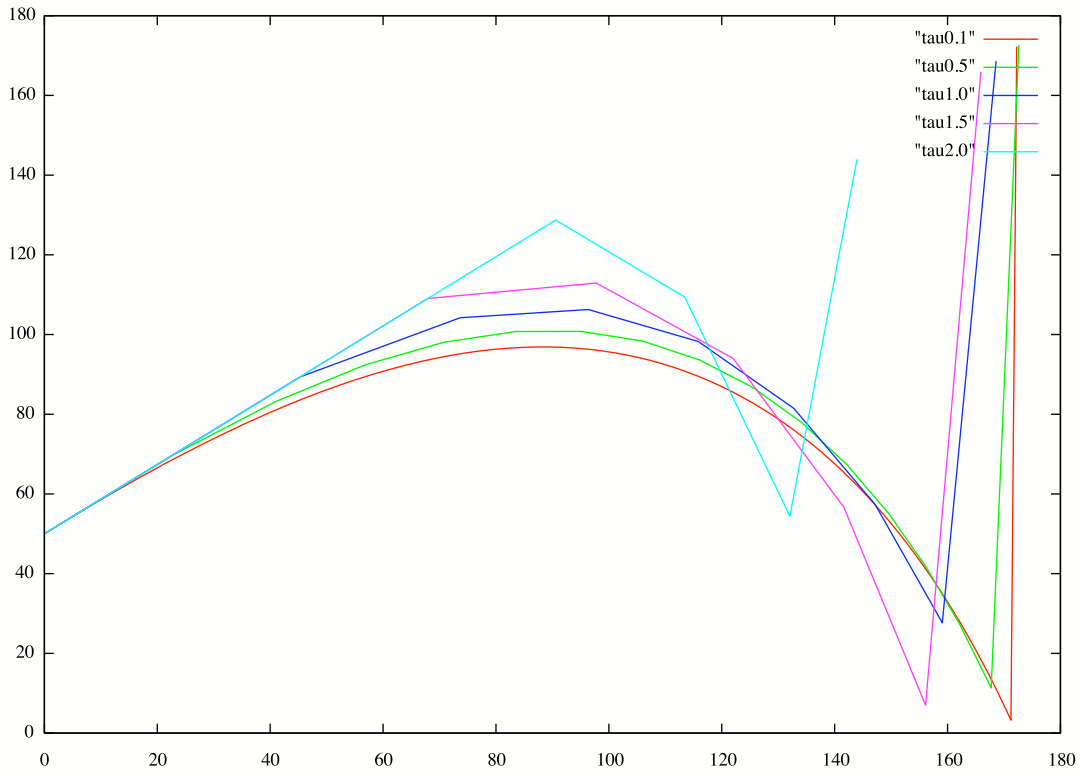
*一部改変

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


    //* 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 + 1 ] = r[1];
yplot[ iStep + 1 ] = r[1];
break;
}
}


    //* 最大到達高さと滞空時間を表示する
cout ≪ “最大到達高さは” ≪ r[1] ≪ “メートル” ≪ endl;
//* 滞空時間の表示をここに書く
cout ≪ “fly lange : “ ≪ r[1] ≪ “m” ≪ endl;
cout ≪ “fly time : “ ≪ (iStep)*tau ≪ “sec” ≪ endl;
cout ≪ “writting opelations” ≪ endl;



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


    int i;
```

```
for( i=1; i¡=iStep+1; i++ ) {
xplotOut ≪ xplot[i] ≪ endl;
yplotOut ≪ 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

    }
```

結果を次の図に示す。



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。

解答

問 1 でも述べた様に、時間刻み τ の値によって精度が大きくかわるので
計算結果も大きく異なるのである。

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。

解答

ルンゲ・クッタ法

この方法は数値計算のときに「中間点」を使って精度を上げるもので、計算も簡単なため広く使われている。
中間点を求め使うことで誤差を少なくする。

参考 URL

微分方程式の数値計算ールンゲ・クッタ法
http://www.sm.rim.or.jp/ shishido/bibun2.html