計算機実験 1

自然情報学講座 高須夫悟

takasu@ics.nara-wu.ac.jp

1. 微分方程式の数値解 その1
2. 微分方程式の数値解 その2
3. 数値積分
4. 乱数生成
5. モンテカルロ法

毎回レポート提出

微分方程式

自然科学ではあらゆる分野で微分方程式が用いられる

その理由は、状態の時間変化（微分）を記述する法則の多くが、
状態 x、時間 t、

$$\frac{dx}{dt},\quad \frac{d^2x}{dt^2},\quad \frac{d^3x}{dt^3}\cdots$$

の間の関係として定まるからである

重力による自由落下

$$m\frac{d^2x}{dt^2} = -mg$$

摩擦力のあるバネの振動

$$m\frac{d^2x}{dt^2} = -b\frac{dx}{dt} - cx$$

放射性元素の崩壊

$$\frac{dA}{dt} = -kA$$

常微分方程式

微分方程式の定義

変数 t とその関数 $x(t)$ と、その有限階の導関数についての関数

$$\boxed{F(t, x, x', x'', \cdots, x^{(n)}) = 0} \qquad x' = \frac{dx}{dt} \quad x'' = \frac{d^2x}{dt^2} \qquad x^{(n)} = \frac{d^nx}{dt^n}$$

を、$x(t)$ に関する微分方程式と呼ぶ。

導関数の最高階数 n をこの微分方程式の階数という
この式を満たす関数 $x(t)$ を、この微分方程式の解という

t を独立変数、$x(t)$ を従属変数と呼ぶこともある

独立変数の数が 1 つの時、偏微分方程式と区別するため常微分方程式と呼ぶ

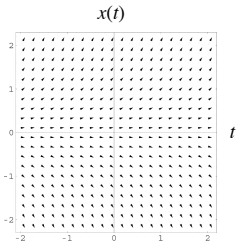
1 階常微分方程式

$F(t, x, x') = 0$ が x' について陽に解けている場合（正規型）

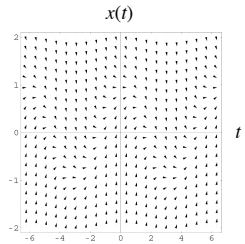
$$\boxed{\frac{dx}{dt} = f(t, x)}$$

幾何学的には、関数 $x(t)$ の勾配が $f(t, x)$ に等しいことを意味する。

$f(t, x) = x$



$f(t, x) = x + \cos t$



常微分方程式は一般に無数の解を持つ

解の存在

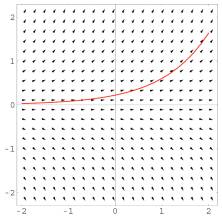
常微分方程式の初期値問題

$$\frac{dx}{dt} = f(t,x)$$
$$x(t_0) = x_0$$

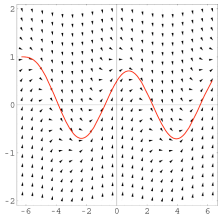
関数 $f(t,x)$ が Lipschitz 条件を満たすなら
初期値問題の解は一意に決まる

t に無関係な $K>0$ が存在して次を満たす　$|f(t,x_1) - f(t,x_2)| \le K|x_1 - x_2|$

$f(t,x) = x$



$f(t,x) = x + \cos t$



常微分方程式の初等解法（求積法）

与えられた常微分方程式から、四則演算、微分・積分、関数の合成・逆関数などの組み合わせによって解を求める方法

1 階常微分方程式（正規型）の初等解法

変数分離型

$$\frac{dx}{dt} = f(t,x) = g(t)h(x) \qquad \frac{dx}{dt} = t(1-x^2)$$

同次型

$$\frac{dx}{dt} = f(t,x) = \varphi\left(\frac{x}{t}\right) \qquad t\frac{dx}{dt} = x + \sqrt{t^2 + x^2}$$

線形型

$$\frac{dx}{dt} = f(t,x) = p(t)x + q(t) \qquad \frac{dx}{dt} = -\frac{x}{1+t} + \cos t$$

常微分方程式の数値解法

Lipschitz 条件により、初期値問題の解の存在が保証されていても、解析解を求めることは一般に困難（求積法で解ける場合もある）

$$\frac{dx}{dt} = f(t,x)$$

その場合、初期値問題を数値的に解くしかない（数値解）

数値解法の基本的な考えは、連続量 t を差分化し、微分を差分式で置き換えるところにある（t の刻み幅 $h>0$ を設定し、差分化）

$$t \to t_n = t_0 + nh \quad (n = 0, 1, 2, \cdots)$$

$$\frac{dx}{dt} \to \frac{x(t_n+h) - x(t_n)}{h}$$

オイラー法

$$\frac{dx}{dt} = f(t,x)$$

$$\Downarrow$$

$$\frac{x(t_n+h) - x(t_n)}{h} = f(t_n, x(t_n)) \Longrightarrow x(t_{n+1}) = x(t_n+h) = x(t_n) + hf(t_n, x(t_n))$$

$x(t_n) = x_n$　と表記すると　$\boxed{x_{n+1} = x_n + hf(t_n, x_n)}$

初期値 x_0 を与えると、漸次 $x_1, x_2, x_3, \ldots$ が決まる。

この方法をオイラー法 (Euler) という。

オイラー法の精度

テイラー展開により

$$x(t_{n+1}) = x(t_n + h) = x(t_n) + hx'(t_n) + \frac{h^2}{2}x''(\xi) \quad t_n < \xi < t_{n+1}$$

よって t を h だけ進める際の誤差は、

$$x(t_{n+1}) - \underbrace{\{x_n + hf(t_n, x_n)\}}_{\text{オイラー法による値}} = \frac{h^2}{2}x''(\xi) = O(h^2) \quad h \to 0$$

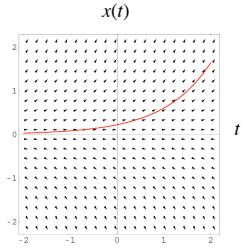
真の値

区間 $[\,t_0, t_0 + T\,]$ を刻み幅 h で分割したとき、終端 $t_0 + T$ での誤差の累積は

$$O(h^2) \times \frac{T}{h} = O(h)$$

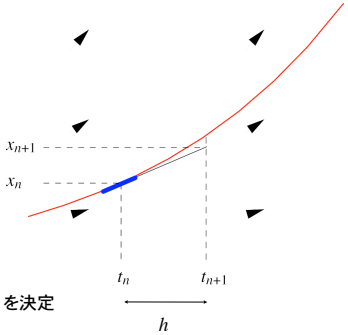
刻み幅 h を 1/10 にすれば累積誤差もだいたい 1/10 になる（計算量は 10 倍になる）

オイラー法の図解



$$\frac{dx}{dt} = f(t, x)$$

$$x_{n+1} = x_n + hf(t_n, x_n)$$



t_n における勾配 $f(t_n, x_n)$ のみを用いて x_{n+1} を決定

プログラム例（オイラー 1 変数）

1) 関数およびパラメータ（刻み幅 h や初期値など）の設定

2) t を h だけ進めることを繰り返す

```
#define dt 0.01        /* 時間刻み幅 */
#define STEP_MAX 1000 /* 実時間 10.0 まで */
double fn(double, double); /* 導関数 */
void euler(double, double, double*, double);

main()
{
  long step;
  double t, x, x_next;

  x=0.1;    /* 初期値設定 */

  for(step=0; step<STEP_MAX; step++){
    t = step*dt;
    euler(t, x, &x_next, dt);
    x = x_next;
  }
}
```

1 変数の場合

$$\frac{dx}{dt} = f(t, x)$$

$x(t+h)$ は、引数 x_next を介して返却（参照渡し）

関数 euler（1変数）

関数 euler を呼び出すと、独立変数 t が刻み幅 h だけ進むようにしたい。

関数に引数 t, $x(t)$, h を与えると、$t + h$ の状態 $x(t+h)$ を引数を解して関数呼び出し元に返すものとする

$$\frac{dx}{dt} = f(t, x) \quad \Longrightarrow \quad x(t+h) = x(t) + hf(t, x(t))$$

現時刻　現状態　次ステップの状態　刻み幅

```
void euler(double t, double x, double *x_out, double h)
{
  /* t, x, h から t+h の状態を計算して x_out を介して返却 */
  double tmp;
  tmp = x + h*fn(t, x);
  *x_out = tmp;
}

double fn(double t, double x)
...
```

数値解の視覚化

1) 計算結果を以下の形式でデータファイルに保存（C プログラミング）

t ↓　$x(t)$ ↓

```
0.000 1.00
0.001 1.02
0.002 1.05
...
```

$$\frac{dx}{dt} = f(t, x)$$ 1 変数の場合

スペースで区切る

2) Mathematica によるデータの読み込みと視覚化

Mathematica の起動はターミナルから　　% /usr/local/bin/mathematica &

データファイルがあるディレクトリを SetDirectory コマンドで指定

SetDirectory[“~/keisanki-1-2009/”]　　Mathematica は大文字と小文字を区別するので注意！

~ はホームディレクトリ=

続き

データファイルの読み込み

data = ReadList[“data_file”, {Real, Real}]

ReadList コマンドを用いて、ファイル名 data_file のファイルから
(x 座標, y 座標) の対でデータを数値として読み込む。読み込んだもの（リスト）に
data という名前を付ける

データをグラフに描く

ListPlot[data, PlotJoined->True, PlotRange->All]

(x 座標, y 座標) の形式のデータをグラフとして描く
PlotJoined->True はデータ各点を結ぶオプション
PlotRange->All はグラフの範囲（縦軸）はすべての範囲とするオプション

PlotRange->{0, 10} とすると、縦軸の範囲が 0 から 10 となる

Mathematica による数値解

Mathematica を用いて常微分方程式を数値的に解くことができる。C プログラミングによる結果と比較してみるとよいだろう

deq = { x’[t] == (1 - x[t])x[t], x[0] == 0.01}

Mathematica では等式は == であることに注意

右の初期値問題を deq として定義。

$$\frac{dx}{dt} = (1 - x)x$$
$$x(0) = 0.01$$

sol = NDSolve[deq, x[t], {t, 0, 10}]　　初期値問題 deq を 区間 $0 \le t \le 10$ で数値的に解き、sol と名付ける。

Plot[Evaluate[x[t]/.sol] , {t, 0, 10}]　　数値的に解いた結果を t の関数として描く。
/. は右辺のルールを左辺に代入することを表す

x[t]/.sol/.t->5　　$x(5)$ を数値的に求める。

問題 1

次の常微分方程式の初期値問題を刻み幅 $h = 0.1$ 及び $h = 0.05$ のオイラー法で解き、解析解（自分で求めよ）と比較せよ。比較は下記の形式で出力すること。

$$\frac{dx}{dt} = x \quad 0 \le t \le 1$$
$$x(0) = 1$$

	数値解		解析解	誤差
n	t_n	x_n	$\exp t_n$	$x_n - \exp t_n$
0	0.0	1.0	1.0	0.0
1	?	?	?	?
2	?	?	?	?

$$\frac{dx}{dt} = -tx + t^3 \quad 0 \le t \le 1$$
$$x(0) = 1$$

	数値解		解析解	誤差
n	t_n	x_n	Exact solution	Error
0	0.0	1.0	1.0	0.0
1	?	?	?	?
2	?	?	?	?

刻み幅 h を半分にすると解の精度はどの程度改善するか確認せよ
また数値解をグラフとして視覚化せよ

多変数の常微分方程式

多変数 1 階常微分方程式

$$\frac{dx_1}{dt} = f_1(t, x_1, x_2, \cdots, x_n)$$
$$\frac{dx_2}{dt} = f_2(t, x_1, x_2, \cdots, x_n)$$
$$\vdots$$
$$\frac{dx_n}{dt} = f_3(t, x_1, x_2, \cdots, x_n)$$

ベクトル表示

$$\frac{d\mathbf{x}}{dt} = \mathbf{f}(t, \mathbf{x})$$

$$\mathbf{x} = \begin{pmatrix} x_1(t) \\ x_2(t) \\ \vdots \\ x_n(t) \end{pmatrix}$$

$$\mathbf{f}(t, \mathbf{x}) = \begin{pmatrix} f_1(t, \mathbf{x}) \\ f_2(t, \mathbf{x}) \\ \vdots \\ f_n(t, \mathbf{x}) \end{pmatrix}$$

オイラー法

$$\boxed{\mathbf{x}(t+h) = \mathbf{x}(t) + h\mathbf{f}(t, \mathbf{x}(t))}$$

関数 euler(多変数)

関数 euler を呼び出すと、独立変数 t が刻み幅 h だけ進むようにしたい。

関数に引数 t, $\boldsymbol{x}(t), h$ を与えると、$t+h$ の状態 $\boldsymbol{x}(t+h)$ を引数を解して関数呼び出し元に返すものとする

$$\frac{d\mathbf{x}}{dt} = \mathbf{f}(t, \mathbf{x}) \quad \Longrightarrow \quad \mathbf{x}(t+h) = \mathbf{x}(t) + h\mathbf{f}(t, \mathbf{x}(t))$$

現時刻　現状態　次ステップの状態　次元　刻み幅

```
void euler(double t, double x[], double x_out[], int n,
                double h)
{
  /* t, x[], n, h から t+h の状態を計算して x_out[] を介して返却 */
  /* n 個の導関数の値を求める必要がある */
}

void derivs(double t, double x[], double derivatives[], int n)
{
  ...
}
```


多変数の導関数の計算

$$\frac{d\mathbf{x}}{dt} = \mathbf{f}(t, \mathbf{x}) \qquad \mathbf{x} = \begin{pmatrix} x_1(t) \\ x_2(t) \\ \vdots \\ x_n(t) \end{pmatrix} \qquad \mathbf{f}(t, \mathbf{x}) = \begin{pmatrix} f_1(t, \mathbf{x}) \\ f_2(t, \mathbf{x}) \\ \vdots \\ f_n(t, \mathbf{x}) \end{pmatrix}$$

n 個の導関数の値を引数を介して返す関数を定義して用いる。

時刻　状態　導関数の値　次元

```
void derivs(double t, double x[], double derivatives[], int n)
{
  /* t, x[], n から導関数を計算して derivatives[] を介して返却 */
}
```


関数 euler (多変数)の骨子

```
#define DIM    2  /* 次元(変数の数)の設定 */

main()
{
  ...
}

void euler(double t, double x[], double x_out[], int n, double h)
{
  int i;
  double x_tmp[DIM], derivatives[DIM];

  derivs(t, x, derivatives, n);  /* 導関数の値を求める */

  for(i=0; i<DIM; i++)           /* Euler 法 */
    x_tmp[i] = x[i] + h*derivatives[i];

  for(i=0; i<DIM; i++)
    x_out[i] = x_tmp[i];         /* 引数を介して演算結果を返却 */
}
```

Euler 法プログラム例（多変数）

```
#define DIM 2          /* 次元 */
#define dt 0.01        /* 時間刻み幅 */
#define STEP_MAX 1000 /* 実時間 10.0 まで */
void derivs(double, double[], double[], int);
void euler(double, double[], double[], int, double);

main()
{
  long step;
  double t, x[DIM], x_next[DIM];

  x[0]=0.1;    /* 初期値設定 */
  x[1]=0.2;
  for(step=0; step<STEP_MAX; step++){
    t = step*dt;
    euler(t, x, x_next, DIM, dt);
    x[0] = x_next[0];
    x[1] = x_next[1];
  }
}
...
```


問題 2

Euler 法により下記の2変数微分方程式を数値的に解け

$$\frac{dx_1}{dt} = \frac{1}{\epsilon}(x_2 - \frac{1}{3}x_1^3 + x_1) \qquad \varepsilon > 0$$

$$\frac{dx_2}{dt} = -x_1$$

van der Pol の振動子

$\varepsilon = 1.0,\ 0.5, 0.1$ とすると解の振る舞いはどうなるか調べよ（初期値は適当）

横軸を t, 縦軸を x_1 と x_2 としたグラフと、
横軸を $x_1(t)$, 縦軸を $x_2(t)$ とした相平面上の解軌道の二つのグラフを描け

2 変数常微分方程式の視覚化

1) 計算結果を以下の形式でデータファイルに保存

t $x_1(t)$ $x_2(t)$

```
0.000 1.00 2.00
0.001 1.02 1.97
0.002 1.05 1.94
...
```

$$\frac{d}{dt}\begin{pmatrix} x_1(t) \\ x_2(t) \end{pmatrix} = \begin{pmatrix} f_1(t, x_1, x_2) \\ f_2(t, x_1, x_2) \end{pmatrix}$$

スペースで区切る

2) データファイルの読み込み

data = ReadList[“data_file”, {Real, Real, Real}] ;

ファイル data_file から (t, x_1, x_2) を対として数値データを読み込む
読み込んだものに data という名前を付ける

コマンドの最後にセミコロン ; を付けると実行結果を出力しない

続き

data は $\{\{t_0, x_0, y_0\}, \{t_1, x_1, y_1\}, \{t_2, x_2, y_2\}, \ldots\}$ という形式になっている

Transpose（転置）コマンドにより、data の成分を転置したものを dataT とする

dataT = Transpose[data];

これにより dataT は $\{\{t_0, t_1, t_2, \ldots\}, \{x_0, x_1, x_2, \ldots\}, \{y_0, y_1, y_2, \ldots\}\}$ となる

dataT の第 1 成分と第 2 成分を組み合わせて転地し、新たに
($t, x(t)$) のデータ dataX とする

dataX = Transpose[{dataT[[1]], dataT[[2]] }];

同様にして ($t, y(t)$) のデータ dataY 、($x(t), y(t)$) のデータ dataXY をつくる

dataY = Transpose[{dataT[[1]], dataT[[3]] }];

dataXY = Transpose[{dataT[[2]], dataT[[3]] }];

続き

時系列データをグラフに描くコマンド ListPlot を用いて、横軸を時刻 t 、縦軸をそれぞれ $x(t)$, $y(t)$ としたグラフを描き、名前を付ける

```
gx = ListPlot[ dataX, PlotJoined->True, PlotRange->All]
```

```
gy = ListPlot[ dataY, PlotJoined->True, PlotRange->All]
```

2 つのグラフを Show コマンドで重ね合わせて表示

```
Show[gx, gy]
```

横軸を $x(t)$, 縦軸を $y(t)$ としたグラフ（相平面上の解の軌道）を描く

```
ListPlot[ dataXY, PlotJoined->True, PlotRange->All]
```


3 変数常微分方程式の視覚化

1) 計算結果を以下の形式でデータファイルに保存

t $x(t)$ $y(t)$ $z(t)$

```
0.000 1.00 2.00 3.00
0.001 1.02 1.97 3.03
0.002 1.05 1.94 3.08
...
```

$$\frac{d}{dt}\begin{pmatrix} x_1(t) \\ x_2(t) \\ x_3(t) \end{pmatrix} = \begin{pmatrix} f_1(t, x_1, x_2, x_3) \\ f_2(t, x_1, x_2, x_3) \\ f_3(t, x_1, x_2, x_3) \end{pmatrix}$$

スペースで区切る

2) データファイルの読み込み

```
data = ReadList[ "data_file", {Real, Real, Real, Real} ];
```

ファイル data_file から ReadList コマンドを用いて(t, x, y, z) を対として数値データを読み込む
読み込んだものに data という名前を付ける

続き

data = $\{\{t_0, x_0, y_0, z_0\}, \{t_1, x_1, y_1, z_1\}, \{t_2, x_2, y_2, z_2\}, \ldots\}$ という形式になっている

Transpose コマンドにより、data の成分を転置したものを dataT とする

```
dataT = Transpose[data];
```

これにより dataT = $\{\{t_0, t_1, t_2, \ldots\}, \{x_0, x_1, x_2, \ldots\}, \{y_0, y_1, y_2, \ldots\}, \{z_0, z_1, z_2, \ldots\}\}$

dataT の第 1 成分と第 2 成分を抜き出し、新たに dataX とする

```
dataX = Transpose[ {dataT[[1]], dataT[[2]] } ];
```

同様にして $(t, y(t))$ のデータ dataY 、$(t, z(t))$ のデータ dataZ 、および $(x(t), y(t), z(t))$ のデータ dataXYZ を作る

```
dataY = Transpose[ {dataT[[1]], dataT[[3]] } ];
```

```
dataZ = Transpose[ {dataT[[1]], dataT[[4]] } ];
```

```
dataXYZ = Transpose[ {dataT[[2]], dataT[[3]], dataT[[4]] } ];
```


続き

横軸を時刻 t 、縦軸を $x(t), y(t), z(t)$ としたグラフを描く。

```
gx = ListPlot[ dataX, PlotJoined->True, PlotRange->All]
```

```
gy = ListPlot[ dataY, PlotJoined->True, PlotRange->All]
```

```
gz = ListPlot[ dataZ, PlotJoined->True, PlotRange->All]
```

3 つのグラフを重ね合わせて表示

```
Show[gx, gy, gz]
```

3 次元空間の軸をそれぞれ $x(t), y(t), z(t)$ としたグラフ（相空間内の解軌道）を描くには、3 次元グラフパッケージをあらかじめ読み込んでおく

```
<<Graphics`Graphics3D`
```

```
ScatterPlot3D[dataXYZ, BoxRatios->{1,1,1}, ViewPoint->{0.810, -2.475, 2.160}]
```

オプション ViewPoint は視点を指定（省略可）

Mathematica による数値解

deq2 = { x1'[t] == –x2[t] – x1[t]^2, x2'[t] == 2 x1[t] – x2[t], x1[0] == x2[0] == 1}

右の初期値問題を deq2 として定義

$$\frac{dx_1}{dt} = -x_2 - x_1^2 \qquad x_1(0) = 1$$
$$\frac{dx_2}{dt} = 2x_1 - x_2 \qquad x_2(0) = 1$$

sol2 = NDSolve[deq2, {x1[t], x2[t]}, {t, 0, 20}]　　区間 $0 \le t \le 20$ で数値的に解く

Plot[Evaluate[x1[t]/.sol2], {t, 0, 20}]　　横軸を t としたグラフを描く

Plot[Evaluate[x2[t]/.sol2], {t, 0, 20}]

ParametricPlot[Evaluate[{x1[t], x2[t]}/.sol2], {t, 0, 20}]　　相平面上の解軌道 ($x_1(t), x_2(t)$) を描く

グラフを LaTeX に貼り込む方法

Mathematica で描いたグラフを EPS 形式でファイルに保存。(EPS: Encapsulated PostScript)

保存したいグラフを選択して、Mathematica のメニューから、
Edit ---> Save Selection As ---> EPS

LaTeX ドキュメントのプリアンブルに下を追加。

```
\usepackage{graphicx}
```

EPS グラフ graph.eps を取り込む

```
\includegraphics{graph.eps}
```

図を中央そろえ、横幅 5cm に縮小拡大して表示する場合

```
\begin{cetner}
\includegraphics[width = 5cm]{graph.eps}
\end{center}
```