
プログラミング基礎演習

第12回

金(5限)

長谷川禎彦

連絡事項

- 課題提出者はHP参照
 - 課題プログラム例もHP参照
 - 次回は1/27(火). **次回は課題は出ないので出席は自由.** 演習時間はほぼ小課題レポート課題ソフト2課題取り組みや質問タイムに当てる
 - レポート2の締め切りが授業期間の終了後なので、不備があった場合授業で連絡できない. そこで、提出後もHPをチェックせよ.
-

今日の復習事項

- 関数ポインタ

- ポインタ, 構造体と併せて, これが理解できればプログラミング言語の基礎概念はほとんどマスターしている
 - 残る重要概念はオブジェクト指向くらい
- 最近のプログラミング言語では, 関数ポインタの概念を前面に導入した言語が多い

関数ポインタ

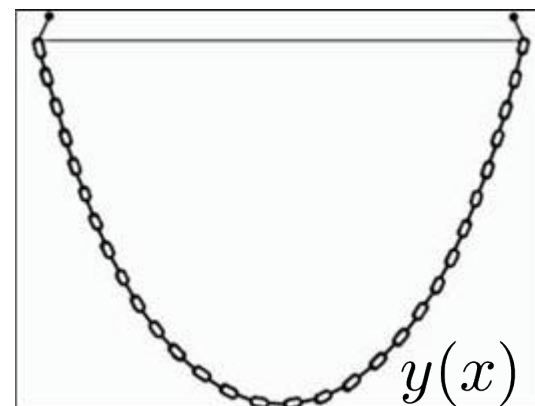
- ポインタの関数版
 - ラムダ式, クロージャ, 純関数なども同様の概念
- 関数ポインタを引数として取る関数がある
 - プログラミングでは高階関数とよぶ
 - 数学・物理では汎関数(functional)

$$U[y] = mg \int_P^Q y(x) \sqrt{1 + y'(x)^2} dx$$

この引数は「関数」!
エネルギーUは「汎関数」

エネルギー最小となる関数 $y(x)$ を求める(変分法)

懸垂曲線



関数ポインタが必要な例

数値積分関数

$$N(f, \text{from}, \text{to}) = \int_{\text{from}}^{\text{to}} f(x) dx$$

積分したい関数f

積分開始値

積分終値

```
double N(f, double from, double to) {  
    double area = 0;  
    double h = 0.01;  
    for (double x=from; x<=to; x+=h) {  
        area += f(x) * h;  
    }  
    return area;  
}
```

関数ポインタが必要な例

数値積分関数

$$N(f, \text{from}, \text{to}) = \int_{\text{from}}^{\text{to}} f(x) dx$$

積分したい関数 f

積分開始値

積分終値

```
double N(f, double from, double to) {
```

```
    double area
```

```
    double h
```

```
    for (double x = from; x < to; x += h)
```

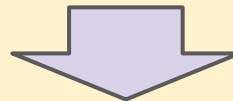
```
        area += f(x) * h;
```

```
}
```

```
return area;
```

```
}
```

f の型は？



関数ポインタ

関数ポインタのイメージ

- 関数ポインタは、関数自体を変数のように扱う方法
 - 変数のポインタとは少しイメージが違う

関数ポインタの宣言

返値の型 (*変数名) (引数...);

例1

doubleを引数に取り, doubleを返す関数fを宣言する場合

```
double (*f) (double);
```

```
f = sin;
```

```
printf("%f\n", f(1)); // sin(1)
```

doubleを引数にとって, doubleを返す関数ならなんでも代入できる

関数ポインタの宣言

例2


二つのintをとって, floatを返す関ポインタgの宣言

```
float (*g)(int, int)
```

プロトタイプ宣言に書く「型」は変数名を除いたもの

例1: `double (*) (double)`

例2: `float (*) (int, int)`

これ全部で, 一つの型を表す

数値積分の場合

$$N(f, \text{from}, \text{to}) = \int_{\text{from}}^{\text{to}} f(x) dx$$

```
double nintegrate(double (*f)(double), double from, double to) {  
    double sum = 0;  
    double v = 0;  
    double h = (to - from) / 1000;  
    for (v = from; v <= to; v += h) {  
        sum += h * f(v);  
    }  
    return sum;  
}
```

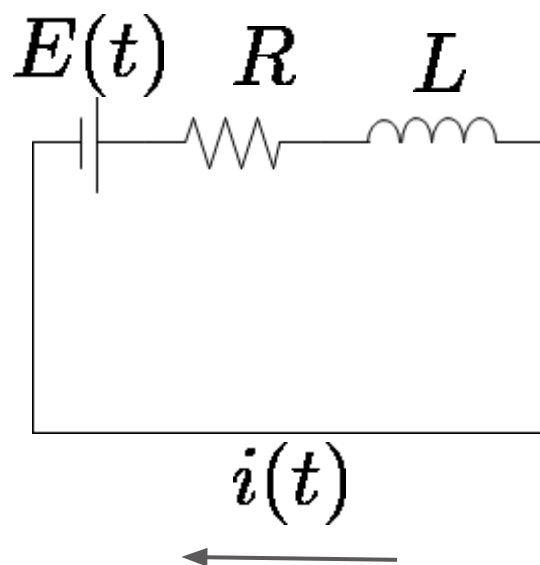
```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
double nintegrate(double (*)(double), double, double);
double test(double x) {
    return exp(-x * x);
}
int main() {
    double (*func)(double);
    func = test;
    printf("%f\n", nintegrate(func, -10, 10));
    return 0;
}
double nintegrate(double (*f)(double), double from, double to) {
    double sum = 0;
    double v = 0;
    double h = (to - from) / 1000;
    for (v = from; v <= to; v += h) {
        sum += h * f(v);
    }
    return sum;
}
```

関数ポインタの「型」は変数名を除いたもの

課題1

微分方程式の数値ソルバーを書け(オイラー法などで. Runge-KuttaなどでもOK). ただし, 計算する方程式を関数ポインタの形で与えよ. 次元の場合のみを考えて良い.

例: RL回路



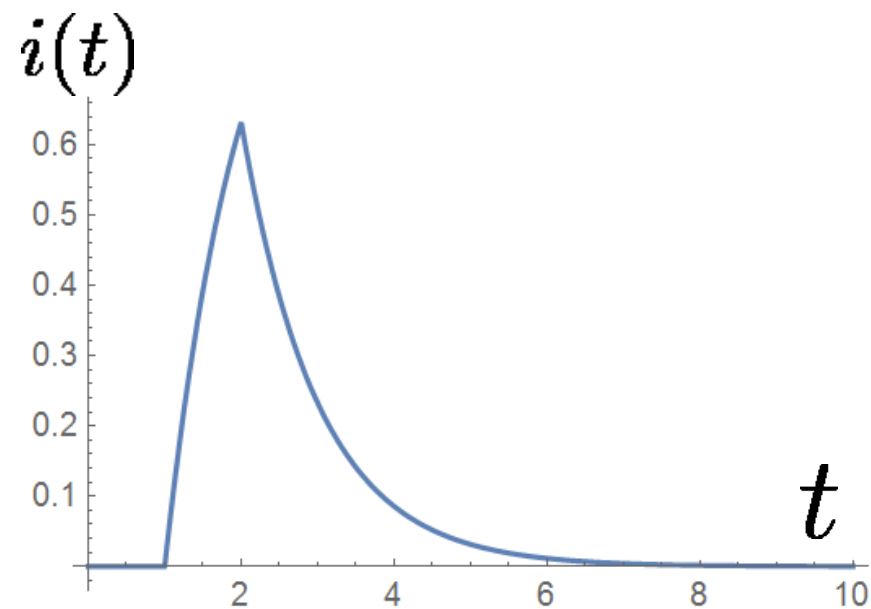
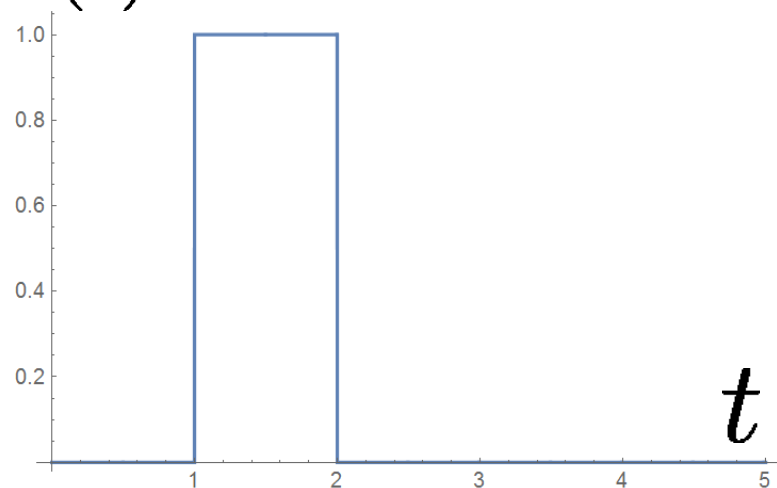
$$L \frac{di}{dt} = E - Ri$$

過渡応答

$$L \frac{di}{dt} = E - Ri$$

$$L = 1, R = 1$$

$E(t)$ 矩形入力



```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#define h 0.001
void ndsolve(double (*)(double, double), double, double, double);
/*RL回路の微分方程式(ODE)右辺*/
double RL(double i, double t) {
    /*....*/
}
int main() {
    ndsolve(RL, 0, 0, 10); // 数值的に解く
    return 0;
}
/*
```

微分方程式を数值的に解いて、各時刻の値を逐次 printfする

f : 関数ポインタ

x0 : 初期値

t0 : 初期時間

tend : 終時間

```
*/
void ndsolve(double (*f)(double, double), double x0, double t0, double
tend) {
    /*埋める*/
}
```

使いたくない人はこのひな形を使わなくてもよい

二つの引数はxとt

オイラー法(復習・ソフト2の資料参照)

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



$$x(t + h) \simeq x(t) + h \times f(x(t), t)$$

h は微小量

この式を繰り返し解くことで、 $x(t)$ の時間変化を計算する

ndsolveの出力

時間と, その時の値をタブで区切って表示

```
printf("%f\t%f\n", t, x);
```

以下のようになる

0.000000	0.000000
----------	----------

0.010000	0.027368
----------	----------

0.020000	0.327368
----------	----------

...



tの値

xの値

補足:リダイレクション

- `$./kadai > out.txt`
 - プログラムkadaiの出力をファイルout.txtに出力する



```
$ ./kadai  
0.00    1.00  
1.00    2.50
```

ファイルの入出力を書かなくても、ファイルに書き込むことが出来る

というprintfによる表示をするプログラムの場合、
`./kadai > out.txt`
とすると、画面に表示される代わりに、ファイルout.txtに出力される。

補足: gnuplotでプロット

out.txtにタブ区切りのデータ(out.txt)がある場合

```
0.00 1.00 4.00
```

```
0.01 0.97 3.98
```

```
0.02 0.78 4.90
```

...

gnuplotと打って起動して、以下のコマンドを打つ.

```
gnuplot> plot "out.txt" using 1:2 w l
```

```
gnuplot> replot "out.txt" using 1:3 w l
```

プロットは線で行う. w lp
だと点と線を用いる.

1列目, 3列目をプロット
する

1列目, 2列目をプロット
する

蛇足: Mathematicaでプロット

読むファイルのパス

Table形式で読む

```
data=Import[“/foo/bar/hoge/data.txt”,“Table”];  
ListLinePlot[ data[;; , {1, 2}] ]
```

data[...]のように
二重括弧に注意

「;;」は行は全て使う
ということ

データの1列目と2列
目を使う場合

```
ListLinePlot[{data[;; , {1, 2}], data[;; , {1, 3}]]}
```

とすると, 2つのデータを一つのグラフに出来る

課題2（自由課題）

N 次元の微分方程式に対して数値的に解くように
課題1を拡張せよ.

そのうえで, 例えば振動子の微分方程式を解いて
みよ($t = 0 \sim 500$ くらいで).

主となる関数の雛形

```
/*
```

微分方程式を数值的に解いて、各時刻の値を逐次printfする

f: 関数ポインタのポインタ(ポインタ配列)

x0: 初期値の配列

t0: 初期時間

tend: 終時間

N: 系の次元

```
*/
```

```
void ndsolve(double (**f)(double*, double), double* x0,
```

```
double t0, double tend, int N) {
```

```
/*ここを埋めてみる*/
```

```
}
```

この雛形を使わなくても
も良い

引数はxの配列とt

関数ポインタのポインタ

```
void show(double (**f)(double)) {  
    printf("%f\n", f[0](1));  
    printf("%f\n", f[1](1));  
}
```

関数ポインタを配列のように扱うことが出来る

関数ポインタのポインタ

```
double (**f) (double);  
f = (double (**) (double))  
    malloc(sizeof(double(*) (double)) * 2);  
f[0] = sin;  
f[1] = cos;  
show(f); // sin(1), cos(1)が表示される
```

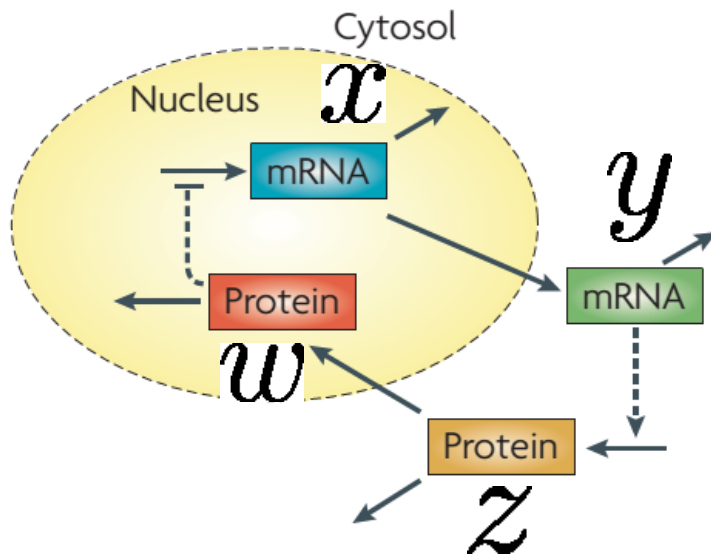

関数ポインタの配列

```
double (*f2[]) (double) = {sin, cos};  
show(f2); // sin(1), cos(1)が表示される
```

関数ポインタのポインタと、関数ポインタの配列は似ている。

例: 生体振動子

生物が持つ「24時間周期の時計」はタンパク質などの化学反応で起きる



Nat. Rev. Mol. Cell Biol., 2008, 9, 981-991

$$\frac{dx}{dt} = \frac{1}{1 + w^6} - 0.1x$$

$$\frac{dy}{dt} = 0.1x - 0.1y$$

$$\frac{dz}{dt} = 0.1y - 0.1z$$

$$\frac{dw}{dt} = 0.1z - 0.1w$$

例: Rössler chaos

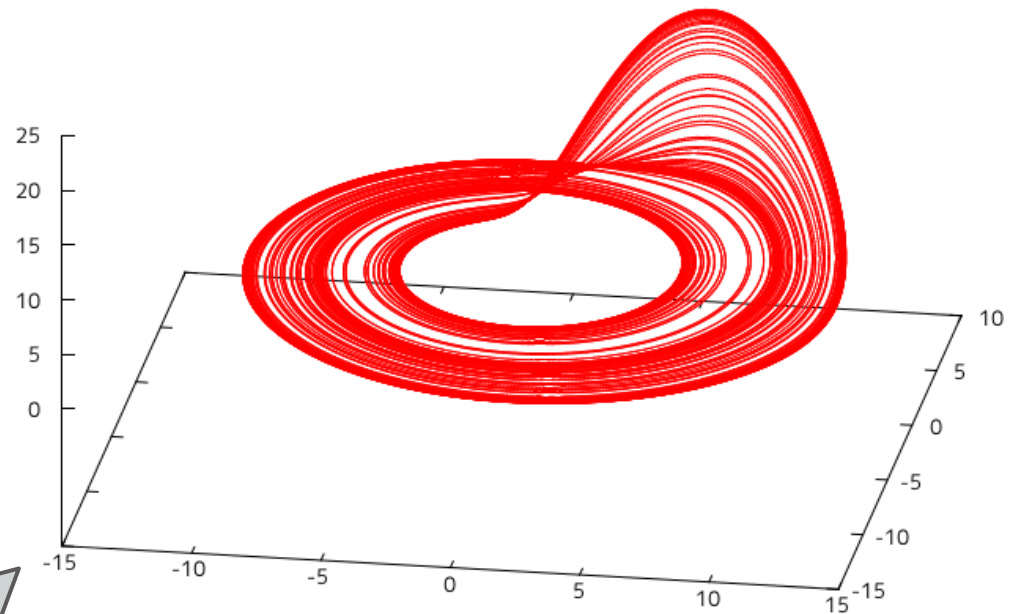
$$\frac{ds_1}{dt} = -s_2 - s_3,$$

$$\frac{ds_2}{dt} = s_1 + c_1 s_2,$$

$$\frac{ds_3}{dt} = c_2 + s_3(s_1 - c_3).$$

$$c_1 = 0.15, c_2 = 0.2, c_3 = 7.1$$

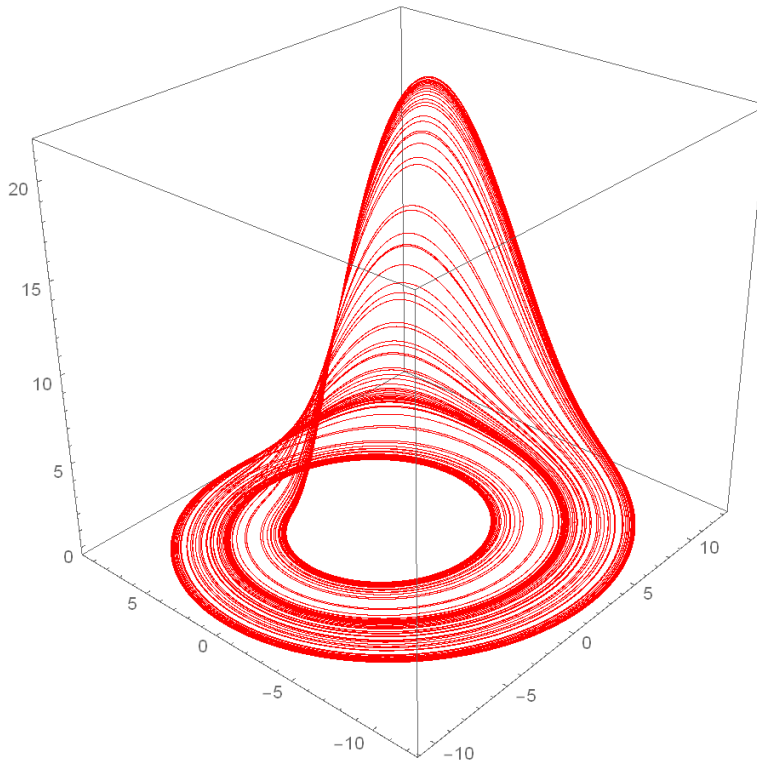
'Rossler.txt' using 2:3:4



例: `splot 'filename.txt' using 2:3:4 w l`

蛇足: Mathematicaで3Dの線を表示

```
d = Import["ファイルのパス", "Table"];  
Graphics3D[{Red, Line[d[[;; , {2, 3, 4}]] ]}, Axes -> True]
```



2,3,4列目のデータを用いる

課題提出方法

- 未完でも提出する
- 課題xの答えをkandai0x.cファイルに書く.
 - xは1,2
- 締め切りは**火曜日**の23:59
- ファイルをzipファイルにまとめる
 - ファイル名:学籍番号.zip
 - 例:学籍番号が341234の場合, 341234.zipとする.
- 学籍番号.zipファイルのみを提出する
 - 提出は <http://goo.gl/hXsfLl>
 - フォームの「課題」から「小課題」を選択