ソフトウェア２
第５回
（2013/1/17）

鶴岡 慶雅

連絡用ページ

- URL

  http://www.logos.t.u-tokyo.ac.jp/~tsuruoka/lecture/software2/

  ユーザ名 :　soft2

  パスワード :　ee2012

- 資料
  - 講義スライド（ppt, pdf）
  - サンプルプログラム

今日の内容

- C言語入門
  - 関数へのポインタ
  - 分割コンパイル
  - ヘッダファイル
  - Makefile

- 多変数関数の最適化
  - 勾配法
  - 再急降下法

関数へのポインタ

- 関数ポインタ
  - 関数の実行コードが格納されているアドレスを指す

```c
#include <stdio.h>

int add_one(int x)
{
  return x + 1;
}

int main()
{
  int (*fp)(int);
  fp = add_one;

  int x = 1;
  int y = (*fp)(x);

  printf("%d¥n", y);
}
```

fp は、引数が int ひとつで返り値が int の関数へのポインタ

fp に、add_one 関数のアドレスを代入

fp で指されている関数（add_one 関数）を呼び出す

関数へのポインタ

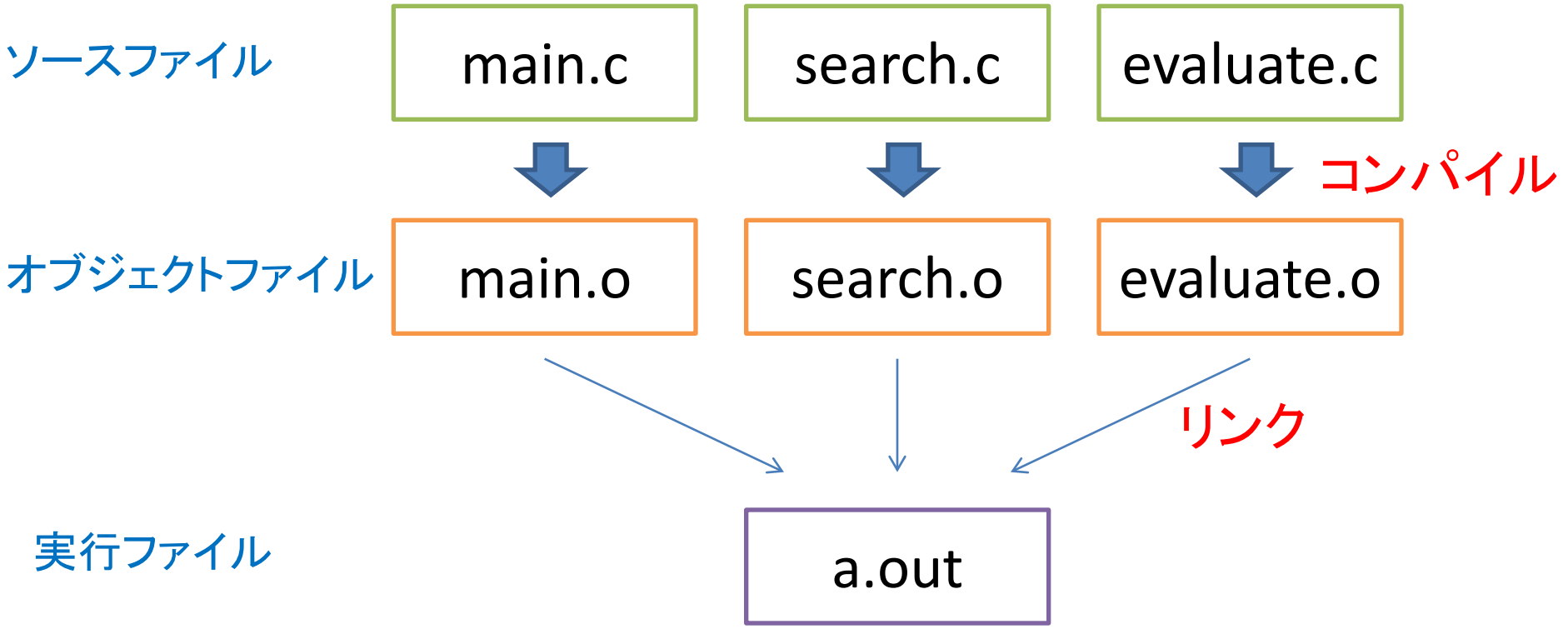
- 使い方の例
  - 関数に関数を渡して動作を変える

```c
#include <stdio.h>

#define SIZE 10

int add_one(int x)
{
  return x + 1;
}

int add_two(int x)
{
  return x + 2;
}

void apply(int v[SIZE], int (*fp)(int))
{
  int i;
  for (i = 0; i < SIZE; i++) {
    v[i] = (*fp)(v[i]);
  }
}
```

```c
int main()
{
  int i;
  int v[SIZE];

  for (i = 0; i < SIZE; i++) {
    v[i] = i;
  }

  apply(v, add_one);
  apply(v, add_two);

  for (i = 0; i < SIZE; i++) {
    printf("%d, %d¥n", i, v[i]);
  }
}
```

分割コンパイル

- 大規模なプログラム
  - 複数のソースファイルでプログラムを構成



分割コンパイル

- コンパイル

```
$ gcc –c main.c
$ gcc –c search.c
$ gcc –c evaluate.c
```

- リンク

```
$ gcc main.o search.o evaluate.o
```

ヘッダファイル（header files）

- 構造体や関数のプロトタイプ宣言などを記述

evaluate.h

```
#ifndef _EVALUATE_H_
#define _EVALUATE_H_

int eval_func(int board[8][8]);

#endif
```

二重に include されるのを防止

関数のプロトタイプ宣言
コンパイラに関数の情報（引数、返り値など）を与える

search.c

```
#include <stdio.h>
#include "evaluate.h"

    :
```

evaluate.h をインクルード
eval_func() 関数が search.c の中で使えるようになる

Makefile

- プログラムをコンパイルするのに必要な情報を指定したファイル
  - コンパイルに必要なファイル群
  - ファイル同士の依存関係
  - コンパイルオプション

- make コマンド
  - Makefile を参照してプログラムをコンパイル

最適化（optimization）

- 多変数関数の最適化
  - 入力はベクトル、出力はスカラー
  - 関数の最小値およびそれを実現する入力を計算

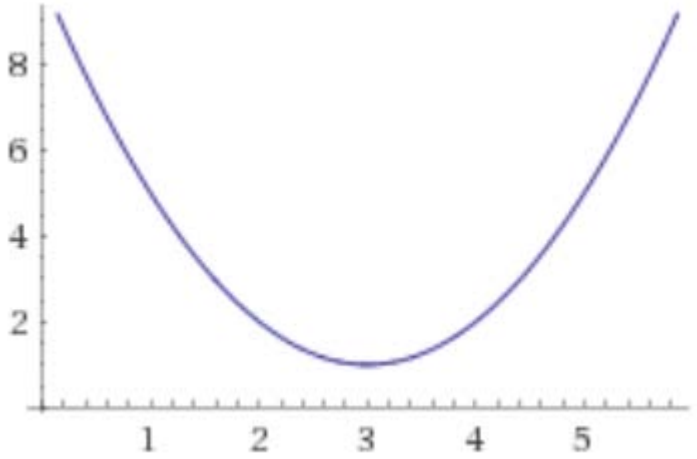
- 勾配（gradient）がわかっている場合
  - 勾配法
  - 最急降下法

１変数関数

- $f(x)$ が最小になる x を求めたい

$$f(x)=(x-3)^2+1$$

$$f'(x)=2(x-3)$$



$f'(x)=0$ が代数的に解けない場合の素朴な数値解法

- 適当に x を決める
- $f'(x) < 0$ なら右に移動
- $f'(x) > 0$ なら左に移動

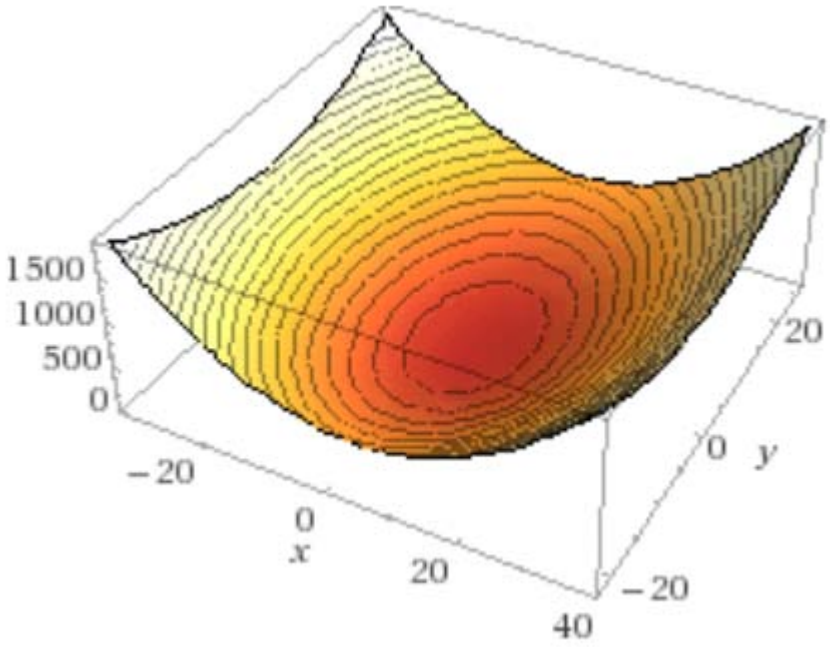
繰り返す

2変数関数の最適化

- $f(x, y)$ が最小になる (x, y) を求めたい

$$f(x,y)=(x-3)^2+(y-2)^2$$

$$\nabla f=\begin{pmatrix}\dfrac{\partial f}{\partial x}\\ \dfrac{\partial f}{\partial y}\end{pmatrix}=\begin{pmatrix}2(x-3)\\ 2(y-2)\end{pmatrix}$$



gradient（勾配）: 関数の値を最も大きく増加させる方向

→ 適当な (x, y) から出発して gradient と反対方向に進んでいけばよい

多変数関数の最適化

最急降下法（gradient descent）

1. 適当に初期位置 $\mathbf{x}$ を決める
2. $\nabla f(\mathbf{x})$ を計算
   $|\nabla f(\mathbf{x})|$ が十分小さければ終了
3. $\mathbf{x} \leftarrow \mathbf{x} - \alpha \nabla f(\mathbf{x})$ として $\mathbf{x}$ を更新
4. 2に戻り繰り返す

サンプルプログラム

- 展開

```
$ tar xvzf optimize.tar.gz
$ cd optimize
```

- コンパイル＆実行

```
$ make
$ ./a.out
```

サンプルプログラム

- Makefile

```
CC = gcc
CFLAGS = -Wall -g
LDLIBS = -lm
OBJS = main.o func.o optimize.o

a.out:  $(OBJS)
        $(CC) $(OBJS) $(LDLIBS)

clean:
        rm -f *~ *.o a.out
```

コンパイルするときのオプションは –Wall -g

a.out は main.o func.o optimize.o に依存

a.out の作り方は、
gcc main.o func.o optimize.o -lm

オブジェクトファイルや、余計なファイルを消したいときは make clean

ソースファイル(*.c)とオブジェクトファイル(*.o)の依存関係は make が自動的に判断

課題（締め切り1/23）

1. optimize.c を修正し、最適化の終了条件を厳しくせよ
   - make したときに optimize.c しかコンパイルされないことを確認
   - プログラムを添付すること（ファイル名は “optimize1.c”）

2. 最適化の各ステップで関数の値 $f(\mathbf{x})$ も表示するように拡張せよ
   - 関数 optimize() に関数 f_value() のポインタを渡すようにする
   - プログラムを添付すること（ファイル名は “optimize2.c”）

3. 入力が3次元の関数を何か考え、極小値を計算せよ
   - 用いた関数と得られた極小値について簡単に説明すること
   - プログラムを添付すること（ファイル名は “func1.c”）

4. ［発展課題］ より収束の速いアルゴリズムを考え、実装せよ
   - プログラムを添付すること（ファイル名は “optimize3.c”）
   - 直線探索、共役勾配法、ニュートン法、疑似ニュートン法、etc
   - 用いた手法とその効果を簡単に説明すること

課題の提出方法

- 宛先
  - software2@logos.t.u-tokyo.ac.jp
- Subject
  - 形式: SOFT-MM-DD-NNNNNNX
    - MM: 月
    - DD: 日（授業が行われた日）
    - NNNNNNX: 学籍番号
- 本文
  - 冒頭に学籍番号、氏名を明記