情報数学C

Mathematics for Informatics C

第5回 最適化問題
(黄金分割探索,最急降下法,
準ニュートン法,シンプレックス法)

情報メディア創成学類
藤澤誠

今回の講義内容

- **今日の問題**
- 3分割法, 黄金分割法
- 最急降下法, 準ニュートン法
- 線形計画問題について
- シンプレックス法

今回の講義で解く問題

$$\arg\min_{x} f(x)$$

今回の講義で解く問題

最小値・最大値探索

例) $f(x) = 2x^2 - 9x + 14 - \frac{9}{x} + \frac{2}{x^2} \quad (x > 0)$

のすべての極値を求めよ.

(2017年度筑波大学前期日程 数学問題より)

[数学での解き方] $f(x)$を微分して増減表を作る

$$f'(x) = 4x - 9 + \frac{9}{x^2} - \frac{4}{x^3} = \frac{1}{x^3}(x-1)(x+1)(4x^2 - 9x + 4)$$

なので, $f'(x) = 0$となるのは, $x = 1,\ \frac{9 \pm \sqrt{17}}{8}$

($x > 0$であることに注意)

今回の講義で解く問題

最小値･最大値探索

例) $f(x) = 2x^2 - 9x + 14 - \frac{9}{x} + \frac{2}{x^2} \quad (x > 0)$

のすべての極値を求めよ.

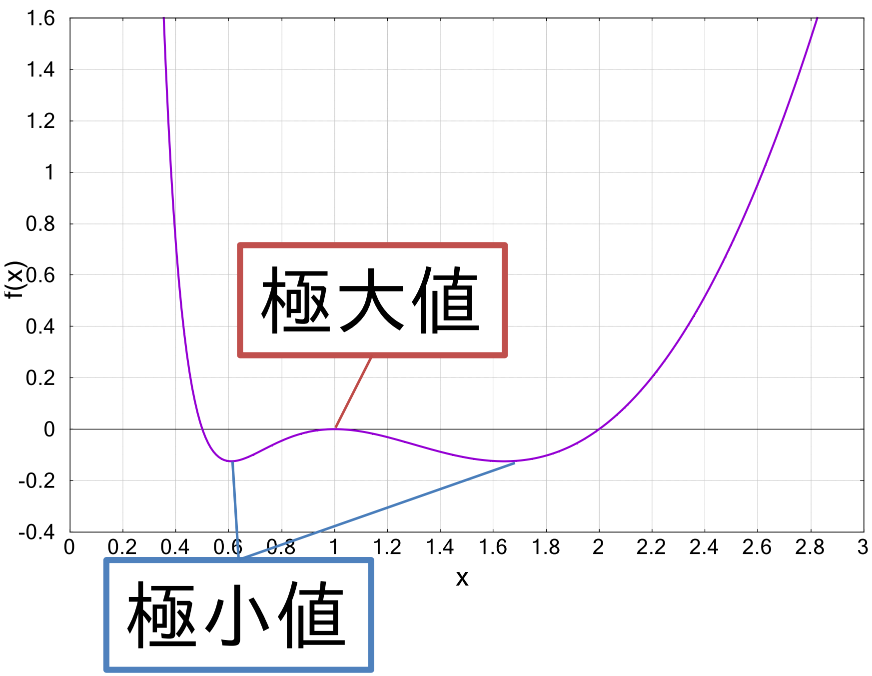
(2017年度筑波大学前期日程 数学問題より)

増減表

x	$\cdots$	$\frac{9-\sqrt{17}}{8}$	$\cdots$	1	$\cdots$	$\frac{9+\sqrt{17}}{8}$	$\cdots$
$f'(x)$	$-$	0	$+$	0	$-$	0	$+$
$f(x)$	↘		↗	0	↘		↗

極大値 : $x = 1$ で $f(x) = 0$

極小値 : $x = \frac{9\pm\sqrt{17}}{8}$ で $f(x) = -\frac{1}{8}$



今回の講義で解く問題

最小値・最大値探索

- 最大値・最小値/極値探索は，**最適化問題**の一種
- ある条件下において最も最適な値，方法などを探すために行われる
- $f(x)$の形や条件の有無でいくつか種類がある*
  - $f(x)$が非線形：**非線形計画問題**
  - $f(x)$が線形+条件付：**線形計画問題**
  - $f(x)$が非線形+条件付き：**制約付き非線形計画問題**(ラグランジュの未定乗数法で単純な極値問題になることも ⇒ 解析II(2020年度から微分積分B))

*他にも離散最適化問題として組み合わせ最適化など

今回の講義で扱う問題

最適化問題は**どんなところで使われる?**

自然科学, 社会科学, 工学など非常に様々なところで用いる基本的な問題の一つ

物理/CG/画像処理など

エネルギー最小化問題として, 設定したエネルギー関数や誤差関数が最小になる値を求めることで, 動画処理(動き検出など), オブジェクト検出(AR/MRへの応用も), 分類分け(k-means,NN), 3次元表面形状補間, 物理シミュレーションなどなど非常に多岐にわたって使われている.

近年発展している深層学習を用いた人工知能も内部的には最適化問題を解いている

今回の講義で解く問題

この授業での最適化問題について

- 最大化問題は$\arg\min_{x}(-f(x))$とすると最小化問題となるので, 基本的に**最小化問題のみを考える**
- この授業で扱うのは**連続最適化問題**のみ*
- $f(x)$を**目的関数**と呼び**, 目的関数を最小化する解を**最適解or最小解**と呼ぶ
- 最適化問題に関してさらに知りたい人はシステム数理II(連続最適化問題), システム数理III(離散最適化問題)の講義を受けよう!

*これに対して離散最適化問題(組合せ最適化問題)もある(巡回セールスマン問題など)
**分野によってはエネルギー関数(物理,画像処理など), 損失関数(深層学習など)とも呼ばれる

今回の講義内容

- 今日の問題
- **3分割法, 黄金分割法**
- 最急降下法, 準ニュートン法
- 線形計画問題について
- シンプレックス法

3分割法

まずは, **非線形最適化問題**について考えていく

⇒ 前回の求根問題も$f(x)$が0となる解を探すという違いがあるもののやりたいことは同じようなもの

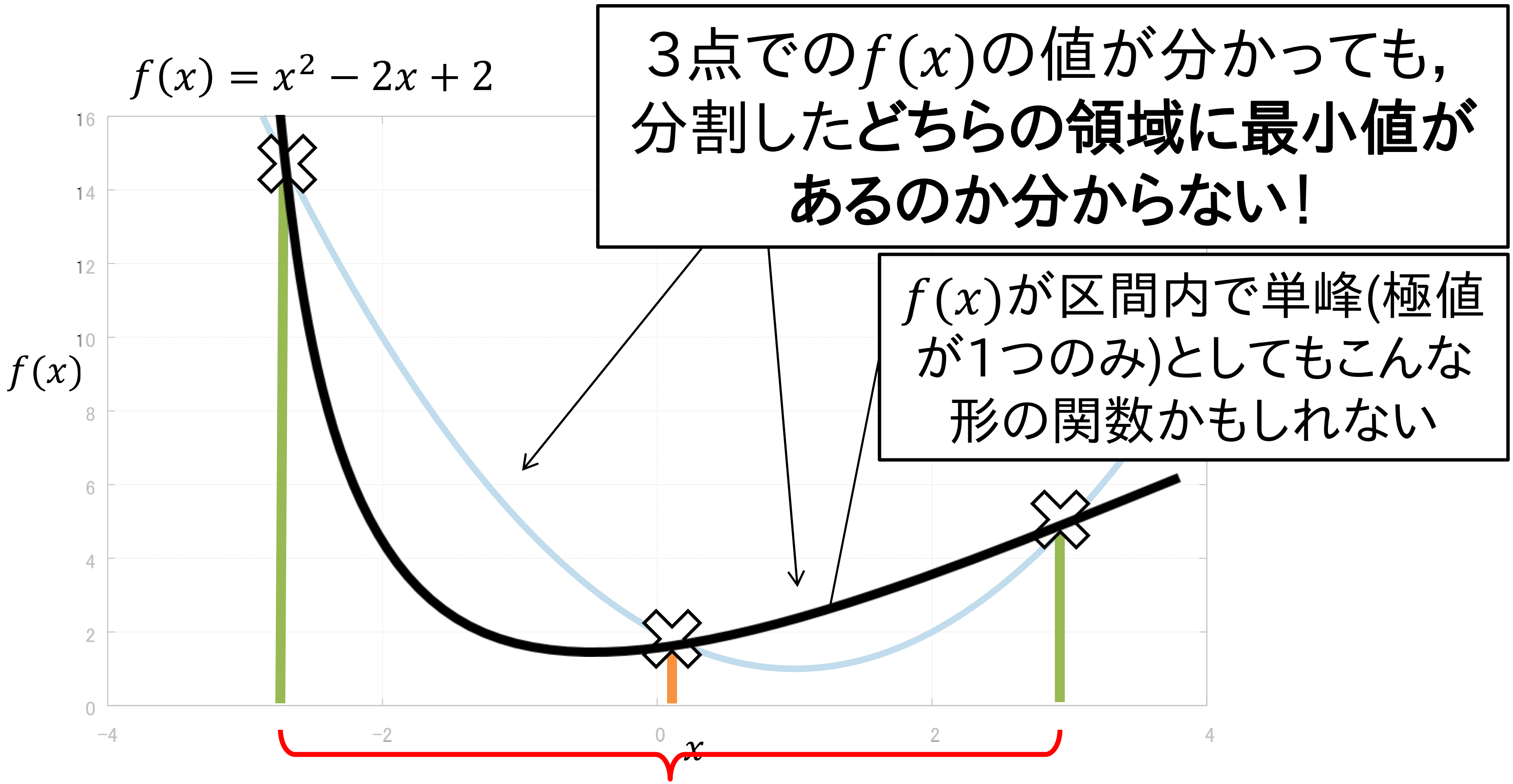
⇩

求根問題での2分法のように**分割していく**ことで最小値探索ができないか？

3分割法

2分法で最適化問題を解く?

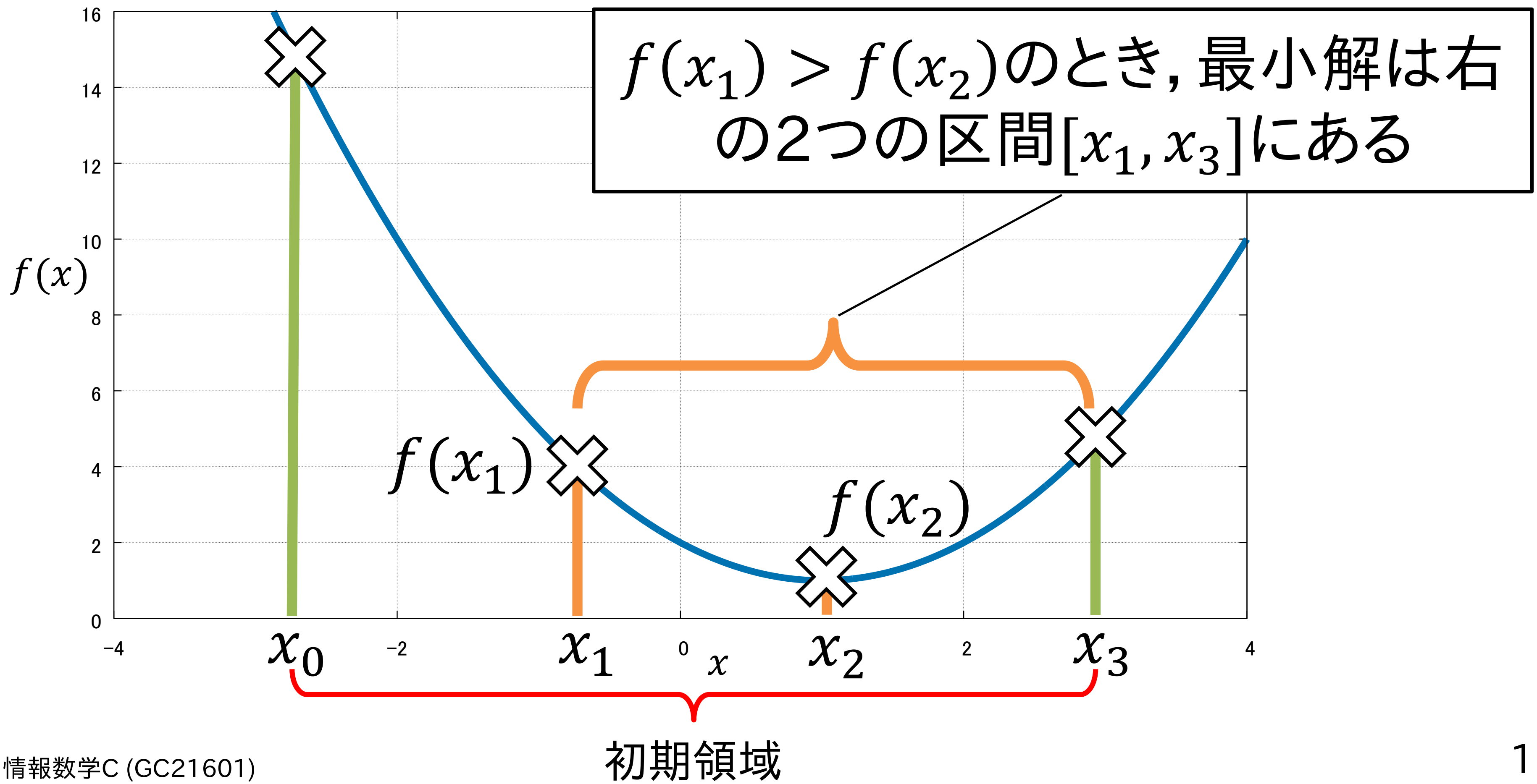
$f(x) = x^2 - 2x + 2$



2分法の初期領域

3分探索

2分割でだめなら**3分割**ではどうか?
⇒ 初期領域内では単峰(凸関数)であることを仮定



3分探索

「$f(x_1) > f(x_2)$のとき, 最小解は区間$[x_1, x_3]$にある」

証明

背理法を用いて証明する. もし, $f(x_1) > f(x_2)$で区間$[x_0, x_1]$に最小解x_{min}があるとすると, x_{min}とx_2の間にx_1がある

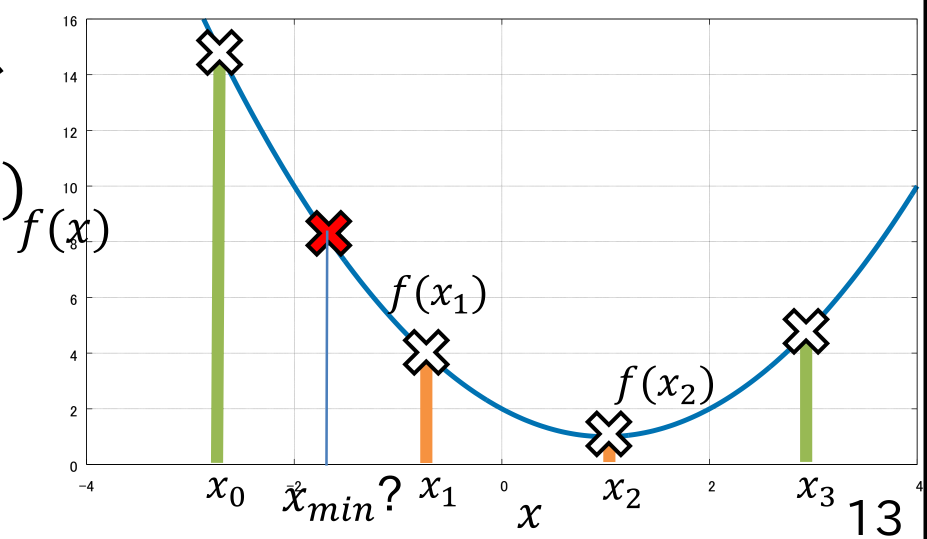
$$\Rightarrow f(x_1) = \underline{f(\alpha x_{min} + (1-\alpha)x_2) \leq \alpha f(x_{min}) + (1-\alpha) f(x_2)}$$

凸関数の性質

ただし, $0 \leq \alpha \leq 1$
また, $f(x_{min}) \leq f(x_2)$ なので

$$\alpha f(x_{min}) + (1-\alpha) f(x_2) \leq f(x_2)$$

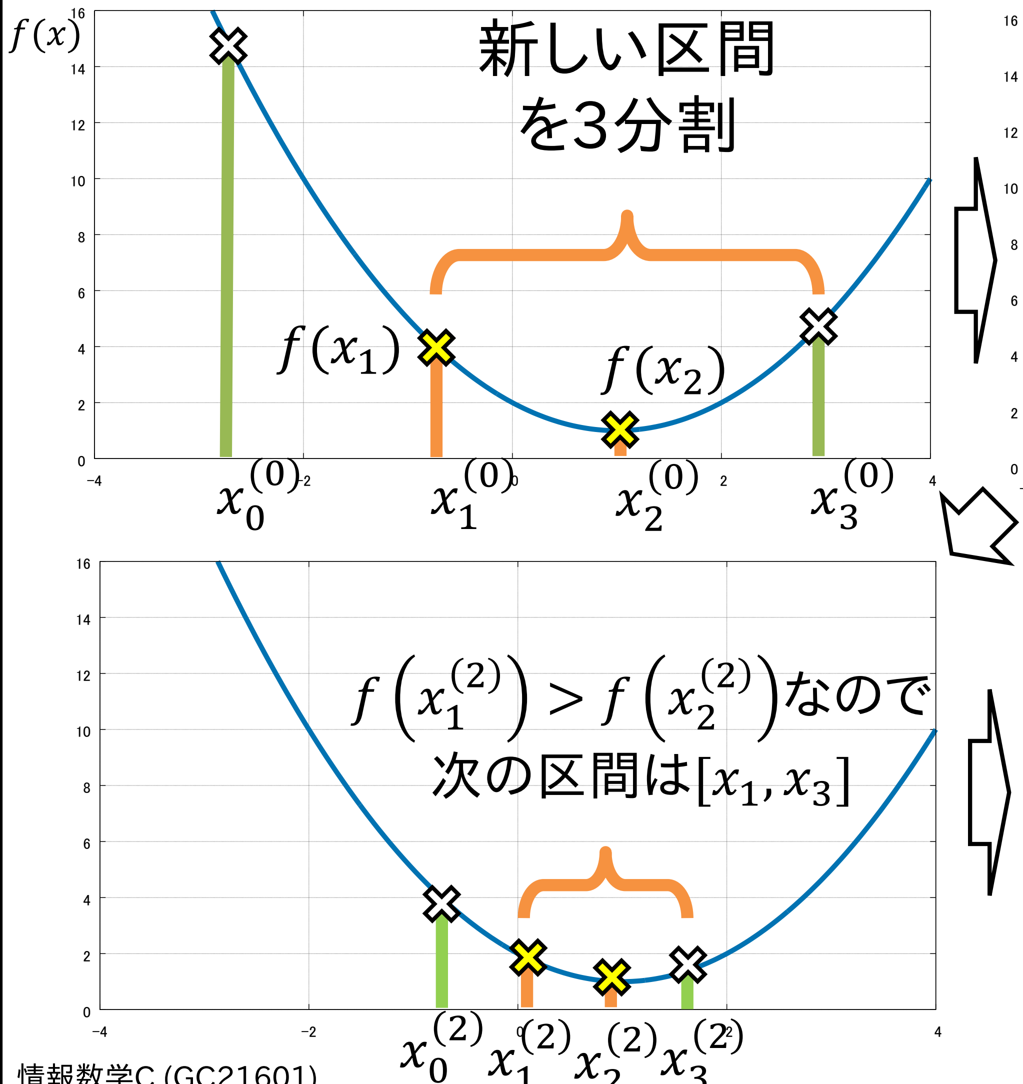
$f(x_1) \leq f(x_2)$となり
$f(x_1) > f(x_2)$と**矛盾!**



(証明終了)

3分探索

3分探索の手順



これを区間幅が

$$\left|x_3^{(k)} - x_0^{(k)}\right| < \varepsilon$$

となるまで反復処理

3分探索

3分探索の計算手順

1. 初期探索領域$[x_0^{(0)}, x_3^{(0)}]$を設定し, $f\left(x_0^{(0)}\right), f\left(x_3^{(0)}\right)$を計算
2. 以下を反復処理($k = 0,1, \dots$)
   1. 分割点$x_1^{(k)}, x_2^{(k)}$ と $f\left(x_1^{(k)}\right), f\left(x_2^{(k)}\right)$を計算
   2. $f\left(x_1^{(k)}\right) < f\left(x_2^{(k)}\right)$ なら $x_0^{(k+1)} = x_0^{(k)}, x_3^{(k+1)} = x_2^{(k)}$,
      $f\left(x_1^{(k)}\right) > f\left(x_2^{(k)}\right)$ なら $x_0^{(k+1)} = x_1^{(k)}, x_3^{(k+1)} = x_3^{(k)}$
   3. $\left|x_3^{(k+1)} - x_0^{(k+1)}\right| < \varepsilon$ なら反復終了

3分割探索

3分探索の問題点:

毎反復少なくとも2点での関数値計算が必要

⇒ 2分法のようにこれを1回にできないか?

⇩

分割幅を上手く設定すると**1回で済む**

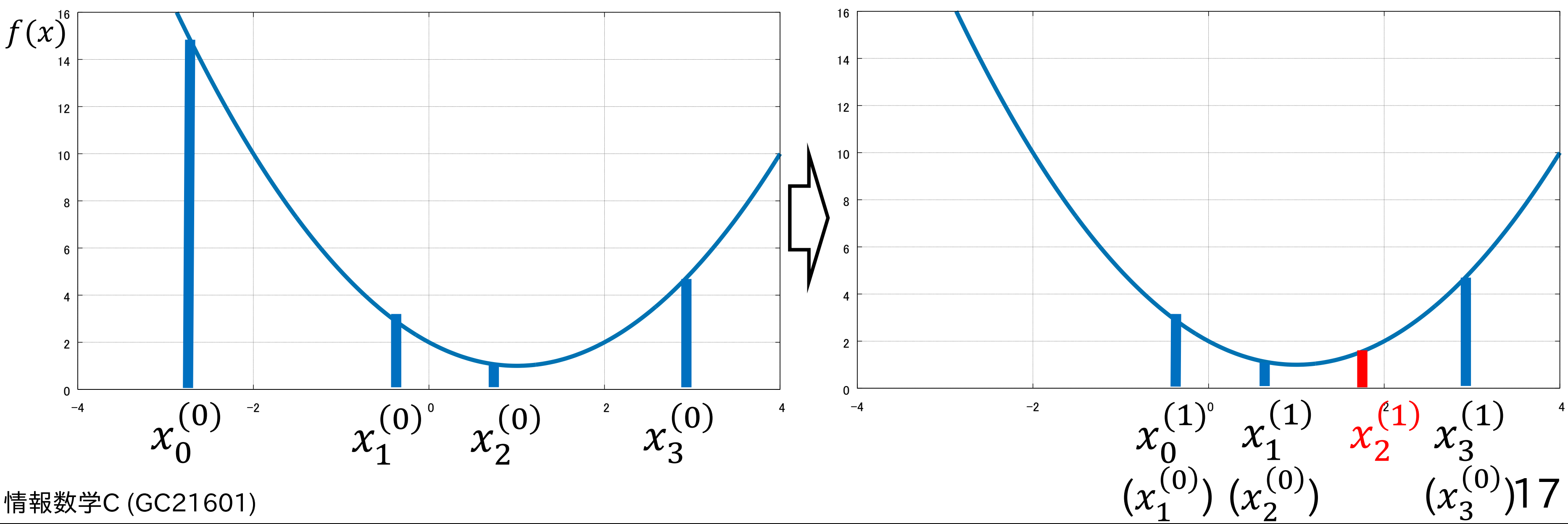
黄金分割法

3分探索で区間を均等に分割せず, 分割幅を分割区間毎に変えることを考える.

下の例だと, $x_1^{(0)} \to x_0^{(1)}$, $x_2^{(0)} \to x_1^{(1)}$とすることで

新たに$f(x)$を計算しなければならないのは$x_2^{(1)}$のみ

⇒ 計算は1回だけで済む



黄金分割法

どのように分割すれば良いのか？**分割比**をどうする？

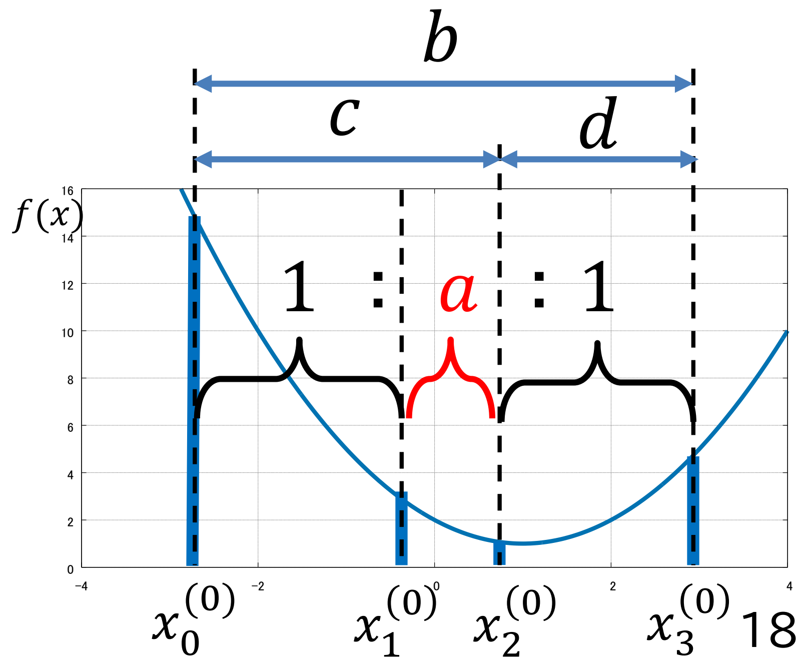
⇒ 分割比が極端(1:10とか)になると収束が不安定になる

反復を繰り返しても**分割比が変わらない**方がよい

黄金分割比

$$\frac{b}{c} = \frac{1+\sqrt{5}}{2}$$

比率の導出は[付録参照]



黄金分割法

右下の図より$b/c = (2+a)/(1+a)$であり, これを黄金分割比の式に代入してaについて解くと:

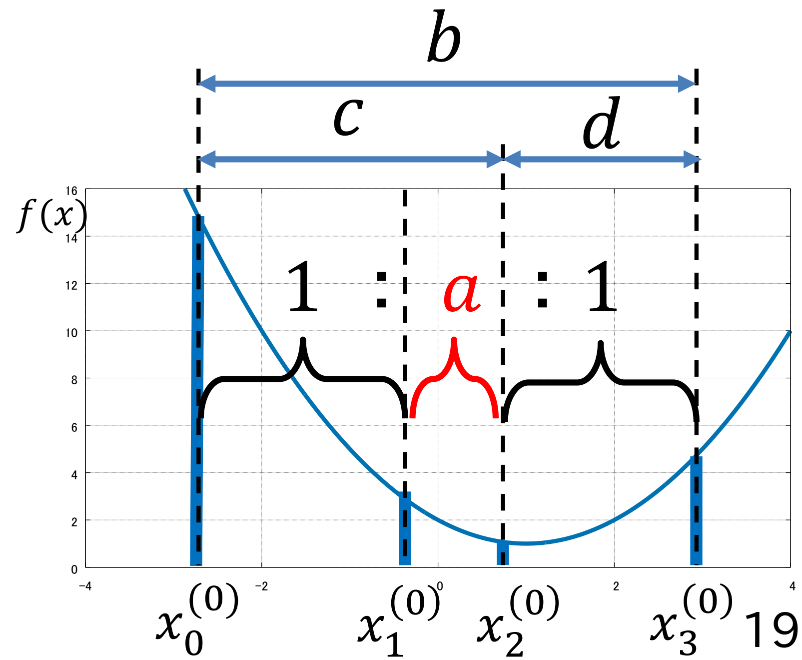
$$a = \frac{3-\sqrt{5}}{\sqrt{5}-1}$$

実際には新しい分割幅設定のためにbとdの比率 $\frac{d}{b} = \frac{\sqrt{5}-1}{\sqrt{5}+1}$ が使われる

黄金分割比で3分割探索する方法を

黄金分割法

という



黄金分割法

黄金分割法の計算手順

1. 初期範囲$[x_l, x_r]$から各種初期値を計算

   $e = \frac{d}{b} = \frac{\sqrt{5}-1}{\sqrt{5}+1}$ (dとbの比), $b^{(0)} = x_r - x_l$, $d^{(0)} = e \times b^{(0)}$,

   $x_0^{(0)} = x_l$, $x_1^{(0)} = x_l + d^{(0)}$, $x_2^{(0)} = x_r - d^{(0)}$, $x_3^{(0)} = x_r$,

   $f_1 = f\left(x_1^{(0)}\right), f_2 = f\left(x_2^{(0)}\right)$

2. 以下の手順を収束するまで繰り返す($k = 0,1,\ldots$)

   a. $f_1 < f_2$の場合：$x_2^{(k+1)} = x_1^{(k)}$, $x_1^{(k+1)} = x_2^{(k)} - d^{(k)}$, $x_3^{(k+1)} = x_2^{(k)}$,

   $f_2 = f_1$, $f_1 = f\left(x_1^{(k+1)}\right)$

   $f_1 \geq f_2$の場合：$x_1^{(k+1)} = x_2^{(k)}$, $x_2^{(k+1)} = x_1^{(k)} + d^{(k)}$, $x_0^{(k+1)} = x_1^{(k)}$,

   $f_1 = f_2$, $f_2 = f\left(x_2^{(k+1)}\right)$

   b. bとdの更新：$b^{(k+1)} = b^{(k)} - d$, $d^{(k+1)} = e \times b^{(k+1)}$

   c. $b < \varepsilon$なら反復終了

関数値$f(x)$の計算は毎ステップ1回のみ

黄金分割法

黄金分割法のコード例(前処理)

```
const double e = (SQRT5-1.0)/(SQRT5+1.0);
double b = xr-xl;
double d = e*b;

double x[4];
x[0] = xl;
x[1] = xl+d;
x[2] = xr-d;
x[3] = xr;

// x[1],x[2]における関数値
double f1 = func(x[1]);
double f2 = func(x[2]);

int k;
for(k = 0; k < max_iter; ++k){
    ・・・// 反復処理(次ページ)
}
```

eの式は$e = \frac{d}{b} = \frac{1}{a+2}$として計算できる. eを使ってdを更新していく

4つの分割位置$x_0 \sim x_3$の初期化

関数値の計算. 最初だけ$f(x_1)$と$f(x_2)$の2つを計算しているけど, 反復中はどちらか1つのみ計算

黄金分割法

黄金分割法のコード例(反復処理)

```
for(k = 0; k < max_iter; ++k){
    if(f1 < f2){ // 区間[x2,x3]に最小値なし
        double x2 = x[2];
        x[2] = x[1]; x[1] = x2-d; x[3] = x2;
        f2 = f1; f1 = func(x[1]);
    }
    else{ // 区間[x0,x1]に最小値なし
        double x1 = x[1];
        x[1] = x[2]; x[2] = x1+d; x[0] = x1;
        f1 = f2; f2 = func(x[2]);
    }

    b -= d;// 新しい[x0,x3]の長さ
    d = e*b; // 新しい[x0,x1](or[x2,x3])の長さ

    if(b < eps) break;
}
```

$f(x_1) < f(x_2)$の場合の処理.
左の2つの区間を新たな区間にする(x_0に変更なし)

$f(x_1) \geq f(x_2)$の場合の処理.
右の2つの区間を新たな区間にする(x_3に変更なし)

区間の幅bと分割後の小さい区間の幅dの更新

黄金分割法

黄金分割法の実行結果例

$f(x) = x^2 - 2x + 2$　　最小解 $x_{min} = 1$

許容誤差$\varepsilon = 1 \times 10^{-6}$,初期探索範囲$[0, 2]$

```
0 : 0         0.763932  1.236068 2,         b = 2
1 : 0.763932  1.236068  1.527864 2,         b = 1.236068
2 : 0.763932  1.055728  1.236068 1.527864,  b = 0.763932
3 : 0.763932  0.9442719 1.055728 1.236068,  b = 0.472136
4 : 0.9442719 1.055728  1.124612 1.236068,  b = 0.2917961
5 : 0.9442719 1.013156  1.055728 1.124612,  b = 0.1803399
6 : 0.9442719 0.9868444 1.013156 1.055728,  b = 0.1114562
      ⋮ (省略)
26 : 0.9999977 1.000001   1.000002 1.000005,  b = 7.368029e-06
27 : 0.9999977 0.9999995  1.000001 1.000002,  b = 4.553693e-06
28 : 0.9999995 1.000001   1.000001 1.000002,  b = 2.814337e-06
29 : 0.9999995 1          1.000001 1.000001,  b = 1.739356e-06
30 : 0.9999995 0.9999999  1        1.000001,  b = 1.074981e-06
x = 0.9999998
```

31回の反復で処理終了

黄金分割法

黄金分割法は1次元関数$f(x)$に対する最小化アルゴリズム

⇒ **多次元**の場合は？ (e.x. $f(x, y, z)$)

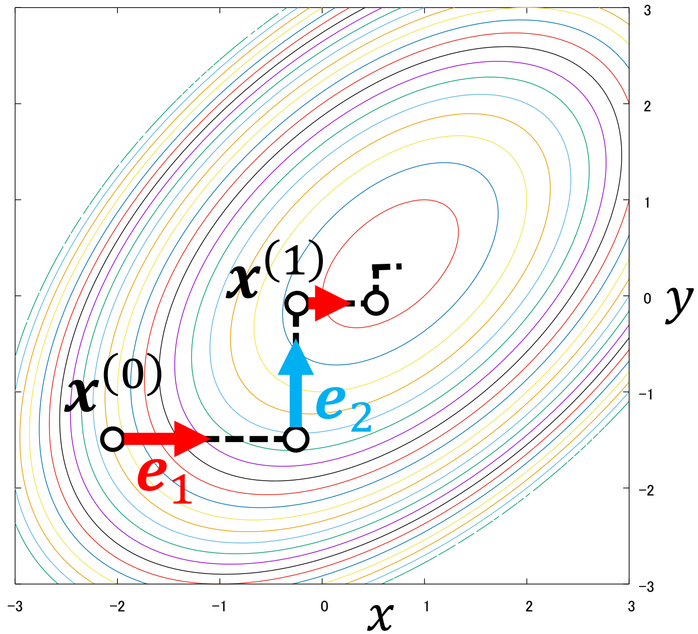
- 多次元空間内でも**ある直線方向**に限れば, **1次元探索で最小値を見つけられる**
- ある方向で最小値を見つける → その点から別の方向に探索して最小値を見つける
  を繰り返せば多次元でも最小値探索できそう

Powellの方法

Powellの方法

Powellの方法の手順(2次元の場合)

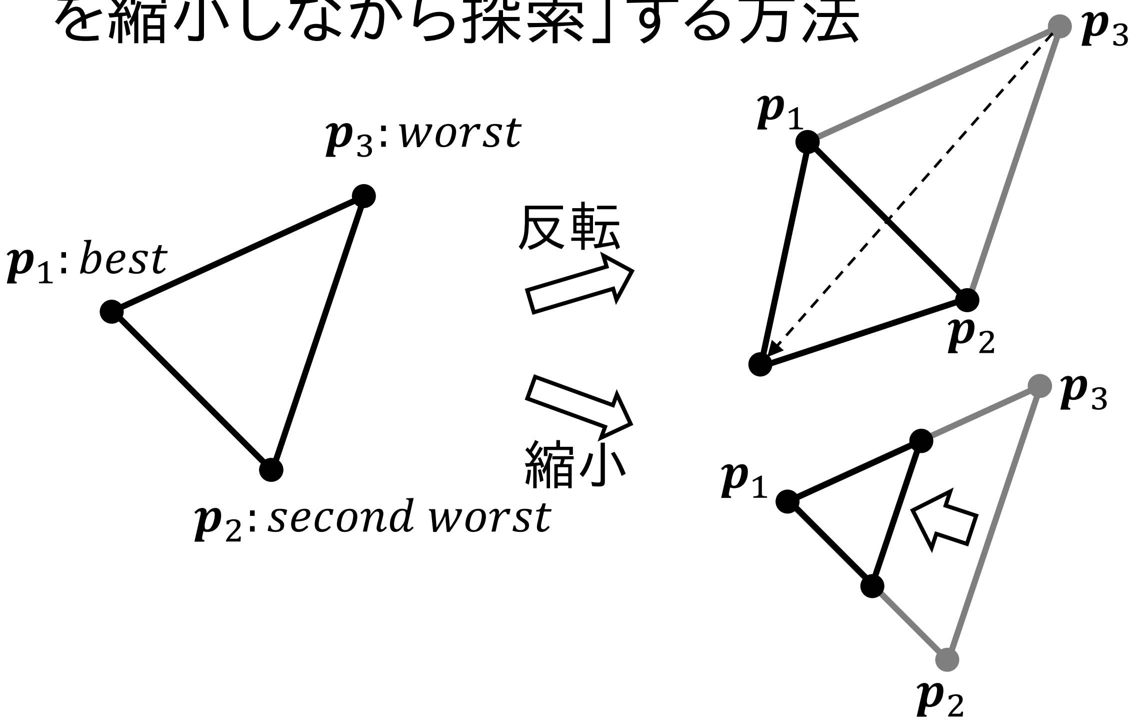
1. 初期位置$\boldsymbol{x}^{(0)}$を設定
2. 以下の処理を繰り返し($k=0,1,\ldots$)
   a. x軸方向に黄金探索法などで探索
      (関数値は軸方向$\boldsymbol{e}_1=(1,0)$と探索点までの距離tを使って$f(\boldsymbol{x}^{(k)}+t\boldsymbol{e}_1)$として計算できる)
   b. 手順aでの最小解を始点としてy軸方向($\boldsymbol{e}_2=(0,1)$)に探索.
      探索結果を$\boldsymbol{x}^{(k+1)}$とする
   c. $t<\varepsilon$となったら反復終了



$f(x,y)=x^2+y^2-xy-x-1$
を等高線を使って表示

その他の多次元最適化法

- **滑降シンプレックス法**(Nelder-Mead法)
  1次元の黄金分割法が「解を含む範囲を縮小しながら探索」であるのに対して，「解を含む**空間を囲む多面体**を縮小しながら探索」する方法



多面体の頂点を$f(\boldsymbol{p})$の値で分類分け，解を含むように**反転**や**縮小**操作していく

この講義の後半に説明するシンプレックス法とは全く別ものなので注意

今回の講義内容

- 今日の問題
- 3分割法, 黄金分割法
- **最急降下法, 準ニュートン法**
- 線形計画問題について
- シンプレックス法

導関数を使わない方法

黄金分割法もPowell法も導関数(微分)を必要としない方法

⇒ 求根問題の時のように**導関数が使える**ならもっと**効率的に探索**できるのでは?

- **最急降下法**: 勾配(微分)が最も急な方向に探索していく方法
- **準ニュートン法** : 2階微分(ヘッセ行列)を用いる方法

最急降下法

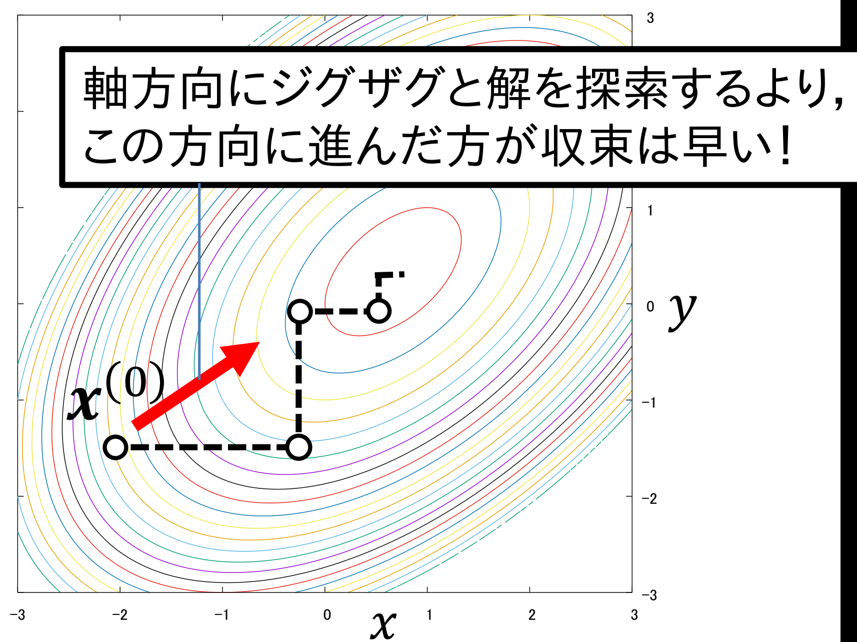
関数の微分(勾配)を用いる**勾配法**の中でも
最も単純な方法

- 現在の探索点から関数値が**最も急に降下する方向**に探索する
- 最も急な降下方向
  = 関数の勾配(df/dx)

最急降下法の更新式

$$\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} - \alpha \nabla f(\boldsymbol{x}^{(k)})$$

α：1回の更新で進む量を
決めるパラメータ



最急降下法

n次元での勾配∇fについて:

1次元での位置xによる微分は $\frac{df(x)}{dx}$

n次元での位置$\boldsymbol{x} = (x_1, x_2, \dots, x_n)^T$による微分を

$$\nabla f(\boldsymbol{x}) = \begin{pmatrix} \partial f / \partial x_1 \\ \partial f / \partial x_2 \\ \vdots \\ \partial f / \partial x_n \end{pmatrix}$$

と表し(grad $f(\boldsymbol{x})$と表記することもある),
$\nabla f(\boldsymbol{x})$を$f(\boldsymbol{x})$の**勾配**という.

注) $f(\boldsymbol{x})$はスカラー値を返す関数だが, $\nabla f(\boldsymbol{x})$は**ベクトル**となる

最急降下法

最急降下法の計算手順

1. 初期探索点$\boldsymbol{x}^{(0)}$を設定
2. 以下の手順を収束するまで繰り返す($k = 0,1, \ldots$)
   a. $\boldsymbol{x}^{(k)}$での勾配$\nabla f(\boldsymbol{x}^{(k)})$を計算
   b. 探索点の更新
   $$\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} - \alpha \nabla f\left(\boldsymbol{x}^{(k)}\right)$$
   c. $|\nabla f(\boldsymbol{x}^{(k)})| < \varepsilon$なら反復終了

最小解では$\nabla f = \mathbf{0}$となることを使って収束判定

最急降下法では関数値$f(x)$の計算はなく，
勾配$\nabla f(x)$の計算のみ

最急降下法

最急降下法のコード例

次元数をnとして, x, dxはn次元配列(ベクトル)

```
double norm_dx; // 勾配ベクトルのノルム(収束判定用)
int k;
for(k = 0; k < max_iter; ++k){
    dx = dfunc(x);
    norm_dx = 0.0;
    for(int i = 0; i < n; ++i){
        dx[i] *= alpha; // 勾配に係数αを掛ける
        x[i] -= dx[i];  // 勾配方向に探索点を移動
        norm_dx += dx[i]*dx[i];
    }

    // 勾配ベクトルのノルムで収束判定
    norm_dx = sqrt(norm_dx);
    if(norm_dx < eps) break;
}
```

関数の勾配∇fの計算(実際には勾配値を返す関数を呼んでいるだけ)

最急降下法の更新式の計算:
$\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} - \alpha \nabla f(\boldsymbol{x}^{(k)})$

最急降下法

最急降下法の実行結果例

$f(x) = x^2 + y^2 - xy - x - 1$ 最小解 $\boldsymbol{x}_{min} = (\frac{2}{3}, \frac{1}{3})$

許容誤差$\varepsilon = 1 \times 10^{-6}$,初期探索点$(0, 0)$, $\alpha = 0.5$

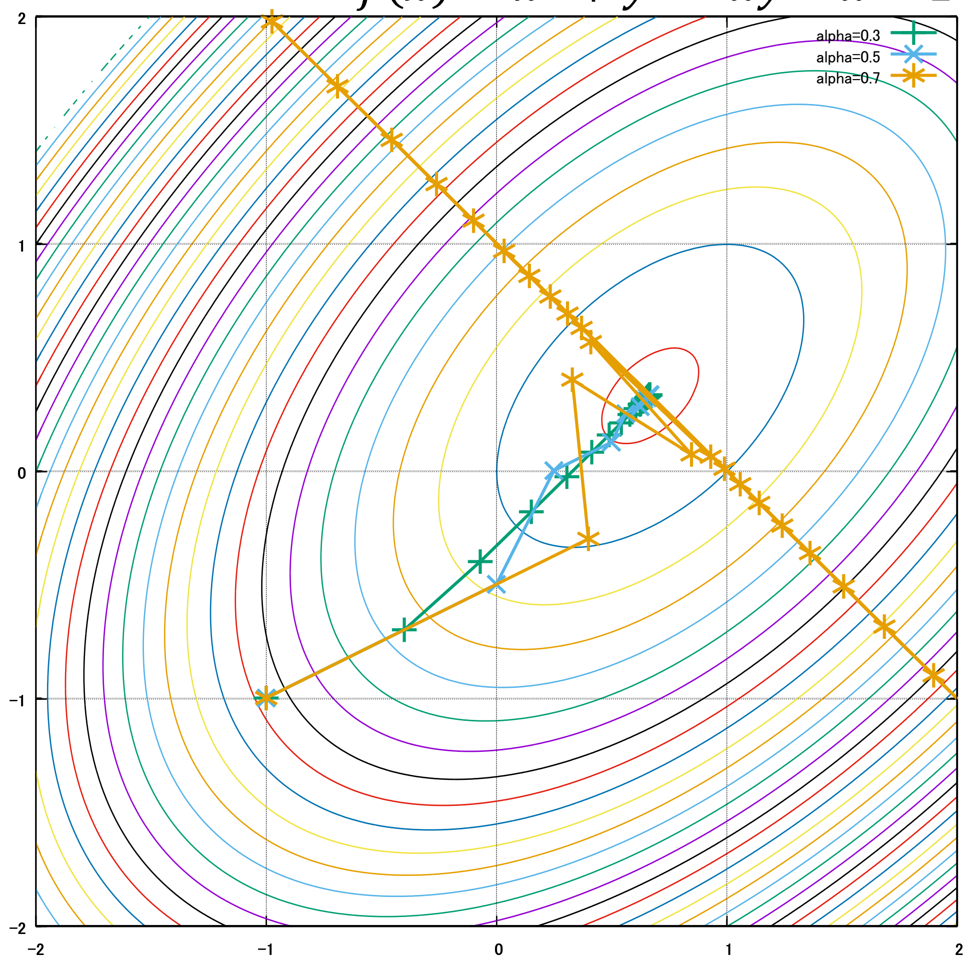
```
0 : 0,       0,        e = 1
1 : 0.5,     0,        e = 0.5
2 : 0.5,     0.25,     e = 0.25
3 : 0.625,   0.25,     e = 0.125
4 : 0.625,   0.3125,   e = 0.0625
5 : 0.65625, 0.3125,   e = 0.03125
6 : 0.65625, 0.328125, e = 0.015625
     ⋮ (省略)
15 : 0.666656, 0.333313, e = 3.05176e-05
16 : 0.666656, 0.333328, e = 1.52588e-05
17 : 0.666664, 0.333328, e = 7.62939e-06
18 : 0.666664, 0.333332, e = 3.8147e-06
19 : 0.666666, 0.333332, e = 1.90735e-06
x,y = 0.666666, 0.333333
```

20回の反復で処理終了

最急降下法

係数αをかえた結果

$f(x) = x^2 + y^2 - xy - x - 1$



初期点 $\boldsymbol{x}^{(0)} = (-1, -1)$

最小解 $\boldsymbol{x}_{min} = (\frac{2}{3}, \frac{1}{3})$

αの値	反復回数
0.3	39
0.5	22
0.7	収束せず

$\alpha = 0.5$の回数が前ページと違うのは左グラフを分かりやすくするために初期値を変えたため

最急降下法

係数αをどう設定すれば良いのか？

αが大きすぎると**発散**し, αが小さすぎると**収束が遅く**なる

反復の最初の方は大きく, **徐々に小さくする**と良さそう

- **指数関数**的に減衰させたり, 反復回数の**逆数**を使う
- **焼きなまし法**：徐々に小さくするとともに, **局所最適解**の回避のためにαが大きいときにたまに(確率的に)逆方向に探索する

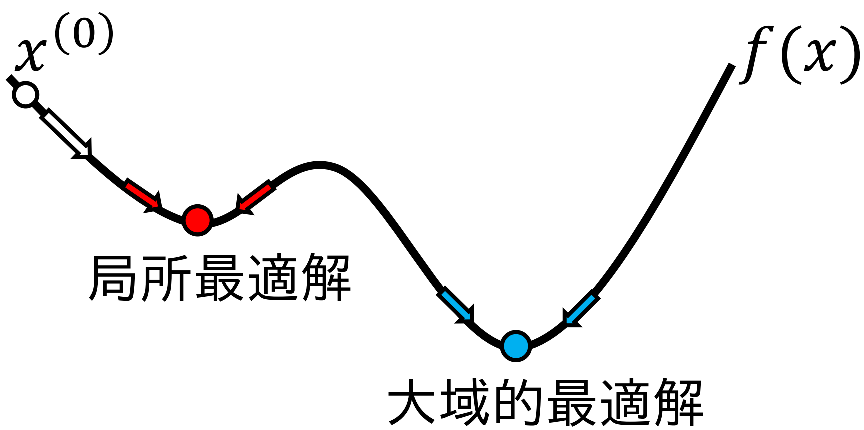
*「焼きなまし」とは金属を焼き入れして硬くした後にもう一度熱した後徐々に冷やすことで欠陥を少なくする作業

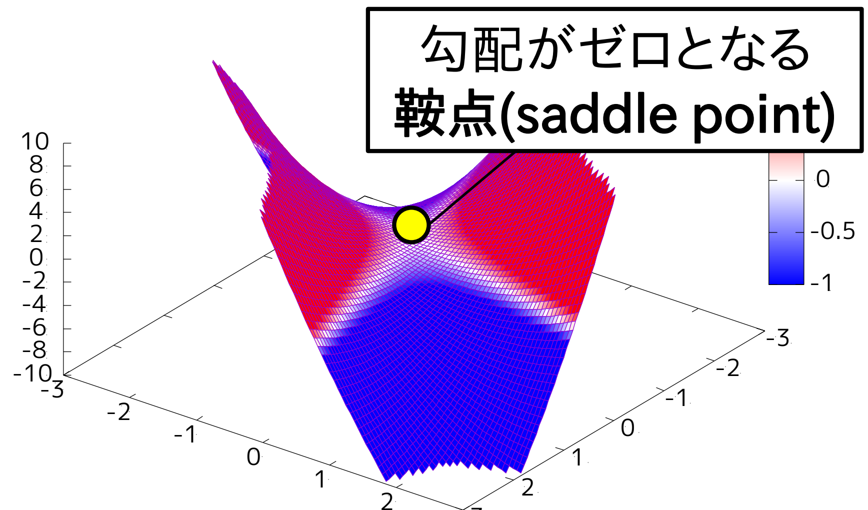
最急降下法

局所最適解とは？

目的関数全体の最小ではなく，局所的な**極小値**を示す解のこと．全体の最小は**大域的最適解**という．

また，極小ではないが勾配がゼロとなる**鞍点**(あんてん)も問題．





特に**次元数**n**が大きい**ときに局所最適解や鞍点に陥りやすい

最急降下法

大域的最適解のための最急降下法の改良

- **焼きなまし法**(シミュレーテッド・アニーリング:SA法)*：
温度パラメータTから求めた確率pによっては目的関数が大きくなるような方向にも進む

[焼きなまし法の手順]
1. 温度Tの初期値と停止条件, 変数$\boldsymbol{x}$の初期値を設定
2. 現在の解$\boldsymbol{x}$の近傍解$\boldsymbol{x}^*$をランダム抽出
3. $\Delta f = f(\boldsymbol{x}^*) - f(\boldsymbol{x}) < 0$ なら$\boldsymbol{x}^*$を新しい解に, そうでない場合でも一様乱数$r \in [0,1)$が確率$p = \exp(-\Delta f / T)$以下なら$\boldsymbol{x}^*$を新しい解にする
4. 温度Tを減少させ, 停止条件になっていなければ2に戻る

*Simulated Annealing, annealingが焼きなましの意味で, そのまま訳すと疑似焼きなまし法. 日本では単に焼きなまし法と呼ぶことが多い

最急降下法

大域的最適解のための最急降下法の改良

- **確率的勾配降下法** : 複数のデータf_iからなる目的関数$F(\boldsymbol{x})$を最小化するときに, データを1つもしくはいくつか**ランダムに選んで,** そこから**勾配を近似して**探索(学習)する. **大規模な機械学習**で非常によく使われる.

$$F(\boldsymbol{x}) = \sum_i f_i(\boldsymbol{x}) \quad \Rightarrow \quad \boldsymbol{x} \leftarrow \boldsymbol{x} - \eta_i \nabla f_i(\boldsymbol{x}^{(k)})$$

- すべてのデータのランダムシャッフル(オンライン) or いくつかのデータのランダムピックアップ(ミニバッチ) + 最急降下法 の繰り返し
- 学習率ηの決め方で**確率的近似法, モメンタム法, AdaGrad, RMSProp, AdaDelta, Adam**などの改良法が提案されている*

*深層学習で有名なGoogleのTensorFlowではAdaGrad, AdaDelta, Adam, RMSPropなどが使われている

導関数を使わない方法

黄金分割法もPowell法も導関数(微分)を必要としない方法

⇒ 求根問題の時のように**導関数が使える**ならもっと**効率的に探索**できるのでは?

- **最急降下法**：勾配(微分)が最も急な方向に探索していく方法
- **準ニュートン法**: 2階微分(ヘッセ行列)を用いる方法

準ニュートン法

求根アルゴリズムであるニュートン法では
$f(x)=0$となるxを求めるために$f'(x)$を用いた

最小値探索では$f'(x)=0$となるxを探せば良い

⇩

$g(x)=f'(x)$としてニュートン法で$g'(x)=f''(x)$
を使って$g(x)=0$となるxを見つければよいのでは？

[問題点] $f''=\nabla^2 f$ (ヘッセ行列)の**逆行列計算**
が難しい(計算時間がかかる)

⇒ $\nabla^2 f(x)$ を x や $\nabla f(x)$ で近似

準ニュートン法

準ニュートン法

目的関数の勾配$\nabla f(\boldsymbol{x})$を1次項までテイラー展開

$$\nabla f(\boldsymbol{x}+\Delta\boldsymbol{x}) \approx \nabla f(\boldsymbol{x}) + H\Delta\boldsymbol{x}$$

$H = \nabla^2 f(\boldsymbol{x})$は[ヘッセ行列]. **最小値では勾配が$\boldsymbol{0}$**
なので, $\nabla f(\boldsymbol{x}+\Delta\boldsymbol{x})$が$\boldsymbol{0}$となるためには:

$$\Delta\boldsymbol{x} = -H^{-1}\nabla f(\boldsymbol{x}^{(k)})$$

この$\Delta\boldsymbol{x}$を使って, $\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} + \Delta\boldsymbol{x}$ で解を更新

$\nabla f(\boldsymbol{x}+\Delta\boldsymbol{x}) = \nabla f(\boldsymbol{x}) + H\Delta\boldsymbol{x}$ (secant条件)
を満たすHの近似値を考える

準ニュートン法

secant条件を満たすHの近似式(H^{-1}が容易に計算可能なことも条件)

- **BFGS法***

  $\boldsymbol{y}_k = \nabla f\left(\boldsymbol{x}^{(k+1)}\right) - \nabla f\left(\boldsymbol{x}^{(k)}\right), \boldsymbol{x}_k = \boldsymbol{x}^{(k)}, \boldsymbol{s}_k = \Delta \boldsymbol{x}_k$とすると:

$$H^{(k+1)} = H^{(k)} + \frac{\boldsymbol{y}_k \boldsymbol{y}_k^T}{\boldsymbol{y}_k^T \boldsymbol{s}_k} - \frac{H^{(k)} \boldsymbol{s}_k \boldsymbol{s}_k^T H^{(k)}}{\boldsymbol{s}_k^T H^{(k)} \boldsymbol{s}_k}$$

⇩ 逆行列H^{-1}の更新式

$$\boxed{H^{-1} \leftarrow H^{-1} + \frac{(\boldsymbol{s}_k^T \boldsymbol{y}_k + \boldsymbol{y}_k^T H^{-1} \boldsymbol{y}_k)(\boldsymbol{s}_k \boldsymbol{s}_k^T)}{(\boldsymbol{s}_k^T \boldsymbol{y}_k)^2} - \frac{H^{-1} \boldsymbol{y}_k \boldsymbol{s}_k^T + \boldsymbol{s}_k \boldsymbol{y}_k^T H^{-1}}{\boldsymbol{s}_k^T \boldsymbol{y}_k}}$$

*BFGS法は多くの数値計算ライブラリ(GSLやSciPy,Eigenなど)で採用されている. 他にはDFP法,ブロイデン法, SR1など.

準ニュートン法

BFGS法の計算手順

1. 初期値$\boldsymbol{x}_0$を設定し, ヘッセ行列の逆行列H^{-1}を単位行列で初期化
2. 以下の手順を収束するまで繰り返す($k = 0,1,\ldots$)
   a. 探索方向$\boldsymbol{p}^{(k)}$の計算：$\boldsymbol{p}^{(k)} = -H^{-1}\nabla f(\boldsymbol{x}_k)$
   b. 黄金分割法などの1次元探索で探索幅$\alpha^{(k)}$を計算：
   $$\alpha^{(k)} = \arg\min_{\alpha} f(\boldsymbol{x}^{(k)} + \alpha \boldsymbol{p}^{(k)})$$
   c. 探索方向ベクトル$\boldsymbol{s}_k = \Delta \boldsymbol{x}_k = \alpha^{(k)}\boldsymbol{p}^{(k)}$を計算して, 探索位置を更新：$\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} + \boldsymbol{s}_k$
   d. 勾配変化$\boldsymbol{y}_k$の計算：$\boldsymbol{y}_k = \nabla f\left(\boldsymbol{x}^{(k+1)}\right) - \nabla f(\boldsymbol{x}^{(k)})$
   e. $|\nabla f(\boldsymbol{x}^{(k+1)})| < \varepsilon$なら反復終了
   f. **前ページ青枠の式**でH^{-1}を更新

準ニュートン法

BFGS法のコード例

```
int k;
for(k = 0; k < max_iter; ++k){
    // 探索方向dの計算(d = -H▽f)
    vector<double> p(n, 0.0);
    for(int i = 0; i < n; ++i){
        for(int j = 0; j < n; ++j) p[i] -= H[i][j]*g[j];
    }

    // 黄金分割法で探索方向d上の最小値までの距離を求める
    goldensection(func, x, p, 0.0, 3.0, alpha, 20, 1e-4);

    // sの計算とxの更新
    vector<double> s(n, 0.0);
    for(int i = 0; i < n; ++i){
        s[i] = alpha*p[i]; x[i] += s[i];
    }

    // yの計算
    vector<double> gprev = g;
    g = dfunc(x);
    vector<double> y(n, 0.0);
    for(int i = 0; i < n; ++i) y[i] = g[i]-gprev[i];
```

探索方向$\boldsymbol{p} = -H^{-1}\nabla f$の計算

黄金探索法で$\boldsymbol{p}$方向に最小値探索して, そこまでの距離をαとする.
1反復で進む距離をパラメータ設定しなくてよく, 収束も早くなる

探索位置xの更新:$x \leftarrow x + s, s = \alpha p$

勾配の差:$y = \nabla f\left(x^{(k+1)}\right) - \nabla f\left(x^{(k)}\right)$

準ニュートン法

BFGS法のコード例(前ページからの続き)

```
gnorm = 0.0;
for(int i = 0; i < n; ++i) gnorm += g[i]*g[i];
if(sqrt(gnorm) < eps) break; // 収束判定

// H*yの計算(行列×縦ベクトル⇒縦ベクトル)
vector<double> Hy(n, 0.0);
for(int i = 0; i < n; ++i){
    for(int j = 0; j < n; ++j) Hy[i] += H[i][j]*y[j];
}
// (s^T)*yと(y^T)*(H*y)の計算(どちらもベクトル同士の内積)
double sy = 0.0, yHy = 0.0;
for(int i = 0; i < n; ++i){
    sy += s[i]*y[i]; yHy += y[i]*Hy[i];
}

// H^-1の更新
for(int i = 0; i < n; ++i){
    for(int j = 0; j < n; ++j){
        H[i][j] += s[i]*s[j]/sy + yHy*s[i]*s[j]/(sy*sy) -
                   Hy[i]*s[j]/sy - s[i]*Hy[j]/sy;
    }
}
}
```

$|\nabla f|$による収束判定

H^{-1}の更新のために $H^{-1}\boldsymbol{y}_k$ と $\boldsymbol{s}_k^T\boldsymbol{y}_k$, $\boldsymbol{y}_k^T H^{-1}\boldsymbol{y}_k$を先に計算しておく

H^{-1}の更新

準ニュートン法

BFGS法の実行結果例

$$f(x) = x^2 + y^2 - xy - x - 1$$

最小解 $\boldsymbol{x}_{min} = (\frac{2}{3}, \frac{1}{3})$

許容誤差$\varepsilon = 1 \times 10^{-6}$,初期探索点$(0, 0)$

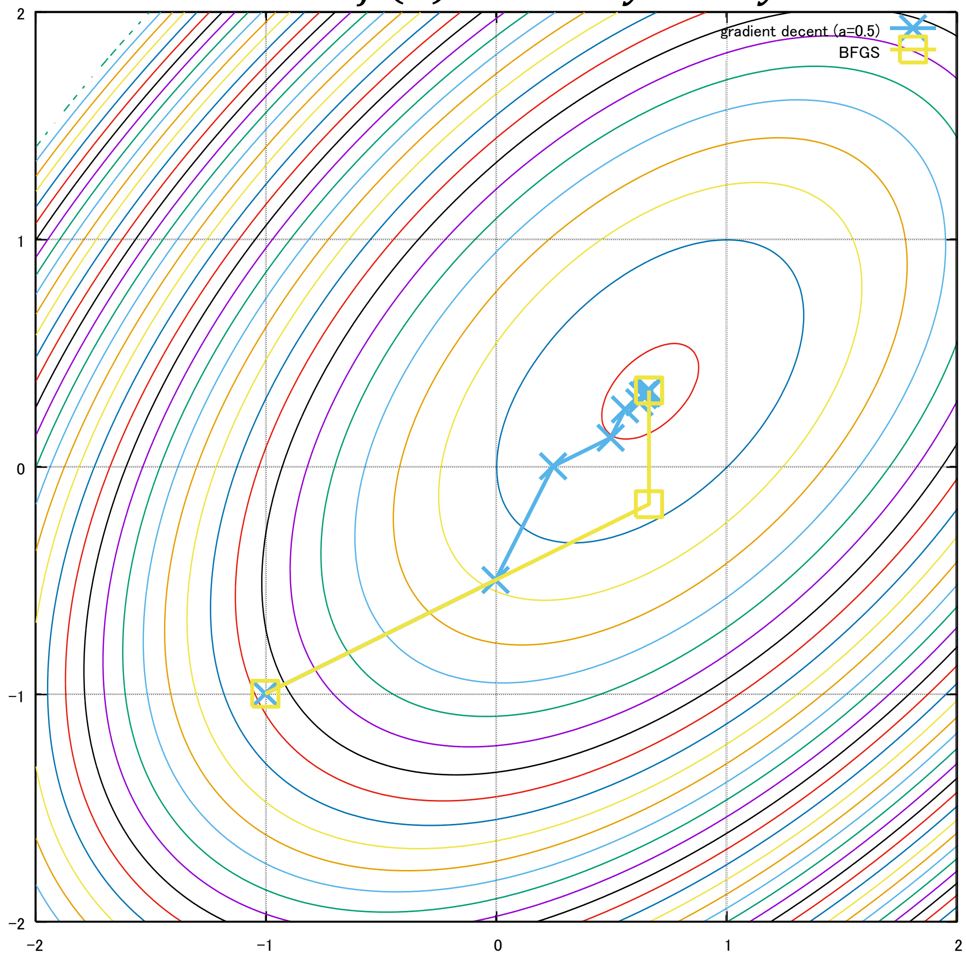
```
f(0,        0)        = -1
f(0.499948, 0)        = -1.25
f(0.666636, 0.333307) = -1.33333
f(0.666667, 0.333333) = -1.33333
(x,y) = (0.666667,0.333333),  f = -1.33333
```

4回の反復で処理終了

準ニュートン法

最急降下法との比較

$$f(x) = x^2 + y^2 - xy - x - 1$$



初期点 $\boldsymbol{x}^{(0)} = (-1, -1)$

最小解 $\boldsymbol{x}_{min} = \left(\frac{2}{3}, \frac{1}{3}\right)$

方法	反復回数
最急降下法 $(\alpha = 0.5)$	22
$BFGS$法	4

$BFGS$法は初期値を$(-1, -1)$に変えたため反復回数が少し増えている

準ニュートン法

その他の微分を使う最適化法

- Levenberg-Marquardt法(レーベンバーグ·マルカート法)

  最急降下法と**ガウス·ニュートン法***を**組み合わせた**方法. 解から遠いときは最急降下法, 局所的に凸関数近似できるようになったらガウス·ニュートン法を使う. **非線形最小自乗問題****を解く場合によく使われる

*ガウス·ニュートン法は準ニュートン法と同じくヘッセ行列を近似する手法で, 非線形最小自乗問題に限定したもの(凸関数限定).
**非線形回帰分析とも言われる. 得られたデータに対してそれに合った非線形方程式を探し出す問題(次回の「補間法」で最小自乗問題として出てきます)

今回の講義内容

- 今日の問題
- 3分割法, 黄金分割法
- 最急降下法, 準ニュートン法
- **線形計画問題について**
- シンプレックス法

今回の講義で解く問題 その2

$$\arg\max_{x} f(x)$$

$$\text{subject to } x \in F$$

ただし $f(x)$ は線形

今回の講義で解く問題 その2

より具体的には

$$
\begin{aligned}
&\arg\max_{x} \boldsymbol{c}^T \boldsymbol{x} \\
&\text{s.t. } A\boldsymbol{x} \le \boldsymbol{b} \\
&\qquad x_j \ge 0 \quad (j = 1, \dots, n)
\end{aligned}
$$

ここで $\boldsymbol{c} = (c_1, \dots, c_n)^T, x = (x_1, \dots, x_n)^T,$
$A = \{a_{ij}\}, b = (b_1, \dots, b_n)$

今回の講義で解く問題 その2

線形制約付きの線形最適化問題：**線形計画問題**
(Linear Program : LP)

例) x, yが不等式$2x + y \leq 8, x + 3y \leq 9, x \geq 0, y \geq 0$を満たすとき, $f(x, y) = x + y$ の値の最大値を求めよ.

実際の問題に置き換えると:
製品A,Bを作るのに以下の表に示した量の部品①,②を必要とする(表は在庫も含む). 製品A,Bはそれぞれ1万円の利益を出す. このとき利益を最大とするそれぞれの生産量を求めよ.

部品	製品Aに必要な数	製品Bに必要な数	在庫
①	2	1	8
②	1	3	9

今回の講義で解く問題 その2

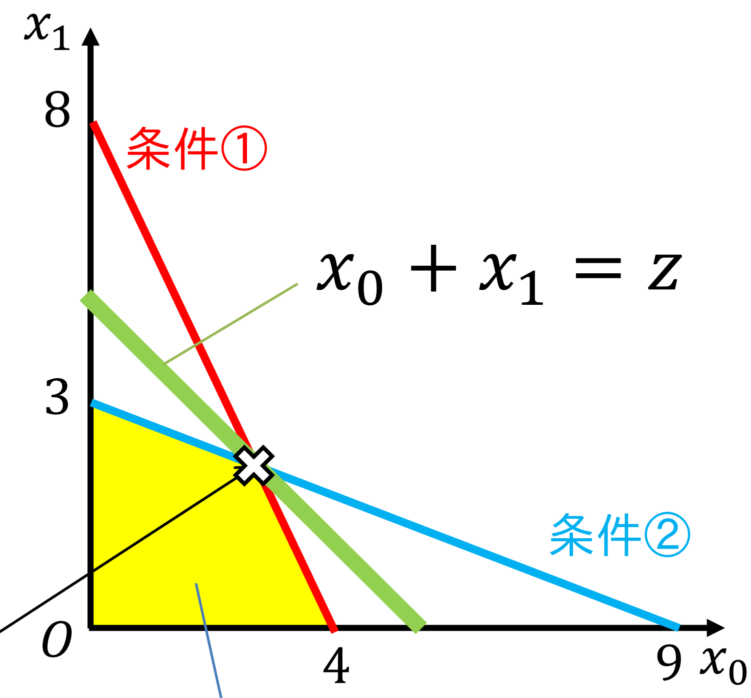
例) x, yが不等式$2x + y \leq 8, x + 3y \leq 9, x \geq 0, y \geq 0$を満たすとき, $f(x, y) = x + y$ の値の最大値を求めよ.

目的関数 : $z = x_0 + x_1$
制約条件 :

$$2x_0 + x_1 \leq 8 \quad \cdots ①$$
$$x_0 + 3x_1 \leq 9 \quad \cdots ②$$
$$x_0 \geq 0,\ x_1 \geq 0$$

制約条件を満たしつつ, zが最大となるのは目的関数がこの交点を通るとき

$\Rightarrow (x_0, x_1) = (3, 2)$で$z_{max} = 5$



$x_0 \geq 0, x_1 \geq 0$を含めて**すべての制約条件**を満たす領域

今回の講義で解く問題 その2

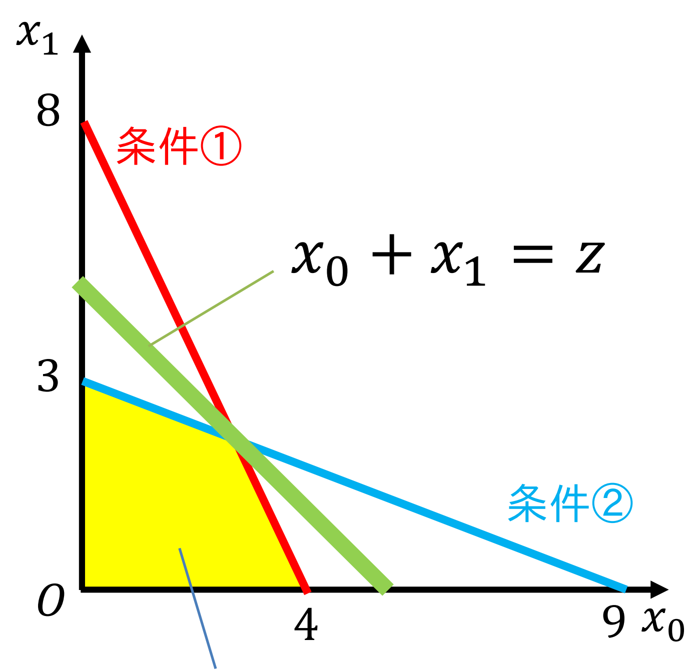
例) x, yが不等式$2x + y \leq 8, x + 3y \leq 9, x \geq 0, y \geq 0$を満たすとき, $f(x, y) = x + y$ の値の最大値を求めよ.

制約条件式が作る**すべての交点を目的関数が通る**ときのzの値を調べれば**解ける**

⇒ 変数や条件の数が増えると**交点数が膨大**になる

効率的に解ける方法

シンプレックス法



今回の講義内容

- 今日の問題
- 3分割法, 黄金分割法
- 最急降下法, 準ニュートン法
- 線形計画問題について
- **シンプレックス法**

シンプレックス法

手順1：スラック変数の導入

不等式が含まれると扱いづらい
⇒ **スラック変数**x_2, x_3を導入して**条件式を等式**にする

目的関数：$z = x_0 + x_1$
制約条件：

$$2x_0 + x_1 \leq 8 \quad \cdots ①$$
$$x_0 + 3x_1 \leq 9 \quad \cdots ②$$
$$x_0, x_1 \geq 0$$

⇒

$$z = x_0 + x_1$$
$$2x_0 + x_1 + x_2 = 8 \quad \cdots ①$$
$$x_0 + 3x_1 + x_3 = 9 \quad \cdots ②$$
$$x_0, x_1, x_2, x_3 \geq 0$$

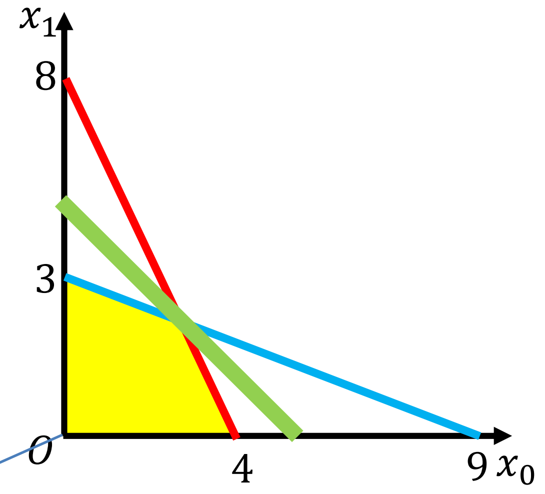
左辺の量が少ないので, 左辺に何らかの正の数x_2, x_3
を足して等式にするということ

シンプレックス法

手順2：何でもいいので基底可能解を1つ見つける

基底可能解：すべての制約条件を満たし, 手順1で作成した方程式が成り立つ解(x_0, x_1) (スラック変数は解に含めない)

$$
\begin{aligned}
&z = x_0 + x_1 \\
&2x_0 + x_1 + x_2 \qquad = 8 \quad \cdots ① \\
&x_0 + 3x_1 \qquad + x_3 = 9 \quad \cdots ② \\
&x_0, x_1, x_2, x_3 \geq 0
\end{aligned}
$$



原点, つまり$(x_0, x_1) = (0, 0)$なら条件を満たす領域内で方程式も成り立つ(この場合, $x_2 = 8, x_3 = 9$とすれば方程式が成り立つ)

シンプレックス法

手順3：現在の解が最適解か調べる

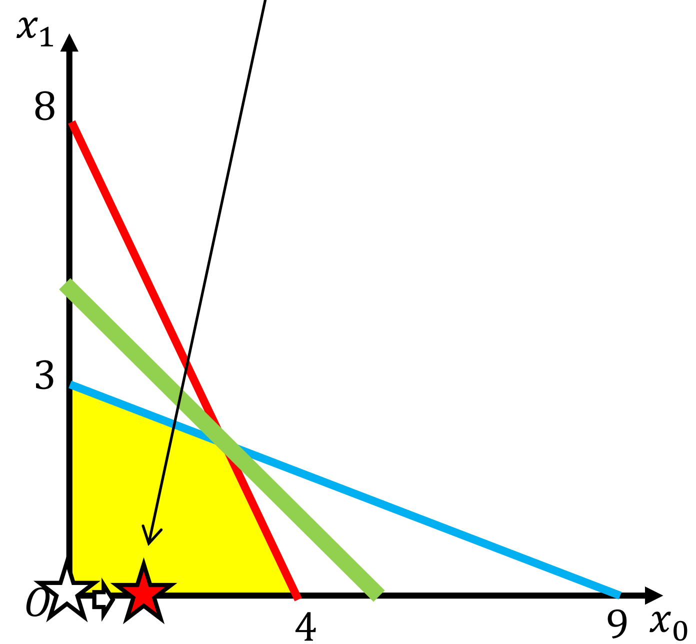
現在の解$(x_0, x_1) = (0, 0)$ で例えば, x_0を1増やしても基底可能解のままで, zの値は大きくなる

⇒ **最適解ではない**

最適解ならば, $z = x_0 + x_1$**で x_0, x_1 の数を増やしてもzは増えない.**

例えば, $z = -x_0 - x_1$なら x_0, x_1を増やしてもzは増えないので$(0, 0)$が最適解となる

$(x_0, x_1 \geq 0$という条件込みで$)$



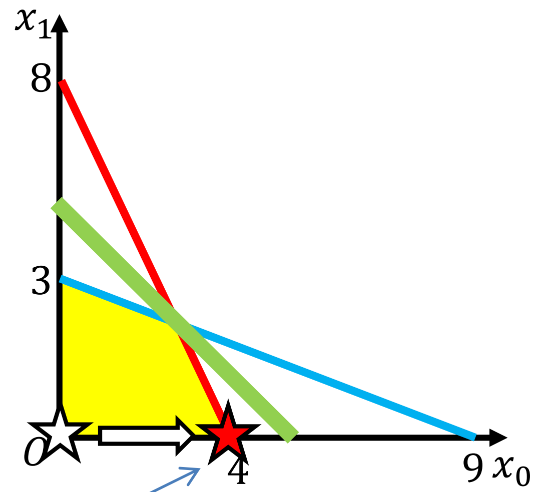
シンプレックス法

手順4：最適解でなければ解を改良して手順3に戻る

$(x_0, x_1) = (\alpha, 0)$ として, x_0をαだけ増やすことを考えてみる.
式①, ②に$(x_0, x_1) = (\alpha, 0)$を代入すると:

$$2\alpha + x_2 = 8 \Rightarrow x_2 = 8 - 2\alpha \geq 0$$
$$\alpha + x_3 = 9 \Rightarrow x_3 = 9 - \alpha \ \ \underline{\geq 0}$$

$x_2, x_3 \geq 0$より



$\alpha \leq 4, \alpha \leq 9$ の両方を満たすαの
最大値は $\alpha = 4$

⇩ 上の式に代入して現在の解を更新

$(x_0, x_1, x_2, x_3) = (4, 0, 0, 5)$ で $z = 4$

シンプレックス法

手順3(2回目)：現在の解が最適解か調べる

現在の解$(x_0, x_1, x_2, x_3) = (4, 0, 0, 5)$ で0となっているx_1, x_2について値を増やしてzが大きくなるか調べてみる

$$
\begin{aligned}
z &= x_0 + x_1 \\
2x_0 + x_1 + x_2 &= 8 \quad \cdots ① \\
x_0 + 3x_1 + x_3 &= 9 \quad \cdots ② \\
x_0, x_1, x_2, x_3 &\geq 0
\end{aligned}
$$

式①より：

$$x_0 = 4 - \frac{1}{2}x_1 - \frac{1}{2}x_2$$

⇩

式①以外に代入して，x_0の影響を消去

$$
\begin{aligned}
z &= 4 + \frac{1}{2}x_1 - \frac{1}{2}x_2 \\
2x_0 + x_1 + x_2 &= 8 \quad \cdots ① \\
\frac{5}{2}x_1 - \frac{1}{2}x_2 + x_3 &= 5 \quad \cdots ②' \\
x_0, x_1, x_2, x_3 &\geq 0
\end{aligned}
$$

zの式から，x_1を増やすとzが大きくなる

⇒ **まだ最適解ではない**

シンプレックス法

手順4(2回目)：最適解でなければ解を改良して手順3に戻る

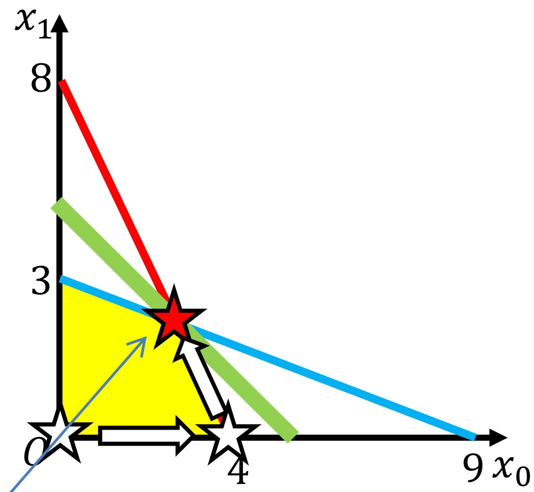
$(x_1, x_2) = (\alpha, 0)$ として, x_1をαだけ増やすことを考えてみる.
式①, ②'に$(x_1, x_2) = (\alpha, 0)$を代入すると:

$$2x_0 + \alpha = 8 \Rightarrow x_0 = 4 - \frac{1}{2}\alpha \geq 0$$
$$\frac{5}{2}\alpha + x_3 = 5 \Rightarrow x_3 = 5 - \frac{5}{2}\alpha \ \geq 0$$

$\alpha \leq 8, \alpha \leq 2$ の両方を満たすαの最大値は $\alpha = 2$

⇩ 上の式に代入して現在の解を更新

$(x_0, x_1, x_2, x_3) = (3, 2, 0, 0)$ で $z = 5$



シンプレックス法

手順3(3回目)：現在の解が最適解か調べる

現在の解$(x_0, x_1, x_2, x_3) = (3, 2, 0, 0)$ で0となっているx_2, x_3について値を増やしてzが大きくなるか調べてみる

式②'より：

$$x_1 = 2 + \frac{1}{5}x_2 - \frac{2}{5}x_3$$

式②'以外に代入して，
x_1の影響を消去

$$\begin{aligned} &z = 4 + \tfrac{1}{2}x_1 - \tfrac{1}{2}x_2 \\ &2x_0 + x_1 + x_2 = 8 \quad \cdots ① \\ &\tfrac{5}{2}x_1 - \tfrac{1}{2}x_2 + x_3 = 5 \quad \cdots ②' \\ &x_0, x_1, x_2, x_3 \geq 0 \end{aligned}$$

$$\begin{aligned} &z = 5 - \tfrac{2}{5}x_2 - \tfrac{1}{5}x_3 \\ &2x_0 + \tfrac{6}{5}x_2 - \tfrac{2}{5}x_3 = 6 \quad \cdots ①' \\ &\tfrac{5}{2}x_1 - \tfrac{1}{2}x_2 + x_3 = 5 \quad \cdots ②' \\ &x_0, x_1, x_2, x_3 \geq 0 \end{aligned}$$

zの式から，x_2, x_3を増やしてもzは大きくならない

⇒ **最適解！**

シンプレックス法

無事に最適解$(x_0, x_1) = (3, 2)$が得られたけど
プログラムにするのが難しそう...

⇒ **単体表**を使うことで機械的に解けるようになる!

スラックス変数x_2, x_3を導入して, 目的関数
含めて問題を置き換え

$$
\begin{aligned}
&z = x_0 + x_1 \\
&\quad 2x_0 + x_1 + x_2 \qquad = 8 \\
&\quad x_0 + 3x_1 \qquad + x_3 = 9 \\
&\quad x_0, x_1, x_2, x_3 \geq 0
\end{aligned}
$$

$$
\begin{aligned}
2x_0 + \ \ x_1 + x_2 \qquad &= 8 \\
x_0 + 3x_1 \qquad + x_3 &= 9 \\
z - x_0 - \ \ x_1 \qquad &= 0
\end{aligned}
$$

線形システム$(A\boldsymbol{x} = \boldsymbol{b})$になった?

シンプレックス法

手順1: **単体表(シンプレックス表)の作成**

各係数+最後の列に基底可能解
を並べた表を作る.

$$
\begin{aligned}
2x_0 + x_1 + x_2 \quad &= 8 \\
x_0 + 3x_1 \quad + x_3 &= 9 \\
z - x_0 - x_1 \quad &= 0
\end{aligned}
$$

基底可能解$(x_0, x_1) = (0, 0)$とすると
右辺項が(x_2, x_3)の基底可能解になる

基底変数	z	x_0	x_1	x_2	x_3	基底可能解
x_2	0	2	1	1	0	8
x_3	0	1	3	0	1	9
z	1	-1	-1	0	0	0

基底可能解が0
以外の変数+zを
基底変数にする

⇩

初期基底変数は
スラック変数になる

係数を並べて入力

シンプレックス法

手順2: 現在の解が最適解か調べる

一番下の行で, 非基底変数(現在はx_0とx_1)のところに**負の値**が入っていたらまだ**最適解ではない**

基底変数	z	x_0	x_1	x_2	x_3	基底可能解
x_2	0	2	1	1	0	8
x_3	0	1	3	0	1	9
z	1	-1	-1	0	0	0

非基底変数x_0, x_1の列の一番下に負の値があるので, まだ最適解ではない

シンプレックス法

手順3:最適解でなければ解を改良して手順2に戻る

負の値となっている非基底変数で**絶対値最大のもの**を選択し, **基底変数にする** (絶対値が同じならどちらでもよい. 今回は左側を選択)

⇒ 前のやり方での x_0をαだけ増やすという手順に相当

基底変数	z	x_0	x_1	x_2	x_3	基底可能解
x_2	0	2	1	1	0	8
x_3	0	1	3	0	1	9
z	1	-1	-1	0	0	0

ピボット要素

ピボット要素の列の値で基底可能解の列の値を割ったときに値が最小となる行の基底変数を新しい基底変数に置き換え

(今回は基底可能解を割ると$\frac{8}{2} = 4, \frac{9}{1} = 9$で, $x_2 \to x_0$とする)

シンプレックス法

手順3(続き):最適解でなければ解を改良して手順2に戻る

置き換えた行を**ピボット要素で割る**

⇒ αの最大値を求めるための前段階

基底変数	z	x_0	x_1	x_2	x_3	基底可能解
x_0	0	2	1	1	0	8
x_3	0	1	3	0	1	9
z	1	-1	-1	0	0	0

青枠の値でx_0の行を割っていく

基底変数	z	x_0	x_1	x_2	x_3	基底可能解
x_0	0	1	1/2	1/2	0	4
x_3	0	1	3	0	1	9
z	1	-1	-1	0	0	0

シンプレックス法

手順3(続き):最適解でなければ解を改良して手順2に戻る

ピボット要素の**他の列がすべて0**になるように, 置き換えた行を**何倍かして他の行から引く**

⇒ 前のやり方でαの最大値を求めて新しい解を得ることに相当

基底変数	z	x_0	x_1	x_2	x_3	基底可能解
x_0	0	1	1/2	1/2	0	4
x_3	0	1	3	0	1	9
z	1	−1	−1	0	0	0

⇩ 青枠の値を何倍かして他の行から引く

基底変数	z	x_0	x_1	x_2	x_3	基底可能解
x_0	0	1	1/2	1/2	0	4
x_3	0	0	5/2	−1/2	1	5
z	1	0	−1/2	1/2	0	4

シンプレックス法

手順2と3(2回目): 現在の解が最適解か調べて改良

負の値となっている非基底関数で**絶対値最大のもの**を選択し, **基底変数にする** (絶対値が同じならどちらでもよい. 今回は左側を選択)

⇒ 前のやり方での x_1 を α だけ増やすという手順に相当

基底変数	z	x_0	x_1	x_2	x_3	基底可能解
x_0	0	1	1/2	1/2	0	4
x_3	0	0	5/2	−1/2	1	5
z	1	0	−1/2	1/2	0	4

ピボット要素で基底可能解を割った値から
基底変数 x_3 を x_1 にしてその行を割る

基底変数	z	x_0	x_1	x_2	x_3	基底可能解
x_0	0	1	1/2	1/2	0	4
x_1	0	0	1	−1/5	2/5	2
z	1	0	−1/2	1/2	0	4

シンプレックス法

手順2と3(2回目): 現在の解が最適解か調べて改良

ピボット要素の**他の列がすべて0**になるように, 置き換えた行を**何倍かして他の行から引く**
⇒ 前のやり方でαの最大値を求めて新しい解を得ることに相当

基底変数	z	x_0	x_1	x_2	x_3	基底可能解
x_0	0	1	1/2	1/2	0	4
x_1	0	0	1	−1/5	2/5	2
z	1	0	−1/2	1/2	0	4

ピボット要素で基底可能解を割った値から
基底変数x_3をx_1にしてその行を割る

基底変数	z	x_0	x_1	x_2	x_3	基底可能解
x_0	0	1	0	3/5	−1/5	3
x_1	0	0	1	−1/5	2/5	2
z	1	0	0	2/5	1/5	5

シンプレックス法

手順2と3(2回目): 現在の解が最適解か調べて改良

一番下の行で, 非基底変数(現在はx_2とx_2)のところに**負の値**が入っていないので**最適解に収束**

基底変数	z	x_0	x_1	x_2	x_3	基底可能解
x_0	0	1	0	3/5	−1/5	**3**
x_1	0	0	1	−1/5	2/5	**2**
z	1	0	0	2/5	1/5	**5**

最適解$(3, 2)$

最大値$z = 5$

負の値がないから
最適解に収束している

この列はzの係数列として
入れたけど, 値は変わって
いないので**実際のプログラ**
ムでは省いています

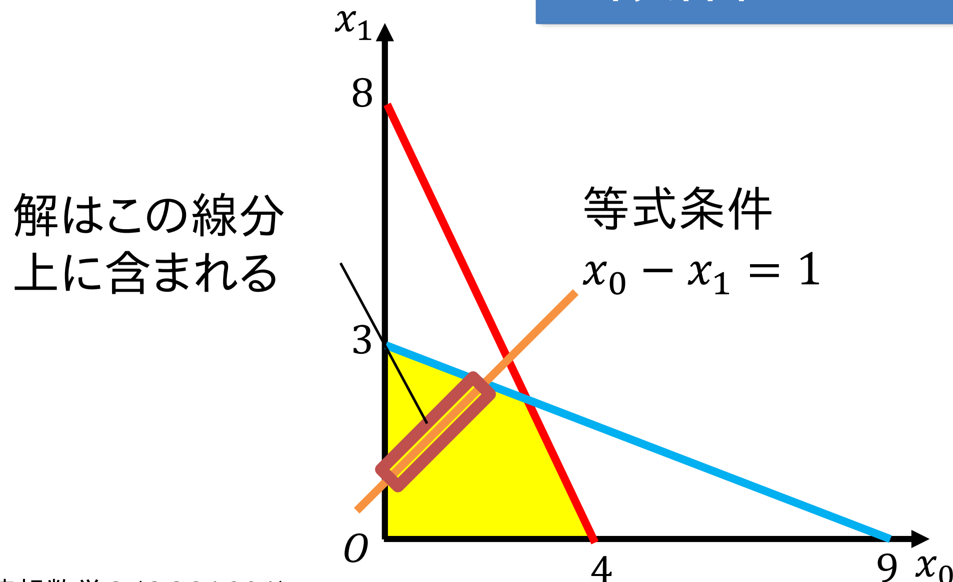
シンプレックス法

単体表を使った方法の問題

条件式に不等式でなく, 等式(例えば, $x_0 - x_1 = 1$)が入っていると, 最初の基底可能解$(0, 0)$が成り立たない

⇒ スラック変数とは別に**人工変数**を更に追加して,
基底可能解を求めた後, 通常の単体表を使って解く

2段階シンプレックス法



等式が含まれる場合だけでなく, 条件を満たす領域に原点が含まれない場合はこちらを使う
(条件不等式に≤と≥の両方が含まれる場合など)

シンプレックス法

シンプレックス法のコード例(単体表の作成部分)

```
int n = n_cond+1;       // 式の数(条件式の数+最適化式の数)
int m_slack = n_cond;   // スラック変数の数
int m_all = m+m_slack; // スラック変数を含む全変数の数
// 単体表の作成
vector< vector<double> > s(n); // 単体表の行数は条件式の数+1=n, 最後の行が目的関数
vector<int> xi(n); // それぞれの行の基底変数(x1=0,x2=1,...とインデックス値を格納)
for(int i = 0; i < n; ++i){
    s[i].resize(m_all+1); // 単体表の列数はスラック変数を含む全変数の数+1,最後列は基底可能解
    if(i != n-1){ // 条件式の行(0～n-2行)
        xi[i] = i+m; // 初期基底変数はスラック変数
        int sgn = (a[i][m] < 0 ? -1 : 1); // 条件式の右辺項の符号
        for(int j = 0; j < m; ++j) s[i][j] = sgn*a[i][j];// 変数の係数を入れていく
        // スラック変数部分の初期係数は単位行列のような形に(ただし条件式の符号が>=の場合は-1をセット)
        for(int j = m; j < m_all; ++j) s[i][j] = (xi[i] == j ? sgn*eqn[i] : 0);
        s[i][m_all] = sgn*a[i][m];// 右辺項を初期基底可能解としてセット
    }
    else{// 最終行は最適化式(ここだけ別で設定)
        xi[i] = -1; // 最適化式の変数インデックスには-1を格納しておく
        for(int j = 0; j < m; ++j) s[i][j] = -a[i][j];// 最終行の最適化式は係数の符号を反転
        // 最後の行の最適化式では初期基底可能解に0をセット
        for(int j = m; j < m_all+1; ++j) s[i][j] = 0;
    }
}
```

シンプレックス法

シンプレックス法のコード例(反復処理部分，ピボット要素の探索まで)

```
int k;
for(k = 0; k < max_iter; ++k){
    // 非基底変数(初期状態ではスラック変数以外)から負で絶対値最大のものを選択
    int b = -1;
    double xmax = 0.0;
    for(int j = 0; j < m_all; ++j){
        if(s[n-1][j] < 0 && -s[n-1][j] > xmax){ b = j; xmax = -s[n-1][j]; }
    }
    if(b == -1) break; // 負の変数が見つからなければ収束したとしてループを抜ける

    // ピボット要素の探索
    int p = 0;
    double ymin = s[0][m_all]/s[0][b];
    for(int i = 1; i < n-1; ++i){
        // 基底可能解をb列の値で割った値が最小のものを選択
        double y = s[i][m_all]/s[i][b];
        if(y < ymin){ p = i; ymin = y; }
    }

    // ピボット要素行を選択された非基底変数(b)で置き換えて，ピボット要素でその行を割る
    ・・・(次ページ参照)・・・
}
```

シンプレックス法

シンプレックス法のコード例(反復処理部分, ピボット要素による処理)

```
int k;
for(k = 0; k < max_iter; ++k){
    // ピボット要素の探索
    ・・・(前ページ参照)・・・

    // ピボット要素行を選択された非基底変数(b)で置き換えて, ピボット要素でその行を割る
    xi[p] = b;
    double xp = s[p][b];// ループ内で値が変わるので一時変数に値を確保しておく
    for(int j = 0; j < m_all+1; ++j){
        s[p][j] /= xp;
    }

    // ピボット行(p)以外の行を非基底変数(b)の値が0になるようにピボット行の係数をx倍して引く
    //  -> s[p][b]は1になっているのでs[p][j]にs[i][b]を掛けて引けば良い
    for(int i = 0; i < n; ++i){
        if(i == p) continue; // ピボット行は飛ばす
        xp = s[i][b];// ループ内で値が変わるので一時変数に値を確保しておく
        for(int j = 0; j < m_all+1; ++j){
            s[i][j] -= s[p][j]*xp;
        }
    }
}
```

シンプレックス法

シンプレックス法の実行結果例

目的関数 : $z = x_0 + x_1$
制約条件 :

$$2x_0 + x_1 \leq 8 \quad \cdots ①$$
$$x_0 + 3x_1 \leq 9 \quad \cdots ②$$
$$x_0, x_1 \geq 0$$

最適解 $(x_0, x_1) = (3, 2)$で
目的関数の値$z = 5$

```
初期単体表
x3:       2       1       1       0       8
x4:       1       3       0       1       9
 z:      -1      -1       0       0       0
処理後の単体表
x1:       1       0     0.6    -0.2       3
x2:       0       1    -0.2     0.4       2
 z:       0       0     0.4     0.2       5
f(3,2) = 5
```

講義内容のまとめ

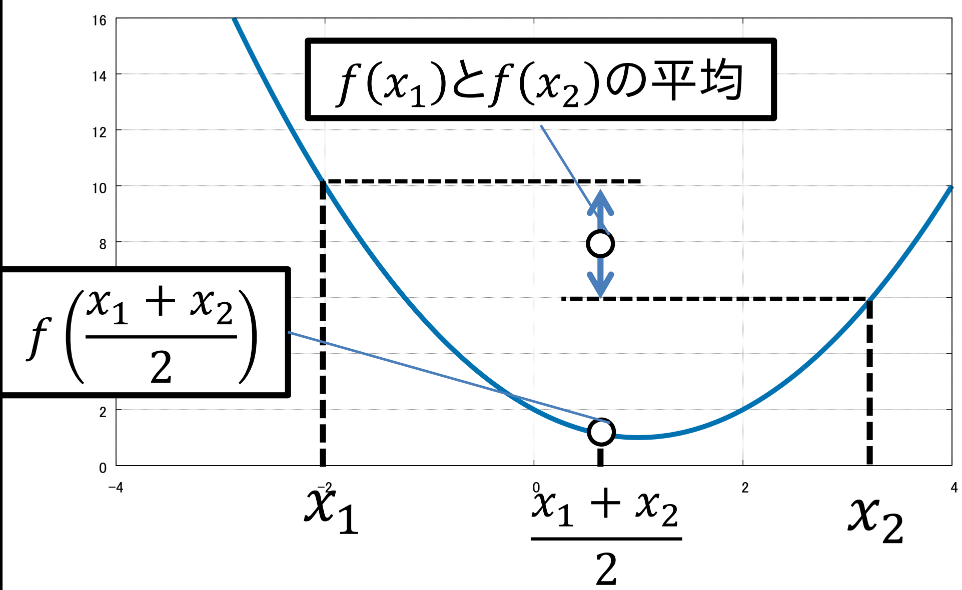
- 今日の問題：$\arg\min_{x} f(x)$
- 3分割法, 黄金分割法
  - 導関数を用いない最適化法
- 最急降下法, 準ニュートン法
  - 導関数を用いた最適化法
- 線形計画問題について
  - 制約条件付きの線形最適化
- シンプレックス法
  - 単体表を用いた解き方

Appendix

(以降のページは補足資料です)

凸関数

凸関数(convex function)：
2次関数$f(x) = ax^2 \ (a > 0)$のような下に凸*な形状の関数



下に凸であるならば，x_1, x_2における関数値の平均とその中点である$(x_1+x_2)/2$における関数値は以下の関係を持つ

$$f\left(\frac{x_1+x_2}{2}\right) \leq \frac{f(x_1)+f(x_2)}{2}$$

区間内で常に下に凸であるならば中点に限らず$[x_1, x_2]$の間のすべての点で上記が成り立つ$(0 \leq \alpha \leq 1)$:

$$f(\alpha x_1 + (1-\alpha)x_2) \leq \alpha f(x_1) + (1-\alpha) f(x_2)$$

*上に凸な場合は凹関数と呼ばれる

黄金分割比の導出1

右下の図のように長さ

$$b = x_3^{(k)} - x_0^{(k)}, \qquad c = x_2^{(k)} - x_0^{(k)}, \qquad d = b - c$$

を定義する.

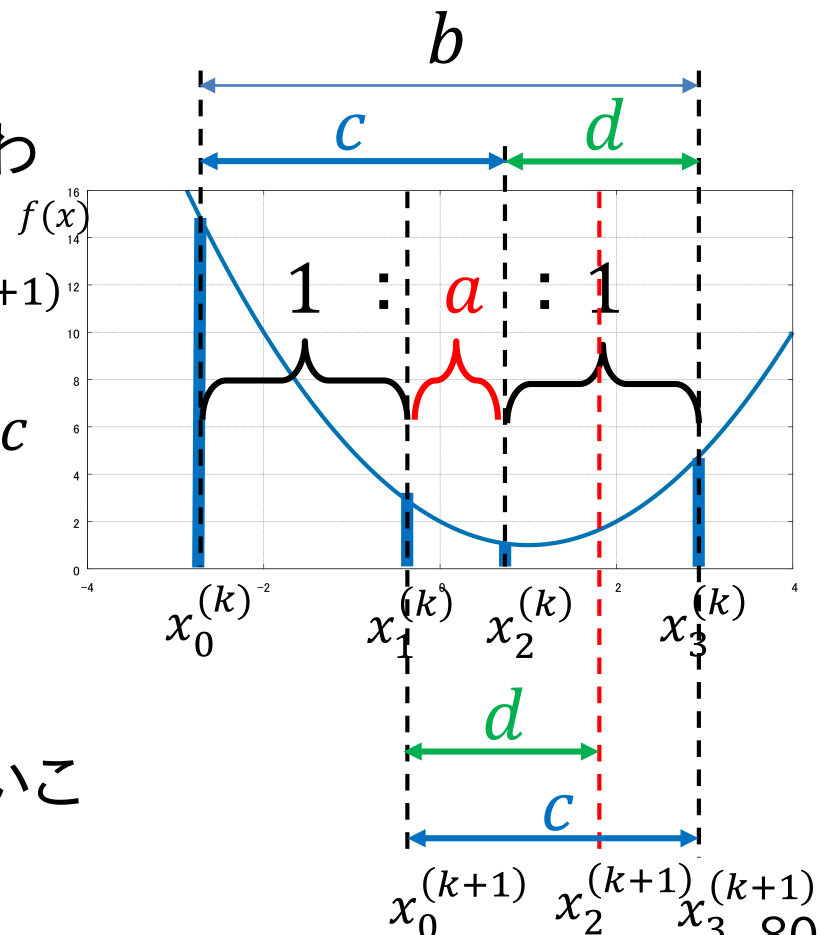
1回分割後($k \to k + 1$)に比率が変わらないならば,

$$\mathrm{c} = x_3^{(k+1)} - x_0^{(k+1)}, d = x_2^{(k+1)} - x_0^{(k+1)}$$

であり, $b : c = c : d$となる. $d = b - c$なので

$$\frac{b}{c} = \frac{c}{b - c}$$

が成り立つ比率$b : c$を求めれば良いことになる.



黄金分割比の導出2

$\frac{b}{c} = \frac{c}{b-c}$を変形すると，$\frac{b}{c}$に関する2次方程式にできる:

$$\left(\frac{b}{c}\right)^2 - \frac{b}{c} - 1 = 0$$

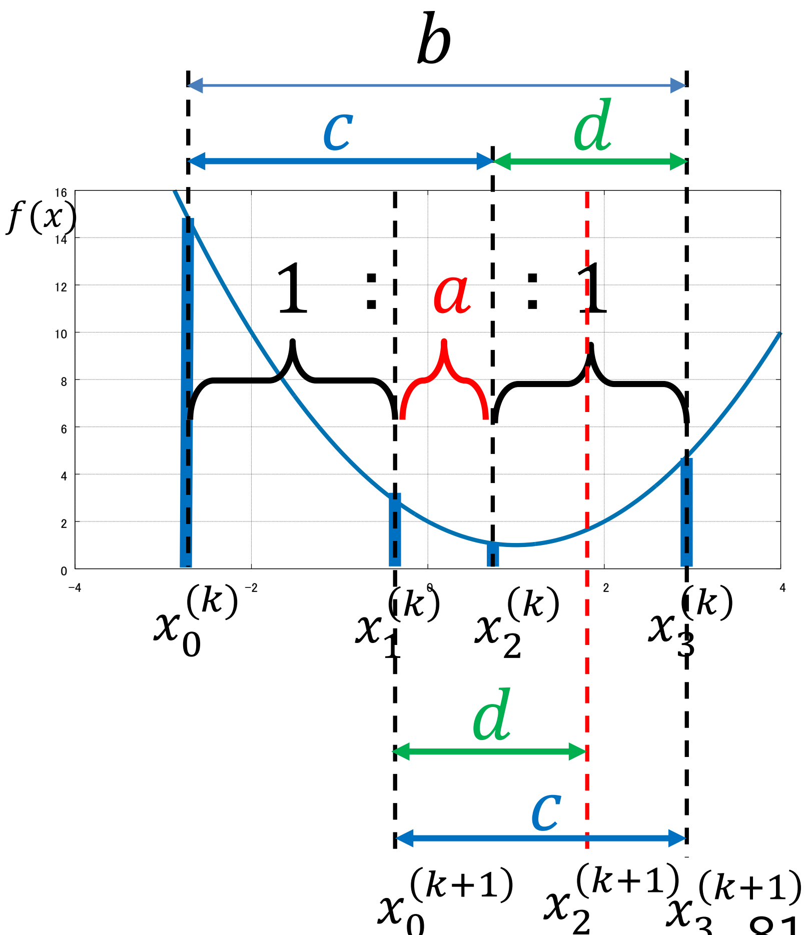
解の公式に当てはめると:

$$\frac{b}{c} = \frac{1 \pm \sqrt{5}}{2}$$

長さの比率なので正の値$(1 < \sqrt{5})$

$$\frac{b}{c} = \frac{1 + \sqrt{5}}{2}$$

(導出終了)



ヘッセ行列

関数 $f(x_1, x_2, \dots, x_n)$ がどの変数でも2階微分可能であるとき,

$$H(f) = \nabla^2 f$$

をヘッセ行列(Hessian)と呼ぶ. 行列Hのi行j列目の要素は,

$$H_{i,j} = \nabla_i \nabla_j f(\boldsymbol{x}) = \frac{\partial^2}{\partial x_i \partial x_j} f(\boldsymbol{x})$$

行列として書き下すと

$$H(f) = \begin{pmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \frac{\partial^2 f}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{pmatrix}$$