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)

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
- 制約付き最小化問題

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.

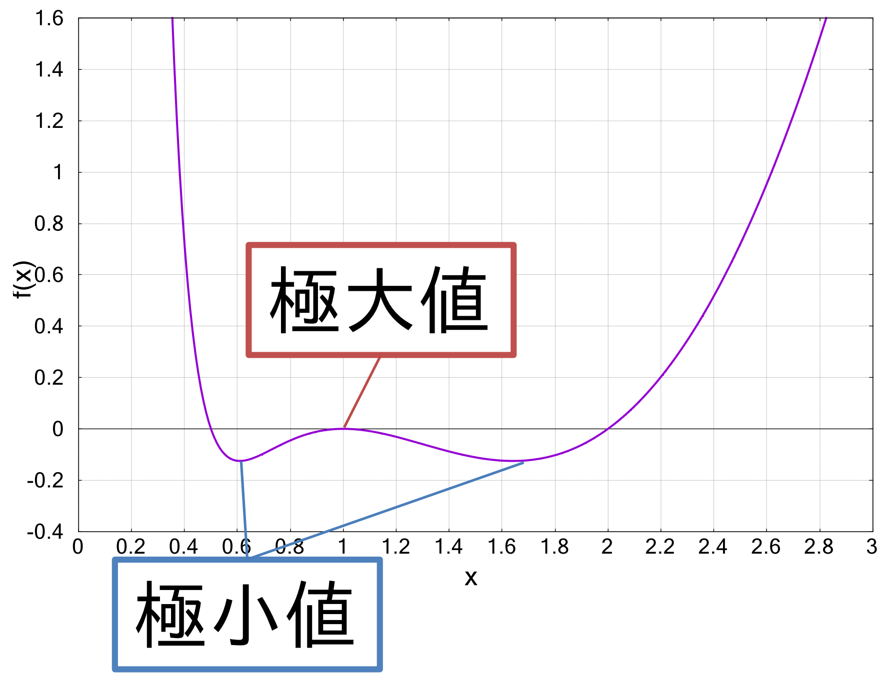
(2017年度筑波大学前期日程 数学問題より)

増減表

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

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**

*他にも離散最適化問題として組み合わせ最適化など
**線形計画問題についてはシステム数理IIで扱っている

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
- 制約付き最小化問題

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

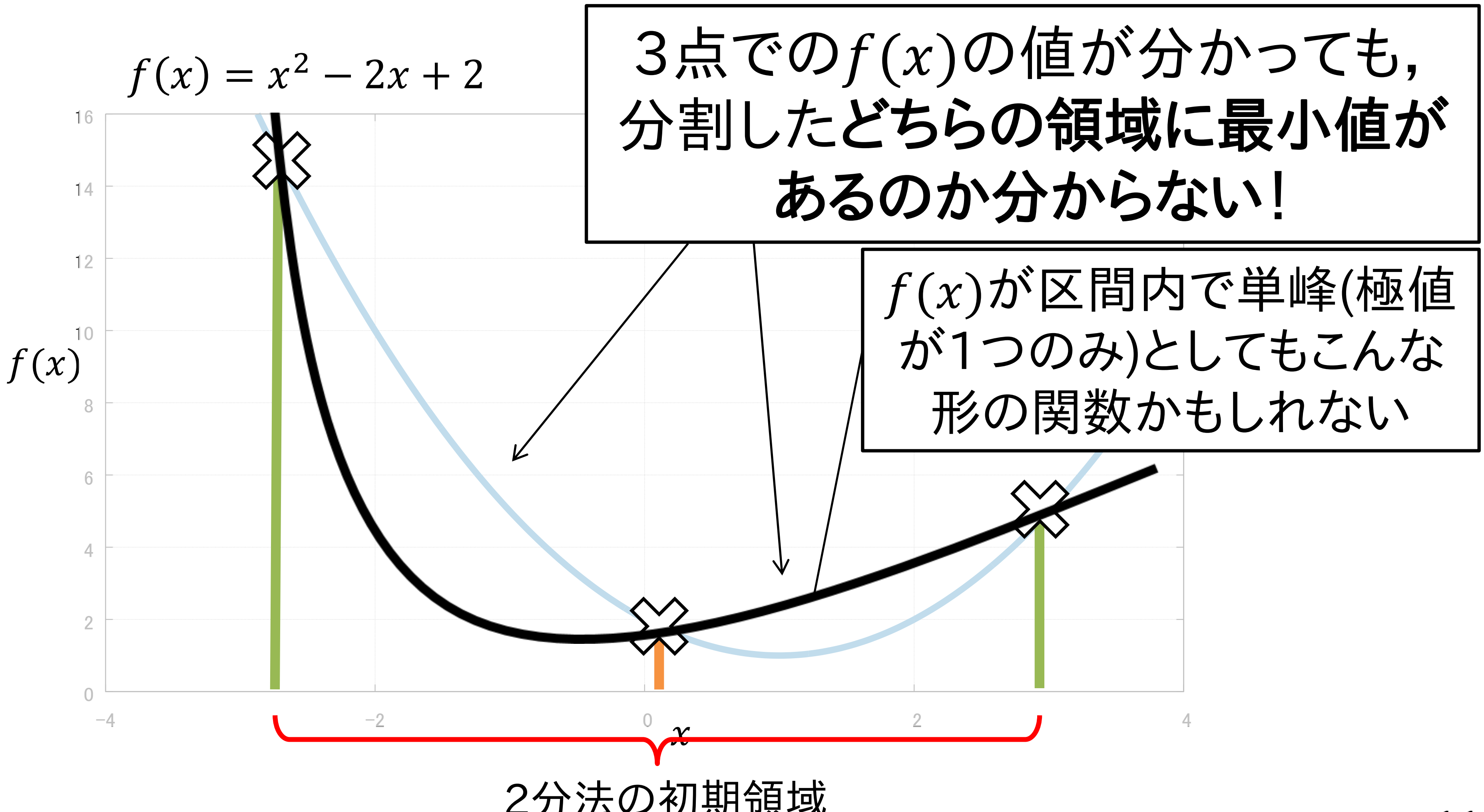
⇩

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

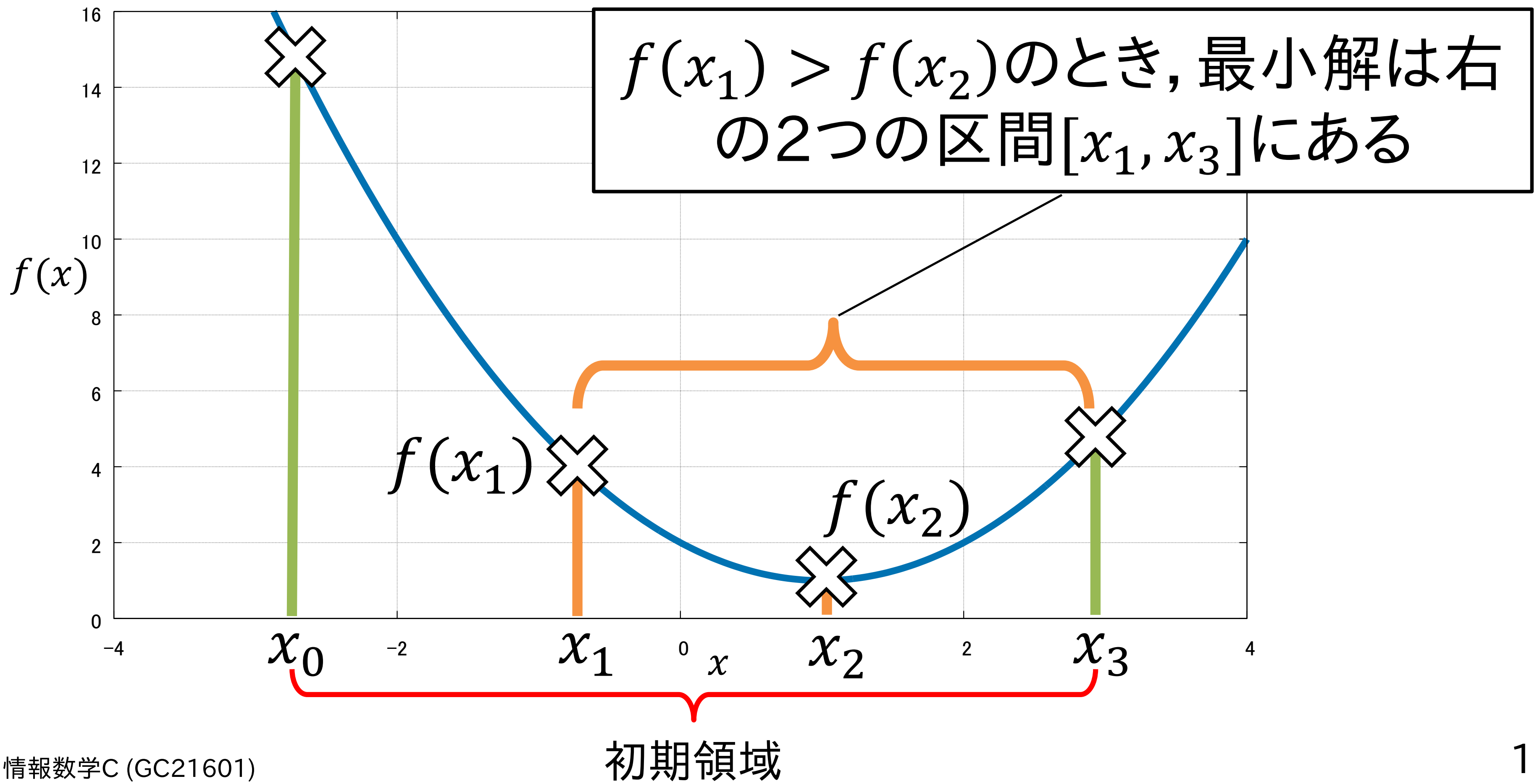
2分法で最適化問題を解く?

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



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

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

凸関数の性質

ただし, $0 \le \alpha \le 1$

また, $f(x_{min}) \le f(x_2)$ なので

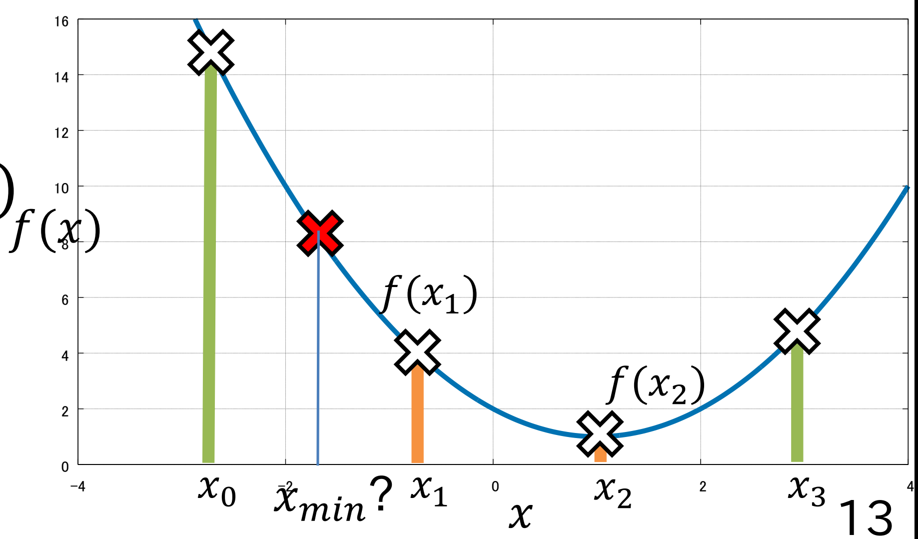
$$\alpha f(x_{min}) + (1-\alpha) f(x_2) \le f(x_2)$$

⇩

$f(x_1) \le f(x_2)$となり

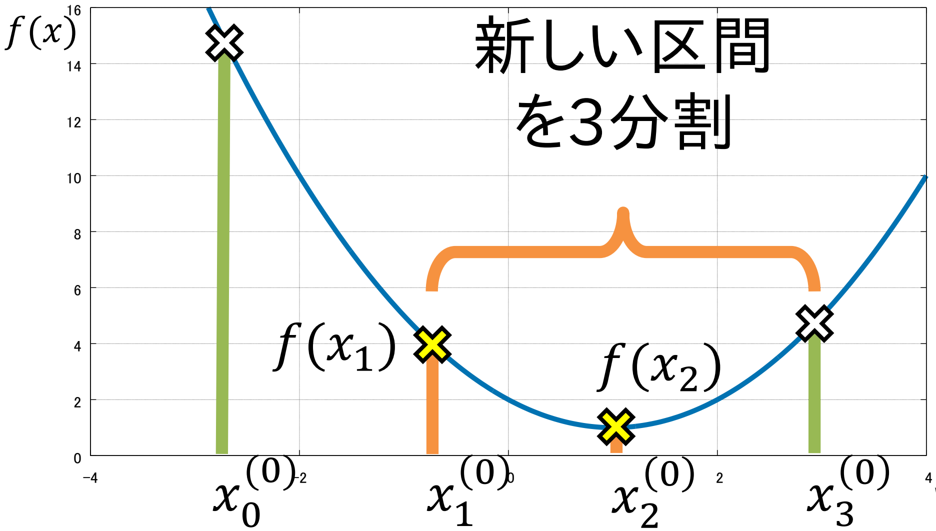
$f(x_1) > f(x_2)$と**矛盾!**

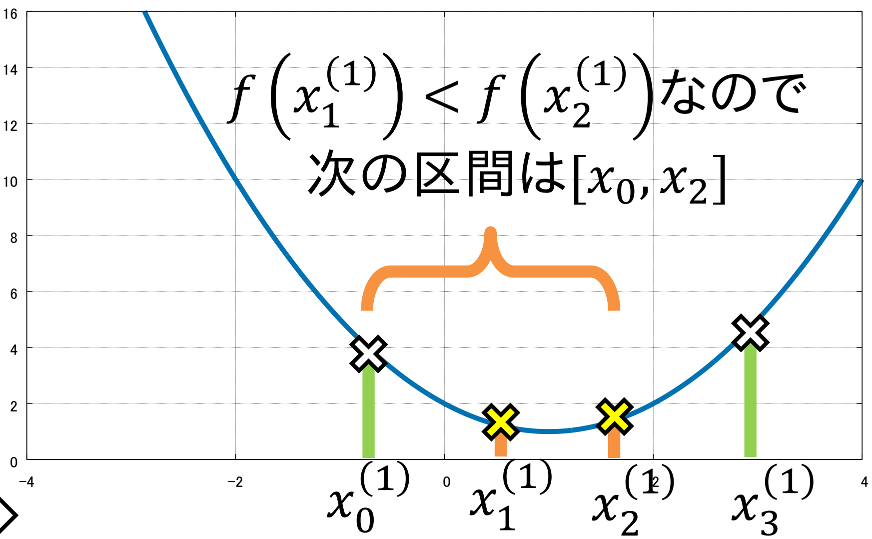
(証明終了)

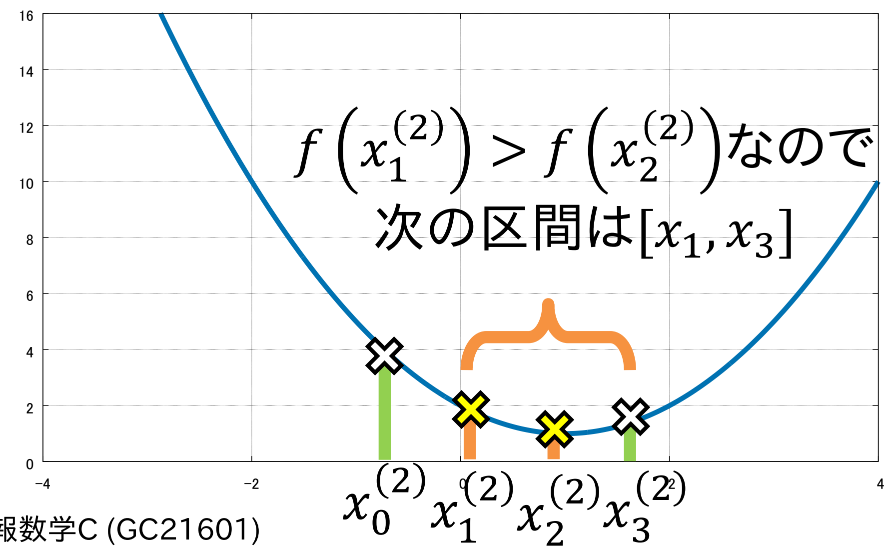


3分探索

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**

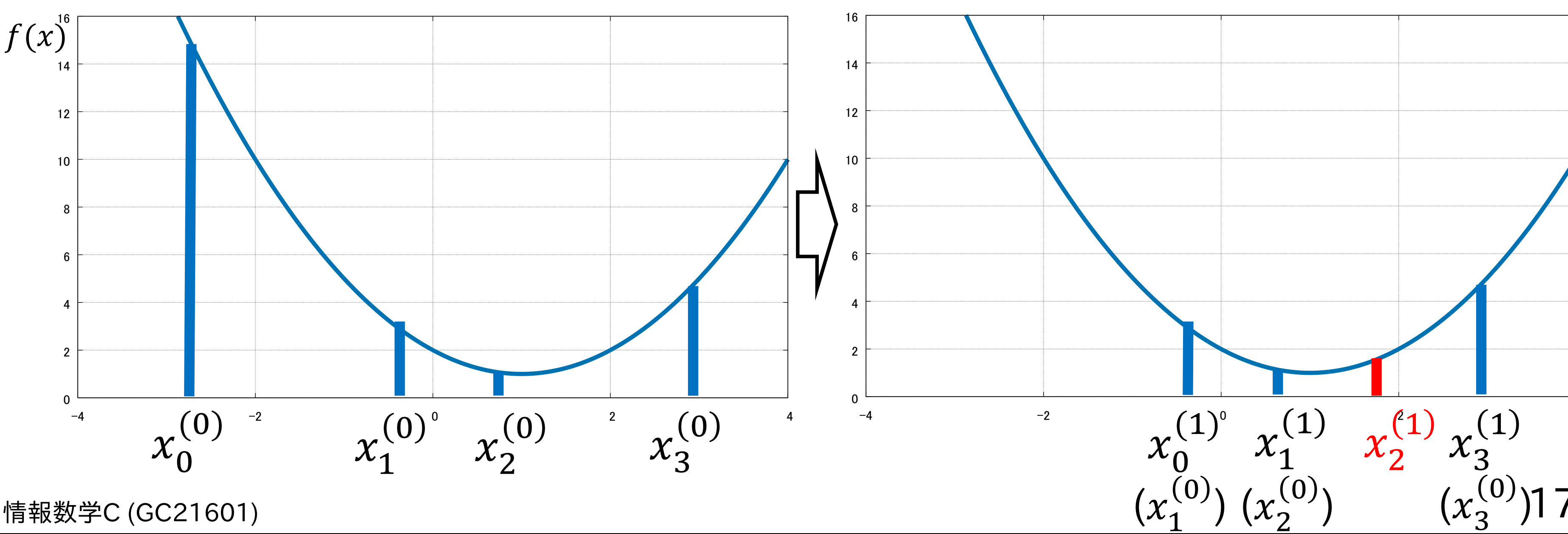
黄金分割法

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?

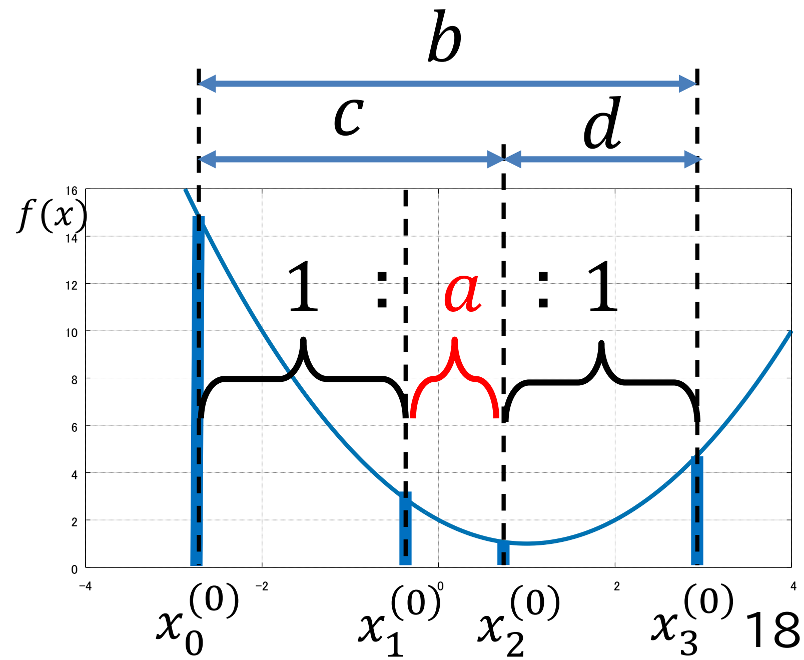
⇒ 分割比が極端(1:10とか)になると収束が不安定になる

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:

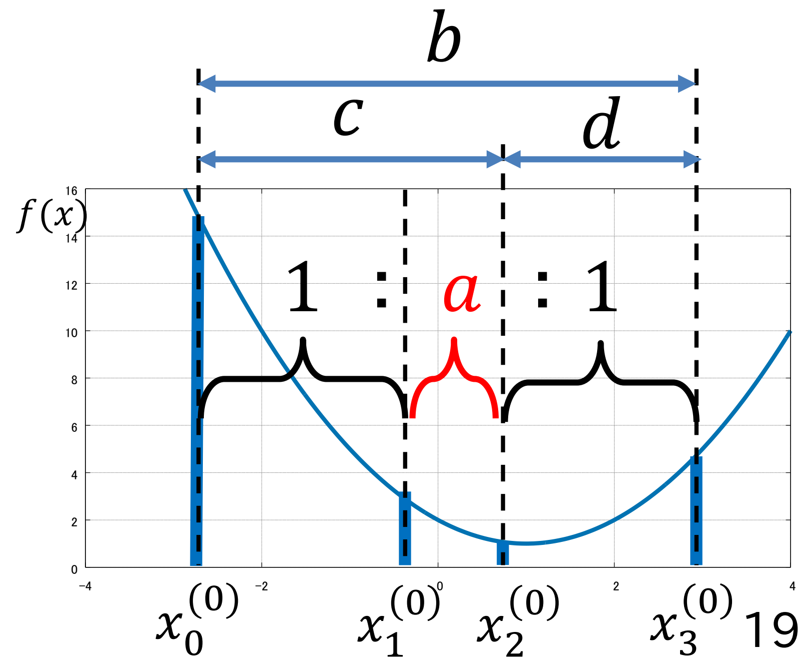
$$a = \frac{3-\sqrt{5}}{\sqrt{5}-1}$$

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. $f(x, y, z)$)

⇩

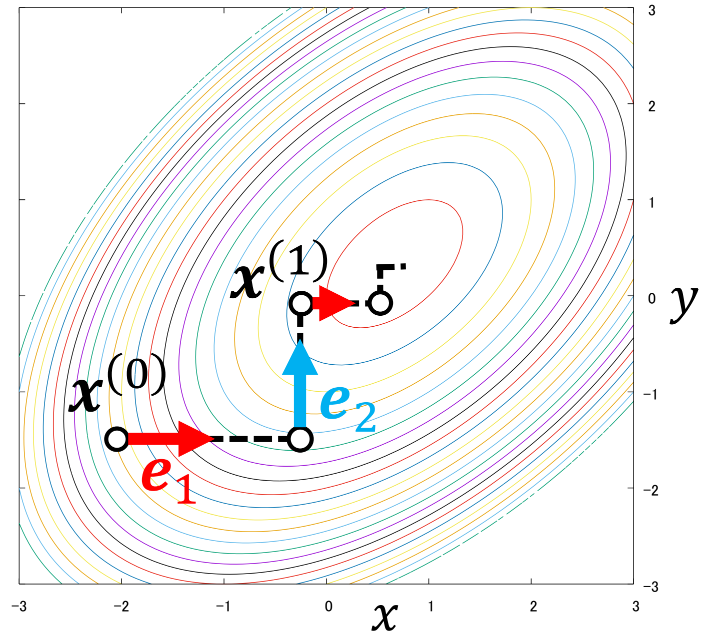
- 多次元空間内でも**ある直線方向**に限れば, **1次元探索で最小値を見つけられる**
- ある方向で最小値を見つける → その点から別の方向に探索して最小値を見つける
  を繰り返せば多次元でも最小値探索できそう

Powellの方法

Powellの方法

Powellの方法の手順(2次元の場合)

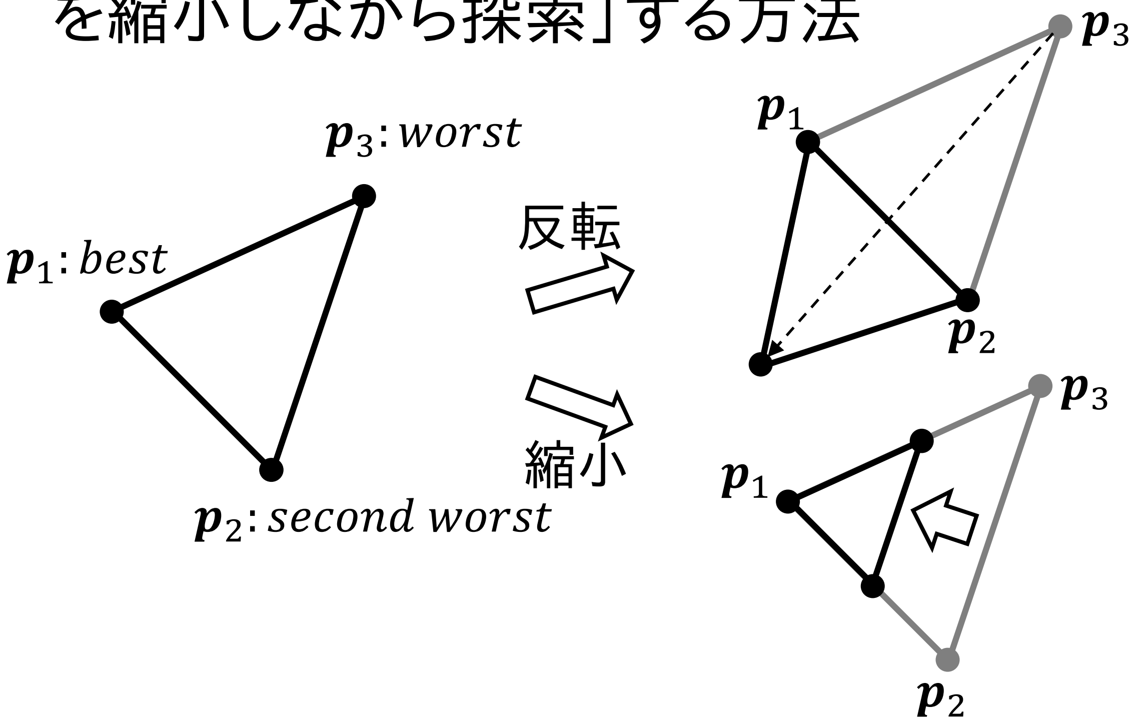
1. 初期位置$\boldsymbol{x}^{(0)}$を設定
2. 以下の処理を繰り返し($k = 0,1, \dots$)
   a. x軸方向に黄金探索法などで探索
   (関数値は軸方向$\boldsymbol{e}_1 = (1, 0)$と探索点までの距離$t$を使って$f(\boldsymbol{x}^{(k)} + t\boldsymbol{e}_1)$として計算できる)
   b. 手順aでの最小解を始点としてy軸方向($\boldsymbol{e}_2 = (0,1)$)に探索.
   探索結果を$\boldsymbol{x}^{(k+1)}$とする
   c. $t < \varepsilon$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

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

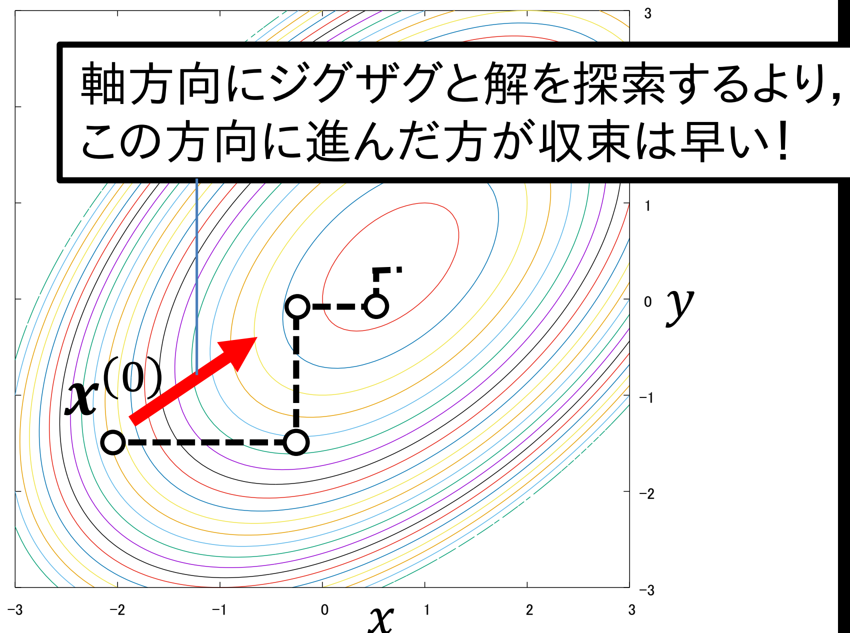
関数の微分(勾配)を用いる**勾配法**の中でも
最も単純な方法

- 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

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

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$(0, 0)$, $\alpha = 0.5$

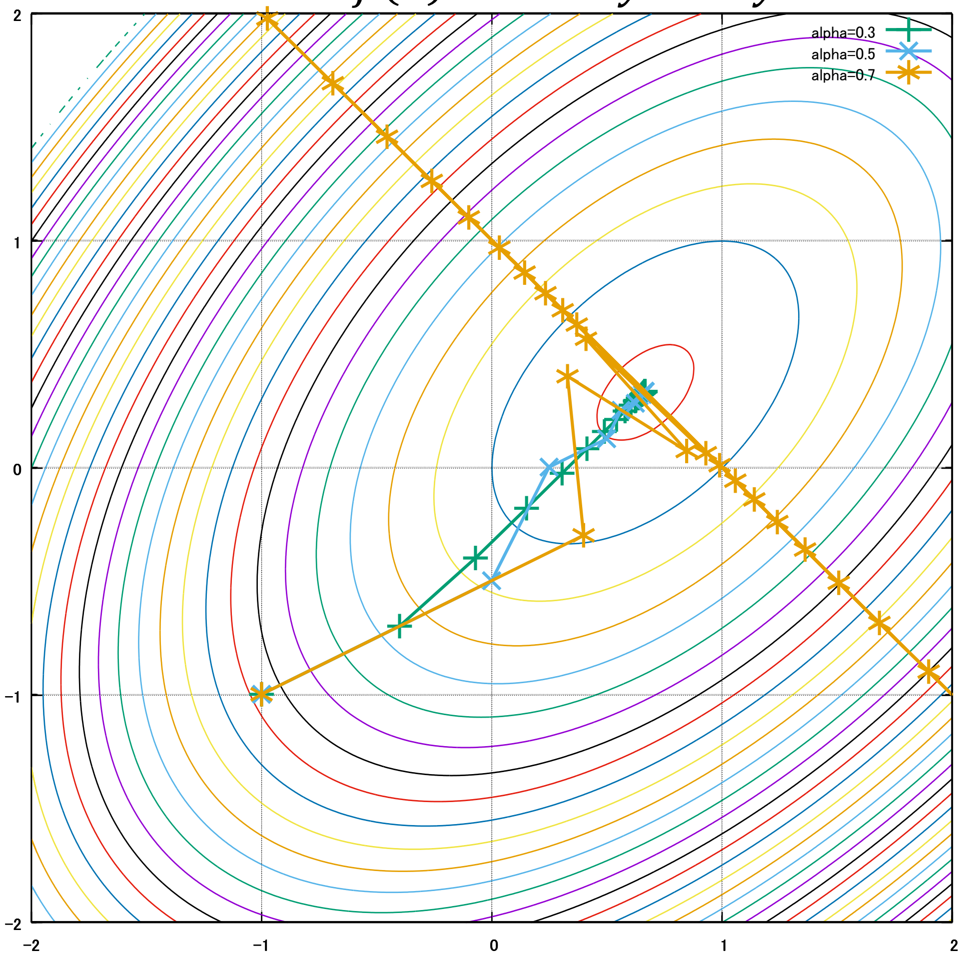
```
0 : 0,        0,        e = 1
1 : 0.5,      0,        e = 0.5
2 : 0.5,      0.25,     e = 0.25
3 : 0.625,    0.25,     e = 0.125
4 : 0.625,    0.3125,   e = 0.0625
5 : 0.65625, 0.3125,    e = 0.03125
6 : 0.65625, 0.328125, e = 0.015625
      ⋮ (省略)
15 : 0.666656, 0.333313, e = 3.05176e-05
16 : 0.666656, 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

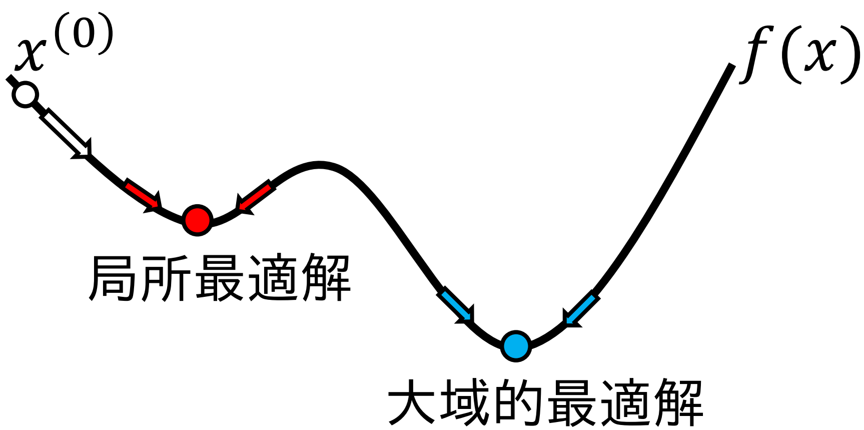
- **指数関数**的に減衰させたり, 反復回数の**逆数**を使う
- **焼きなまし法** : 徐々に小さくするとともに, **局所最適解**の回避のためにαが大きいときにたまに(確率的に)逆方向に探索する

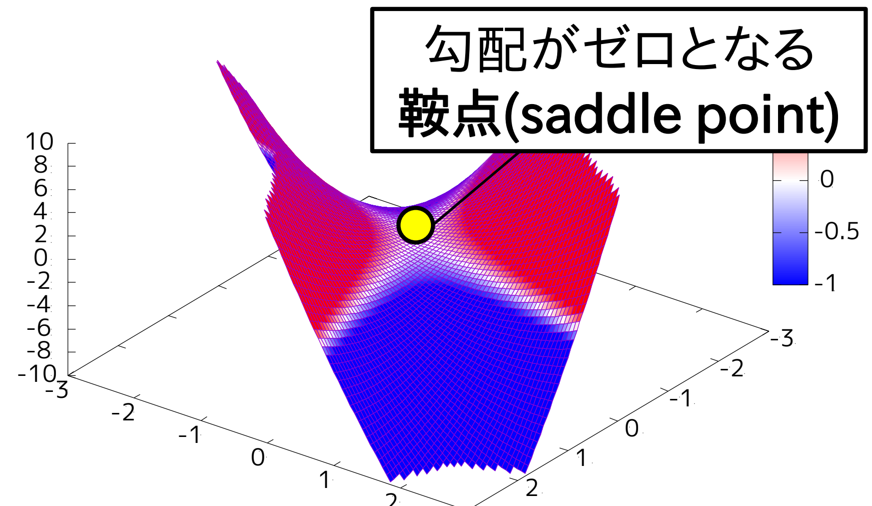
*「焼きなまし」とは金属を焼き入れして硬くした後にもう一度熱した後徐々に冷やすことで欠陥を少なくする作業

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)

$\Rightarrow$ $\nabla^2 f(x)$ 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$(0, 0)$

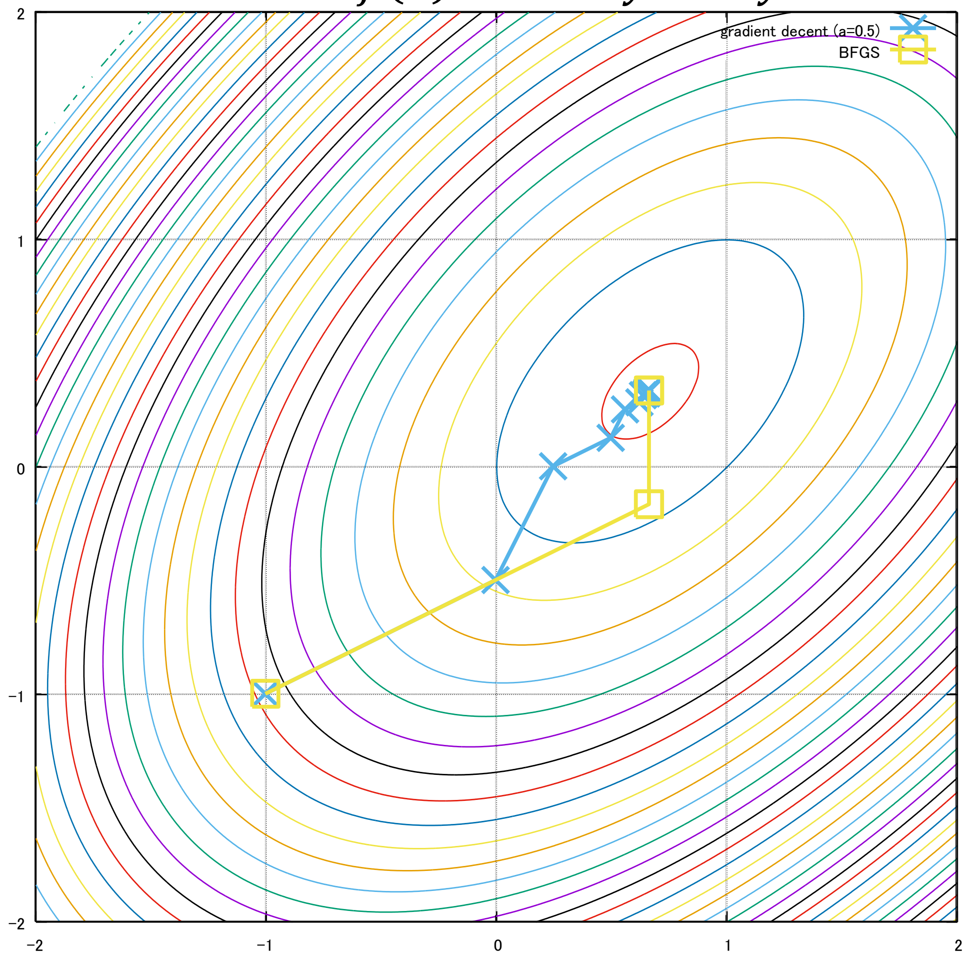
```
f(0,        0)        = -1
f(0.499948, 0)        = -1.25
f(0.666636, 0.333307) = -1.33333
f(0.666667, 0.333333) = -1.33333
(x,y) = (0.666667,0.333333),  f = -1.33333
```

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

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
- **制約付き最小化問題**

今回の講義で解く問題 その2

$$\operatorname*{arg\,min}_{x} f(x)$$

$$\text{subject to } g_i(x) = 0$$

and/or

$$h_j(x) \leq 0$$

$$(i = 1, \dots, m)$$

$$(j = 1, \dots, l)$$

今回の講義で解く問題 その2

$$\arg\min_{x} f(x)$$:**目的関数**(最小化したい式)

$$\text{s.t.}\quad g_i(x) = 0$$:**等式制約**

and/or

$$h_j(x) \leq 0$$:**不等式制約**

$$(i = 1, \dots, m)\ (j = 1, \dots, l)$$

$\boxed{\text{制約付き最小化問題}}$

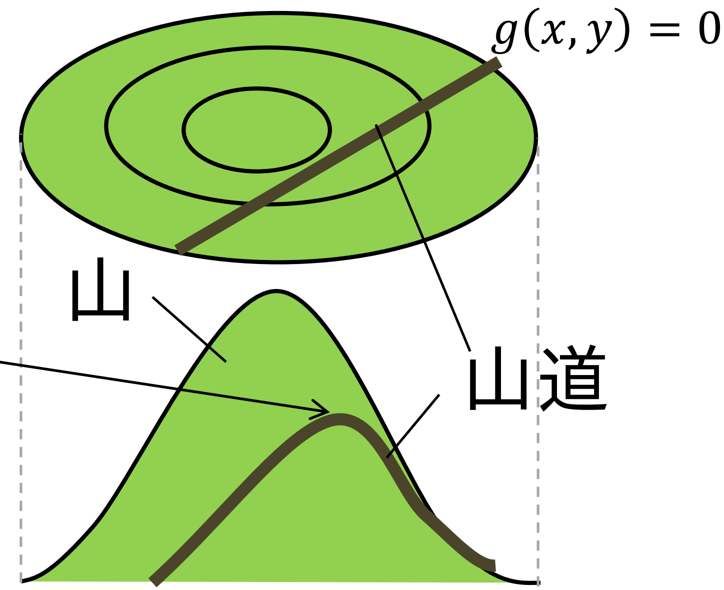
制約付き最小化問題の具体例

具体的にはどんな問題がこれに当てはまる？

例) 山の高さが$f(x,y)$で表されるとき,
この山にかかっている山道の軌道がxy
平面上で$g(x,y)=0$と表される.
このとき, **<u>山道で一番高い箇所</u>**を求めよ.

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

$$\text{s.t.}\quad g(x,y)=0$$



他にも, 気温分布からある範囲における最小値を求める,
決められた予算内で効果が最大となる製品の生産量
は？などなど,
色々な問題が制約付き最適化問題として表される.

ラグランジュの未定乗数法を用いた解き方

制約付き最適化問題は**微分積分**の授業(微分積分B,2年次春AB,**必修**)
ですでに解き方は学んでいる ⇒ **ラグランジュの未定乗数法**

復習

$$\arg\min_{\boldsymbol{x}} f(\boldsymbol{x}) \qquad \text{s.t.} \quad g_i(\boldsymbol{x}) = 0$$

(目的関数) (制約条件)

ラグランジュ関数: $L(\boldsymbol{x}, \boldsymbol{\lambda}) = f(\boldsymbol{x}) + \sum_{i=1}^{m} \lambda_i g_i(\boldsymbol{x})$

最適解を$\boldsymbol{x}^*$とすると以下の式を満たす$\boldsymbol{\lambda} = (\lambda_1, \dots, \lambda_m)^T$
が存在する(必要条件*)

$$\nabla L(\boldsymbol{x}^*, \boldsymbol{\lambda}) = \nabla f(\boldsymbol{x}^*) + \sum_{i=1}^{m} \lambda_i \nabla g_i(\boldsymbol{x}^*) = 0$$

$\boldsymbol{\lambda}$: ラグランジュの未定乗数
(Lagrange multiplier)

*f, gが凸関数ならば十分条件

ラグランジュの未定乗数法を用いた解き方

ラグランジュの未定乗数法

等式条件が1つのみ($m = 1$)ならば

$$\boxed{\nabla_{\boldsymbol{x}} L(\boldsymbol{x}^*, \boldsymbol{\lambda}) = 0} \rightarrow \nabla f(\boldsymbol{x}^*) + \lambda \nabla g(\boldsymbol{x}^*) = \boldsymbol{0}$$

$$+$$

$$\boxed{\nabla_{\boldsymbol{\lambda}} L(\boldsymbol{x}^*, \boldsymbol{\lambda}) = 0} \rightarrow g(\boldsymbol{x}^*) = 0$$

← $-g(\boldsymbol{x}) = 0$として，$\nabla f - \lambda \nabla g = 0$とする場合もある

$\boldsymbol{x} = (x_1, \ldots, x_n)^T$とする($f\colon \mathbb{R}^n \rightarrow \mathbb{R}$)と, 式の数が$n+1$個, 未知数($\boldsymbol{x}$と$\lambda$)が$n+1$個

⇒ すべて1次式になる場合：**線形システム**を解けば良い(第2,3回の講義参照)

⇒ 非線形式が含まれる：**多次元のニュートン法など**で解けば良い(第4回の講義参照)

ラグランジュの未定乗数法を用いた解き方

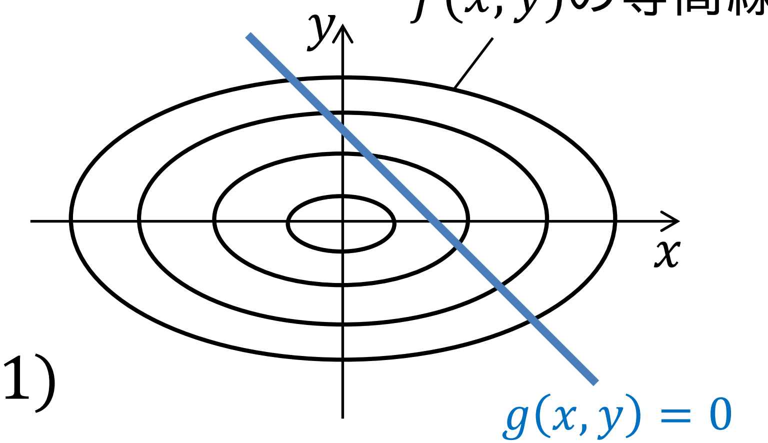
ラグランジュの未定乗数法を用いた解法の例

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

$$s.t.\ g(x,y) = x + y - 1 = 0$$

⇩ ラグランジュ関数

$$L(x,y,\lambda) = x^2 + 2y^2 + \lambda(x + y - 1)$$



最適解は以下の式を解けば求められる

$$-\nabla f(x,y) = \lambda \nabla g(x,y)$$

$$x + y = 1$$

$\nabla f(x,y) = (2x, 4y)^T, \nabla g(x,y) = (1,1)^T$となるので,

$$\begin{aligned} 2x + \lambda &= 0 \\ 4y + \lambda &= 0 \\ x + y &= 1 \end{aligned} \quad \Rightarrow \quad x = \frac{2}{3}, y = \frac{1}{3}, \lambda = -\frac{4}{3}$$

(線形システム)

解)

$(x,y) = \left(\frac{2}{3}, \frac{1}{3}\right)$で

最小値$f(x,y) = \frac{2}{3}$

等式制約+不等式制約

ラグランジュの未定乗数法は

等式制約($g(\boldsymbol{x})=0$)付きの最適化問題

⇩

ラグランジュ関数Lの停留点*を求める問題

と**変換**することで解く方法

- **制約条件が2つ以上**の場合は?

$$\begin{cases} \nabla f(\boldsymbol{x}) + \lambda_i \nabla g_i(\boldsymbol{x}) = \boldsymbol{0} \\ g_i(\boldsymbol{x}) = 0 \end{cases} \quad (i = 1 \sim m)$$

⇒ 条件の数$m>1$でも式の数と未知数は$n+m$個で一致しているので
数値計算法で停留点を求めれば良い

- **不等式条件**が入る場合は? ⇨ KKT条件

*停留点:微分が0となる，つまり関数が変化しなくなる点(極小点,極大点,鞍点)

KKT条件

不等式制約を含む最適化問題

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

$$\text{s.t.} \quad g_i(\boldsymbol{x}) = 0 \qquad (i = 1 \sim m)$$

$$h_j(\boldsymbol{x}) \leq 0 \qquad (j = 1 \sim l)$$

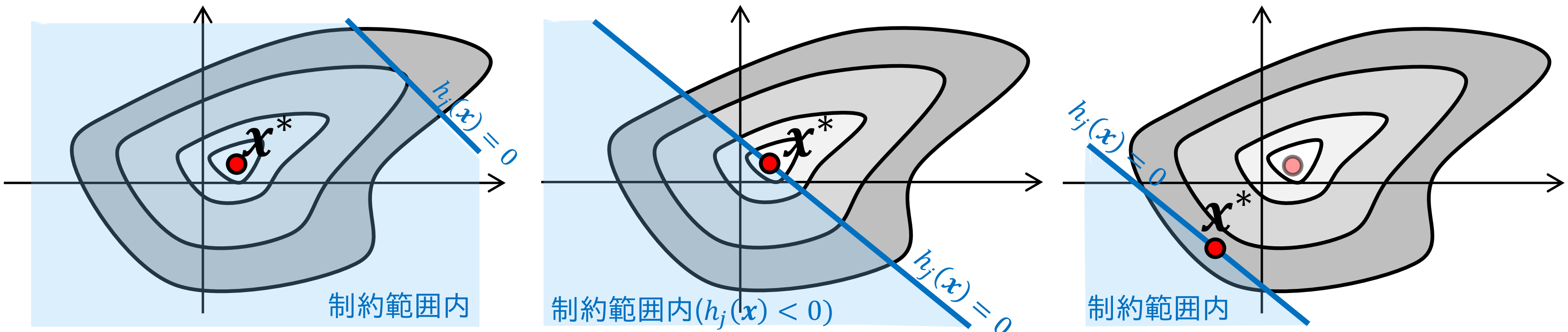
⇩ 不等式条件を含めたラグランジュ関数

$$L(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}) = f(\boldsymbol{x}) + \sum_{i=1}^{m} \lambda_i g_i(\boldsymbol{x}) + \sum_{j=1}^{l} \mu_j h_j(\boldsymbol{x})$$

$\nabla L(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}) = 0$ に条件が**「効いているか？効いていないか？」**
に関する式を追加する

相補性条件

目的関数に対してその不等式条件は意味がある？



不等式制約は
最適解$\boldsymbol{x}^*$に**影響していない**

この制約式は点$\boldsymbol{x}^*$で
効いていない(inactive)

$$h_j(\boldsymbol{x}^*) \neq 0$$

弱く効いている
(weakly active)

不等式制約は明確に
最適解$\boldsymbol{x}^*$に**影響している**

この制約式は点$\boldsymbol{x}^*$で
効いている(active)

$h_j(\boldsymbol{x}^*) = 0$が成り立つ

効いてないときは$\mu_j = 0$としたい $\Rightarrow \mu_j h_j(\boldsymbol{x}^*) = 0$

相補性条件(complementarity condition)

KKT条件

Karush-Kuhn-Tucker(KKT)条件

$$
\begin{cases}
\nabla_x L(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}) = \nabla f(\boldsymbol{x}) + \sum_{i=1}^{m} \lambda_i \nabla g_i(\boldsymbol{x}) + \sum_{j=1}^{l} \mu_j \nabla h_j(\boldsymbol{x}) = \boldsymbol{0} \\
\nabla_\lambda L(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}) = g_i(\boldsymbol{x}) = 0 \quad (i = 1 \sim m) \\
\nabla_\mu L(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}) = h_j(\boldsymbol{x}) \leq 0 \quad (j = 1 \sim l) \\
\mu_j \geq 0 \quad (j = 1 \sim l) \\
\mu_j h_j(\boldsymbol{x}) = 0 \quad (j = 1 \sim l)
\end{cases}
$$

ラグランジュ関数の停滞点条件

相補性条件

($l = 0$でラグランジュの未定乗数法と同じ)

μ_jを**KKT乗数**, KKT条件を満たす点$(\boldsymbol{x}^*, \boldsymbol{\lambda}^*, \boldsymbol{\mu}^*)$を**KKT点**と呼ぶ

元の条件付き最適化問題の解は**KKT条件を満たす**

双対問題と凸性

ラグランジュ未定乗数法, KKT条件ともに**必要条件**

(KKT条件の解 ∊ 最適化問題の解)

⇒ イコールではない(最適化問題の解すべてがKKT条件で求められるわけではない)

最適化関数が凸性を持つときは**十分条件**となり,
イコールと考えられる

(凸性を持つなら広域最適解1つだけを持つということになるので)

元の最適化問題を**主問題**とすると, KKT条件などは**補問題**となり, どちらか一方の解が両方の問題の解となる場合,
補問題は**双対問題**と呼ばれる.

KKT条件

Karush-Kuhn-Tucker(KKT)条件

$$
\begin{cases}
\nabla_x L(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}) = \nabla f(\boldsymbol{x}) + \sum_{i=1}^{m} \lambda_i \nabla g_i(\boldsymbol{x}) + \sum_{j=1}^{l} \mu_j \nabla h_j(\boldsymbol{x}) = \boldsymbol{0} & \\
\nabla_\lambda L(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}) = g_i(\boldsymbol{x}) = 0 & (i = 1 \sim m) \\
\nabla_\mu L(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}) = h_j(\boldsymbol{x}) \leq 0 & (j = 1 \sim l) \\
\mu_j \geq 0 & (j = 1 \sim l) \\
\mu_j h_j(\boldsymbol{x}) = 0 & (j = 1 \sim l)
\end{cases}
$$

ラグランジュ関数の停滞点条件

相補性条件

($l = 0$でラグランジュの未定乗数法と同じ)

不等式がまだ含まれていないか?**コンピュータでどうやって計算しよう…**

⇒ 基本的には**数学的に解く方法**(線形計画問題だとシンプレックス法などもあるが)

数値計算を使った条件付き最適化問題の解き方

ここまでの話は**条件付き最適化問題**をどちらかというと**数学的に解く方法**(問題をKKT乗数などを使って置き換えた後に, 数値計算法で解く場合もあるけど).

- 元の問題の最適化式や条件で**解き方が変わってしまう(アルゴリズム化が難しい).**
- 条件が境界を表していたときにその**境界形状が複雑**だと解くのが大変になる(式で表すのが難しい)

数値解法

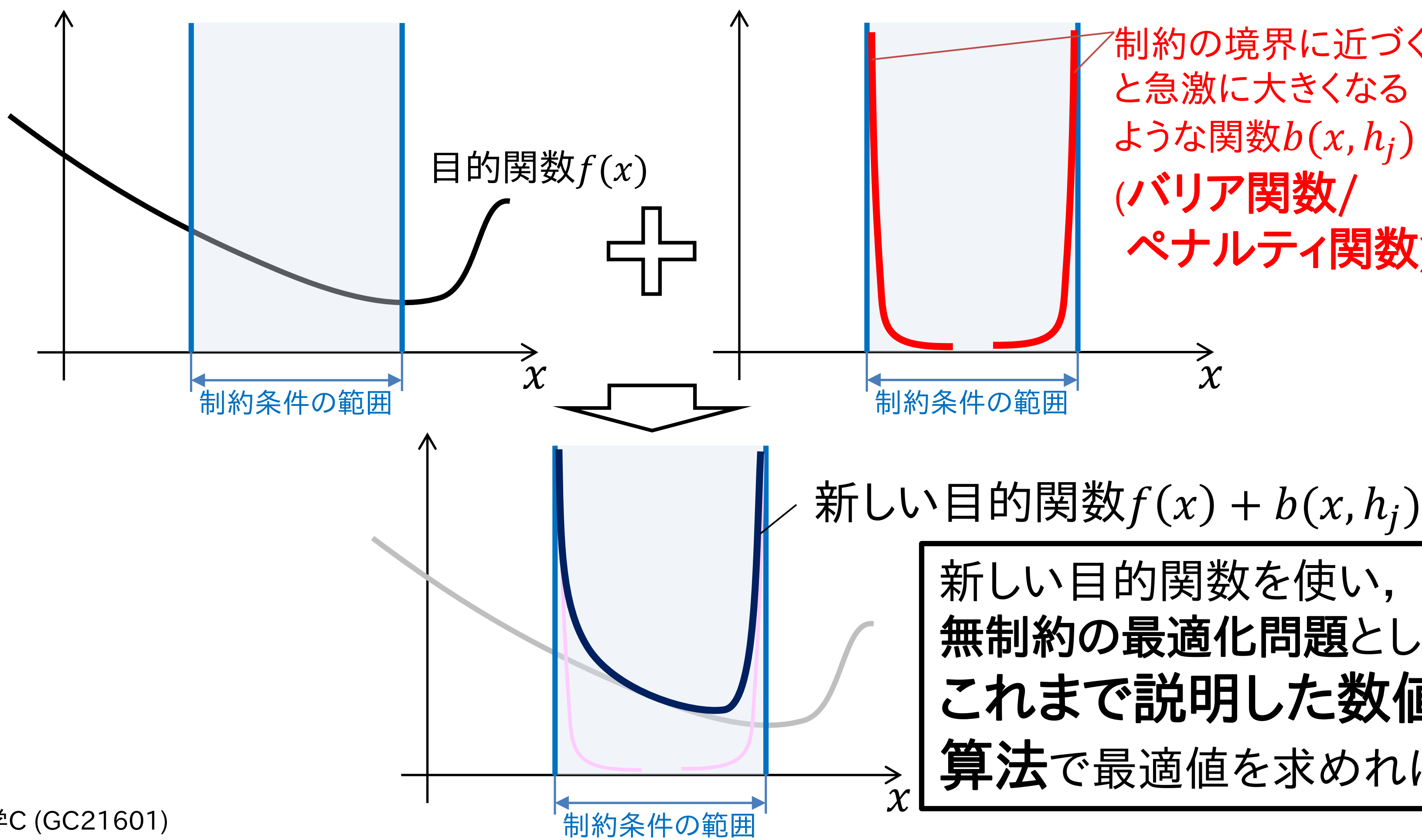
↓

ペナルティ関数法, バリア関数法

数値計算を使った条件付き最適化問題の解き方

ペナルティ関数法/バリア関数法

　ペナルティ関数/バリア関数を使って「制約付きの問題を制約なし問題に変換」して解くという方法

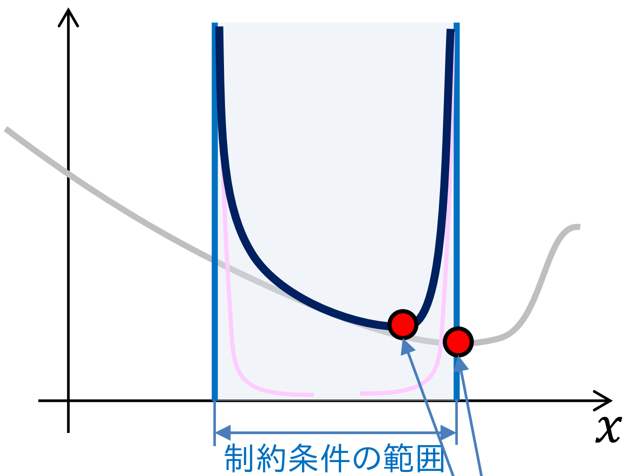


新しい目的関数を使い，
無制約の最適化問題として
これまで説明した数値計算法で最適値を求めれば良い

ペナルティ関数法/バリア関数法

内点ペナルティ関数法(バリア関数法)

領域の境界に近づくと無限大に発散するような**バリア関数**(barrier function)$b(x, h_j)$を目的関数に加える方法



$$\begin{cases} \arg\min_{\boldsymbol{x}} f(\boldsymbol{x}) \\ \text{s.t.} \quad h_j(\boldsymbol{x}) \le 0 \end{cases} \Rightarrow \arg\min_{\boldsymbol{x}} f(\boldsymbol{x}) + b(\boldsymbol{x}, h_j)$$

これだと最適解が一致しなさそう
⇒ $\mu \to 0$となるまで反復処理する

バリア関数の例

$$b(\boldsymbol{x}, h_j) = \mu \sum_{j=1}^{l} \frac{1}{h_j(\boldsymbol{x})^2},$$

μ：バリアパラメータ

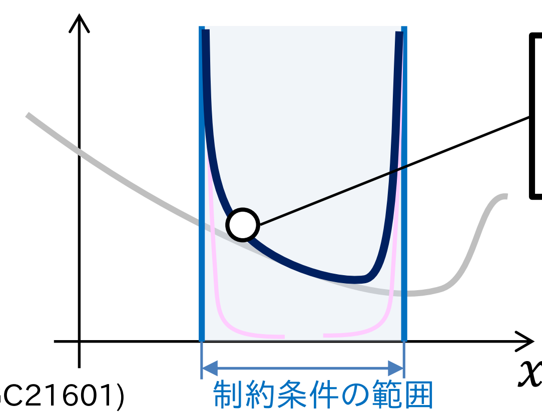
$$b(\boldsymbol{x}, h_j) = -\mu \sum_{j=1}^{l} \log\left(-h_j(\boldsymbol{x})\right)$$

ログバリア関数(log-barrier function)

ペナルティ関数法/バリア関数法

内点ペナルティ関数法(バリア関数法)のアルゴリズム
(これまで説明した数値計算法を使うだけになるのでコード例はなし)

1. 制約範囲内の**内点**を初期点$\boldsymbol{x}^{(0)}$として選択し,
   バリアパラメータの初期値$\mu^{(0)} > 0$を設定
2. $k = 1 \sim k_{max}$まで以下を反復処理
   a. $f(x) + b(x, h)$を目的関数として最小解$\boldsymbol{x}^{(k)}$を求める
      (これまで説明した数値計算法を使う.初期点として$\boldsymbol{x}^{(k-1)}$が使える)
   b. $\mu^{(k)} < \varepsilon$ ならば$\boldsymbol{x}^{(k)}$を解として反復終了
   c. $0 < \mu^{(k+1)} < \mu^{(k)}$となるような$\mu^{(k+1)}$を計算

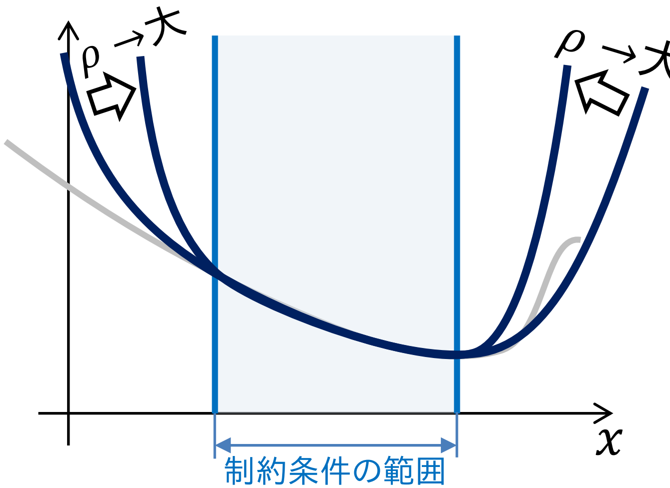


バリア関数の形から初期値として**内点**を選ぶ必要があることに注意

ペナルティ関数法/バリア関数法

外点ペナルティ関数法

領域内では目的関数は$f(x)$と同じ, **領域外では境界から離れると無限大に発散**するように目的関数を設定する方法



$$
\begin{cases}
\arg\min_{\boldsymbol{x}} f(\boldsymbol{x}) \\
\text{s.t.} \quad g_i(\boldsymbol{x}) = 0, h_j(\boldsymbol{x}) \leq 0
\end{cases}
$$

⇩

$$
\arg\min_{\boldsymbol{x}} f(\boldsymbol{x}) + \rho\left(\sum_{i=1}^{m} |g_i(\boldsymbol{x})|^{\alpha} + \sum_{j=1}^{l} \left(\max\left(0, h_j(\boldsymbol{x})\right)\right)^{\beta}\right)
$$

$\alpha, \beta \geq 1$：ユーザ設定パラメータ，ρ：ペナルティパラメータ

$\rho \to \infty$となるように反復処理. 初期値は内点でなくても良い

主双対内点法

バリア関数法(内点法)の問題

バリア関数付近で不安定になりやすく, バリアパラメータμと最適解$x^{(k)}$の両者の最適化が必要(2重の反復計算になる)

主双対内点法

バリア関数法+双対問題を解くKKT条件

⇒ KKT条件の不等式はどうする? ⇒ **スラック変数**を導入する

$$\arg\min_{\boldsymbol{x}} f(\boldsymbol{x}) - \underbrace{\mu \sum_{j=1}^{l} \log(s_j)}_{\text{バリア関数}} \qquad \mathrm{s.t.} \quad g_i = 0, h_j(\boldsymbol{x}) + \underbrace{s_j}_{\text{スラック変数}} = 0$$

バリア関数は$\mu > 0$でs_iが0に近づくほど無限に発散.
この式の停留点方向に移動しながら, 徐々にμを0に近づけていくということを反復することで元の問題の解を見つける.

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
- 条件付き最適化問題
  - ラグランジュの未定乗数法, KKT条件
  - ペナルティ関数, バリア関数

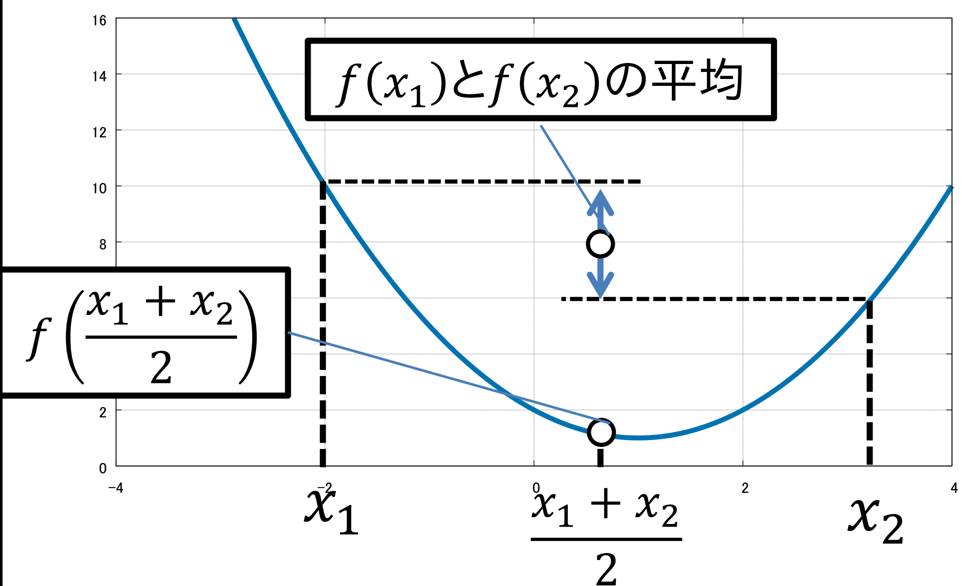
Appendix

(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.

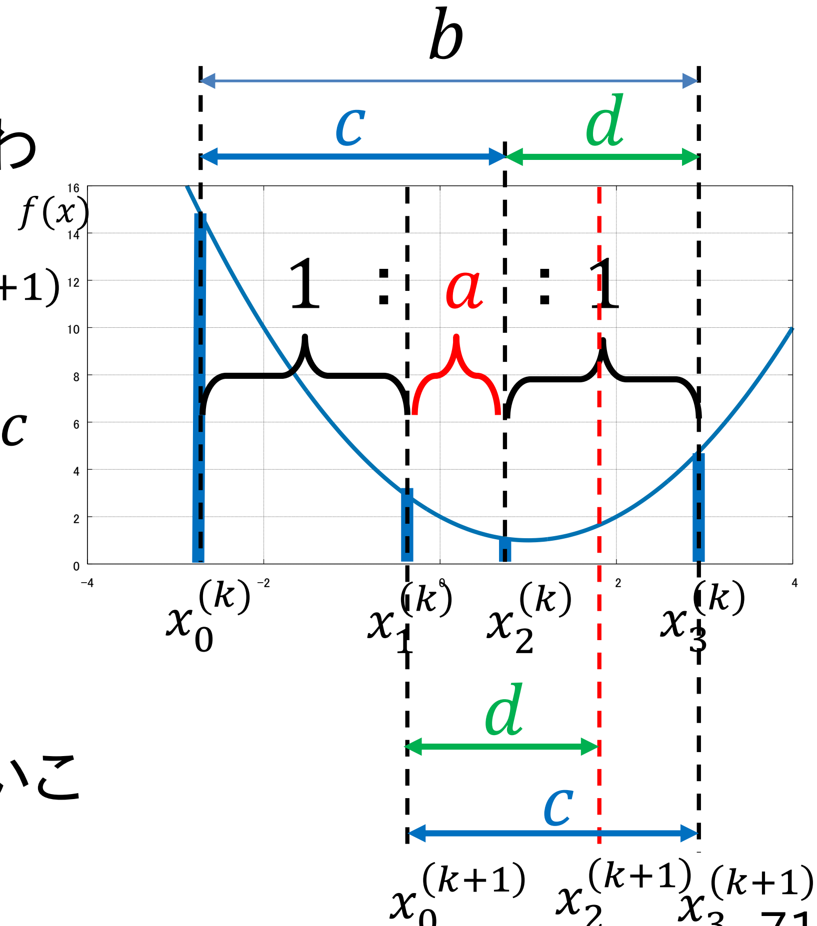
1回分割後($k \to k+1$)に比率が変わらないならば,

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

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:

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

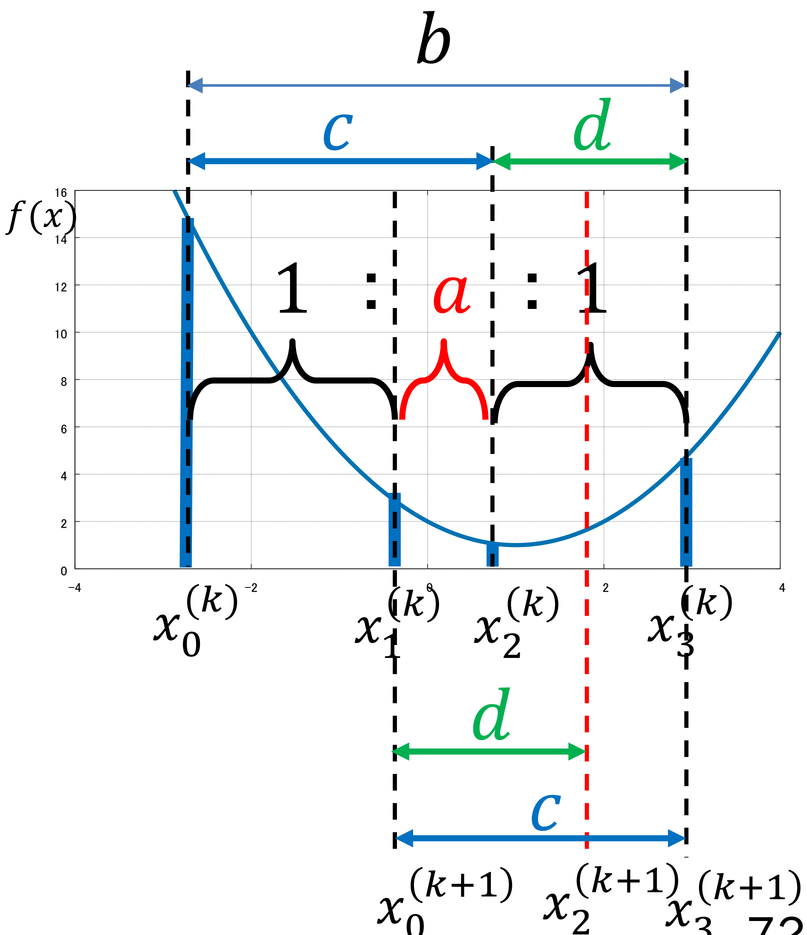
解の公式に当てはめると:

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

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

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