数値解析

12　数理計画法
（黄金分割法と最急降下法）

2008 年 6 月 30 日

環境都市工学部 都市システム工学科
山川 栄樹

数理計画問題

- 定義： 与えられた条件を満たす変数の組のなかで，
  ある関数の値を最小にするものを求める問題
- 例： 都市計画（道路網の整備計画立案）
  - ＊変数：各道路の拡幅量
  - ＊条件：利用者の目的地までの到達時間をある値以下
  - ＊関数：道路の拡幅にかかる費用の総額
- 数学的表現：

目的関数： $f(x_1, \ldots, x_n) \rightarrow$ 最小

制約条件： $g_1(x_1, \ldots, x_n) \geqq 0$

$$\vdots$$

$$g_m(x_1, \ldots, x_n) \geqq 0$$

（注意）「最小」は「最大」でもよい．$\geqq$ は $\leqq$ でもよい．

１変数関数の最小化（制約条件なし）

- 問題： 目的関数：$f(x) \rightarrow$ 最小

＊前提条件：関数 f は連続／微分可能とは限らない

区間 $[a, b]$ 内に唯一の最小点をもつ.

- 考え方：非線形方程式に対する

2分法の変形

＊区間 $[a, b]$ 内に点 $p \leqq q$ をとる.

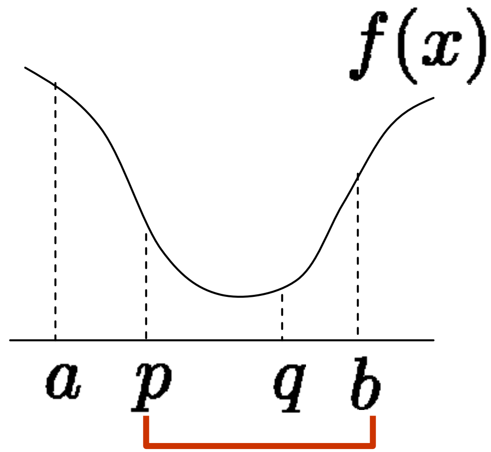
＊ $f(p) \geqq f(q)$ であるとき,

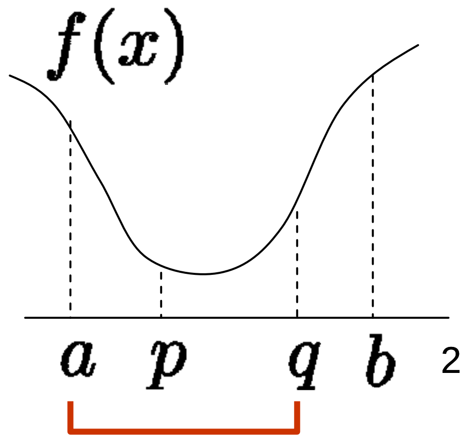
区間を $[p, b]$ に縮小する.

＊ $f(p) < f(q)$ であるとき,

区間を $[a, q]$ に縮小する.

＊区間幅が十分小さくなれば終了.

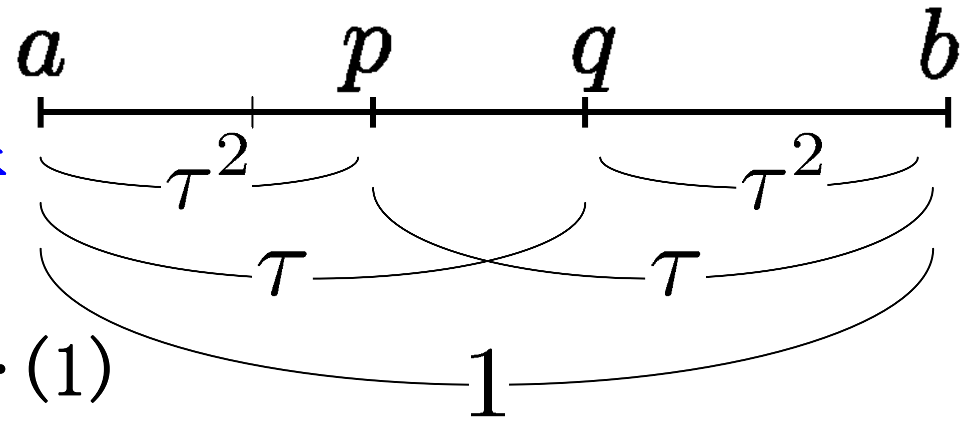




黄金分割

- 分割の方法：

  ＊区間幅が毎回 τ (<1) 倍に縮小するようにする.



$$(q-a)=\tau(b-a) \quad \cdots(1)$$

$$(b-p)=\tau(b-a) \quad \cdots(2)$$

さらに

$$(p-a)=\tau(q-a) \quad \cdots(3)$$

$$(b-q)=\tau(b-p) \quad \cdots(4)$$

(1), (3)より

$$(p-a)=\tau^2(b-a) \quad \cdots(5)$$

(2) + (5)

$$(b-a)=(\tau^2+\tau)(b-a)$$

すなわち

$$\tau^2+\tau-1=0$$

$\tau>0$ より

$$\tau=(-1+\sqrt{5})/2$$

$$\approx 0.618$$

黄金分割比

黄金分割法

- アルゴリズム：

ｽﾃｯﾌﾟ1: $a<b$ なる初期点 a, b を選ぶ.
$\tau=(\sqrt{5}-1)/2$, $\varepsilon>0$ (十分小)を定める.

ｽﾃｯﾌﾟ2: $p=b-\tau(b-a)$, $q=a+\tau(b-a)$ とおき,
$f(p)$, $f(q)$ を計算する.

$b-a\geqq\varepsilon$ の間繰返し

ｽﾃｯﾌﾟ3: $b-a<\varepsilon$ ならば終了する. (解は $x^*=(a+b)/2$)

ｽﾃｯﾌﾟ4: $f(p)\geqq f(q)\Rightarrow a=p,\ p=q,\ q=a+\tau(b-a)$

$f(p)$: 直前の $f(q)$ の値

$f(q)$ を計算する.

$f(p)<f(q)\Rightarrow b=q,\ q=p,\ p=b-\tau(b-a)$

$f(q)$: 直前の $f(p)$ の値

$f(p)$ を計算する.

ｽﾃｯﾌﾟ3へ戻る.

☆繰返し実行する計算 → 変数に格納, MATLABの関数に

黄金分割法のプログラム

GoldenSection.m (1/2)

```
function x = GoldenSection(a, b)
tau = (sqrt(5) - 1) / 2;
eps = 0.000001;
w   = b - a;
p   = b - tau * w;
q   = a + tau * w;
fp  = fval(p);
fq  = fval(q);
disp([a p q b fp fq]);
while(w  >= eps)
  if (fp >= fq)
    a  = p;
    w  = b - a;
    p  = q;
    fp = fq;
    q  = a + tau * w;
    fq = fval(q);
```

初期点の表示

(2/2)

```
  else
    b  = q;
    w  = b - a;
    q  = p;
    fq = fp;
    p  = b - tau * w;
    fp = fval(p);
  end
  disp([a p q b fp fq]);
end
x = (a + b) / 2;
```

探索経過の表示

黄金分割法

（練習）　目的関数：$\boldsymbol{f(x) = e^{x} - \log x}$ → 最小

の $(0, 1]$ にある解を，黄金分割法で求めよう．

- o $f(x) = e^x - \log x$ の値を計算する MATLAB の関数(**fval.m**)を作って動作を確かめよう．

```
function z = fval(x)
z = exp(x) - log(x);
```

- o 初期区間を **[0, 1]** として実行しよう．

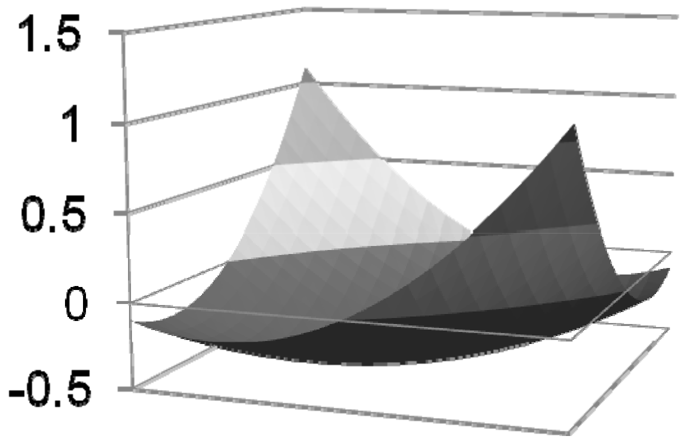
```
>> GoldenSection(0, 1)
```

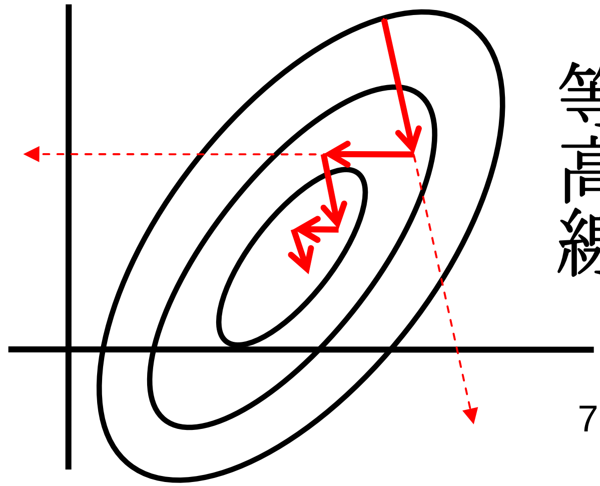
多変数関数の最小化（制約条件なし）

- 問題： 目的関数：$f(x_1, \ldots, x_n) \to$ 最小
  ＊前提条件：関数 f は１回連続的微分可能
- 考え方：
  ＊つぎの２操作を繰返して解に収束する点列を生成

  (1) 探索方向の決定
  どの方向に進むか？
  （ f の値がよく減る方向）

  (2) ステップ幅の決定
  どれくらい進むか？
  （行き過ぎると値は増える）





最急降下法

- 探索方向： 関数値が最もよく減少する方向

  ＊最急降下方向： $\boldsymbol{d}^{(k)} = -\nabla f(x_1^{(k)}, \ldots, x_n^{(k)})$

  o 関数 f の等高線に直交する方向

  ＊ただし，∇f は関数 f の勾配ベクトル

$$\nabla f(x_1, \ldots, x_n) = \begin{pmatrix} \frac{\partial f(x_1,\ldots,x_n)}{\partial x_1} \\ \vdots \\ \frac{\partial f(x_1,\ldots,x_n)}{\partial x_n} \end{pmatrix}$$

- ステップ幅：1変数関数の最小化問題を解いて決定

  目的関数： $\phi^{(k)}(\alpha) = f(\boldsymbol{x}^{(k)} + \alpha \boldsymbol{d}^{(k)}) \ \rightarrow$ 最小

- 点列の生成： $\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} + \alpha \boldsymbol{d}^{(k)}$

黄金分割法によるステップ幅の決定

- 1変数関数の最小化　　直線探索 (line search) という

　目的関数：$\phi^{(k)}(\alpha) = f(\boldsymbol{x}^{(k)} + \alpha \boldsymbol{d}^{(k)}) \to$ 最小

の $\alpha \in [0, 1]$ における解を求める.

(練習) $f(x_1, x_2) = 2x_1^2 - x_1 x_2 + x_2^2 - 2x_1 - 3x_2$ とする.

$\boldsymbol{x}^{(0)} = (0, 0)^\top$, $\boldsymbol{d}^{(0)} = -\nabla f(\boldsymbol{x}^{(0)}) = (2, 3)^\top$ のとき,

$\phi^{(0)}(\alpha) = f(\boldsymbol{x}^{(0)} + \alpha \boldsymbol{d}^{(0)})$ を最小にする $\alpha \in [0, 1]$を

黄金分割法を用いて求めなさい.

- **fval.m** を作り替える.
- **lineSearch([0;0], [2; 3])**.

直線探索のプログラム

lineSearch.m　(1/2)　GoldenSection.m を改造

```
function alpha = lineSearch(x, d)
tau = (sqrt(5) - 1) / 2;
eps = 0.000001;
a   = 0; % ステップ幅は必ず
b   = 1; % 0以上1以下
w   = b - a;
p   = b - tau * w;
q   = a + tau * w;
fp  = fval(x + p * d);
fq  = fval(x + q * d);
while(w  >= eps)
  if (fp >= fq)
    a  = p;
    w  = b - a;
    p  = q;
    fp = fq;
    q  = a + tau * w;
    fq = fval(x + q * d);
```

(2/2)

```
  else
    b  = q;
    w  = b - a;
    q  = p;
    fq = fp;
    p  = b - tau * w;
    fp = fval(x + p * d);
  end
end
alpha = (a + b) / 2;
```

最急降下法のプログラム

steepest.m

```
function x = steepest(x_0)
  eps = 0.000001;
  x = x_0;
for k = 1: 1000
  disp([x.', fval(x)]);
  d = - gval(x);
  if (norm(d) < eps)
    disp('optimal');
    break;
  end
  alpha = lineSearch(x, d);
  x = x + alpha * d;
end
```

引数は初期点.

反復回数に上限設定

変数と関数値を表示

最急降下方向の計算

収束判定

norm(d): $\|\nabla f(x^{(k)})\|$

黄金分割法による
ステップ幅の計算

反復点の更新

最急降下法のプログラム

(練習) 黄金分割法でステップ幅を求める最急降下法で

目的関数: $f(x_1, x_2) = 2x_1^2 - x_1 x_2 + x_2^2 - 2x_1 - 3x_2 \to$ 最小

の解を求めよう.

- $f(x_1, x_2)$ の勾配ベクトルを計算するMATLAB の関数 **gval.m** を作ろう.

gval.m

```
function z = gval(w)
x  = w(1);
y  = w(2);
fx = 4 * x - y - 2;
fy = - x + 2 * y - 3;
z = [fx; fy];
```

- Mファイル **steepest.m** を作ろう.

- 初期点を $\boldsymbol{x}^{(0)} = (0, 0)^\top$ として実行しよう.

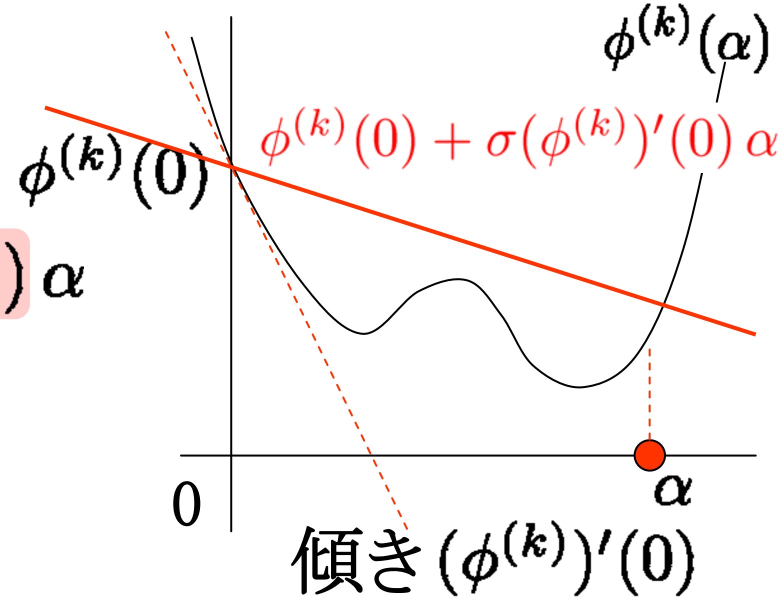
ステップ幅の計算（もう一つの方法）

- Armijo の規則

値は負

$$* \ \phi^{(k)}(\alpha) \leqq \phi^{(k)}(0) + \sigma(\phi^{(k)})'(0)\,\alpha$$

を満たすできるだけ大きな
$\alpha \in [0,1]$ を試行錯誤的に
求める. $(0 < \sigma < 1)$



$\alpha = 1$ が上の不等式を満たすか？ さもなければ,
$\alpha = 0.5$ が上の不等式を満たすか？ さもなければ,
$\alpha = 0.25$ が上の不等式を満たすか？ さもなければ,
$\vdots$

$$\boxed{f(\boldsymbol{x}^{(k)} + \alpha \boldsymbol{d}^{(k)}) \leqq f(\boldsymbol{x}^{(k)}) + \{\sigma \nabla f(\boldsymbol{x}^{(k)})^\top \boldsymbol{d}^{(k)}\}\alpha}$$

最急降下法のプログラム

```
function x = steepest(x_0)
  sigma = 0.1;  % Armijo の規則で使うパラメータ
  beta  = 0.5;  % Armijo の規則で使うパラメータ
  eps   = 0.000001;
  x = x_0;
  f = fval(x);
for k = 1: 1000
  disp([x.', f]);
  d = - gval(x);
  w = - sigma * (d.' * d);   % −σ∇f(x^(k))^T d^(k)
  if (norm(d) < eps)
    disp('optimal');
    break;
  end
  alpha = 1;
```

例（最急降下法）

```
  for h = 1: 100
    new_x = x + alpha * d;
    new_f = fval(new_x);        % f(x^(k) + αd^(k))
    if (new_f <= (f + w * alpha))
       break;
    end
    alpha = alpha * beta;
  end
  x = new_x;
  f = new_f;
end
```

（練習）上のプログラムでつぎの問題の解を求めよう.

目的関数: $f(x_1,x_2)=2x_1^2-x_1x_2+x_2^2-2x_1-3x_2 \rightarrow$ 最小

＊12ページで実行した結果と比較してみよう.