最適化ゼミ

指導者 窪田　耕明
チーフ 長谷　佳明
サブチーフ 角　美智子 奥田　環

第 I 部
最適化問題

1　最適化問題とは？

最適化[1]は, 日々の活動において常に行っている行動である. 例えば, 目的地に速く行きたい時, または目的地に最も安く行きたい時. これらの活動は, どちらも最適化に他ならない. 学問的には, 古くは数学の一部門として研究され, 第二次世界大戦後には, その経済発展を支えるシステム工学の中心的話題となった. そして, 最適化問題は, 以下のように定義することができる.

最適化　… 与えられた制約条件[2]の元で, ある目的関数[3]を最大 (最小) にする解 (最適解[4]) を求めること

この定義にそって考えたとき, 第一に疑問となるのが制約条件であるが, これは例えば, ”電車を使う ”, ”一番安く ”, ”一番速く ”, ”耐震性能がが震度７以上 ”など, 人が節々で求める要求である. そして第二に疑問となるのが目的関数である. これは, ”地元に帰る ”, ”パソコンを買う ”などは最適にするべき ”対象 ”であり ”目的 ”と考えることができる.

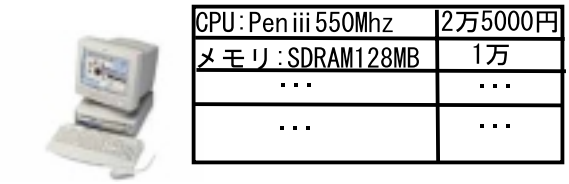

Fig. 1　私に最適な性能とは？

2　最適化問題のコンピュータへの導入

最適化問題をコンピュータを使って解く際にはどうすれば良いのか？コンピュータによって解かせる最短の方法は問題を数式化することである. それに従って, 問題に付随する制約条件も数式, もしくは数値化する必要がある. これらをすることによって問題は扱いやすい形にすることができる. そして, 数式化する際に気が付くことが以下のことがらであろう.

問題には種類があるのではないか？

その直感は正しい, 後ほど説明するが最適化問題には数式化するとはっきりとするが, 種類が存在する. 種類とは大きく分けると連続最適化問題, 離散最適化問題の二種類である. さらに, 前者は導かれる数式によって線形計画問題, 非線形計画問題に分けられる. これらを Table 1 に示す. これら数式化の後に, 各問題に適したアルゴリズムを用いて問題を解くことになる. 代表的なアルゴリズムとしては, 二分法, ニュートン法などがあり, これらを表にすると Table 2 のようになる.

[1] optimization
[2] constraint
[3] objective variable
[4] optimal solution

Table 1　最適化問題の分類

連続最適化問題	線形計画問題 非線形計画問題	探す値が連続的に分布している
離散最適化問題		探す値が離散的に分布している

Table 2　代表的アルゴリズム

連続最適化問題の解法	二分法, ニュートン法 最急降下法
離散最適化問題の解法	遺伝的アルゴリズム シミュレーテッドアニーリング

ただし,SA[5]に関しても GA[6]に関しても最近ではこれを連続最適化問題に適用しようとする研究が活発になされている.

3　最適化問題の例

ここで, 一つの最適化問題の例を挙げてみることとする.

例えば食費をできるだけ抑えたいとする. 選択肢は, 食品 1, 食品 2 だとする. 考慮するものは, 栄養素 A, 栄養素 B, 栄養素 C の最低摂取量を越えることである. これらの摂取量に関しては,Table 3 のようになる. 各食品の 100g あたりの価格はそれぞれ,200 円,300 円だとする.

まず, 最適化すべき目的関数はコストに関して求め, その式は以下のようになる.

$$f = 200x_1 + 300x_2 \quad \rightarrow \quad min \tag{1}$$

この際に, もうすでに言葉としては使用しているが, この関数のことを目的関数,x_1,x_2 のことを設計変数と呼ぶ. これでは考えにくいので,100 で割って,

$$f = 2x_1 + 3x_2 \quad \rightarrow \quad min \tag{2}$$

$$x_2 = -\frac{2}{3}x + F \tag{3}$$

これを最小化することを考える. すると F を最小にすることは,x_1 と x_2 の関係は, 傾きとして保存されているので, なるべく x_1,x_2 軸の交わった点である原点からちかければそれだけ F も小さくなる. つまりは, 原点から以下の式を平行移動させて制約条件を満たす領域で持っともはじめに交わる点が最適な点となる. Table 3 から, 各栄養素から導か

Table 3　食品 100g と必要な栄養素の関係

	食品 1	食品 2	最低必要量
栄養素 A	4	2	20
栄養素 B	1	8	10
栄養素 C	6	4	16

れる制約条件は次の 3 式である. 第一に, 栄養素 A は, 最低でも 20 必要であり, 食品 1 からは 100g あたり 4, 食品 2 か

[5]Simulated Annealing
[6]Genetic Algorithm

らは 100g あたり 2 だけとることができるので, 以下の式のように制約条件を求めることができる.

$$g_1 : 4x_1 + 2x_2 \geq 20 \tag{4}$$

同様にして栄養素 B,C についても制約条件を求めることがきる.

$$g_2 : x_1 + 8x_2 \geq 10 \tag{5}$$

$$g_3 : 6x_1 + 4x_2 \geq 16 \tag{6}$$

制約条件から x_1 と x_2 の満たす範囲は絞られるが, それは Fig.4 の色ずけされた部分である.

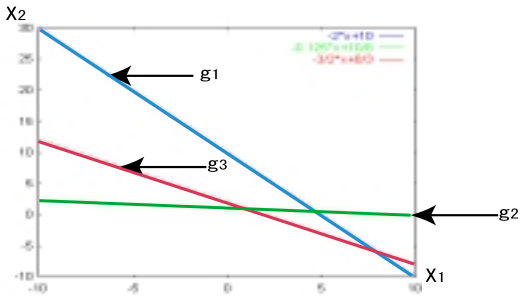

Fig. 2　制約条件式の関係

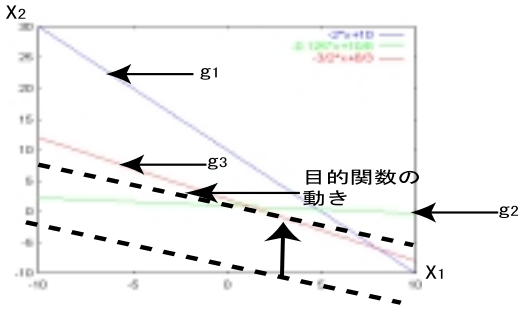

Fig. 3　制約条件と目的関数の関係

すると以上から最適値が求められる. これらはを実際には各種アルゴリズムを用いてコンピュータに解かせるわけ

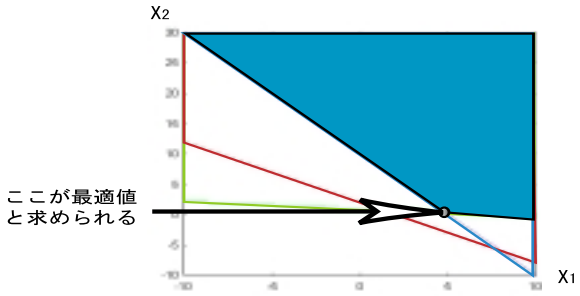

Fig. 4　求められる最適値

である. ちなみに, この最適化問題の解は,

$$(x_1, x_2) = (\frac{14}{3}, \frac{2}{3}) \tag{7}$$

と求められる.

第 II 部
最適化問題の分類

4　最適化問題の分類

最適化問題は，その数学的な性質によっていくつかの問題に分類することができます．この章では，それらのうち，線形問題，非線形問題，離散問題という基本的な問題について紹介し ます.

4.1　連続最適化問題

4.1.1　線形計画問題

線形計画問題（Linear programing, LP）とは，モデルの目的関数と制約条件が，すべて線形である（変数に関する 1 次式で表現できる）ような問題を指し ます．前章で取り上げた例題も線形問題です.

線形問題は，一般的に次のようにあらわすことができます.

$$\text{目的関数 } f = c_1x_1 + c_2x_2 + \ldots + c_nx_n \longrightarrow max(ormain) \tag{8}$$

$$\begin{aligned} \text{制約条件 } & a_{11}x_1 + a_{12}x2 + \ldots + a_{1n}x_n \leq b_1 \\ & a_{21}x_1 + a_{22}x2 + \ldots + a_{2n}x_n \leq b_2 \\ & \vdots\!\vdots \\ & a_{m1}x_1 + a_{m2}x2 + \ldots + a_{mn}x_n \leq b_m \end{aligned} \tag{9}$$

これを行列・ベクトル表現を用いて簡潔に表すことができます.

A, b, c, x をそれぞれ

$$A = \begin{pmatrix} a_{11} & a_{12} & \ldots & a_{1n} \\ a_{21} & a_{22} & \ldots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \ldots & a_{mn} \end{pmatrix}, b = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{pmatrix}, c = \begin{pmatrix} c_1 \\ c_2 \\ \vdots \\ c_m \end{pmatrix}, x = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{pmatrix} \tag{10}$$

と置くと線形問題は,

> 制約条件 $Ax \leq b$ のもとで，目的関数 $f(x) = c^\top x$ を最小化するような変数ベクトル $x = x_1, x_2, \ldots, x_n{}^\top$ を見いだす問題

であるといえます.

線形計画問題のアルゴリズム

線形計画問題に対するアルゴ リズムでもっとも一般的なものは，**シンプレックス法**(simplex method) です．シンプレックス法は，一般の LP 問題に対してなるべく効率よく，解の現れる端点を探し出そうとするアルゴ リズムです．シンプレックス法では，解が存在するか否かをはじ めに判定する過程，および解が有界か否かを判定する過程も含むように工夫されており，収束性も保証されていることから，極めて優れた一般性のあるアルゴ リズムであるといえます．ただし，設計変数の数が多い大規模線形計画問題には適さないという欠点もあります.

4.1.2 非線形計画問題

非線形計画問題(Nonliner programming problem) とは，問題関数または制約条件の少なくとも一方が非線形であるような最適化問題のことを指します.

非線形計画問題のアルゴリズム

非線形計画問題に対するアルゴリズムとしては，Newton 法や準 Newton 法が一般的です．これは目的関数の勾配ベクトルと，Hesse 行列を利用することで直線上の最小化を繰り返し，最適解を効率的に求めるアルゴリズムです. Newton 法は，解の近傍での収束速度は非常に早いのですが，初期点を解の近傍にとらなければ収束性が保証されないという決定があります．また，Hesse 行列の計算や線形代数方程式を解かなければならず，計算量が多いことも問題です.

4.2 離散問題

離散的計画問題 (discrete programming problem) とは，目的関数や制約条件が離散集合である最適化問題のことを指します．離散的なものの例としては，整数の集合，グラフにおけるノードやアークの集合などが挙げられます．離散集合では，連続性，微分，積分などの概念を直接適用することができないという特徴があります.

ここで，離散的計画問題の代表例を表 4 いくつか挙げておきます．このように，離散的計画問題は工学のほとんどすべての分野に登場する非常に重要な問題です．しかしながら，離散問題には，連続性や微分といった古典的な手法をとることができない難しさがあります．また，各分野で生じる問題は多様であるため，それらに共通する構造を見抜き一般化することも困難であるといえます．このような数学的なモデル化が困難な問題を解決するのに GA（遺伝的アルゴリズム）が有効であることが知られています.

最短経路問題	A 点から B 点までの最短の経路を見いだす問題
巡回セールスマン問題 (TSP)	複数の都市を巡回するための最短のルートを見出す問題
ナップザック問題	総重量 x 以下という条件の下で，より多くのものをナップザックに詰めるためにはどうすればよいのかという問題
スケジューリング問題	m 種の機械を用いて n 個の作業を実行するとき，各機械上での作業順序を決定する問題
工場設置問題	工場をどこに設置するのが建設費と製品の輸送コストの観点から最適化を問う問題

Table 4　代表的な離散的計画問題

4.3 その他

よく使う微係数

目的関数 $f(x)$ が微分可能であるとき，最適化のアルゴリズムで目的関数の 1 次または 2 次の微係数を用いることがあります．そこでは，

$$\nabla f \triangle \left(\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \dots, \frac{\partial f}{\partial x_n} \right)^\top \qquad (n \text{ ベクトル}) \tag{11}$$

という**勾配ベクトル**(gradient vecter), および

$$H \triangle \nabla^2 f = \left(\frac{\partial^2 f}{\partial x_i \partial x_j} \right) \qquad (n \times m \text{ 行列}) \tag{12}$$

という *Hesse* **行列**(Hessian) がよく登場します．なお，ベクトルや行列の転置は，右肩に添字 $\top$ をつけて表します.

大域的最小点と局所的極小点

最適化問題における難問の一つに，大域的最小点を局所的最小解の問題があります．制約条件 L を満たす設計関数 $x_0, (x_0 \in L)$ について

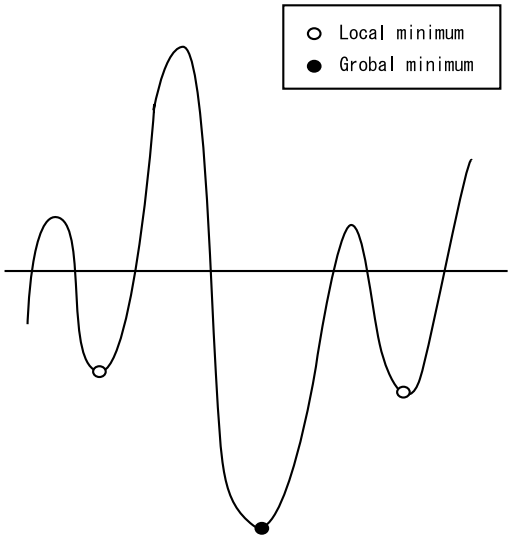

Fig. 5　大域的最小点と局所的最小点

$$f(x_0) \leq f(x) \qquad (\forall x \in L) \tag{13}$$

であるとき，x_0 を**大域的最小点**(global minimum) といいます．それに対して，$x_0 \in L$ について，x_0 の近くにあるすべての $x \in L$ に対してのみ

$$f(x_0) \leq f(x) \tag{14}$$

であるとき，x_0 を**局所的最小点**(local minimun) といいます.

一般に局所的最小点は互いに離れていていくつも存在し ます．一つの局所的最小点を求めることは比較的簡単ですが，すべての局所的最小点を求めて，その中から大域的最小点，つまり真の最適解を求めることは非常に困難です.

5　宿題

以下の式の最小値を自分の知っているアルゴ リズムで解いてください. もちろんコンピュータを用いてです. これは, 予習です.

$$y = 0.1x^4 - x^2 - 5 \tag{15}$$

第 III 部
アルゴリズム紹介

ここでは,EC ゼミで紹介されないアルゴ リズムであるが, 最適化問題で使用されるアルゴ リズムについて紹介する.

6　二分法

最適化問題において二分法を用いるには, その目的関数が連続関数である必要がある. その理由は, そもそも二分法が数学的に以下の関係を用いて編み出されたアルゴ リズムである. そして, 二分法の目的は f(x)=0 となる x を求めることである.

$$f(x_1)\cdot f(x_2) < 0 \tag{16}$$

これが保証されるとき,x_1 と x_2 の間に少なくとも一つの f(x)=0 となる点が存在することが保証されるからである.

実際には, 上の式の関係を満たす初期点を求めた上で,f(x)=0 となる点を求めるわけである. 範囲の絞込みには, 以下の関係を用いる.

$$x_m = \frac{1}{2}(x_1 + x_2) \tag{17}$$

ここで f(x_m) の符号を調べ,

$$f(x_1)\cdot f(x_m) > 0 \tag{18}$$

であれば,x_1 を x_m で置き換える. もし,

$$f(x_1)\cdot f(x_m) < 0 \tag{19}$$

であれば,x_2 を x_m で置き換える.

これを繰り返すことにより, 求めるべき値に近づくほど,

$$|x_1 - x_2| < \xi \tag{20}$$

ξ がある程度小さくなると,x_1 が x_2 ほとんど等しくなる, その点が求めるべき値である. これらを一連の図で表すと, 以下の Fig.6,Fig.7,Fig.8,Fig.9,Fig.10 ようになる.

7　ニュートン法

ニュートン法は, 二分法と同じく f(x)=0 を解くことができるのみならず, 未知数が二個以上ある場合にも同様に適用することができる. ここでは, 一般的である未知数が一個の場合を例に挙げ, ニュートン法を紹介する. ニュートン法は簡単にいえば傾きを利用した解法である.

ランダムな初期値である x_0 からはじめて以下の式を適用する.

$$d_k = -\frac{f(x_k)}{f^{'}(x_k)} \tag{21}$$

その後,

$$x_{k+1} = x_k + d_k \qquad (k = 0, 1, 2, \dots) \tag{22}$$

これを繰り返すことで, 解は収束に向かう. 但し, 目的関数によっては振動など, 収束できなくなることがある. そして解の判定は以下の式で行われる.

$$|x_{k+1} - x^*| \leq C|x_k - x^*|^2 \tag{23}$$

ここで x^* は, 求める解である. 求める解の近くでは,x_{k+1} にしても x_k にしても, その差異に関してほとんど変化しない状態. つまり, それくらい解に近づいていることをこの式は示す一つの鍵となっている. しかしながら, この安易な解の判定は解が大域的解ではなく, 局所的解に陥ってしまう危険性をはらんでいるといえる.

これらの動きを図で表すと,Fig.11 のようになる. ただし, これはうまくいく例である.

8 最急降下法のアルゴリズム

8.1 最急降下法

初期点 x_k を定め，その点における微分係数 (勾配)$f'(x_k)$ を求める．勾配は，その点において目的関数 $f(x)$ の値が大きく増加する方向であるので，関数 $f(x)$ の値を減少させるため，勾配と逆の方向 $d_k = -f'(x_k)$ に進めばよい. (Fig.12 参照)

$$x_{k+1} = x_k + \alpha_k d_k \tag{24}$$

を次の点とすればよい．ここで α_k はステップ幅という.

8.2 直線探索

ステップ幅を具体的に求めるために，直線探索を行う．初期点 x_k より，上で求めた探索方向に沿って，最小点を通り越すまでだんだん増加して進んでいく．ここで微小な距離 s を設定し，$s, 2s, 4s$ のように進める．各々の点で関数値 $f(x_i), n$:自然数を計算し，$f(x_i) < f(x_{i+1})$ となった時点で $x_{i-1} = x_1, x_{i+1} = x_2$ として黄金分割法に進む. (Fig.13 参照)

8.3 黄金分割法

黄金比 $\dfrac{1+\sqrt{5}}{2}$ を τ とする．x_1 と x_2 の距離を d とし，新たな点

$$x_3 = x_1 + \frac{\tau - 1}{\tau} d, x_4 = x_1 + \frac{1}{\tau} d \tag{25}$$

を求める．関数値 $f(x_3), f(x_4)$ を求め，$f(x_3) < f(x_4)$ ならば $x_1 = x_3$ とし，$f(x_3) < f(x_4)$ ならば $x_2 = x_4$ とする．これを繰り返し，あらかじめ定めた ϵ に対して $d < \epsilon$ が成り立てばその時点で打ち切る．(Fig.14 参照)

8.4 課題

$f(x) = (x-1)^2 + 1$ の極小値を求めよ.

9 スケジューリング問題

いくつかの仕事をいくつかの機械で処理するとき，すべての仕事を完了時間等の評価値（目的関数値）が最小となるように各機械上での仕事の処理順序を決定する問題です.

機械の台数や仕事の内容によっていくつかの問題に分けられます.

- 1 機械型：機械が一台のみの場合
  多くの場合理論的にあるいは経験的に良い解法が知られています.
- 並列機械型：機械が複数台ある場合
  どの機械も同じ機能を持ち，各仕事はどの機械でも処理可能です.
- ショップ型：機会が複数ある場合
  機械での処理順序によりさらに 3 つに分類されます.
  - フローショップ型：すべての仕事で処理順序が同じ場合
  - ジョブショップ型：機械の処理順序が仕事ごとに異なる場合
  - オープンショップ型：どの仕事も機械の処理順序が指定されていない場合

[二分法参考プログラム]----------

```
#include<stdio.h>
#include<math.h>

/*式はここに入れる*/
float f(float x)
{
return (x*x*x-8*x*x+10*x);
}

int main()
{
float x1,x2,eps,f1,f2,xm,ff;
int i=0;
printf("\tBisection method\n");
printf("\t 初期値 x1 x2 , 終了条件\n");
for(;scanf("%g%g%g",&x1,&x2,&eps)!=EOF;){
f1 = f(x1); f2 = f(x2);
if(f1>0){
xm=x1; x1=x2; x2=xm;
ff=f1; f1=f2; f2=ff;
}
printf("\nFinding a root between x1 = %g and x2 = %g \n",x1,x2);
printf("f(x1)=%g f(x2)=%g eps=%g\n",f1,f2,eps);
if(f2<0){
printf("共に負の値です. 初期点としてふさわしくありません\n");
continue;/*for*/
}
else if(f1>0 && f2 >0){
printf("共に正の値です. 初期点としてふさわしくありません\n");
continue;
}
for(;fabs(x1-x2)>eps;){
xm = (x1+x2)/2;
ff=f(xm);
i++;
printf("%2d %15.6e %15.6e %15.6e\n",1,x1,x2,xm,ff);
if(ff<0)
x1=xm;
else
x2=xm;
}
printf("A root found between %g and %g\n",x1,x2);
break;
}
return 0;
}
```

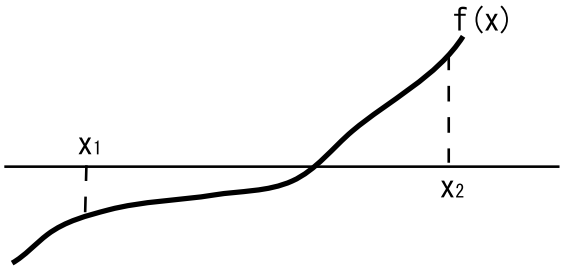

Fig. 6　二分法その 1

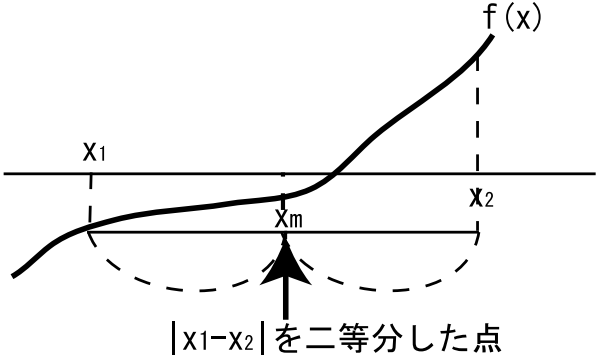

Fig. 7　二分法その 1

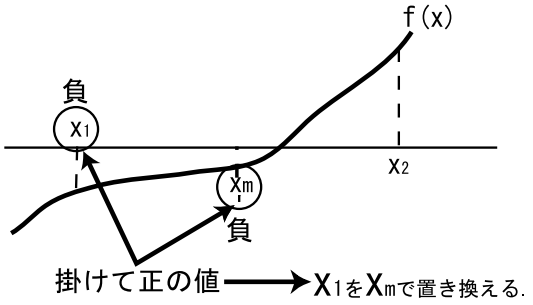

Fig. 8　二分法その 3

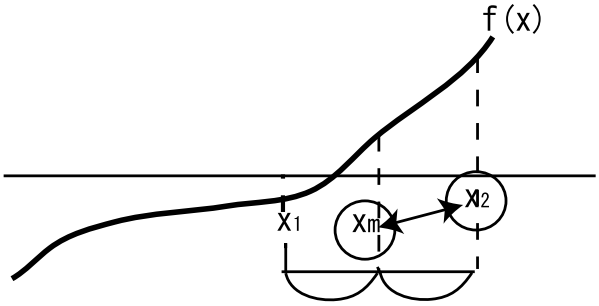

Fig. 9　二分法その 4

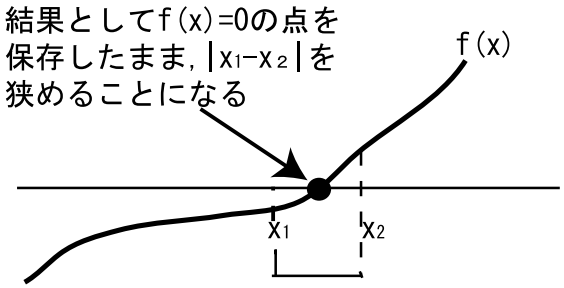

Fig. 10　二分法その 5

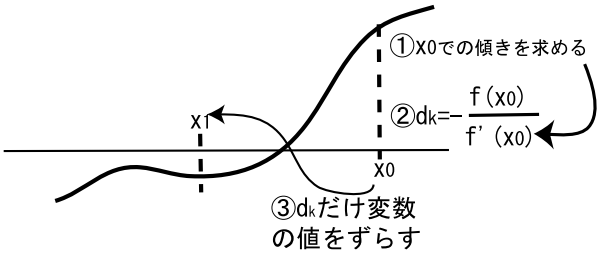

Fig. 11　ニュートン法

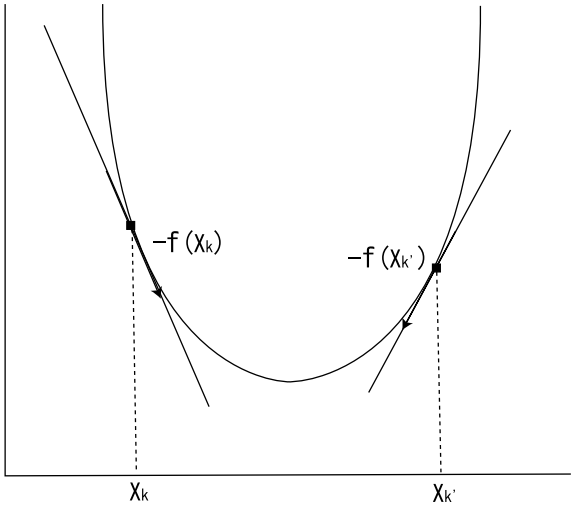

Fig. 12　最急降下法

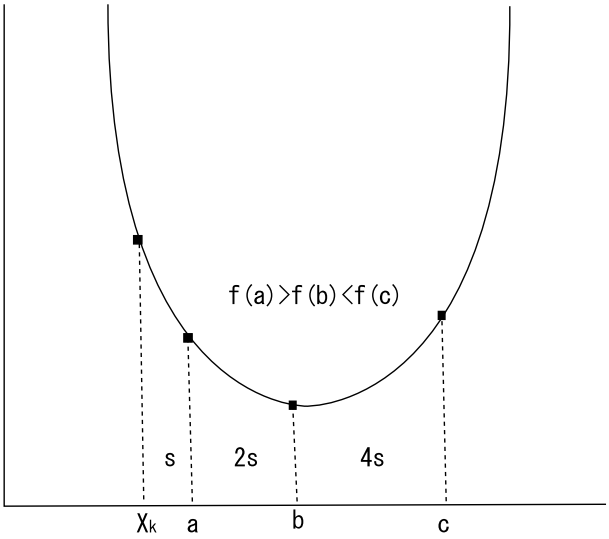

Fig. 13　直線探索

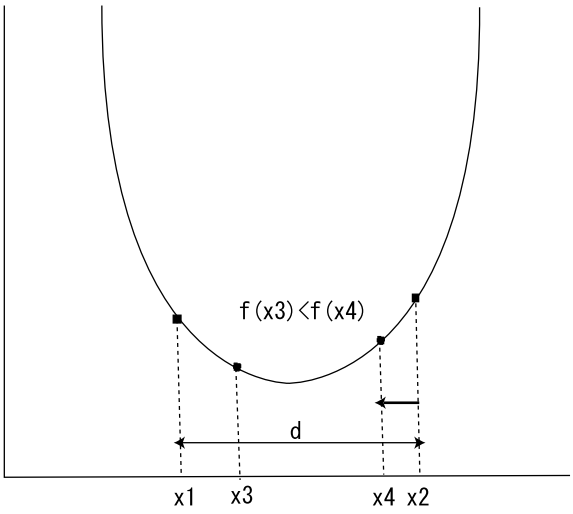

Fig. 14　黄金分割法