＜特別講演＞　函数の最大値の効果的な探し方

古　瀬　大　六*

昨日の懇親会の席上で，多田さんからＯＲ学会の地方学会は，その主要な目的にエレクリェーションである。おしゃべりはなるべくわかりやすく，そして数式などは出ないほうがいいというようなお話があり，私もその趣旨に賛成でありまして，なるべく数式的なことは申し上げないようにしたいと思います。それは私ができるのにやらないということでなしに，私自身能力がないからやらないということであり，数学的な表現はこの中にも多数の専門の方もおりますので，そういう方にお任せする。そして私のほうは直感的な，一種の suggestion を並べるということになるわけで，厳密性を尊ばれる方にはいろいろ不愉快な点もあると思いますが，こういうような話もあるんだということでお聞き願えれば結構だと思います。

函数の最大値，最小値，最大の点あるいは探すということはＯＲの基本的なテクニックでありまして解析的な方法から，ＬＰであるとかＤＰであるとか，いろいなテクニックが出ているわけであります。

しかし，私は数学者ではありませんので，元来は経営学畑でございますので，主として経営学的な立場からこういう問題を考えてみたいと思います。で，経営学的な立場に立った場合に函数の極値を探すということはどういうことかと申しますと，これもいろいろな場合があると思いますが，その一つは，経営を合理的にやっていくということは，結局経営者が，自分のきめるべきdecision variablesの最適値をどうやって探していくかという問題になってくるわけであります。

その際に自分の企業のおかれているあらゆる状況が数学的に与えられておって，かつその目的函数がはってりしているということであれば，これは問題をすべて数学者に任せても一向差しつかえはないわけであります。ところが残念ながら現実はそうじゃない，むしろそういう explicit な数学的表現を許さない場合のほうが通常であると考えてよろしいわけであります。

そうなってきますと，ある函数の最適値を探すということは，決して数学的に式を立ててそれを解くという問題ではない。むしろそれ以前の問題があるんだということになる。何か函数的な関係はあるには違いないが，それを式の形にもっていくことはできない。しかし，現実というものがある以上，現実の中にはなんらかの函数的な関係は存在するわけであります。

そういう与えられた環境の中でなんらかの合理的な行動をやっていこうということになると，その最適値を探すという場合にも，それを数学的に明らかな形に取り出すということをなるべくやらないで最適値を探したいという要求が出てくるわけであります。

---

*小樽商大　昭和38年10月３日　第14回　研究発表会「経営科学」第７巻第３号

これをごく簡単な例をとって申し上げてみましょう。周囲の環境あるいは目的函数それ自体が時間に依存する場合もありえますが，一応単純化するために時間に依存しないものを考えると，問題は，二次元の場合で考えれば，山の最高点，あるいは谷底の一番低い点を探すという問題になってくるわけであります。

ここでは一応山登りをやるんだということにしておきましょう。山登りをやる場合にその函数の形が完全に与えられているということはどういうことかといいますと，結局地図が完全に存在する，あるいはレーダーがあるとか，あるいは望遠鏡，肉眼でもよろしいのですが，とにかく周囲の状況が完全に見通せる，そして自分の現在立っている位置がどこであり，どういう方向へ行けばどこへ行くということがはっきりわかって，頂上の存在地点もはっきり見えるというような場合であります。で，地図の上で適当と思われる線を引っぱってその通り歩けば最高点に到達することができるわけであります。

ところが，企業内部のいろいろな事情，あるいは周囲の経済的な事情というものは，いつも完全に見通せるというわけではない。自分の近所はある程度見えますけれども，遠くへ行けば行くほどかすんで見えないという状態になっていくわけであります。極端な場合を考えますならば，めくらである。めくらが山を登るのにはどうしたらいいかというようなアナロジーで問題を考えていくことができるのではないかと思います。

その際に一番単純なケースとしては，その函数自体がなめらかな格好をしている。山で申しますと茶わんを伏せたような格好になっているという単純な場合が考えられます。そこへ行くにはどうしたらいいかというと，めくらがこういうなめらかな山のてっぺんに行くにはどういうふうに行動したらいいかということになって参ります。それはご存じのように例のgradient motionで進めばよろしい。と申しますのは，自分の立っている付近の傾斜を探すわけであります。これは横から見た場合ですから，上から見おろした場合を考えれば，高さの等しいところを結んだいくつかの等高線ができまして，それが内側に行けば行くほどその値は大きいという性格を持っており，これら等高線の全部がなめらかであり，そして，重なり合っていないという場合であります。

この場合でありますと，現在自分のいるところが，この地図のどこであるかということがわからなくてもいい，とにかくどこかにいたとすると，この点から東西に傾斜をはかる，それから南北に傾斜をはかる。これならば手探りでもできるので目が見えなくてもできるわけです。自分の立っている足元のローカルな状況に対する手操りは目が見えなくてもできるわけです。

その測定をやった上でx方向への傾斜が$+0.1$，y方向への傾斜が$+0.2$なら$(0.1,\ 0.2)$のヴエクトルを考えて，その方向へ動いていけばいい。そういうことを移動しながらしょっちゅう自分の足元について測定していくと結局等高線に直角に動いて最高点に到達することができるわけであります。何らかのポテンシャルを持った一般の物理学的な運動の場合にもこれと似たような動きが現れます。

それではどんな場合であっても，こういうように gradient motion で進めば，ローカルなインフォメーションだけで，全体としてのグローバルな最高点に行くことができるかというと，これはいつもそういうふうにうまくいくとは限らない。数学者というのは自分の都合のいい場合ばかりみますから，そういう都合のいい場合にはうまくいきますが，現実にはうまくいかない場合があるわけです。たとえば，等高線が凹んでいればどういうことになるか。等高線が滑らかであれば凹んでいても一向差しつかえないわけであり，凹みがあっても，やはりこの最高点が一つしかない限りこれはどこからスタートしても，この gradiənt motion に従って最高点に行くことができる。めくらであっても足元を見ながら行けばいい。どこからスタートしても必ず山の頂上に行けるということになるわけです。

ところが，頂上が一つであればいいが，峯が二つある場合もあるわけです。この場合スタートの位置のいかんによって，ある所からスタートすれば一つの頂上へ行くが他の位置からスタートすれば他の頂上へ行く。問題はどちらがほんとうのグローバルな最高点であるかという問題になるわけです。

もし目が見えるならば，いったん頂上に上がって望遠鏡で，自分の立っている水面線を眺め渡す。そしてそれにひっかかるところがあるかないかということを調べれば，これがローカルなマキシマムであるかどうかの判定がつくわけですが，残念ながら目が見えませんから，頂上へ登って眺めるということもできない。従って，このような場合にはローカルな情報だけで行動したとしますと，あるところからスタートした場合には，ある地区の中での最高点には行きますが，他の地区を含めた全体としての最高点に行き着くことはできないということになる。

従って，どうしてもほんとうの最高点へ行こうということになれば，やはりある程度見通しがなければだめだということになって参ります。それをどのように現実的に処理していくかということはあとのほうで触れたいと思います。

次の難関は壁がある場合であります。山で申しますと国境がある。国境にもまっすぐなのと曲ったのといろいろあると思いますが，この場合は一応まっすぐということにすると，この国境の内側にほんとうの最高点があれば問題はありませんが，外側にある場合もあるわけです。後者の場合，ある所からスタートして gradient motion で進んでいたとしますと，壁にぶつかるかもしれません。

それでは，この東西・南北の壁で囲まれた領域の中だけを考えた場合の最適点はどこにあるか，それは申すまでもなく，壁の上にあるわけです。gradient motionに従って進行していき，この壁にぶつかった場合，そのぶっかった点における gradient をはかって，そのうちの壁の外側の方向に押し出そうとする運動はそこでいったんカットして，壁に沿って動くことのできるコンポネントだけについて動かしていく。

つまり，壁にぶっかったならばそれを越えてはいけないわけですから，その壁に沿っての分力のうち，上にいける分力があれば，そちらへ進む。下に下がる分力があればそちらへ行けばいい，

進む際にも，その途中の，すべての点におきまして gradient をはかっていることになります。ですから力としては，いつも壁に押しつけるような力が働いているわけであります。そういう力が働いている場合には，壁に押しつけられながら，ほんとうの意味における最適点に到達することができます。

以上のように壁がありましても一向差しつかえありません。ただし山の格好がひょうたん型をしておりますと，やはり前と同じように工合のわるいことが起きて参ります。

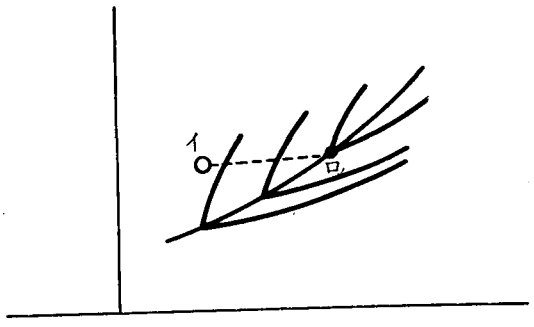
山の頂上が1つしかなくてその等高線がきれいにふくらんだ格好をしている場合であれば，この壁に制約された場合の最適点はいつもただ1つしかありません。

けれども，このように凸凹といいますか，ひょうたん型の部分がある場合には，大域的な最適点のほかにローカルな最適点がまた幾つか存在する可能性があります。あるところからスタートすると，1つの最適点へいく，別のところからスタートすると他の最適点へいってしまう。ほんとうの最適点がどれかということは遠くを眺めてみなければならないという場合もあるわけであります。

しかし，そういう特殊なケースを除いて，等高線が，ふくらんでいるという通常の場合であれば，仮に壁があったとしても，この gradient motion の動き方に少し工夫を加えて壁にぶつかったらこうするんだというふうにしておけば，その壁の中における最適点に到達することは可能になって参ります。この場合にもやはり，完全に目が見えないということであって一向差しつかえはありません。

それから今度は，凸型でありますけれども，函数が piece-wise continuous な導函数をもっている場合，すなわち，その等高線に角があって，滑らかでない場合もあり得ると思いますが，この場合でありましても，やはり grdient motion で進んで一向差しつかえありません。厳密に申しますと，この角の点における 函数の値が不明確になるなけですけれども，現実的にいえば，やはり gradient motion で進めるわけであります。

角まで一応原則通りに進みまして，それからこの角の上の点あたりに到達いたしましたならば，今度はこちら側とむこう側の傾斜，数学的な言葉でいえば左導函数と右導函数とを両方とも計算して，左側からこれを越えていくときには右導函数のほうを使う，反対側から越えていくときには右導函数を使う，というふうに約束しておけば差しつかえないわけであります。



A 図

それから，もう1つの変数につきましても，やはり同様の操作をやる。こうすれば，めくらであっても，壁や峯が途中にあっても一向差しつかえないわけであります。

しかし，この場合には工合のわるいケースも出てくる。それはA図のような場合であります。仮に点イからスタートして，右方向に進んでいく，そして峯の稜線上の点ロのところまでいく。そこで，ちょうどそこで手探りをしてみるわけであります。そうするとこの右方向への傾斜は，点ロまで増加して，それから先は減少するわけです。ですから，gradient motion をやったとすると，点ロでは止まらなければならない。右方向への分力はないということになります。次に縦方向への分力を考えますと，やはり同様であります。従って縦横の分力ともゼロである，だから点ロで止まるということになりますが，実際はこの峯の稜線にそっていけばどんどんふえる，よりよい点に到達することができるわけです。だから通常の意味における gradient motion はこういうようなケースでは動けないということになってしまう。

数学的にいえばそうですが，実際問題としては動けないというのはおかしいわけで，なんらかの工夫をすれば当然動けるわけであります。それを動かすようにするにはどうしたらよいかと申しますと，いろいろ対策があると思います。その一つは，この動けない点に到達したならば，その点を中心としてある小さい半径の円を描いてみて，その自分の脚元を一応全部捜索してみる。二次元の場合であれば全面的な捜索は容易です。それがN次元なると全部探すのは大変なことですが，一応二次元の場合であれば，こういうことも可能になるわけであります。実際問題としては，自分の足許に短かい半径の円周を描いて，その上の高さを測るということをやればよい。

そうするとその中で必ずどこか一番高いところがあるはずです。この円周上の一番高いところへ向って引いた，中心点からのヴェクトルの傾斜が，その点における gradient になります。

従来の意味における導函数で定義された gradient でありますと，こういう場合には失敗いたしますけれども，今申しましたように定義を変えて，その現在点を中心にして，ある小さい球あるいは円を描いて，その上で最高点を与えるようなヴェクトルの上での傾斜が gradient だと定義しておけば，このような場合でありましてもひっかからないで，さらにこの陵線に沿ってどんどん進むことができ，ほんとうの最高点にどこかで到達するという可能性が出て参るわけであります。

ただ，この次元数が少ない場合はいいのですが，次元数が多くなると現実問題としていろいろ困難があるでしょう。

次元数が高くてもやりたいというならば，その変数の decision variables の一つ一つを受け持っている担当者の間に適当な情報の交換をやらなければならない。しかしこれはローカルな狭い範囲の情報でありますから，独立的と考えても，それほど誤差がないと考えますならば，変数が多くなってもまあまあやれないこともないという見解も成り立つかと思います。

今申しました例は2つの連続な函数があるところで一方から一方へパッと不連続に移っている場合ですが，それが3つ以上の互に異った連続函数があって，それらがある一点で不連続に一緒になっているというような場合であっても差しつかえないわけです。やはりこの論法で完全に山の頂上に到達することは可能であります。

現実問題としては，例の結合需要（上着を一つ売ればズボンが必ず二着くっついて売られるというような場合）がありますと，こういった問題が現実に成り立つことになります。

もう一つ，こういう状況を打破する一つの方法として，次のようなことも，考えられると思います。それは，ここの gradient motion で動けなくなったならば，over-shoot すればよい。あるところでひっかかったならば，その点の gradient が普通の方法では計算できないとしても，ここで従来とっていたと同じ速度をそのまま保ちながら若干進んでみる。そうすれば，ここのへりからはずれますから，ノーマルな領域に入りまして，そこで従来の gradient motion が可能になるわけであります。ここで gradient に従ってもう一度戻る。で，戻ったところでまたひっかかりますから，そこでもう一度 over-shoot をやる。行ったりきたりということをやりながら，結果においてはだんだん上昇していく。途中でいったん減りますけれども，あとでは必ずふえるわけであります。

それからもう一つ別の解決方法といたしまして， random shock を与えるという方法もあるような気がします。これの与え方としては，その変数を決定する人自身が，自分のきめた変数に random shock を与えてもいいし，また函数の中自体に random shock が built-in がされていてガタガタ動くということであっても同じだと思いますが，いずれの場合でありましても，動けなくなったところから，どちらの方向でもいいのですから，ランダムに離してしまう。そうすると，その離された位置のいかんによりましては， gradient に従って動くことが可能になってくるわけであります。

一般に壁の形がこのようにカミソリの刃のような形の場合でなく，もっとブロードな場合であればあるほど，ランダムショックを受けた結果，元の値よりも低いところに行くという好ましくない結果を生ずる。しかし，極端にこちらのほうが狭くカミソリの刃のようになったといたしましても，前よりもよくなるところと，前よりもわるくなるところとの比率は 50：50 であって，後者のほうが50％以上になるということは絶対にない。ですから運がわるいといつもいつも減少減少ということになって下がっていく可能性がないわけではありませんが，しかし，通常の場合はこういうことを繰返していくことによって上に上がっていくという可能性も相当ある。むしろその可能性のほうが多いと考えていいのではないか。これを厳密に数学的にやるといろいろ面倒なことになりそうな気がいたしますけれども，そういうような解釈もあり得るのではないかと思います。

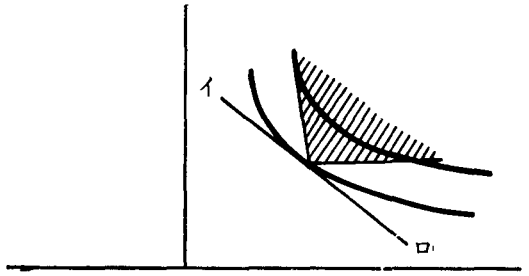
それから，さらに面倒くさい場合で，二つの斜面のつぎ目が絶壁になっている。函数自体が不連続であるという場合ですと，先ほどの over-shoot の方法は必ずしもうまくはいかないわけです。この辺からスタートして，グレイジエントに従って動くとこういうふうに動くことになります。そこでオーバーシユートして，ちょっとこっちに出ますと，今度はこれをまた元へ引き戻すような力が働くことになります。結局この辺でガタガタやりながら最高点に行ける…。行けない場合もあり，ここからオーバー・シユートして行くとこっちへ戻されて，戻されたところからまた行くということで，この上に乗って逆のほうに押し戻されるような場合も出てくるはずであります。こ

ういうふうに落差があると，つまり函数自体に不連続なところがありますと，random shockとか，あるいは over-shoot のやり方でやってうまくいかない場合が出てくるかと思います。

それから次の問題は，gradient に従って動くということは一体どういうことを意味しているかということです。われわれの目的は決して数式を解くこと自体でなくて，なるべく函数の値の高いところ，すなわち等高線の高いところへ行こうということが目的ですから，正確にグレイディエントに従って動く必要は必ずしもない。それから若干離れた方向に進んだとしても，やはりいくらか高いところに行けるわけです。さらにこの現在おります位置における等高線の接線の方向よりも少しでも内側に向いておれば，そういう運動は山の必ずより高いところに到達する可能性を持つております。

ですから，ごく狭い範囲で考えますと，この接線の真上はいけませんが，それより少しでも内側であればどちらへ行っても函数をふやすという目的は達成できるわけです。ただし現実の経営管理という面から考えますと，こういう方向へ進むという決定をするということは通常非常に困難ではないか。と申しますのは，企業の中では，この decision をできるだけ簡単にするために decentralize することが有利であります。完全に decentralize したとしますと，1つ1つの変数の担当者には全空間の中のある一次元だけしか見えない，従って，全部の分担者の情報を集めても，座標軸外の空間は見えないわけです。だから gradient を測定する場合にも変数 x の担当は横軸上だけを見て，その方向への分力を測定する。y のほうもそうやって判定するわけです。そしてその2つの情報を合わせて，しかもこの場合はこちらへ来てもいいと，すなわち x は逆の方向に行った方がいいという結論を出すためにはこの y についての情報を同時に必要とするわけであります。だからこの場合にお互いに情報交換が必要であります。しかし，情報交換をやらなければならないということは管理上あまり望ましいことではありませんから，なるべく一人一人がばらばらにきめながら，しかも全体として望ましいところに行くことのできるメカニズムがあれば，そのほうが望ましいということになります。

このような decentralization を重視するという観点からすると，ここのところはちょっと具合がわるい。それぞれが decentralize された決定を行う，すなわち自分のところで得られる情報だけについて，自分の所属している変数を動かす。しかも全体としては多少なりとも高い等高線上に到達できるという領域が存在する。言いかえれば，現在自分の立っている位置でgradientをそれぞれ decentralizeされた形で分担して決定，測定ということをやる，その点で，各変数の gradient がプラスに出たとすれば，その絶対値は何であろうと一向差しつかえないそのプラスの符号だけを知っておればよい。x はとにかくふやせ。また y の gradient がプラスに出れば，それを増加させ，マイナス



B 図

に出れば減らすということ，それだけのことをやればよろしいわけで，その絶対値を幾らにしなければいけないということは必ずしもないわけです。とにかく，傾斜の符号だけを考えて，しかも decentralize された方式で動いていくとすれば，全体としてはB図の斜線の中のどこかに落ちる。

実際にSをふやすことができるのは，この接線イロの右側の半空間全部であります。この半空間の白地の部分まで入ってこようとすると，これは情報の交換を必要としますが，斜線を引いた部分への方向の範囲であれば，gradient についての定量的な情報だけでいいということになります。

以上申し上げました定性的な方法が実際問題としてどういう意味を持っているかというと，この decentralize した場合に，それぞれの変数の分担者としては，x の分担者は強気で，y の分担者は弱気であるという，性格的な差があっても一向に差しつかえないことになります。ただそれが完全にバランスがとれておればスピードが早い，しかしスピードが違ってもそのために目的点にいけないという心配はないと言ってよろしいわけです。

このような定性的な gradient に従った進行を，先ほどのような不連続がある場合あるいは壁がある場合などに拡張して適用するということも可能になって参ります。

定性的方法はもう一つの利点をもっております。普通の定量的な gradient motion ですと，茶碗を伏せたような形をしている場合には，ふもとの急傾斜な所では非常によく進めますが，頂上に近附くにつれて一段とスピードが遅くなるわけです。デイジタルコンピューターでやっておりますと，頂上のそばまではすぐ行きますが，頂上附近でモタモタして，何万回やっても収束しないということをよく経験します。頂上の近所に行ったならば，スピード・アップしてやるということをやれば，やらない場合よりも早くこの地点に行けることになる。定量的な gradient motion の場合でも，このスピードを傾斜が少し緩やかになってきたところで速めてやるという方法がとれると思いますし，今申した定性的な方法でやっていけば，この辺のスピードもこの辺のスピードも全然変らないわけです。厳密にいえば，頂上の近所では永久に頂上に到達できないというような数学的な結論が出る場合もあると思いますけれども，とにかく頂上の近所まで行くスピードは定性的に行ったほうが早いということもあり得ると考える次第であります。

さらに問題を現実にいろいろ近付けていくと，数学的に考えた場合に，どうしても動けないような場合であっても，現実的に考えれば動けるはずだというようなケースがいろいろあるかと思うのであります。

条件附き極値問題の場合であって，その元の函数が完全に concave でないというような場合でありますと，条件をつけますと，ローカルな optimum point があっちこっちに現れてうまく進行できないという場合がときどき出てきますが，その場合の対策としては，もう一つこういうような対策も考えられます。ある分権的管理者が自分の分担の変数について非常に強力な情報網を持っていて，この変数を瞬間的に全変域にパッと動かして，それにともなう目的函数の変化

を見ることができる。全部の変数についてやるということは無理ですが，ある特定の変数については，こういうことができる。そういうことであると先ほどのようなケースであっても，このlocal な optimum point から，この一変数だけについての全域的な optimum point に跳び移ることができる。そういう瞬間的なアジヤストメントが可能になると，場合によっては local optimum があっちこっちにあっても，それをポンと飛び越えて行くということも可能になる。そういうことが考えられと思います。

そのほか，今申したような瞬間的なアジヤストメントについて，x と y というように一つ一つ時分割的に繰り返してやっていくという探し方も可能であります。まずここで x について maximum を出して，次に y についてを出す。その結果，有限ステップで全体としての最適点に到達できる場合もありますが，一般的に無限ステップを必要とするかと思います。

このような tâtonnement process は，純数学的な興味の対象であるよりも，むしろ現実の企業の経営，組織，その中の情報網をどのように設計していくかということを考えます場合に非常に重要なことになってくるのではないでしょうか。

そういうことを頭においてやられると非常にありがたいと考えております。まとまりのない話でお聞き苦しかったと思いますが，一応私の話を終らせていただきたいと思います。

以　上

IFAC東京シンポジウム

近頃，東京で国際的な学会が続々と開催されますが，1965年春または秋に東京で国際自動制御連合（IFAC）の「自動制御におけるシステム工学」というシンポジウムが開催されることになり，兼重寛九郎氏を委員長とする準備会が去る2月18日結成されました。準備会の主体は日本学術会議の自動制御研究連絡委員会であるが，主題の内容に鑑みORとも関連が深いので本会からは常務理事の近藤東大教授が委員として加わることになりました。関心の深い会員も多いことと存じますが詳細は同氏に御問合わせ下さい。なおIFAC東京シンポジウムの事務局は計測自動制御学会内に置かれています。

1964年度秋季研究発表会

大　阪

本誌第7巻第2号で予告いたしました本会の秋季大会を富山市で開催する件は其後地元の御都合により将来に延期することにして関西支部の所在地大阪地区に変更になりました。

大阪は本会の前身である関西経営科学協会の発祥の地でもあり，活動的なORワーカーが多数居住して居られます．盛大な会合になることが期待されます。

期日は今のところ未定ですがオリンピック東京大会の後，多分11月初旬の見込みであります。