



Title	科学計算用サブプログラム・ライブラリィ（SSL）の追加について
Author(s)	
Citation	大阪大学大型計算機センターニュース. 1974, 13, p. 16-28
Version Type	VoR
URL	https://hdl.handle.net/11094/65237
rights	
Note	

The University of Osaka Institutional Knowledge Archive : OUKA

<https://ir.library.osaka-u.ac.jp/>

The University of Osaka

科学計算用サブプログラム・ライブラリ (SSL) の追加について

49年4月現在、下記のプログラムが当大型計算機センターのSSLとして登録されました。これらは、京都大学大型計算機センターから譲り受けたものです。都合でディスクには登録されていませんので、利用者はソースカードで入力しなければなりません。御希望の方には、ソースプログラムのリストをさしあげますので共同利用掛に申し出て下さい。

ライブラリ名 LMINF

LINEAR MINIMIZATION WITHOUT USING DERIVATIVE

直線上の極小化（微係数を使用しない方法）

作成者：星野 聡

1. 概 要

n 次元空間中のベクトルに沿って、関数の極小化を行なう。

関数計算回数を節約するために、内挿において、より小さい関数値を見出せば、RETURN する。

2. 使用法

2.1 呼出し手順

CALL LMINF (TEST, N, X, F, YS, EPS, IER, STEP, INIT, X0, X1, X2)

引 数	型 と 種 類	内 容
TEST	サブルーチン名	<p>極小化を行なう関数を定義するサブルーチン名である。 このサブルーチンは SUBROUTINE TEST (N, X, F, INIT) の形をもつこと。ここで</p> <p> $\left\{ \begin{array}{l} N : X \text{ の次元数, 整数型変数名, 入力} \\ X : \text{関数 } F \text{ に対する位置ベクトル, 実数型配列名, 大きさは } X(KK), \\ \quad KK \geq N. \\ F : \text{実数型変数名, 関数値を入れること。} \end{array} \right.$ </p> <p>INIT INIT=0 ならば関数の計算の他に、必要な initialization も行なうようにすること。 INIT \neq 0 のときは関数の計算のみ行なう。整数型変数名。</p> <p>⑩ TEST は呼ぶ側のプログラムで EXTERNAL 宣言をする必要がある。</p>
N	整 数 型	極小化を行なう関数の次元数。入力、保存される。

引 数	型 と 種 類	内 容
X	実 数 型 1 次 元 配 列 名	N次元の位置ベクトル。 linear searchの開始点の位置ベクトルを set して, LMINF に入ること。 returnの際には, linear searchの結果として求められた, 位置ベクトルが入っている。 入力および出力。大きさは X(KK), $KK \geq N$ 。
F	実 数 型 変 数 名	極小化しようとしている関数の値。 linear searchの開始点における関数値を set して, LMINF に入ること。 returnの際には, linear searchの結果として求められた位置における関数値が入っている。 入力および出力。
YS	実 数 型 1 次 元 配 列 名	linear searchの方向ベクトル。大きさは YS(KK), $KK \geq N$ 。 サブルーチン LMINF を call する際に set しておくこと。 LMINF 内で符号が変えられることがある。 LMINFで行なわれる最初の searchの大きさは, LMINF が call されたときの引数 STEPの大きさと $\ YS\ $ の積にひとしくとられる。入力および出力。
STEP	実 数 型 変 数 名	LMINFで行なわれる最初の searchの大きさを決める。従って LMINF を call する際に set しておかねばならない。外挿が行なわれるたびに 2 倍され, 内挿が行なわれるたびに $\frac{1}{2}$ 倍される。入力および出力。
EPS	実 数 型	内挿を行なった際の search による変位の大きさが EPS の $\frac{1}{100}$ 以下になると, 内挿を打ち切って return する。この際 IER は -2 に set される。 この値の選び方は関数の性質により異なるが, テストにおいては, 単精度の場合 10^{-6} , 倍精度の場合 10^{-12} を使用した。入力, 保存される。
IER	整 数 型 変 数 名	return condition を示す。出力。 IER = -1 : LMINF で zero division が起ったときに set される。 通常, 極小点に十分近いときに発生する。 IER = -2 : LMINF で内挿が 10 回以上行なわれたとき, および search による変位の大きさが EPS の $\frac{1}{100}$ 以下になったときに, 内挿を打ち切り, IER = -2 として return する。 この時, STEP は 1.0 に set される。 IER = 0 : 上記以外の時
INIT	整 数 型 変 数 名	LMINF 内でサブルーチン TEST の CALL において引数として用いられる。(TEST 参照)。入力, 保存される。
X0 X1 X2	実 数 型 1 次 元 配 列 名	作業用として用られる。大きさは X0(KK), X1(KK), X2(KK), $KK \geq N$ 。

2.2 精 度

関数の性質により異なる。

3. 計算方法

n 次元空間中のベクトルに沿って minimization を行なう。

まず, このベクトルに沿って 2 点 (そのうち 1 点は search の開始点) をとる。新しくとられた点での関数値が開始点の関数値より大きければ, これと反対方向で同距離の点での関数値を求める。この点の関数値も, 開始点での関数値より大きければ, 第 1 の step (extrapolation)

は終る。

もし、開始点の関数値より小さい関数値が現れると、stepを2倍にして、新たな点での関数値を求める。これが前の点より大きい関数値をもてば、extrapolationを終る。逆の場合には、さらにstepを2倍にしてすすむ。これをくり返し行なって、図1のような関係をもつ3点 a , b , c を求める。

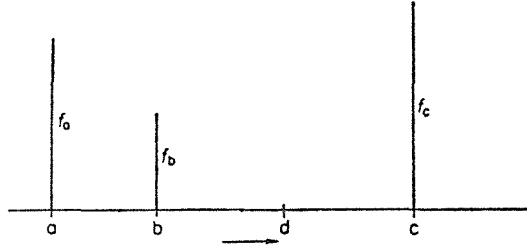


図 1

$$f_a > f_b, \quad f_b < f_c$$

次に b , c の中央の点 d で関数値 f_d を計算して

$$f_a < f_d \quad \text{ならば} \quad \text{区間 } bc$$

$$f_d \geq f_b \quad \text{ならば} \quad \text{区間 } ad$$

を考える。

次に内挿を行なう。これは関数を2次式で近似して微係数がゼロになる所を計算し、区間長を減少させて行く。

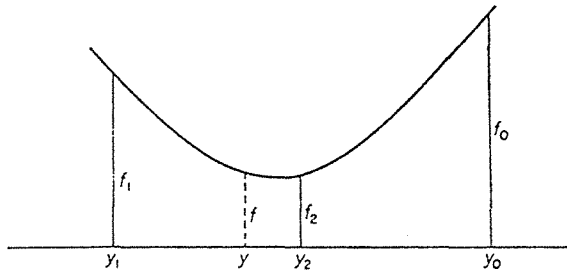


図 2

とするとき、

$$y - y_2 = \frac{\frac{f_0 - f_2}{y_0 - y_2}(y_1 - y_2) - \frac{f_1 - f_2}{y_1 - y_2}(y_0 - y_2)}{2 \left\{ \frac{f_0 - f_2}{y_0 - y_2} - \frac{f_1 - f_2}{y_1 - y_2} \right\}}$$

で、微係数がゼロになる点の新しい近似とし、ここで関数 f を計算する。

$$f > f_2 \quad \text{ならば} \quad \text{区間 } (y, y_0)$$

$f \leq f_2$ ならば 区間 (y_1, y_2)

を新しい区間とする。

この操作をくり返して行なう。

内挿の打ち切りは

1. 内挿の繰返し回数 (LIM=10回以上)
2. 内挿による変位の大きさ (EPST=EPS*0.01以上)

によって行なっている。

[備 考]

サブルーチン POW から LMINF を call することもある。

[参考文献]

W.H. Swann. Report on the development of a new direct searching method of optimization, C.I.
L. Research Note 64/3.

ライブラリ名 POW

MINIMIZATION OF FUNCTION WITHOUT USING DERIVATIVE

関数の極小化（微係数を使用しない方法）

作成者：星野 聡

1. 概 要

関数の極小を求める。関数の偏微係数は必要としない。unconstrained の場合である。

n 次元空間内の点 X から出発して linear search を行ない、より小さい関数値を見出して return する。この subroutine POW をくりかえし call することによって極小を search することができる。linear search の方向は、このサブルーチンの内部で計算される。また、linear search 自身はサブルーチンを内部で call することによって行なわれる。

2. 使用法

2.1 呼出し手順

CALL POW(TEST, N, X, F, YS, EPS, STEP, INIT, IER, NN, RITA, XS, FN, Y,
X0, X1, X2, DD)

引 数	型 と 種 類	内 容
TEST	サブルーチン名	<p>極小化を行なう関数を定義するサブルーチン名である。 このサブルーチンは、 SUBROUTINE TEST (N, X, F, INIT) の形をもつこと。ここで</p> <div style="border-left: 1px solid black; border-right: 1px solid black; padding: 0 10px;"> <p>N : X の次元数、整数型。 X : 関数 F に対する位置ベクトル、1 次元実数型配列名、 大きさは $X(K)$, $K \geq N$。 F : 実数型変数名。関数値を入れること。 INIT : INIT の項、参照。</p> </div> <p>④ TEST は呼ぶ側のプログラムで EXTERNAL 宣言をする必要がある。</p>
N	整 数 型	極小化を行なう関数の次元数。入力、保存される。
X	実 数 型 1 次元配列名	N 次元の位置ベクトル。 最初に、極小化を始める初期値ベクトルの値を set すること。POW から return すると 1 回の linear search 後の位置が X に入る。大きさは $X(K)$, $K \geq N$ 。 入力、出力。
F	実数型変数名	極小化しようとしている関数の値。入力、出力。 POW から return すると linear search 後の関数値が入る。
YS	実 数 型 1 次元配列名	linear search の方向のベクトル。出力。 サブルーチン POW の内部で計算されるものである。 大きさは $YS(K)$, $K \geq N$ 。

引 数	型 と 種 類	内 容
EPS	実 数 型	1 回の full powell step を行なった結果生じた displacement が EPS 以下になると converge したものと判定する。 このとき IER に 1 を set して return する。 EPS はまた linear search 自身を行なうサブルーチン LMINF において、内挿の打ち切りを行なうときの条件としても用いられる。これについては、LMINF の説明を参照のこと。この値の選び方は、関数の性質により異なるが、テストにおいては、単精度の場合、 10^{-6} を、倍精度の場合には 10^{-12} を使用した。入力、保存される。
STEP	実数型変数名	サブルーチン LMINF による linear search で用いられる変数である。 (LMINF 参照) また STEP は POW によって行なわれる最初の search の大きさにひとしい。
INIT	整 数 型	サブルーチン LMINF の実引数となる (LMINF 参照)。 最初 POW を call する前に INIT をゼロにセットしておく必要がある。
IER	整数型変数名	return codition を示す。出力。 IER=1 : converge したことを示す。 すなわち、linear search による displacement が EPS 以下になった場合である。 JER=0 : まだ converge していないことを示す。 つづいて POW を call すればよい。 IER=-1 : サブルーチン LMINF で zero division が起ったことを示す。 通常、充分、解に近いことが多い。 IER=-2 : サブルーチン LMINF で内挿の打ち切りが起ったことを示す。 (LMINF 参照) POW の計算はそのまま続けられよい。
NN	整 数 型	配列 RITA の DIMENSION の第 1 添字。入力、保存される。 $NN \geq N$ 。
RITA	実 数 型 (2 次元配列名)	search の方向 YS の記憶に使用される実数型配列。作業用として用いられる。 大きさは $RITA(NN, KK)$, $KK \geq N$ 。
XS, Y, X0, X1, X2, DD	実 数 型 1 次元配列名	いずれも、作業用として用いられる一次元配列である。 大きさは各々 N 以上であること。
FN	実 数 型 1 次元配列名	作業用として用いられる一次元配列である。大きさは $FN(K)$, $K \geq N+1$

2.2 精 度

関数の性質によって異なる。

2.3 使用ルーチン

サブルーチン LMINF, および極小化しようとする関数を定義するサブルーチンが必要である。

2.4 備 考

収束したと判定した場合でも、確かに所要の極小点であるかどうかは、利用者が判定する必要がある。

3. 計算方法

POWELL の方法により、関数の極小を求める。すなわち、一次独立な方向 $\xi_1, \xi_2, \dots, \xi_n$ と点 p_0 が与えられたとき

- (1) $r=1, \dots, n$ に対して、 $f(p_{r-1} + \lambda_r \xi_r)$ が ξ_r 方向に対して極小になるように λ_r をきめ、

$$p_r = p_{r-1} + \lambda_r \xi_r$$

とする。

- (2) $\{f(p_{m-1}) - f(p_m)\}$ が最大であるような m を見出す。

ただし、 $m=1, 2, \dots, n$ である。

$$\Delta = f(p_{m-1}) - f(p_m) \text{ と定義する。}$$

- (3) $f_3 = f(2p_n - p_0)$ を計算する。

$$f_1 = f(p_0), f_2 = f(p_n) \text{ と定義する。}$$

- (4) $f_3 \geq f_1$ と

$$(f_1 - 2f_2 + f_3) \cdot (f_1 - f_2 - \Delta)^2 \geq \frac{1}{2} \Delta (f_1 - f_3)^2$$

の、少はくとも一方が成り立てば、前回の方向 $\xi_1, \xi_2, \dots, \xi_n$ を次の iteration で再び用いる。また、 p_n は次の iteration における p_0 とする。

その他の場合には、

- (5) $\xi = (p_n - p_0)$ と定義し、 λ を $f(p_n + \lambda \xi)$ が最小なるようにきめる。次の iteration では $\xi_1, \xi_2, \dots, \xi_{m-1}, \xi_{m+1}, \dots, \xi_n$ を新しい search の方向とし、また、 $p_n + \lambda \xi$ を次回の iteration での starting point とする。

[参考文献]

M. J. D. Powell, An efficient methods for finding the minimum of a function of several variables without calculating derivatives, Computer Journal, Vol. 7, (1964)

ライブラリ名 LMIN

LINEAR MINIMIZATION USING DERIVATIVE

直線上の極小化（微係数を使用）

作成者：星野 聡

1. 概 要

n 次元空間中のベクトルに沿って関数の極小化を行なう。

関数計算回数を節約するために、内挿においてより小さい関数値を見出せば return する。

すなわち、 n 次元空間中の一点 X よりベクトル YS 方向（関数が減少する向き）に linear search を行ない、より小さい関数値 F を与える点 X を求める。

2. 使用法

2.1 呼出し手順

CALL LMIN(TEST, N, X, F, G, YS, EPS, IER, INIT)

引 数	型 と 種 類	内 容
TEST	サブルーチン名	<p>極小化を行なう関数を定義するサブルーチン名。 このサブルーチンは SUBROUTINE TEST(N, X, F, G, INIT) の形をもつこと。ここで</p> <p>N : XおよびGの次元数。入力、保存される。 X : 関数Fに対する位置ベクトル。実数型1次元配列名。 大きさはX(KK), $KK \geq N$。 F : 実数型変数名。関数値を入れること。 G : 関数Fに対する gradient ベクトル。実数型1次元配列名。 大きさはG(KK), $KK \geq N$。 INIT: 極小化を行なわせるとき、最初にゼロに set する必要がある整数型変数である。すなわち、サブルーチン TEST では、INIT ≠ 0 の場合には、関数の計算のみ行なうが、INIT = 0 ならば、この他に必要な initialization も行なうようにしておく必要がある。 TEST は call する側で EXTERNAL 宣言をする必要がある。</p>
N	整 数 型	極小化を行なう関数の次元数。入力、保存される。
X	実 数 型 1 次 元 配 列 名	<p>N次元の位置ベクトル。 linear search の開始点の位置ベクトルを set して LMIN に入る。 return の際は、linear search の結果として求められた位置ベクトルが入っている。 大きさはX(KK), $KK \geq N$。</p>
F	実 数 型 変 数 名	<p>極小化しようとしている関数の値。入力、出力。 linear search の開始点における関数値を set しておいて、LMIN を call すること。return の際には linear search の結果として求められた位置における関数値が入っている。</p>

引 数	型 と 種 類	内 容
G	実 数 型 1 次 元 配 列 名	極小化しようとしている関数の gradient の値。入力、出力。 大きさは G(KK), KK \geq N。 linear search の開始点における関数の gradient を set しておいて LMIN を call すること。 return の際には, linear search の結果として求められた位置における関数の gradient の値が入っている。
YS	実 数 型 1 次 元 配 列 名	linear search の方向のベクトル。 サブルーチン LMIN を call する際に set する必要がある。 大きさは YS(KK), KK \geq N。 YS の長さは, このベクトルの方向で search の開始点から極小点までの距離の推定値にひとしくしておくこと。 ただし, LMIN で YS の長さをかえることがある。
EPS	実 数 型	内挿を行なって生じた変位の大きさが, EPS*0.01 以下になれば内挿を打ち切って return する。このとき IER を -4 に set する。 EPS の選び方は, 関数の性質により異なるが, テストにおいては, 単精度の場合 10 ⁻⁶ を, 倍精度の場合には 10 ⁻¹² を選んだ。入力, 保存される。
IER	整 数 型 変 数 名	return condition を示す。出力。 IER=-2 : search の方向の gradient 成分がゼロまたは正のときに set される。ただちに return する。 IER=-3 : 内挿で zero division が発生したときに set される。極小点に近い場合が多い。 IER=-4 : linear search の displacement が EPS*0.01 以下になったときに set される。 linear search の開始点が極小点に充分近い場合である。 IER=0 : 上記以外のとき。
INIT	整 数 型 変 数 名	LMIN 内でのサブルーチン TEST の call において, 引数として用いられる。

2.2 精 度

関数の性質によって異なる。

3. 計算方法

search vector YS の方向に関数値 f が減少していることを確かめてから, 外挿を行なって, 関数の極小が, 区間内にはさまれるようにする。すなわち, より大きい関数値が現われるか, 正の gradient (YS 方向の成分) が現われるまでくり返す。

f を linear search の開始点での関数値, g_b をこの点での gradient の search 方向の成分とする。initial search vector は $\eta \cdot \text{YS}(I)$ で与えられる。ただし,

$$\eta = - \frac{2f}{g_b \parallel \text{YS} \parallel}$$

この η が 1.0 をこえるときは, $\eta = 1.0$ とする。この search vector の長さは, 外挿を行な

うたびに4倍される。

次に cubic interpolationを行なって、区間の両端の関数値よりもより小さい関数値をうるまで内挿をくりかえす。

すなわち、点 $|x\rangle$ より、方向 $|s\rangle$ にそって極小を求める。いいかえれば、関数 f が極小になる $|x\rangle + \alpha |s\rangle$ をきめる。このために一点

$$|y\rangle = |x\rangle + \lambda |s\rangle$$

の情報を用いる。すなわち、 x, y 点での関数値と gradient を $f_x, |g_x\rangle, f_y, |g_y\rangle$ とするとき、

$$\frac{\alpha}{\lambda} = 1 - \frac{\langle g_y | s \rangle + w - z}{\langle g_y | s \rangle - \langle g_x | s \rangle + 2w}$$

ここで、

$$w = (z^2 - \langle g_x | s \rangle \langle g_y | s \rangle)^{1/2}$$

$$z = \frac{3}{\lambda} (f_x - f_y) + \langle g_x | s \rangle + \langle g_y | s \rangle$$

〔参考文献〕

R. Fletcher, M. J. D. Powell, A rapidly convergent method for minimization, The Computer Journal Vol. 6 (1963)

ライブラリ名 DAVID

MINIMIZATION OF FUNCTION USING DERIVATIVE

1. 概 要

非負の関数の極小を求める。関数の偏微係数を必要とする。unconstrained の場合である。

n 次元空間の点 X から出発して linear search を行ない、より小さい関数値を見出して return する。このサブルーチン DAVID をくり返し call することによって、極小を search することができる。linear search の方向は、このサブルーチンの内部で計算される。また linear search 自身はサブルーチン LMIN を内部で call することによって行なわれる。

2. 使用法

2.1 呼出し手順

CALL DAVID (TEST, N, X, F, G, Y, YS, H, NN, EPS, INIT, IER, SIGMA,
TEMP1, TEMP2, TEMP3, DIAG,)

引 数	型 と 種 類	内 容
TEST	サブルーチン名	極小化を行なう関数を定義するサブルーチン名である (C7/LMIN の TEST の項参照)。
N	整 数 型	極小化を行なう関数の次元数。入力, 保存される。
X	実 数 型 1 次元配列名	N次元の位置ベクトル。入力, 出力。 最初は極小化を行なわせる初期値ベクトルの値を set すること。 DAVID から return すると, 1 回の linear search 後の位置が X に入る。 大きさは X(KK), $KK \geq N$ 。
F	実数型変数名	極小化しようとしている関数の値。入力, 出力。 DAVID から return すると linear search 後の関数値が入る。
G	実 数 型 1 次元配列名	極小化しようとしている関数の gradient ベクトル。出力。 DAVID から return すると, linear search 後の gradient の値が入る。 大きさは G(KK), $KK \geq N$ 。
Y	実 数 型 1 次元配列名	linear search の前後における gradient の差が入れられる。出力。 大きさは Y(KK), $KK \geq N$ 。
YS	実 数 型 1 次元配列名	linear search の方向のベクトル。 サブルーチン DAVID の内部で計算されるものである。 大きさは YS(KK), $KK \geq N$ 。出力。
H	実 数 型 2 次元配列名	Jacobian の逆行列が入れられる。出力。 大きさは H(NN, KK), $KK \geq N$ 。
NN	整 数 型	配列 H の DIMENSION の第 1 添字を与える。入力, 保存される。 $NN \geq N$ 。

引数	型と種類	内 容
EPS	実数型	linear searchを行なった結果生じた displacement が EPS 以下になると収束したものと判定する。このとき、IER=1 に set して return する。 EPS は、また linear search 自身を行なうサブルーチン LMIN において、内挿の打ち切りを行なうときの条件として用いられる (C7/LMIN を参照のこと)。この値の選び方は、関数の性質により異なるが、テストにおいては、単精度の場合 10^{-6} を、倍精度の場合は 10^{-12} を使用した。入力、保存される。
INIT	整数型変数名	最初 DAVID を call する前に INIT をゼロに set する必要がある。DAVID では、INIT にゼロでない値を set する。入力、出力。 INIT は DAVID の外部で以後、変化させてはならない。
IER	整数型変数名	return condition を示す。出力。 IER=1 : 収束したことを示す。 IER=0 : 収束していないことを示す。 つづいて DAVID を call すればよい。 IER=-1 : 行列 H の reset がひきつづいて発生したときである。充分解に近いことが多い。 これは次の場合におこる。 (1) $\langle \sigma y \rangle \leq 0$ (2) $\langle y H y \rangle \leq 0$ 行列 H は、対角線上が DIAG にひとしく、それ以外の要素はすべてゼロにひとしい行列に reset して、linear search を再開する。 IER=-2 : サブルーチン LMIN において、search の方向の gradient 成分がゼロまたは正のときに set されるが、DAVID と共に用いる場合には発生しない。 IER=-3 : サブルーチン LMIN において、内挿で zero division が発生したときに set される。 解に充分近い場合が多い。 IER=-4 : LMIN で linear search による変位が $EPS * 0.01$ 以下になったとき set される。 DAVID についてはそのまま計算をすすめればよい。 INIT=0 の状態で最初に DAVID を call すると、IER=0 と initialize される。
SIGMA TEMP1 TEMP2 TEMP3	実数型 1次元配列名	作業用として用いられる。 大きさは各々 N 以上であること。
DIAG	実数型	配列 H は、最初 DAVID が call されるとき (INIT=0 のとき) に、対角要素は、すべて DIAG にひとしく、その他の要素は、ゼロにひとしい行列に set される。通常、DIAG=1.0 としておけばよい。入力、保存される。

2.2 精 度

関数の性質によって異なる。

2.3 使用ルーチン

サブルーチン LMIN および極小化しようとする関数を定義するサブルーチンが必要である。

3. 計算方法

DAVIDON の方法により関数の極小を求める。

すなわち、Jacobian の逆行列の近似行列列 H を iterative に求め、これを用いて Newton 近似を行なう。

Jacobian の逆行列の近似のよりよいに求め、これを用 近似 H_{i+1} は

$$H_{i+1} = H_i + \frac{|\sigma_i\rangle\langle\sigma_i|}{\langle\sigma_i|\sigma_i\rangle} - \frac{H_i|y_i\rangle\langle y_i|H_i}{\langle y_i|H_i|y_i\rangle} \dots\dots\dots(1)$$

により求められる。ここで、

H_i : Jacobian の逆行列の第 i 近似

$|\sigma_i\rangle$: 前の linear search による変位ベクトル

$|y_i\rangle := |g_{i+1}\rangle - |g_i\rangle$

(現在の位置での gradient と、linear search $|\sigma_i\rangle$ を行なわなかった以前の点での gradient の差)

新しい search は

$$|\sigma_{i+1}\rangle = -\alpha_{i+1}H_{i+1}|g_{i+1}\rangle \dots\dots\dots(2)$$

で与えられる。 α_{i+1} は、 $-H_{i+1}|g_{i+1}\rangle$ 方向に linear search を行なって極小を求めた結果で決まる係数である。

なこの process によりえられる search $|\sigma_i\rangle$ は、もし関数 f が位置 $X=(x_1, \dots, x_n)$ の quadratic function ならば互いに conjugate になる。

行列 H は最初は、対角線上がすべて一定数 $\text{DIAG}(>0)$ で、その他の要素がゼロになる行列に set される。(1)式により、行列 H_{i+1} は positive definite である。しかし、演算は有限の桁数で行なわれるために $\langle\sigma_i|\sigma_i\rangle$ と $\langle y_i|H_i|y_i\rangle$ が必ずしも正にはならないことがある。この場合には H_i は、上記の行列 (対角線上が DIAG , その他はゼロ) に reset してから、linear search をつづける。

〔参考文献〕

- (1) W. C. Davidon, Variable metric method for minimization, A. E. C Research and Development Report, ANL-5990 (1959)
- (2) R. Fletcher & M. J. D. Powell, A rapidly convergent descent method for minimization, The Computer Journal, Vol. 6 (1963)