多変数関数の勾配の計算方法

岩田憲和　伊理正夫
（東京大学　工学部）

1. はじめに

関数の計算手続きが与えられたとき，ある点におけるその関数の勾配の値を計算するには，数式処理的に式で微分してから計算するか，あるいは，いくつかの関数値を計算してそれらを用いて数値微分するかの，どちらかが普通の方法であろう．前者の方法では，各偏導関数値の計算の際，同じ計算を繰り返し行う場合が多く，無駄な感じがする．また，関数の計算手続きに，FORTRAN プログラムにおけるGO TO 文，DO文，IF文などの制御機能をもつ部分が含まれていると，数式処理的な微分は面倒である．後者の数値微分による方法では，打ち切り誤差と丸め誤差の取り引き，という困難があり，また，$y=\exp(x)$ の微分は y に等しいというような事が利用できない．

ところで，「関数の計算手続きを，計算グラフ（computational graph）として表現したとき，その勾配の計算手続きが，計算グラフ上に簡単に表現できることを利用し，関数の値と，その勾配の値とを並行して計算する」ならば，次のような利点をもつ計算手続きが得られる．

(1) 数式処理によって予め各偏導関数を求めてから勾配を計算すると同じ計算を繰り返し行う場合があるが，そのような無駄が省ける．
(2) 関数の計算手続きに，FORTRAN プログラムにおけるGO TO 文，DO文，IF文のような制御機能をもつ部分が含まれていてもよい．
(3) 四則，べき，exp，sin等の基本演算の微分は数式処理的厳密さをもつ
(4) 計算の手間は，勾配と関数とを並行して計算しても，理論的には，関数値の計算の手間の7倍以下で済む．この比は，関数の主変数の数だけでなく関数に依存しない．
(5) 関数計算と並行して関数値計算の際の丸め誤差の限界の推定ができる．

これらの利点をもつ勾配の計算方法を用いて，関数の計算手続きが，FORTRANの関数副プログラムとして与えられたとき，そのプログラムを勾配を並行して計算する関数副プログラムに書き換えるパッケージ PADRE（PArtial Derivatives and Rounding Errors）を製作した．その概要について報告する．

2. 勾配の計算方法[1]

2.1 計算グラフ[2]の定義

関数 $f(x_1,x_2,\cdots,x_n)$ の計算過程を表す有向グラフを計算グラフと言う．x_i，f および中間変数 v_k を節点とする無サイクル（acyclic）グラフで各節点には基本演算が付随している．

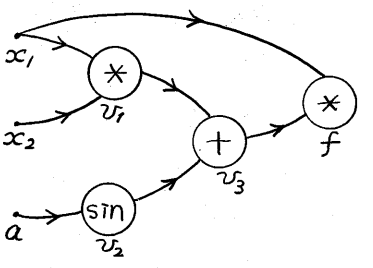

図1　$f=x_1*(x_1*x_2+\sin(a))$の計算グラフ

2.2 要素的偏導関数 (Elementary Partial Derivatives)

$x_1, x_2, \cdots, x_n$ を主変数とする関数 f の計算過程は，$\varphi_1, \varphi_2, \cdots, \varphi_m$ を基本演算として，中間変数 $v_1, v_2, \cdots, v_m$ を順々に計算していく過程として図2のように表せる．ここで各 u_{ij} は次のいずれかである．

(1) $u_{ij} = x_l$ （l は $1, 2, \cdots, n$ のいずれか）
(2) $u_{ij} = v_k$ （$k < i$）
(3) $u_{ij} \in K$ （定数；K は基礎体）

そして，通常 $f = v_m$ であり，基本演算は四則，べき，sin, cos, exp, log 等で二項または単項の演算子である．このとき，中間変数の計算式

$$v_1 = \varphi_1(u_{11}, \cdots, u_{1l_1})$$
$$v_2 = \varphi_2(u_{21}, \cdots, u_{2l_2})$$
$$\vdots$$
$$v_m = \varphi_m(u_{m1}, \cdots, u_{ml_m})$$

図2 計算過程

$$v_i = \varphi_i(u_{i1}, u_{i2}) \quad \text{または} \quad v_i = \varphi_i(u_{i1}) \tag{1}$$

に対して

$$\partial v_i / \partial u_{ij} \qquad (j = 1, 2 \text{ または } j = 1) \tag{2}$$

を要素的偏導関数と呼ぶ．計算過程の各段階 i について要素的偏導関数の値を求めることは，基本演算が，四則，べき，exp, log 等であることから，中間変数 v_j $(j \leq i)$ の値を利用すると，手間はあまりかからない．「中間結果の利用」という点が，この勾配の計算法の基本である．

2.3 勾配の計算

図2の計算過程において df は

$$df = \sum \frac{\partial f}{\partial v_{k_1}} \cdot \frac{\partial v_{k_1}}{\partial v_{k_2}} \cdot \cdots \cdot \frac{\partial v_{k_l}}{\partial x_j} \, dx_j \tag{3}$$

という形に書ける $(k_1 > k_2 > \cdots > k_l)$．各 $\partial v_{k_i} / \partial v_{k_{i+1}}$ は要素的偏導関数である．計算グラフ上で，$v_{k_{i+1}}, v_{k_i}$ を結ぶ枝に $\partial v_{k_i} / \partial v_{k_{i+1}}$ の値を付随させると，例えば，

$$f = (-x_2 + \sqrt{x_2 **2 - 4 * x_3 * x_1}) / (2 * x_1) \tag{4}$$

の場合には図3のように表せる．

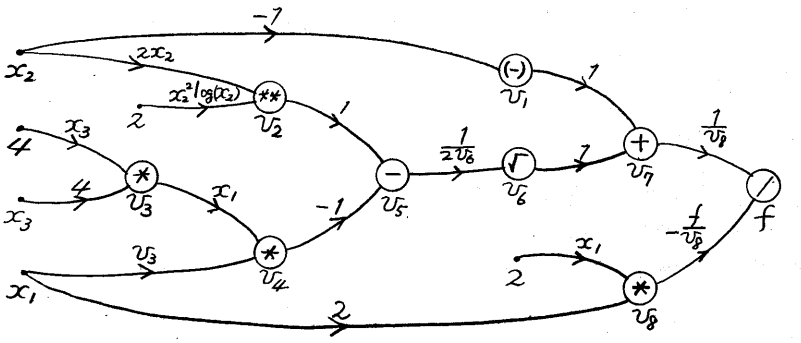

図3 $f = (-x_2 + \sqrt{x_2 **2 - 4 * x_3 * x_1}) / (2 * x_1)$ の計算グラフとその要素的偏導関数

式(3)のように df が表せることから，式(4)の df は図3の要素的偏導関数から

$$df = \frac{1}{v_8} \times (1 \times (-1)\, dx_2 + 1 \times \frac{1}{2v_6} \times (1 \times (2x_2)\, dx_2 + (-1) \times (x_1 \times 4\, dx_3 + v_3\, dx_1))) + (-\frac{f}{v_8}) \times 2\, dx_1 \qquad (5)$$

と表せる．すなわち，計算グラフを f の側から逆にたどりながら，各枝に付随している要素的偏導関数の値を乗じた値を各節点に累積していく．その結果 x_j の各節点に残された値から $\partial f/\partial x_j$ の値が求まるのである．

2.4 関数値が含む丸め誤差の限界の推定

前節で述べた計算法により，各中間変数 v_i についての導関数の値 $\partial f/\partial v_i$ が求まることがわかる．この値を利用して丸め誤差の限界の推定ができる．

$$v = \varphi(u_1, u_2) \quad \text{または} \quad v = \varphi(u_1) \qquad (6)$$

という基本演算の際発生する丸め誤差を δv，v に含まれる誤差を Δv，いわゆる machine-epsilon を ε ($=16^{-6}, 16^{-14}, 2^{-24}, 2^{-56}$ あるいはそれぞれの1/2)とする．ここで，

$$|\delta v| \leqq \varepsilon \cdot |v| \qquad (7)$$

という仮定が許されるとし，すなわち，基本演算は最終ビットまで正しく計算されるとする．さらに丸め誤差の発生，伝播が

$$\Delta v = \delta v + (\partial\varphi/\partial u_1)\cdot\Delta u_1 + (\partial\varphi/\partial u_2)\cdot\Delta u_2 \quad \text{または}$$

$$\Delta v = \delta v + (\partial\varphi/\partial u_1)\cdot\Delta u_1 \qquad (8)$$

で記述できるとする．このとき式(7)，式(8)より，

$$|\Delta f| \leqq \varepsilon \cdot \left(\sum_{i=1}^{m} |\partial f/\partial v_i| \cdot |v_i| \right)$$

（m は計算過程の段階数） (9)

によって，関数値 f の計算過程で丸めによって生じた誤差 Δf の上界の推定ができる[3]．各中間変数の値 v_i，各中間変数に関する微分の値 $\partial f/\partial v_i$ は f の勾配の計算の際に求まる値であるから，Δf の上界の推定のために要する余分な手間は式(9)の $\sum$ の計算だけ，すなわち，f の計算過程の段階数の乗算と加算だけである．

2.5 計算方法のまとめ

ここで本節で述べた計算方法を整理しておくと次のようになる．

Stage1：中間結果を保存したまま関数値を求める．計算グラフ上ではx側からf側へ向かって無サイクルグラフの定める半順序に従って計算することになる．

Stage2：Stage1で保存されている各計算段階での中間結果を用いて要素的偏導関数値を計算する．

Stage3：計算グラフをf側からx側へ逆方向にたどりながら，各中間変数v_i（x_jもその特別な場合とする）に関する偏導関数値$\partial f/\partial v_i$を計算していく．

Stage4：Stage3の結果とStage1の結果とを利用して，式(9)の右辺：丸め誤差の上界，を計算する．

3. 勾配の計算の手間の評価

BaurとStrassenは，「多変数の有理式 $f(x_1,x_2,\cdots,x_n)$ の計算に要する四則演算の回数を$L(f)$，そのすべての偏導関数の値 $\partial f/\partial x_1, \partial f/\partial x_2, \cdots, \partial f/\partial x_n$ をfと同時に計算するのに要する演算の回数を $L(f, \partial f/\partial x_1, \partial f/\partial x_2, \cdots, \partial f/\partial x_n)$ とすると

$$L(f) \leqq L(f, \partial f/\partial x_1, \partial f/\partial x_2, \cdots, \partial f/\partial x_n) \leqq c \cdot L(f) \quad (10)$$

である；ここでcはfによらない定数で，演算回数の定義にもよるが$c = 3 \sim 7$ぐらいである」と述べている[4]．ここでは，2章で述べた勾配の計算方法に関して最も単純な方法で手間の評価を行う．

関数値fを計算する際の加算，減算，符号反転，乗算，除算，その他の基本演算の回数をそれぞれ，a, s, m, t, d, e回とする．§2.5「計算法のまとめ」の各Stageでの演算回数は次のようになる．

Stage1　$a+s+m+t+d+e = L(f)$　(11)

Stage2　要素的偏導関数値の計算は加算，減算，符号反転，乗算に関しては全く手間がいらない．除算に関しては，1回の符号反転と，2回の除算が必要となる．その他の基本演算に関してはその基本演算と高々同程度の手間と考えてよい．Stage2での演算回数は

$$3d + e \quad (12)$$

Stage3　各中間変数v_iに関する偏導関数値$\partial f/\partial v_i$の計算では加法の際の1倍などを実行しないものとする．また四則以外の基本演算は単項演算子と考えると$\partial f/\partial v_i$（$\partial f/\partial x_j$も含む）を計算する演算回数は

$$a+2s+m+3t+3d+e \quad (13)$$

以下である[1]．

$u=a+b$	$du = da + db$
$u=a-b$	$du = da + (-)db$
$u=(-)a$	$du=(-)da$
$u=a*b$	$du = b\,da + a\,db$
$u=a/b$	$du = 1/b\,da + (-)u/b\,db$
$u=a**b$	$du = bu/a\,da + u\log(a)db$
$u=\exp(a)$	$du = u\,da$
$u=\log(a)$	$du = 1/a\,da$
$u=\mathrm{sqrt}(a)$	$du = 1/(2a)da$
$u=\sin(a)$	$du = \cos(a)da$
$u=\cos(a)$	$du = (-)\sin(a)da$
$u=\tan(a)$	$du = (1+u^2)da$
$u=\sinh(a)$	$du = \cosh(a)da$
$u=\cosh(a)$	$du = \sinh(a)da$
$u=\tanh(a)$	$du = (1-u^2)da$
$u=\mathrm{abs}(a)$	$du = \begin{cases} da & (a>0) \\ 0 & (a=0) \\ (-)da & (a<0) \end{cases}$
$u=\max(a_1,a_2,\cdots,a_m)$	$du = \sum_{i=1}^{m} \varepsilon_i\, da_i$

図4　基本演算の微分

以上を加えると

$$L(f, \frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \cdots, \frac{\partial f}{\partial x_n}) \leqq L(f) + a + 2s + m + 3t + 6d + 2e$$

$$\leqq 7L(f) \qquad (14)$$

もし除算がなければ

$$L(f, \frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \cdots, \frac{\partial f}{\partial x_n}) \leqq 4L(f) \qquad (15)$$

になる.

式(16)より，関数値を計算する際，中間結果を保存しておくことにより，勾配の値と関数の値とを並行して計算しても演算回数は，関数値のみ計算するのに比べて7倍までしか増えないことが示された.

4. FORTRANで書かれた関数副プログラムへの適用

FORTRAN言語で書かれた関数副プログラムが与えられたとき，関数値だけでなく，勾配の値も並行して計算する副プログラムに書き換えるパッケージPADREについて述べる.

4.1 PADREの機能

勾配を並行して計算するためにすべき事は，

(1) 中間結果を保存する.
(2) 計算グラフを作成する.
(3) 計算グラフをたどり勾配の値を求める.

の三項目である．それらの実現方法を述べる.

(1) 中間結果を保存する必要があるのは，左辺が実数型の算術代入文だけである．図5のように中間結果が実数型の場合にはZxxxx，そうでない場合にはNxxxxという名前の変数に中間結果を保存するようプログラムを書き換える．ただしREAD文も代入文の一種とみなす.

(2) 中間結果を保存するよう書き換えた関数副プログラムに対して，左辺が実数型である代入文の実行の直後にASSIGNという名前のサブルーチーンの引用が起こるように書き換える．サブルーチーンASSIGNは計算グラフを作成する機能をもちPADREによって与えられる．サブルーチーンASSIGN

```
F=X(1)+X(2)*SIN(X(3))/(I*J)
```

⇩

```
Z0001=SIN(X(3))
Z0002=X(2)*Z0001
N0001=I*J
Z0003=Z0002/N0001
F=X(1)+Z0003
```

図5 中間結果の保存

```
      DO 100 I=1,3
100   S=S+X(I)
```

⇩

```
      DO 100 I=1,3
      S=S+X(I)
      CALL ASSIGN
     & 'S',0,S,
     & '+',
     & 'S',0,S,
     & 'X',I,X(I))
100   CONTINUE
```

図6 計算グラフの作成

の引数は順に，代入文の左辺の英字名，左辺の添字（左辺が変数名の場合は0），左辺の値（左辺そのもの），右辺の基本演算子，右辺の第一オペランドの英字名，第一オペランドの添字，第一オペランドの値，第二オペランドの英字名，第二オペランドの添字，第二オペランドの値である．基本演算子が単項の場合は第二オペランドに関する最後の三つの実引数は，それぞれ，0, 0, 0.0 とする．サブルーチンASSIGNが計算グラフを作成する記憶領域を仮にヒストリーと名づける．ヒストリーは，左辺の英字名，左辺の添字の値，左辺の値，右辺の演算子，右辺のオペランドのヒストリー上での場所，左辺の参照回数を保持し計算グラフを表現する．左辺の参照回数は，左辺が計算グラフ上で対応する節点の，出次数である．ASSIGNが関数副プログラム中の代入文の直後に実行されると，その代入文の両辺の情報がヒストリーに残る．従って

	左辺	添字	値	演算子	右辺1	右辺2	参照回数
#9							
#8							
#7	'S'	0	$\Delta+x_1x_2$	'+'	#6	#3	0
#6	'S'	0	$\Delta+x_1x_2$	'+'	#5	#2	1
#5	'S'	0	$\Delta+x_1$	'+'	#4	#1	1
#4	'S'	0	Δ				1
#3	'X'	3	x_3				1
#2	'X'	2	x_2				1
#1	'X'	1	x_1				1

図7　計算グラフの内部表現

$$S = S + X(I) \qquad (16)$$

の左辺のSと右辺のSとは計算グラフ上では別の節点であるように，別の変数，すなわち，左辺のSは新しい別の変数として扱う．そのため，関数副プログラムの中で実数型の変数，配列要素への代入が実行されるごとに，原則としてヒストリーは高さを増す．

(3) 計算グラフをたどって勾配の値を求めるために，実行時において関数副プログラムでRETURN文の実行が起こる直前にNABLAという名前のサブルーチーンの引用が起こるように書き換える．サブルーチーンNABLAはサブルーチーンASSIGNが作成した計算グラフを関数副プログラムの関数名から出発して，要素的偏導関数値を計算しながらたどり，勾配の値，オプションとして丸め誤差の限界の値の計算を行う．この際，参照回数が2以上の中間変数ひについては $\partial f/\partial$ひの値がきまるまで，一時ひから先へたどることを保留する．このサブルーチーンNABLAはPADREによってもたらされる．

PADREに与えられたFORTRAN関数副プログラムのテキストの書き換えは，UTILISP[5]で書かれたプログラムが受けもつ．UTILISPのプログラムは，関数副プログラムのテキストのFORTRANの一文を読むたびに，書き換えた文を出力し，上述の(1), (2), (3)の処理と英字名のコード化をone passで行い，作業用記憶領域を必要としない．その書き換えた結果に対し，記憶領域の初期化を行うINITという名のサブルーチーンとASSIGN；NABLAとをつけ加えたテキストをPADREは出力する．このテキストを他のプログラム単位とリンクすれば，実行時には関数値と並行して勾配の値，丸め誤差の限界の値が計算できる．

4.2 PADREに入力できる関数副プログラムの仕様

(1) JIS C6201-1982 「電子計算機プログラム言語FORTRAN」[6]の基本水準(subset language)を満足していなければならない．ただし，現在のところ文関数とEQUIVALENCE 文の機能のみ処理できていない．

(2) コメント文，空文でない第一文は

[REAL] FUNCTION $f(m, x, grad\left\{\begin{array}{l}\ \\ ,rnderr)\\ ,rnderr,\cdots)\end{array}\right\}$

でなければならない．ここでfは関数名，xはfの主変数で配列名，$grad$は勾配の値を副作用として返す配列名，mは配列名x，$grad$の配列要素の個数，$rnderr$はオプションでfの値の丸め誤差の限界でmachine-epsilonを乗じてない値を副作用として返す変数名である．その他の引数はいくつあってもよい．

4.3 PADREが出力したプログラムの実行結果についての注意

(1) 基本演算の中間結果は，計算機のレジスタ上に倍精度で保持される場合が多いが，PADRE を使って得た計算結果はすべての中間結果を単精度でしか保持していないので，その差による違いが出ることがある．

(2) COMMON A(2), B(2) と宣言されているとき，B(0)はA(2)と解釈されるが，そのような記憶列による結合を利用しているプログラムに関しては結果は異なる．

(3) 副作用をもつ副プログラム(手続き)を引用した場合，関数値は正しいが，勾配の値は正しいとは限らないことがある．これは，PADREが処理した関数副プログラムの外で，「秘かに」代入が起こることに起因する．

5. 計算例

$$f(x)=\frac{\exp\left(-\sum_{i=1}^{N}\frac{(x_i-m_i)^2}{2\sigma_i^2}\right)}{(2\pi)^{\frac{N}{2}}\prod_{i=1}^{N}\sigma_i} \qquad (17)$$

に関して，いくつかのNの値についてPADREの出力したプログラム[付録]の計算時間などをテストした．

```
      FUNCTION F(N,X,GRAD,EPSLN,M,S)
      REAL X(N),GRAD(N),M(N),S(N)
      REAL DENOMI,SIGMA,PI2
      PI2=3.14159*2.0
      DENOMI=1.0
      DO 001 I=1,N
  001 DENOMI=DENOMI*S(I)
      DENOMI=DENOMI*(SQRT(PI2)**N)
      SIGMA=0.
      DO 002 I=1,N
  002 SIGMA=SIGMA+(X(I)-M(I))**2/(2.0*S(I)**2)
      F=EXP(-SIGMA)/DENOMI
      RETURN
      END
```

図8 関数副プログラム

5.1 計算時間

計算機システムは東大大型センターのHITAC M280-H，コンパイルは，VOS3の最適化FORTRANでNOOPT(最適化しない)，NOIAP(ベクトル命令を使わない)オプションで/ooo 回同じプログラムを連続して実行させ，それに要したCPU 時間を測定した．表1，表2は，それぞれ，関数値と勾配と丸め誤差，関数値と勾配，を1回計算するのに費した時間である．

表1　丸め誤差まで計算した際の計算時間　（単位 ms）

N	(1) PADRE の出力プログラム	(2) 図8 のプログラム	(3) (1)/(2) の比	(4) overhead	(5) {(1)-(4)}/(2) の比	(6) ASSIGN の引用回数	(7) ヒストリー の高さ	(8) NABLA	(9) (8)/(2) の比
2	0.640	0.0256	25.0	0.146	19.2	23	35	0.350	13.7
4	1.04	0.0342	30.4	0.266	22.6	37	55	0.579	16.9
8	1.81	0.0510	35.4	0.456	27.3	65	95	1.02	20.0
16	3.37	0.0833	40.4	0.842	30.3	121	175	1.92	23.1
32	6.41	0.149	43.2	1.61	32.3	233	335	3.69	24.8
64	13.7	0.280	49.0	3.06	38.1	457	655	7.27	25.9
128	25.4	0.541	46.9	6.19	35.5	905	1295	14.7	28.0

表2　勾配と関数だけ求めた際の計算時間　（単位 ms）

N	(1) PADRE の出力プログラム	(2) 図8 のプログラム	(3) (1)/(2) の比	(4) overhead	(5) {(1)-(4)}/(2) の比	(6) ASSIGN の引用回数	(7) ヒストリー の高さ	(8) NABLA	(9) (8)/(2) の比
2	0.436	0.0256	17.0	0.146	11.3	23	23	0.182	7.10
4	0.696	0.0342	20.4	0.266	12.6	37	35	0.297	8.70
8	1.23	0.0510	24.0	0.456	15.1	65	59	0.519	10.2
16	2.29	0.0833	27.5	0.842	17.4	121	107	0.980	11.8
32	4.39	0.149	29.6	1.61	18.7	233	203	1.89	12.7
64	8.62	0.280	30.8	3.06	19.8	457	395	3.72	13.3
128	17.2	0.541	31.7	6.19	20.4	905	779	7.37	14.0

Nは式(17)，図8のN．(1)/(2)の比とはPADREが出力したプログラムの実行時間と図8のプログラムの実行時間の比．overheadとはサブルーチンASSIGN，NABLAが引用される際のoverhead時間の総計である．(5)の{(1)-(4)}/(2)の比は，overhead時間を(1)のPADRE出力から除いた場合，すなわち，ASSIGN，NABLAをインライン展開した際，実現されるであろう計算時間の(2)に対する比．(8)のNABLAはサブルーチンNABLAでの計算時間．(8)/(2)の比はサブルーチンNABLAでの計算時間の(2)に対する比．

5.2　誤差の限界の推定

式(17)でN=5の場合の結果を述べる．

(1) $x_1=11.0$　$x_2=8.03$　$x_3=4.38$　$x_4=34.4$　$x_5=7.02$

$m_1=10.3$　$m_2=7.79$　$m_3=4.40$　$m_4=33.0$　$m_5=6.90$

$s_1=28.2$　$s_2=59.1$　$s_3=8.50$　$s_4=512.$　$s_5=15.9$

machine-epsilon を乗じない誤差の上界の値：0.28132509 7E-08

16^{-6} を乗じた推定値：0.00000016 76　　E-10

f の値（PADRE）　：0.87594445　　E-10

倍精度計算した f　：0.8759433973749631 9　　D-10

(2) $x_1=3.31$ $x_2=8.50$ $x_3=3.31$ $x_4=16.5$ $x_5=2.44$
$m_1=14.9$ $m_2=10.7$ $m_3=7.30$ $m_4=38.2$ $m_5=10.2$
$s_1=11.0$ $s_2=72.2$ $s_3=11.0$ $s_4=272.$ $s_5=8.65$

machine-epsilon を乗じない誤差の上界の値：0.9701373 93E-08
16^{-6} を乗じた推定値：0.000000 0578 E-09
fの値（PADRE）：0.1851221 〃 E-09
倍精度計算したf：0.1851218170924254 1 D-09

以上のように，これらの例に関しては，誤差の限界の推定は満足すべきものといえる．

6. おわりに

この勾配の計算方法は，コンパイラ機能の一部として実現することが可能である．コンパイル時にこの勾配の計算法に必要な計算過程に関する情報の多くが判明しているのである．コンパイラ機能の一部として実現すれば，表1，表2の各々の項目(3)，(9)の比の値はもっと小さくなるに違いない．また，この勾配を関数値と並行して計算する方法によれば，丸め誤差の自動解析が可能となるから，数値計算の結果の精度を調べる，言い換えると，数値計算の品質評価ができることになる．また，関数の値と同時に勾配の値も必要とするような数値計算上の問題は多数あるのでこの計算法の効果は大きいであろう．さらにHessian 等，高階の導関数も同様にして計算できるが，かなり面倒なものとなる．

本報告は研究の中間報告であって，さらに大幅な改良が期待できる点も多く，それらの改良の結果については，いずれまた報告したい．

参考文献

[1] 伊理正夫：Simultaneous Evaluation of Functions, Their Partial Derivatives and Rounding Errors. 研究集会"並列数値計算アルゴリズムとその周辺"，京都大学数理解析研究所，1983年11月24日．

[2] F.L. Bauer：Computational Graphs and Rounding Errors, SIAM Journal on Numerical Analysis, Vol. 11(1974), pp. 87-96.

[3] J.H. Wilkinson：Error Analysis of floating-Point Computation, Numerishe Mathematik, Bd. 2(1960), pp. 319-340.

[4] W. Baur and V. Strassen：The Complexty of Partial Derivatives (extended version, January 1982). unpublished note.

[5] T. Chikayama：UTILISP MANUAL, Technical Reports, METR 81-6, (September 1981), University of Tokyo.

[6] 日本規格協会編：JISハンドブック，Vol.9，情報処理，日本規格協会．

付録 図8のプログラムをPADREが書き換えた結果

```
      FUNCTION F(N, X, GRAD, EPSLN, M, S)
      REAL X(N), GRAD(N), M(N), S(N)
      REAL DENOMI, SIGMA, PI2
      PARAMETER(NH0000=2048)
      NH0001=NH0000-N
      NH0002=NH0001-N
      NH0003=NH0002-N
      NH0004=NH0003-N
      CALL INIT(NH0003, N)
      PI2=3.14159*2.0
      CALL ASSIGN(
     & 1, 0, PI2,
     & 4,
     & 2, 0, 3.14159,
     & 3, 0, 2.0)
      DENOMI=1.0
      CALL ASSIGN(
     & 4, 0, DENOMI,
     & 0,
     & 0, 0, 0.0,
     & 0, 0, 0.0)
      DO 50001 I=1, N
  001 DENOMI=DENOMI*S(I)
      CALL ASSIGN(
     & 4, 0, DENOMI,
     & 4,
     & 4, 0, DENOMI,
     & NH0001, I, S(I))
50001  CONTINUE
      Z0001=SQRT(PI2)
      CALL ASSIGN(
     & 5, 0, Z0001,
     & 12,
     & 1, 0, PI2,
     & 0, 0, 0.0)
      Z0002=Z0001**N
      CALL ASSIGN(
     & 6, 0, Z0002,
     & 6,
     & 5, 0, Z0001,
     & 7, 0, REAL(N))
      DENOMI=DENOMI*Z0002
      CALL ASSIGN(
     & 4, 0, DENOMI,
     & 4,
     & 4, 0, DENOMI,
     & 6, 0, Z0002)
      SIGMA=0.
      CALL ASSIGN(
     & 8, 0, SIGMA,
     & 0,
     & 0, 0, 0.0,
     & 0, 0, 0.0)
      DO 50002 I=1, N
  002 Z0003=X(I)-M(I)
      CALL ASSIGN(
     & 9, 0, Z0003,
     & 2,
     & NH0003, I, X(I),
     & NH0004, I, M(I))
      Z0004=Z0003**2
      CALL ASSIGN(
     & 10, 0, Z0004,
     & 6,
     & 9, 0, Z0003,
     & 11, 0, REAL(2))
      Z0005=S(I)**2
      CALL ASSIGN(
     & 12, 0, Z0005,
     & 6,
     & NH0001, I, S(I),
     & 13, 0, REAL(2))
      Z0006=2.0*Z0005
      CALL ASSIGN(
     & 14, 0, Z0006,
     & 4,
     & 15, 0, 2.0,
     & 12, 0, Z0005)
      Z0007=Z0004/Z0006
      CALL ASSIGN(
     & 16, 0, Z0007,
     & 5,
     & 10, 0, Z0004,
     & 14, 0, Z0006)
      SIGMA=SIGMA+Z0007
      CALL ASSIGN(
     & 8, 0, SIGMA,
     & 1,
     & 8, 0, SIGMA,
     & 16, 0, Z0007)
50002  CONTINUE
      Z0008=(-(SIGMA))
      CALL ASSIGN(
     & 17, 0, Z0008,
     & 3,
     & 8, 0, SIGMA,
     & 0, 0, 0.0)
      Z0009=EXP(Z0008)
      CALL ASSIGN(
     & 18, 0, Z0009,
     & 7,
     & 17, 0, Z0008,
     & 0, 0, 0.0)
      F=Z0009/DENOMI
      CALL ASSIGN(
     & 19, 0, F,
     & 5,
     & 18, 0, Z0009,
     & 4, 0, DENOMI)
      CALL NABLA(
     & N,
     & GRAD,
     & NH0003,
     & 19,
     & EPSLN)
      RETURN
      END
```