25周年記念論文

偏導関数計算と丸め誤差推定の自動化の大規模非線形方程式系への応用†

伊理正夫††　土谷　隆††　星　守†††

計算グラフを利用して関数の勾配を正確に効率良く計算し，同時に関数計算の丸め誤差評価を得ることができる新しい算法の性質を理論的ならびに実験的に解析し，①この算法によれば，従来の数値微分法などに比べ，10倍以上高速に方程式系の正確なヤコビ行列を求めることができること，②この算法は十分精密な丸め誤差評価を与え，手間の観点からも十分に実用的であること，を確認した．そして，本算法によって得られたヤコビ行列と丸め誤差の評価を使い，丸め誤差評価を考慮に入れた正規化ノルムによって収束判定を行うニュートン法が従来のやり方に比べて勝れていることを論じ，具体例で実証した．

1. はじめに

各種の最適化法，ニュートン法など，数値計算に際して，関数とその偏導関数の値を同時に必要とする場合は大変多い．しかし，従来大規模な問題の偏導関数計算には，普通数値微分が使われていた．また，数値計算全般において，丸め誤差を具体的に管理することは，重要であるとの認識はあったものの，理論としてはともかく[1),2)]技術として確立されているとは言い難い．

最近，グラフ理論的な考え方に基づいて，“正確な”偏導関数を求め，同時に関数計算時の丸め誤差を推定する新しい方法が提案され，具体的なプログラム化も試みられている[3)~5)]．L. B. Rall[6)]なども計算グラフ(Kantorovich グラフ)を用いて偏導関数計算を自動的に行う方法を論じているが，その方法は，偏導関数を求めるのに計算グラフを変数側から関数側へと辿るものである．これに対して，新しい方法は，中間結果を保存しながら計算グラフを関数側から変数側へと逆順に辿ることによって，次の二点に関して根本的な技術の改良を実現している：(1)一つの関数のすべての偏導関数が“関数自身を計算する手間”の定数倍の手間で求められる．〔数値微分や Rall などの方法の手間は，“(関数自身を計算する手間)×(入力変数の個数)”程度である．〕(2)従来の区間解析などによるものより，はるかに精密な誤差評価が得られる．一方，新しい方法は時間計算量と同程度の記憶領域を必要とするが，上記の長所を考えるならば，その実用価値は高いと言えよう．

本論文ではこの新しい方法を紹介し，実際に小規模なシステム(トランジスタの Ebers-Moll モデル)と大規模なシステム(蒸溜塔の DPS モデル)に適用して，計算時間，丸め誤差評価などの性能を実測し，かつ大規模な非線形方程式系のニュートン法による解法に応用した場合の効果を調べた結果を報告する．

2. 算　法

たとえば関数

$$f=f(x,y)=(yx+b)x+c$$

について，“入力変数” x, y の値を与えて“関数” f の値を計算する手順は，“中間変数” v_1, v_2, v_3 と“基本演算”(この場合，四則演算)を用いて**図-1** の①～④のような一つの“計算過程”として表される．(以下では入力変数と関数も中間変数に含めることがある．特に入力変数，関数であることを強調する時には，それぞれ記号 x, x_i あるいは f, f_i などを用いる．)

このように，一般に関数計算の過程は，中間変数 v_i と，基本演算 φ_i (四則演算や exp, log, … など)を用いて，“基本計算ステップ”

$$v_i=\varphi_i(v_{i_1},\cdots,v_{i_n})\ (i_1,\cdots,i_n<i) \qquad (1)$$

の列として書ける．中間変数 v_i を点とし，基本計算ステップ(1)式に対応して点 $v_{i_1},\cdots,v_{i_n}$ を点 v_i に枝で結ぶことによって，関数の計算過程を一つのグラフ

† “Automatic Computation of Partial Derivatives and Rounding Error Estimates with Applications to Large-scale Systems of Nonlinear Equations” by Masao IRI, Takashi TSUCHIYA, (Department of Mathematical Engineering and Instrumentation Physics, Faculty of Engineering, University of Tokyo) and Mamoru HOSHI (Chair of Information Processing Engineering, Faculty of Engineering, Chiba University).

†† 東京大学工学部計数工学科
††† 千葉大学工学部情報処理工学

$f=(y\cdot x+b)\cdot x+c$

① $v_1:=x\cdot y$	$\frac{\partial f}{\partial x}:=\frac{\partial f}{\partial x}+\frac{\partial f}{\partial v_1}\cdot\frac{\partial v_1}{\partial x}=v_2+x\cdot y$	⑩
	$\frac{\partial f}{\partial y}:=\frac{\partial f}{\partial y}+\frac{\partial f}{\partial v_1}\cdot\frac{\partial v_1}{\partial y}=0+x\cdot x=x^2$	⑨
② $v_2:=v_1+b$	$\frac{\partial f}{\partial v_1}:=\frac{\partial f}{\partial v_1}+\frac{\partial f}{\partial v_2}\cdot\frac{\partial v_2}{\partial v_1}=0+x\cdot 1=x$	⑧
③ $v_3:=v_2\cdot x$	$\frac{\partial f}{\partial v_2}:=\frac{\partial f}{\partial v_2}+\frac{\partial f}{\partial v_3}\cdot\frac{\partial v_3}{\partial v_2}=0+1\cdot x=x$	⑦
	$\frac{\partial f}{\partial x}:=\frac{\partial f}{\partial x}+\frac{\partial f}{\partial v_3}\cdot\frac{\partial v_3}{\partial x}=0+1\cdot v_2=v_2$	⑥
④ $f:=v_3+c$	$\frac{\partial f}{\partial v_3}:=\frac{\partial f}{\partial v_3}+\frac{\partial f}{\partial f}\cdot\frac{\partial f}{\partial v_3}=0+1\cdot 1=1$	⑤

図-1　関数 f の計算過程と偏導関数の計算過程
Fig. 1　Computational scheme for the function "$f=(yx+b)x+c$" and its derivatives.

("計算グラフ"[2] とか "Kantorovich グラフ"[6] とかと呼ばれる) でも表すことができる．各点 v_i に基本演算 φ_i を対応づけておけば，計算グラフは計算過程を(無意味な順序の入れ換えを除いて) 完全に表現する．たとえば，図-1 の関数の計算過程には 図-2 の計算グラフが対応する．

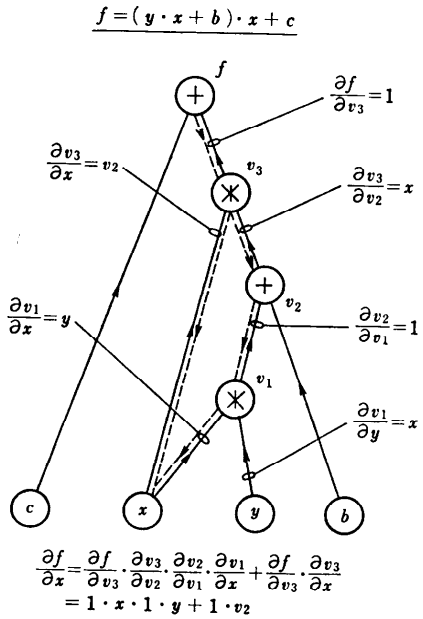
基本演算 φ が単項および二項演算のみであれば，計算グラフの点の数 V と枝の数 A の間には次の関係が成り立つことはグラフ理論では良く知られている：

$$A\leqq 2V. \qquad (2)$$

計算グラフにおいて点 v_{i_k} を点 v_i に結ぶ枝に "要素的偏導関数" $\partial v_i/\partial v_{i_k}=\partial\varphi_i/\partial v_{i_k}$ を対応づける．すると，関数 f の入力変数 x_i に関する偏導関数は，合成関数の偏微分のいわゆる "chain-rule" により，中間変数 v_i を用いて次のように書ける：

$$\frac{\partial f}{\partial x_i}=\sum\frac{\partial f}{\partial v_{i_1}}\cdot\frac{\partial v_{i_1}}{\partial v_{i_2}}\cdot\cdots\cdot\frac{\partial v_{i_n}}{\partial x_i}. \qquad (3)$$

ここで，和 $\sum$ は，計算グラフ上で入力変数 x_i に対応する点から関数 f に対応する点に至る有向道 $(x_i, v_{i_n}, \cdots, v_{i_2}, v_{i_1}, f)$ すべてについて取るものとする．すなわち $\partial f/\partial x_i$ は「計算グラフ上で x_i から f への道に沿って要素的偏導関数の積を作り，その積をそのようなすべての道に関して足し合わせる」ことによって計算される．これは，有向閉路のない (acyclic な) グラフ上の最短路問題と同じ形式を有しているので，それについて知られている事実から「要素的偏導関数が求められていれば，関数のすべての中間変数についての偏導関数は，計算グラフの枝の数 A に比例する手間で求めることができる」ということが分かる．要素的偏導関数は，枝の数 A 程度の手間で求められることは明らかである．(実際には，関数計算が済んだ後では要素的偏導関数値はどれかの中間変数の値に等しいことが多く，それらの計算にはほとんど余分な手間はかからない[3]．) 関数計算には点の数 V に等しい回数の基本演算が必要である．そこで(2)式により，結局「関数 f のすべての偏導関数 (すなわち f の勾配) は f 自身を求める手間の高々定数 (入力変数の数に依らない！) 倍程度で求められる」ことになる．



$$\frac{\partial f}{\partial x}=\frac{\partial f}{\partial v_3}\cdot\frac{\partial v_3}{\partial v_2}\cdot\frac{\partial v_2}{\partial v_1}\cdot\frac{\partial v_1}{\partial x}+\frac{\partial f}{\partial v_3}\cdot\frac{\partial v_3}{\partial x}$$
$$=1\cdot x\cdot 1\cdot y+1\cdot v_2$$

図-2　関数 f の計算グラフ
Fig. 2　Computational graph for the function "f".

関数計算が済んだ後で偏導関数を計算する算法は次の通り．

Step 0：〈要素的偏導関数を求める〉

Step 1：〈f の各中間変数 v_i に関する偏導関数を求める〉

初期化：**for** all v_i **do** $\frac{\partial f}{\partial v_i}:=0$;

$\frac{\partial f}{\partial f}:=1$;

グラフを辿る：(f の計算過程を逆に辿りながら)

各 $v_i=\varphi_i(v_{i_1},\cdots,v_{i_k},\cdots,v_{i_n})$ に対応して

for $k:=1$ **to** n **do**

$$\frac{\partial f}{\partial v_{i_k}}:=\frac{\partial f}{\partial v_{i_k}}+\frac{\partial f}{\partial v_i}\cdot\frac{\partial v_i}{\partial v_{i_k}}.$$

たとえば，図-1 において，関数の計算過程①～④にこの算法を適用すると偏導関数の計算過程⑤～⑩が得られる．

複数の関数が同時に計算される計算過程（計算グラフ）においては，上記の算法の Step 1 だけを各関数ごとに独立に繰り返せばよい．

この算法では関数を計算する向きと偏導関数を計算する向きが逆で，すべての中間変数の値を保存している．したがって，計算グラフの大きさ（V や A）に比例する記憶領域が必要である．しかし，関数 f のすべての中間変数に関する偏導関数も副産物として求められるので，それを利用して，次節に述べるように，f の計算値に含まれる丸め誤差の評価ができる．

3. 丸め誤差解析

以下では，変数記号 v, f などに $\tilde{}$ をつけることによってその変数の計算値を表す．また，基本演算 φ に対応して実際に行われる演算を $\tilde{\varphi}$ と書き，元の演算 φ と区別する．

与えられた入力変数 x_i の値は正確であるとすると，ある中間変数 v_i の計算値 $\tilde{v}_i$ の真値からのずれ Δv_i は，各基本計算ステップ(1)式に対して

$$\begin{aligned}\Delta v_i &\triangleq \tilde{v}_i - v_i \\ &= \tilde{\varphi}(\tilde{v}_{i_1}, \cdots, \tilde{v}_{i_n}) - \varphi(v_{i_1}, \cdots, v_{i_n})\end{aligned} \tag{4}$$

を満足する．(4)式は

$$\left.\begin{aligned}\Delta v_i &= \varphi_i(\tilde{v}_{i_1}, \cdots, \tilde{v}_{i_n}) - \varphi_i(v_{i_1}, \cdots, v_{i_n}) + \delta v_i, \\ \delta v_i &\triangleq \tilde{\varphi}_i(\tilde{v}_{i_1}, \cdots, \tilde{v}_{i_n}) - \varphi_i(\tilde{v}_{i_1}, \cdots, \tilde{v}_{i_n})\end{aligned}\right\} \tag{5}$$

と書けるが，Δv_i などが微小量であれば

$$\Delta v_i \fallingdotseq \sum_{k=1}^{n} \frac{\partial v_i}{\partial v_{i_k}} \Delta v_{i_k} + \delta v_i \tag{6}$$

と書いてよい．（右辺第1項を“伝播誤差”，第2項を“発生誤差”と呼ぶ．）関数 f の計算過程全体に(6)式を適用し，入力変数に対して $\Delta x_i = 0$ であることおよび(3)式を考慮して，f を計算する際に発生する丸め誤差 Δf の次の表式を得る（和は入力変数を除く全中間変数についてとる）：

$$\Delta f = \sum_i \frac{\partial f}{\partial v_i} \delta v_i. \tag{7}$$

計算機上ですべての基本演算がその仮数部の最下位桁まで正しく計算されるとすれば，発生誤差 δv_i は，いわゆる machine epsilon ε を用いて

$$|\delta v_i| \leq |v_i| \varepsilon \tag{8}$$

と上から抑えられる．(7)，(8)式より，$|\Delta f|$ の上からの評価式

$$|\Delta f| \leq \sum_i \left|\frac{\partial f}{\partial v_i}\right| \cdot |\delta v_i| \leq \left(\sum_i \left|\frac{\partial f}{\partial v_i} v_i\right|\right) \varepsilon \tag{9}$$

が得られる．(9)式の最右辺を“Δf の絶対評価”と呼ぶことにする．$\partial f/\partial v_i$, δv_i の符号は実際には揃っていないのが普通であるから，問題の規模が大きくなり中間変数の個数が増えると，(9)式はかなりの過大評価となる虞れがある．そこで(9)式の和を“ピタゴラス和”でおきかえて係数 $1/\sqrt{3}$ をつけた次式も実用上有効である：

$$\frac{1}{\sqrt{3}} \left(\sum_i \left(\frac{\partial f}{\partial v_i}\right)^2 v_i^2\right)^{1/2} \varepsilon. \tag{10}$$

(10)式は次のような統計的意味を有している．すなわち，発生誤差 δv_i が互いに独立な確率変数であるとすると，$(\Delta f)^2$ の期待値は次のように書ける：

$$\begin{aligned}\mathrm{E}[(\Delta f)^2] &= \mathrm{E}\left[\left(\sum_i \frac{\partial f}{\partial v_i} \delta v_i\right)^2\right] \\ &= \sum_i \left(\frac{\partial f}{\partial v_i}\right)^2 \mathrm{E}[\delta v_i^2] \\ &\quad + \sum_i \sum_{j \neq i} \frac{\partial f}{\partial v_i} \frac{\partial f}{\partial v_j} \mathrm{E}[\delta v_i] \mathrm{E}[\delta v_j].\end{aligned} \tag{11}$$

（i） 四捨五入（7捨8入，0捨1入など）の場合：——δv_i が区間 $[-\varepsilon_i, \varepsilon_i]$ $(\varepsilon_i \leq |v_i| \varepsilon)$ 上の一様分布に従うとすると，$\mathrm{E}[\delta v_i] = 0$, $\mathrm{E}[\delta v_i^2] = \mathrm{V}[\delta v_i] = \varepsilon_i^2/3 \leq v_i^2 \varepsilon^2/3$ であるから，(11)式は次のように評価できる：

$$\mathrm{E}[(\Delta f)^2] = \sum_i \left(\frac{\partial f}{\partial v_i}\right)^2 \mathrm{E}[\delta v_i^2] \leq \frac{1}{3} \sum_i \left(\frac{\partial f}{\partial v_i}\right)^2 v_i^2 \varepsilon^2. \tag{12}$$

なお，$\mathrm{E}[\Delta f] = \sum (\partial f/\partial v_i) \mathrm{E}[\delta v_i] = 0$ であるから，$\mathrm{E}[(\Delta f)^2] = \mathrm{V}[\Delta f]$ であり，(10)式は Δf の標準偏差の上界になる．

（ii） 切り捨ての場合：—— δv_i が区間 $[0, \varepsilon_i)$ $(|\varepsilon_i| \leq |v_i| \varepsilon)$ の一様分布に従うとすると，$\mathrm{E}[\delta v_i] \neq 0$ であるから(11)式の最右辺の第2項は0とはならない．しかし，(11)式の第1項は正数の和であるが，第2項は普通正負の値の混ざった和であるから，第2項を無視すれば，次のようにして(11)式を近似できる：

$$\mathrm{E}[(\Delta f)^2] \fallingdotseq \sum_i \left(\frac{\partial f}{\partial v_i}\right)^2 \mathrm{E}[\delta v_i^2]. \tag{13}$$

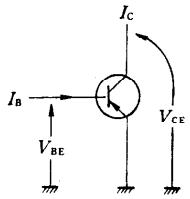
一方，$\mathrm{E}[\delta v_i^2] = \varepsilon_i^2/3 \leq v_i^2 \varepsilon^2/3$ であるから次式を得る：

$$\mathrm{E}[(\Delta f)^2] \lessapprox \frac{1}{3} \sum_i \left(\frac{\partial f}{\partial v_i}\right)^2 v_i^2 \varepsilon^2. \tag{14}$$

そこで(10)を“Δf の確率評価”と呼ぶ．

4. 小規模な例による予備実験

前章までに述べた算法の効率および丸め誤差を解析する際に置いた仮定の正当性を確認するため，小規模



$$I_B=-(1-\alpha_F)\cdot I_{ES}\cdot[\exp(-q\cdot V_{BE}/k\cdot T)-1]$$
$$-(1-\alpha_R)\cdot I_{CS}\cdot[\exp(q\cdot(V_{CE}-V_{BE})/k\cdot T)-1].$$
$$I_C=-\alpha_F\cdot I_{ES}\cdot[\exp(-q\cdot V_{BE}/k\cdot T)-1]$$
$$+I_{CS}\cdot[\exp(q\cdot(V_{CE}-V_{BE})/k\cdot T)-1].$$

I_{ES}, I_{CS}：エミッタ・ベース間およびコレクタ・ベース間の飽和電流
α_F, α_R：電流伝達率
V_{BE}, V_{CE}：ベース・エミッタ電圧およびコレクタ・エミッタ電圧
T：温度
q：電子の電荷
k：ボルツマン定数

図-3　トランジスタの Ebers-Moll モデル
Fig. 3　Ebers-Moll model for a pnp-transistor.

な例としてトランジスタの Ebers-Moll モデルをとりあげ計算機実験を行った．基本的な関数の式を**図-3**に示す．これらの式の計算グラフを用いた解析は文献3)，4) に記されているので，ここでは実験方法および結果のみを述べる．

4.1　計算の効率

（a）　方法：二つの関数 I_B, I_C の計算式をそのまま定義通り書き下したプログラム [OD, F] (OD は OrDinary method の略；F は Functions の略) と，中間変数を用いて最適化した計算過程 [IM, F] (IM は InterMediate-variable method の略) の二通りを実行し，計算に要する時間を測定した．

次に，この問題（9入力変数 V_{BE}, V_{CE}, I_{ES}, I_{CS}, α_F, α_R, T, q, k；2関数 I_B, I_C) のヤコビ行列の 18 個の要素を一々式で書き下したプログラム [OD, F+J] (F+J は Functions & Jacobian matrix の略) と，計算グラフを利用した本算法によるもの [IM, F+J] を実行し，要した時間を測定して比較した．[なお，全計算を通じ，$I_{ES}=1.0\times10^{-9}$A, $I_{CS}=2.0\times10^{-9}$A, $\alpha_F=0.98$, $\alpha_R=0.5$, $V_{BE}\fallingdotseq-0.4$ V, $V_{CE}\fallingdotseq-1.0$ V, $T=300.0$K, $q=1.602\times10^{-19}$C, $k=1.38053\times10^{-23}$ J/K とした．]

（b）　結果：結果の一部を**表-1**に示す．“OD の時間/IM の時間” は関数のみの計算で 1.5〜2，ヤコビ行列まで計算すると 4.5〜5.5 であった．

表-1　計算時間の実験結果
Table 1　Comparison of computation times of the functions and their derivatives by InterMediate-variable method and OrDinary method.

計算時間(μs)	IM	OD
F	10	16
F+J	24	109

時間比(1)	IM : OD
F	1 : 1.6
F+J	1 : 4.5

時間比(2)	IM	OD
F	1	1
F+J	2.4	6.6

注）使用計算機：HITAC M 280-H，使用言語：VOS 3 FORTRAN 77，コンパイル・オプション：OPT (2)・NOIAP．（引数をもつサブルーチン呼び出しを使用．呼び出しに要した時間を含む．1回の呼び出しにかかる時間は 2〜8 μs.)

4.2　丸め誤差解析

（a）　方法：V_{BE} の値を，有効数字の最後の方の桁がランダムになるように，微小な刻み幅で 1000 段階に変化させて，I_B, I_C の計算を単精度，倍精度で行い，両者の差を単精度計算時の丸め誤差 $\Delta I_B, \Delta I_C$ の実測値とした．得られた 1000 個の丸め誤差の実測値の標本に対し，平均と標準偏差を算出した．

一方で，発生丸め誤差の独立性や分布型に関する前節での仮定の妥当性を調べるために，(1) δv_i の分布幅 ε_i を単に $|v_i|\varepsilon$ で抑えるのでなく，v_i の計算機内での内部表現まで考慮して ε_i をより精密に見積もり，

表-2　丸め誤差評価の実験結果（標本 1000 個）
Table 2　Average and standard deviation of rounding errors (1000 samples) for I_B and I_C (observed values vs. estimates).

	演算の形式		16進切り捨て (M 280-H)	2進0捨1入 (VAX-11)
I_B の丸め誤差	I_B の関数値		およそ -1.0×10^{-4}A	
	平均	実測値	-2.01×10^{-10}A	4.64×10^{-11}A
		理論値	-1.73×10^{-10}A	4.94×10^{-11}A
	標準偏差	実測値	3.83×10^{-10}A	5.66×10^{-11}A
		理論値	3.80×10^{-10}A	5.56×10^{-11}A
	実測最大丸め誤差		0.93×10^{-9}A	1.80×10^{-10}A
	丸め誤差絶対評価		5.24×10^{-9}A	3.28×10^{-10}A
I_C の丸め誤差	I_C の関数値		およそ -5.2×10^{-3}A	
	平均	実測値	-0.65×10^{-8}A	0.26×10^{-8}A
		理論値	-0.55×10^{-8}A	0.24×10^{-8}A
	標準偏差	実測値	1.88×10^{-8}A	0.28×10^{-8}A
		理論値	1.87×10^{-8}A	0.27×10^{-8}A
	実測最大丸め誤差		0.42×10^{-7}A	0.92×10^{-8}A
	丸め誤差絶対評価		2.52×10^{-7}A	1.57×10^{-8}A

注）使用計算機は HITAC M 280-H および VAX-11.

(2) V_{BE} を変える際に，その影響を受けない中間変数 ($\partial v_i/\partial V_{BE}=0$ となるような v_i) に対しては δv_i が一点分布に従うものとして，理論的に ΔI_B, ΔI_C の平均，標準偏差を $E[\Delta f]=\sum_i(\partial f/\partial v_i)E[\delta v_i]$ および(11)式により計算し，それらを実測値と比較した．また，前節で述べた丸め誤差の絶対評価の値とも比較した．

(b) 結果：**表-2** の通り．理論値と実測値がよく一致していること，各計算機の数値表現および丸め方式の違いが結果によく反映されていることが見られる．VAX-11 (2進0捨1入) においては，ΔI_B と ΔI_C の共分散も求めたが，実測値に基づく値 7.20×10^{-18} に対して，$\sum_i(\partial I_B/\partial v_i)(\partial I_C/\partial v_i)V[\delta v_i]$ ($V[\delta v_i]=\varepsilon_i^2/3$) により計算した値は 7.06×10^{-18} であった．

5. 大規模システムへの適用例

本章では，メタノール蒸溜塔の数学モデル[7),8)]の一部として現れる108元の非線形連立方程式系を，本論文の方法で解析した結果を述べる．

方程式系は，108個の入力変数と98個の定数を与えて，108個の関数値を計算する924ステップの計算過程として記述されている．中間変数（入力変数，関数，定数は除く）816個も含め，計算グラフ上には1130個の点がある．

5.1 計算時間

この方程式のヤコビ行列の 108^2 個の全要素を計算するのに，数値微分によるときに要する時間と本論文の算法（以下“グラフ算法”と略記する）によるときに要する時間を比較した．

(a) 実験方法：数値微分では，108個の入力変数の値を1個ずつ僅かに変化させて関数の計算過程を109回繰り返し実行し，関数値の差を変数の変化量で割ることにより偏導関数を求めた．グラフ算法では，関数計算と要素的偏導関数の計算は唯1回行い，各関数の偏導関数は，独立にグラフの必要な部分だけを**2章**の方法で辿ることによって計算した．

(b) 結果と考察：**表-3** に見られるように，グラフ算法は数値微分法より約13～14倍速い．数値微分に要する時間の内訳は

(入力変数の個数+1)×(関数計算の時間)ⓐ
+(入力変数の個数)²×(1回の引算，割算)ⓑ (15)

であり，グラフ算法に要する時間の内訳は

(関数計算の時間)+(要素的偏導関数の計算時間)ⓒ+(関数の個数)×(2章の方法でグラフを辿って一つの関数のすべての偏導関数を求める時間)ⓓ (16)

である（“関数の個数＝入力変数の個数＝108”である）．コンパイル・オプションを固定して，ⓐ，ⓑ，ⓒ，ⓓそれぞれに要する時間を測定した結果は**表-4**の通りである．数値微分ではⓐに，グラフ算法ではⓓに，ほとんどの時間がかかっていることが分かる．ⓐもⓓも入力変数の個数に比例する量なので，二つの算法の速さの違いは，基本的には，グラフ算法で“計算グラフを top-down に1回辿り一つの関数のすべての偏導関数を計算する”ことが，“数値微分で計算グラフを bottom-up に辿って全関数を1回計算する”ことよりも高速に行えることによる．グラフ算法においては，一般に問題が大規模になるにつれて一つの関数の偏導関数を求めるのに辿るべき計算グラフの部分の割合は小さくなっていくのが普通であるから，両算法の速さの比は問題が大規模になるにつれてますます大きくなると予想される．

表-3 ヤコビ行列の計算時間（単位s）
Table 3 Comparison of computation times of the Jacobian matrix by ① numerical differentiation and by ② the new method.

	OPT(0)	OPT(2)	OPT(3)
① 数値微分法	6.94	4.64	4.06
② グラフ算法	0.50	0.35	0.29
①/②	13.9	13.3	14.0

注) 使用計算機：HITAC M 280-H，使用言語：VOS 3 FORTRAN 77，コンパイル・オプション：OPT (n) (n=0, 2, 3)・NOIAP

表-4 計算時間の内訳（単位s）
Table 4 Detailed analysis of computation times of the Jacobian matrix.

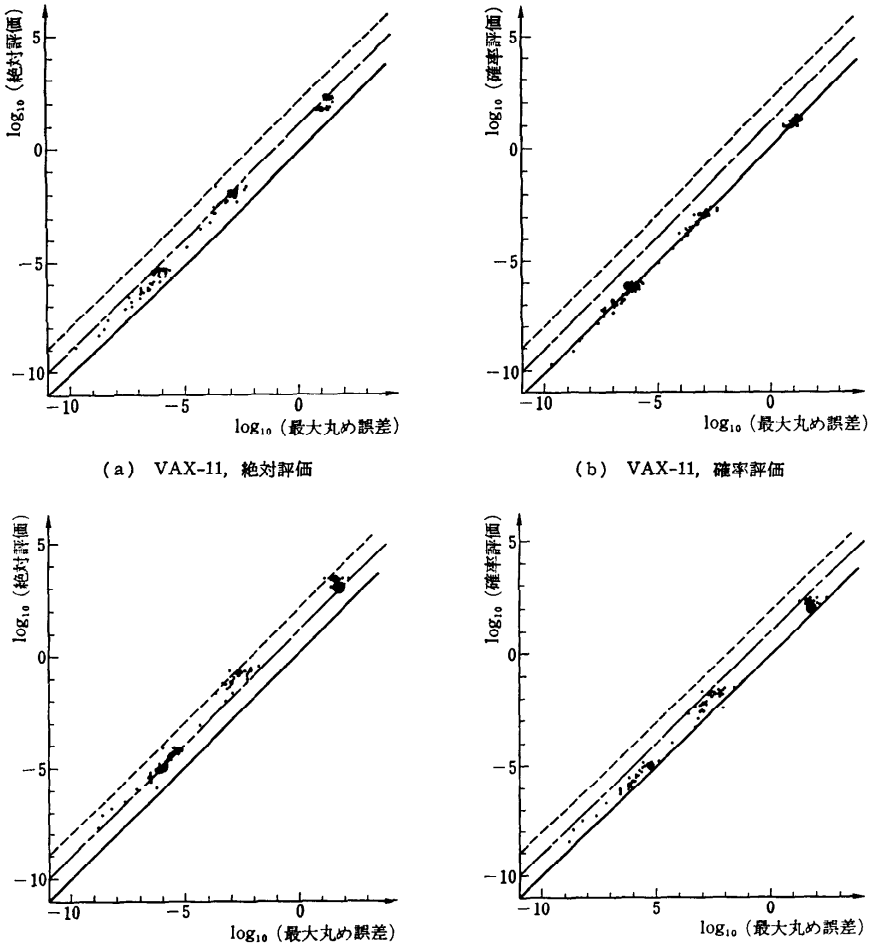
	ⓐ	ⓑ	ⓒ	ⓓ
数値微分法	4.6 (>98%)	<0.1 (<2%)	.....	.....
グラフ算法	.....	.....	0.05 (14%)	0.30 (86%)

注) コンパイル・オプションは OPT (2)・NOIAP，その他の計算環境は表-3 と同じ．

5.2 丸め誤差評価

3章で述べた丸め誤差評価法を自動的に実行するプログラムを作成し，方程式系の108個の関数を評価する際に生じることが予想される丸め誤差を評価した．

(a) 実験方法：**7章**で述べるニュートン法の初期値①のごく近くの5点を乱数を使って選び，各点にお



(a) VAX-11, 絶対評価　(b) VAX-11, 確率評価

(c) M 280-H, 絶対評価　(d) M 280-H, 確率評価

図-4 最大丸め誤差と評価値の比較

Fig. 4 Observed maximum rounding errors vs. the theoretical estimates for the 108 functions. (実線は, "最大丸め誤差＝評価値" の線, 一点鎖線および破線は, それぞれ, "10×最大丸め誤差＝評価値" および "100×最大丸め誤差＝評価値" の線.)

いて, 108 個の関数それぞれについて丸め誤差 (単精度計算) の確率評価, 絶対評価の値を計算し, "(単精度計算の結果)－(倍精度計算の結果)" として算出した丸め誤差の実測値と比較した. 実験は, HITAC M 280-H (16 進切り捨て, $\varepsilon=16^{-5}$), VAX-11 (2 進 0 捨 1 入, $\varepsilon=2^{-24}$) の 2 機種で行った.

(b) 結果: 108 個の関数それぞれについて得られた 5 個の丸め誤差の実測値のうちで絶対値が最大のもの (以下 "最大丸め誤差" と呼ぶ) とその丸め誤差評価値との比較を**図-4**, **図-5** に示す.

(c) 考察: 図-4 から見られることは, 異なる関数に対する丸め誤差の差異が非常に大きく (比が 10^{11}～10^{12} にも達し "桁違い" という語が正に適当する), それに比して, 2 種の丸め誤差評価値がどちらも実測値に近いということである. また, "最大丸め誤差/丸め誤差評価値" の分布のヒストグラム (図-5) からそれらの評価がどのくらい過大評価になっているかが観察できる.

絶対評価についてみると, **図-5**(a), (c) から観察されるように, VAX-11 では平均で約 5 倍 (最小で 2

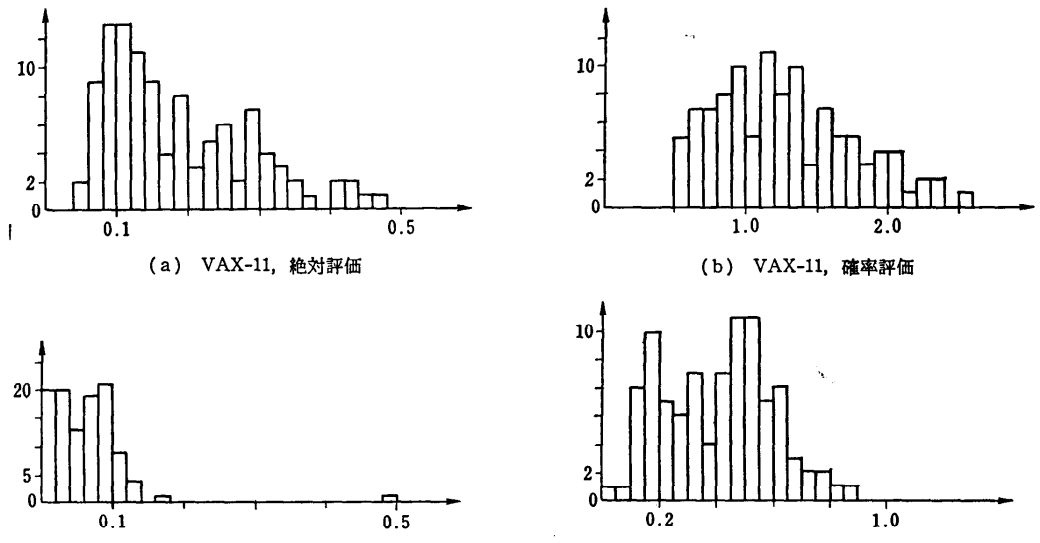


図-5 108 個の関数の (最大丸め誤差/評価値) のヒストグラム

Fig. 5 Histogram of the ratios of the observed maximum rounding errors to the theoretical estimates for the 108 functions.

倍), M 280-H でも平均で約 14 倍 (最小で2倍) 程度の過大評価で済んでいる. これは, 3章の記号で, $|v_i|\varepsilon/\varepsilon_i$ がそれぞれ2および 16 にまでなりうることを考えると, それほど過大な評価ではない. このことは, 丸め誤差の表現式(7)式において, 項数がいくら多くても実際に支配的な項がそう多くはないことにもよると考えられる.

確率評価についてみると, 図-5(b)から観察されるように, VAX-11 では "最大丸め誤差/確率評価" が 0.5～2.7 の範囲に収まっている. 確率評価が丸め誤差の標準偏差を上から抑えているという意味付けに照らすと, これは非常に良い評価であるといえる. M 280-H でも図-5(d)から観察されるように, "最大丸め誤差/確率評価" はおよそ 0.1～0.9 の間にあり, 大雑把にいって数倍の過大評価でしかない.

以上のように, 本論文で述べた丸め誤差評価法 (特に確率評価法) は実用的でかつ十分に良い評価法であることがこの例で観察された.

6. 丸め誤差評価法を応用したニュートン法の制御と収束過程

方程式系

$$f_i(x_1, \cdots, x_n)=0 \quad (i=1, \cdots, n) \tag{17}$$

を解くことは, 理想的には, f_i を x 付近で計算した際の丸め誤差の評価値を $|\Delta f_i(x)|$ として,

$$|f_i(x)| \leq |\Delta f_i(x)| \quad (i=1, \cdots, n) \tag{18}$$

を満たす x を求めることである. ところが, このような系をニュートン法などを用いて解く場合, 従来は

$$\|f\| = \left(\sum_{i=1}^{n} |f_i|^2\right)^{1/2} \tag{19}$$

などのノルム (以後 "通常ノルム" と呼ぶ) があらかじめ与えられたある正の実数より小さくなったら反復を停止する, という "収束判定" を行ってきた. しかし, 前章で取り上げた例では, 5.2 で見たように, 108 個の関数の丸め誤差は 10^{-7}～10^{-9} から 10^{2}～10^{4} に至るまできわめて広い範囲に分布しており, したがって(19)式のようなノルムは理論的にも実際的にもまったく無意味であると言えよう. (18)式の意味での収束を実現するためには, machine epsilon ε を用いて

$$\tilde{f}_i = \frac{f_i}{|\Delta f_i|}\varepsilon \quad (i=1, \cdots, n) \tag{20}$$

とおいて, $\tilde{f}_i$ に対する通常のノルムを用いて収束判定を行うべきであることは明らかである (条件(18)式は $\|\tilde{f}\| \leq \varepsilon$ に対応する). 本論文で述べたように, $|\Delta f_i|$ の良い評価値を高速に得る実用的な方法が確立されたので, このような $\tilde{f}_i$ のノルム (これを以後 "正規化ノルム" と呼ぶ) を用いる計算法が実現可能である.

7. 大規模システムのニュートン法による解の実例

前章で述べた方針に基づいて 5 章で取り上げた 108 元の方程式をニュートン法を使って解くことを試み，従来の方法との比較を行った．

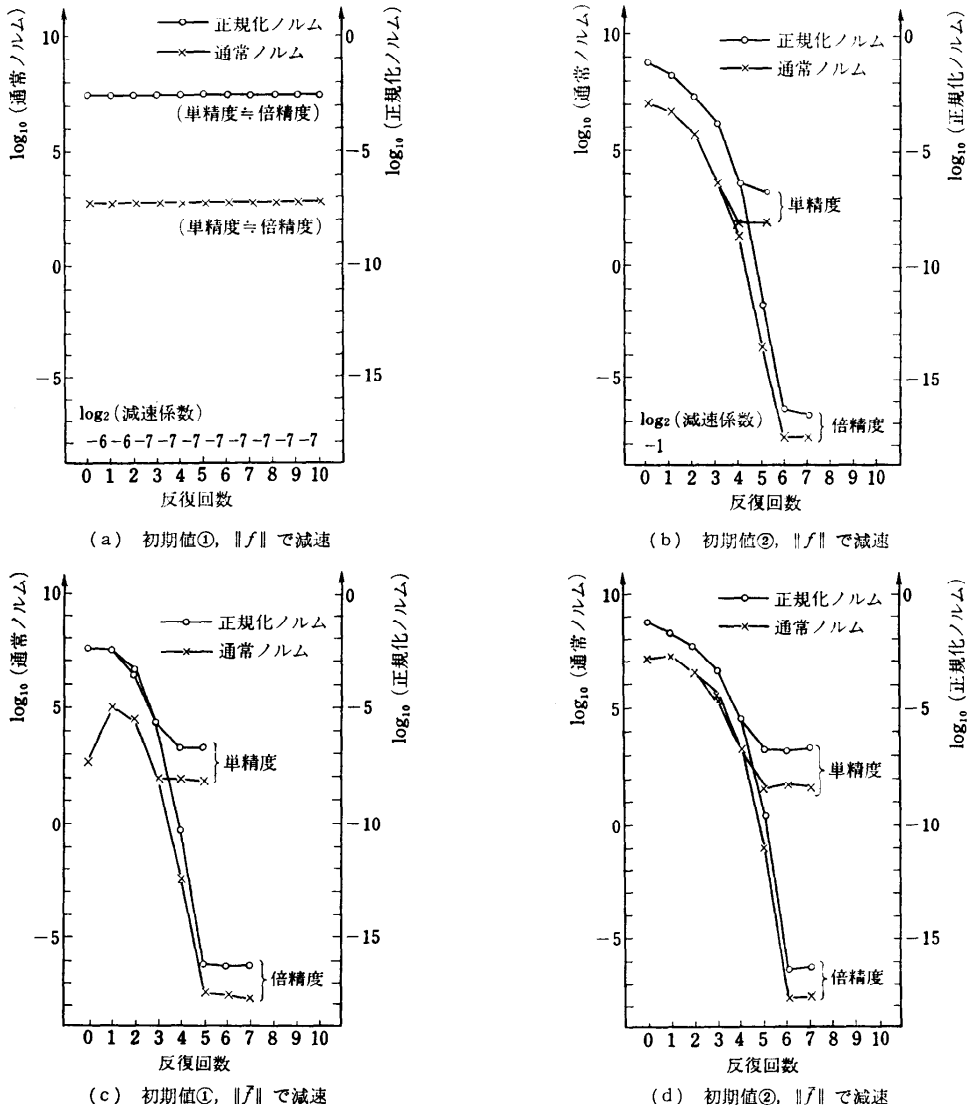
（a） 実験方法：VAX-11 ($\varepsilon=2^{-24}$ (単精度)，$\varepsilon=2^{-56}$ (倍精度)) を用いた．次の 4 種の減速ニュートン法のルーチンを用意した．

（1） ヤコビ行列を求める部分，関数計算の部分，連立一次方程式を解く部分のすべてを倍精度で行う．

(1.1) 通常ノルム $\|f\|$ が減少するように減速する．

(1.2) 正規化ノルム $\|\bar{f}\|$ が減少するように減速する．

（2） ヤコビ行列を求める部分，関数計算の部分は単精度で計算し，連立一次方程式を解く部分のみ倍精



（a） 初期値①，$\|f\|$ で減速

（b） 初期値②，$\|f\|$ で減速

（c） 初期値①，$\|\bar{f}\|$ で減速

（d） 初期値②，$\|\bar{f}\|$ で減速

図-6 ニュートン法の収束過程

Fig. 6 Process of convergence of the Newton method with deceleration for the equations.

度で行う.

(2.1) 通常ノルム $\|f\|$ が減少するように減速する.

(2.2) 正規化ノルム $\|\tilde{f}\|$ が減少するように減速する.

ここで,

$$\|f\|=\left(\sum_{i=1}^{108} f_i^2\right)^{1/2},\quad \|\tilde{f}\|=\left(\sum_{i=1}^{108} \tilde{f}_i^2\right)^{1/2}, \tag{21}$$

$$\tilde{f}_i=\frac{f_i}{|\Delta f_i|}\varepsilon \quad (i=1,\cdots,108) \tag{22}$$

とおき, $|\Delta f_i|$ としては, "丸め誤差の確率評価" の値を用いた.

また, 初期近似解としては,

① 蒸溜塔の常微分方程式系モデルをしばらくの間積分して定めた平衡解に近い解,

② 平衡解の有効数字を2桁に縮めたもの

の二つを用いて実験した.

(b) 結果と考察: それぞれの場合について, ニュートン法の反復回数と $\|\tilde{f}\|, \|f\|$ との関係を図-6(a)～(d)に示す (減速した場合には減速係数も示してある). (1.2), (2.2) のルーチーンでは, 図-6(c), (d)に示すように, 単精度, 倍精度どちらの場合も正規化ノルムが machine epsilon ε (単精度で 6.0×10^{-8}, 倍精度で 1.4×10^{-17}) の程度にまで二乗収束で落ちていて, (18)式の条件がほぼ達成されていることが分かる.

特に注目すべきことは, 両方の初期値とも, 最初の反復では通常ノルム $\|f\|$ が増えているにもかかわらず正規化ノルム $\|\tilde{f}\|$ は減少しており, 後者で見る限りニュートン法が素直な収束列を作り出していることである. このような場合, (1.1), (2.1)のルーチーンを用いて通常ノルムによる減速をかけると, 図-6(a), (b)に示すように, 無用な減速をかけることになるし, 時にはかなり悲惨な結果を招く. この例は, 通常ノルムを用いると, 「関数全体としての減少傾向が大きな丸め誤差をもつ少数の関数の増大によってかき消されてしまう」現象が起きることを示している. また, 反復過程が図-6(a)のような挙動を示した場合, 通常ノルムで見たのでは, 解に収束したかどうかの判断は難しいが, 正規化ノルムで見れば, $\|\tilde{f}\|\gg\varepsilon$ であることから解からは程遠いことが分かる.

8. 数値計算の単位としての丸め誤差

数値計算では, ある量を "0" にしようとする算法が非常に多い. $\mathbf{R}^n$ 上で考える場合, 数学的にはすべてのノルムが同値であると言われているが, 数値解析の立場から見ると, 様相がまったく異なることは前章で見てきた通りである. したがって, 反復過程の制御や収束判定の際, どのようなノルムを用いるべきかは重要な問題である. それに対する一つの答えが, 前章で与えた "丸め誤差を単位として測った正規化ノルム" である. 各成分の "雑音水準" を揃えることにより, 有用な情報と雑音を分離することができ, 丸め誤差の悪影響が抑えられることは前章で示した通りである.

この考え方をさらに発展させて, 関数の丸め誤差 (の評価値) の分散・共分散行列を求めて, それによって, 残差空間の計量を定義し, 系の精密なスケーリングを行ったり, その計量から誘導される解空間の計量を用いて解 x 自体の精度を推定することなども考えられる[9]. また, "雑音水準を揃える" という考え方自体は, 反復法以外の算法に関しても有効なスケーリングの方針を与えるのではないかと思われる.

9. まとめと今後の課題

グラフ理論を応用した新しい偏導関数計算法および関数計算の丸め誤差の評価法を, 蒸溜塔の数学モデルである108元非線形方程式の解法に適用しその効果を調べた. その結果, 新しい方法によれば

① "正確な" ヤコビ行列が従来の算法に比べて10倍以上高速に計算できること

② 精密な丸め誤差評価が得られ, それを利用することによって合理的なノルムが定義できること

が確認された.

大規模なシステムに対しても, 丸め誤差の確率評価の際に, 発生誤差 δv_i の分布幅 ε_i を $|v_i|\varepsilon$ で抑えず正確に見積もることの自動化もそう困難でない. そのような ε_i を用いることによって, より精密な評価が得られ, 異なる関数の計算の丸め誤差の共分散の精密な見積もりも可能となるはずであるので, その方向に研究を進めつつある.

謝 辞 5章の蒸溜塔の数学モデルおよびそれに関連したデータは日本科学技術研修所のご好意により使わせていただいた. お世話になった恒川純吉氏始め日科技研の皆様に感謝致します.

なお, 本研究の一部は文部省科学研究費補助金 (一般研究(B)60460130: 偏導関数計算と丸め誤差評価の自動化のためのプログラム言語と処理系の研究) の援助による.

参 考 文 献

1) Wilkinson, J. H.: *Rounding Errors in Algebraic Processes*. Prentice-Hall Inc., Englewood Cliffs, N. J. (1963).
2) Bauer, F. L.: Computational Graphs and Rounding Errors. *SIAM Journal on Numerical Analysis*, Vol. 11, pp. 87-96 (1974).
3) Iri, M.: Simultaneous Computation of Functions, Partial Derivatives and Estimates of Rounding Errors—Complexity and Practicality. *Japan Journal of Applied Mathematics*, Vol. 1, No. 2, pp. 223-252 (1984).
4) 伊理正夫: 偏導関数計算へのグラフ理論の応用. 電子通信学会技術報告 CAS-84-74 (1984).
5) 岩田憲和: 偏導関数計算の自動化. 東京大学大学院工学系研究科情報工学専門課程修士論文, (Mar. 1984)
6) Rall, L. B.: *Automatic Differentiation: Techniques and Applications*. Lecture Notes in Computer Science, Vol. 120, Springer-Verlag, Berlin (1981).
7) 恒川純吉, 大阿久聰, 小山 潤, 佐々木孝: 応用分野の動向. 情報処理, Vol. 23, No. 2, pp. 130-139 (1982).
8) DPS (V 2) 利用者マニュアル. 情報処理振興事業協会 (1980).
9) 伊理正夫, 土谷 隆: 偏導関数自動計算法の大規模非線形方程式系への応用. 京都大学数理解析研究所研究集会 "数値計算の基本アルゴリズムの研究" (1984年11月29日～12月1日) 予稿集, pp. 11-12.

本論文投稿後, ソヴィエトにおいてほとんど同じ着想に基づいてほとんど同じ高速微分法の研究が独立に始められていることを知った. そこでは, しかし, 最適化計算への応用が当面の主題とされており, 丸め誤差解析や正規化ノルムの導入の研究はまだ行われていないようである. それに関連する文献に, 本論文で触れるべきであったものをつけ加えて, 以下に記す.

10) Baur, W. and Strassen, V.: The Complexity of Partial Derivatives. *Theoretical Computer Science*, Vol. 22, pp. 317-330 (1983).
11) Kim, K. V., Nesterov, Ju. E. and Čerkasskiĭ, B. V.: Ocenka trudoemkosti vyčislenija gradienta. *Doklady Akademii Nauk SSSR*, Matematika, tom 275, no. 6, pp. 1306-1309 (1984) (in Russian).
12) Kim, K. V., Nesterov, Ju. E., Skokov, V. A. and Čerkasskiĭ, B. V.: Èffektivnyĭ algoritm vyčislenija proizvodnyh i èkstremal'nye zadači. *Èkonomika i Matematičeskie Metody*, tom 20, vyp. 2, pp. 309-318 (1984) (in Russian).
13) Kim, K., Nesterov, Yu. and Cherkassky, B.: An Algorithm for Fast Differentiations and Its Applications. *Abstracts of the 12th IFIP Conference on System Modelling and Optimization* (September 2-6, 1985, Budapest), pp. 181-182.

(昭和60年7月1日受付)