多倍長浮動小数点演算と補外を用いた数値微分法に基づく Jacobi 行列計算

幸谷智紀 *
* 静岡理工科大学

Jacobian Matrix Computation based on Numerical Differentiation Using Multiple Precision Floating-point Arithmetic and Extrapolation

Tomonori Kouya*
*Shizuoka Institute of Science and Technology

Abstract. Nowadays, the Jacobian matrix computation is usually based on automatic differentiation(AD). Unless AD can be applied, numerical differentiation is selected. In this case, it generally yields a low precision. However, we can obtain an arbitrary precision Jacobian matrix by using extrapolation and multiple-precision floating-point arithmetic. In this paper, we propose an arbitrary precision numerical computation method of Jacobian matrix based on numerical differentiation using extrapolation, and demonstrate its efficiency through numerical experiments using MPFR [7].

1. 初めに

n 変数関数 $\mathbf{F}(\mathbf{Y})= [F_1(\mathbf{Y}) \ldots F_n(\mathbf{Y})]^T$, $\mathbf{Y} \in \mathbb{R}^n$ の Jacobi 行列 $\partial\mathbf{F}/\partial\mathbf{Y} \in \mathbb{R}^{n\times n}$

$$\frac{\partial \mathbf{F}}{\partial \mathbf{Y}} = \begin{bmatrix} \frac{\partial F_1}{\partial Y_1} & \frac{\partial F_1}{\partial Y_2} & \cdots & \frac{\partial F_1}{\partial Y_n} \\ \frac{\partial F_2}{\partial Y_1} & \frac{\partial F_2}{\partial Y_2} & \cdots & \frac{\partial F_2}{\partial Y_n} \\ \vdots & \vdots & & \vdots \\ \frac{\partial F_n}{\partial Y_1} & \frac{\partial F_n}{\partial Y_2} & \cdots & \frac{\partial F_n}{\partial Y_n} \end{bmatrix} = \left[\frac{\partial F_i}{\partial Y_j} \right]_{i,j=1,2,\ldots,n} \tag{1.1}$$

の近似行列 $J(\mathbf{Y}) \approx \partial\mathbf{F}/\partial\mathbf{Y}$ を，中心差分と補外法を用いて高精度に計算する方法について，本稿では議論する．

一般に，数値微分は真の導関数をソフトウェアを用いて代数的に計算する自動微分 (Automatic Differentiation, AD) [1] が使用できない場合に利用されることが多い．例えば，陰的公式を用いた常微分方程式ソルバや，非線型方程式向けの Newton 法のライブラリ関数には数値微分を用いて求めた $J(\mathbf{Y})$ を用いるオプションが用意されている．しかしこれは，一階差分商を用いたものであり，低精度な近似値しか得られないとされている．

今回提案する手法は，中心差分商と補外法を用いて高精度な近似 Jacobi 行列を求める

ものである．永坂・福井の結果 [3, 8] から，補外法を用いても，初期系列に用いた近似値の丸め誤差が 2 倍未満に抑えられることが判明している．従って，多倍長浮動小数点数を用いて高精度な初期系列の値を得ることができれば，補外法の段数を増やして打ち切り誤差を削っていくことにより $J(\mathbf{Y})$ の各成分も高精度に求められることになる．しかし，実際の計算では計算量の削減と，収束判定の精密さを両方実現する工夫が不可欠である．

本稿ではまず，永坂・福井による一変数関数 $f(x)$ に対する数値微分の誤差解析の結果に基づいた，実用的な収束判定法を述べる．次に，これに基づいて列単位で $J(\mathbf{Y})$ を計算し，要素ごとに収束判定を行う任意精度 Jacobi 行列計算法のアルゴリズムを述べる．最後に，数値実験を行ってアルゴリズムの有用性を確認する．

2. 中心差分商と補外法を用いた任意精度数値微分

無限回微分可能な一変数関数 $f(x)$ の m 階微係数 $f^{(m)}(x)$ を，N 点差分商で近似すると，その近似式は一般に

$$f^{(m)}(x) = \frac{1}{h^m} \sum_{s=1}^{N} b_s f(x_s) + E_T(h) \tag{2.1}$$

と表現できる．ここで b_s は定数，$E_T(h)$ は打ち切り誤差である．この時 $E_T(h)$ は

$$E_T(h) = \sum_{w=0}^{\infty} \frac{f^{(2w+3)}(x)}{(2w+3)!} h^{2w+2}$$

と h の偶数べき乗和として表現できる．

(2.1) を p 進 q 桁の浮動小数点数を用いて数値計算する時に発生する丸め誤差の限界を $E_R(h)$ とすると，総和の計算，h^m の計算，及びそれらを乗じる際に発生する丸め誤差を総合して

$$E_R(h) = \frac{N-1}{h^m} \max_s |b_s f(x_s)| \cdot c \cdot p^{-q} \tag{2.2}$$

と評価できる [8]．ここで c は丸めの方式によって決まる定数で，

$$c = \begin{cases} p & (\text{切り捨て}) \\ \frac{p}{2} & (p/2 - 1 \text{ 捨 } p/2 \text{ 入}) \end{cases}$$

である．

中心差分商を用いて求めた近似値 (2.1) を，Romberg 数列 $2^0, 2^1, \ldots$ を用いた補外法の初期系列 $f^{l,1}$ として使用する．この時

$$f^{l,1} = \frac{\sum_{s=1}^{N} b_s f(x_s)}{h^m / 2^{l-1}}$$

である．

よって，補外計算は

$$
\begin{array}{ccc}
 & & f^{l-1,k-1} \\
 & & \searrow \\
f^{l,k-1} & \rightarrow & f^{l,k} = f^{l,k-1} + \dfrac{f^{l,k-1} - f^{l-1,k-1}}{4^{k-1} - 1} = f^{l,k-1} + R^{l,k}
\end{array}
\tag{2.3}
$$

となる．この時 $f^{l,k}$ に含まれる打ち切り誤差は $O(h^{2l})$ で減少していく．この計算を，以下の表の初期系列から右の下三角成分を計算していくことになる．

初期系列				
$f^{1,1}$				
$f^{2,1}$	$f^{2,2}$			
$\vdots$	$\vdots$	$\ddots$		
$f^{L-1,1}$	$f^{L-1,2}$	$\cdots$	$f^{L-1,L-1}$	
$f^{L,1}$	$f^{L,2}$	$\cdots$	$f^{L,L-1}$	$f^{L,L}$

この補外計算で混入する丸め誤差限界 $E_R^{l,k}$ は

$$
E_R^{l,k} = \frac{1 + 2^{-m}}{4^{k-1} - 1} \cdot \frac{25}{14} E_R(h/2^l)
$$

であることが福井によって証明されている [3]．これにより，補外計算によって初期系列に混入した丸め誤差は，2 倍以内に抑えられることが分かる．同様の議論は Hairer & Wanner [5] にもある．

実際の数値計算ではこの値になる以前に収束したと判定することが望ましい．よって，収束判定は，初期系列 $f^{l,1}$ の丸め誤差評価値

$$
E_R^l = 2E_R(h/2^{l-1}) = E_R(h/2^l) \tag{2.4}
$$

と，ユーザが与えた相対許容度 $\varepsilon_r > 0$ と絶対許容度 $\varepsilon_a \geq 0$ も用い，補外計算 (2.3) において，修正項 $R^{l,k}$ が

$$
|R^{l,k}| \leq \max\left(\varepsilon_r |f^{l,k-1}| + \varepsilon_a, E_R^l\right) \tag{2.5}
$$

を満足した時に収束したと判定する．与えられた ε_r や ε_a が小さすぎる時には，自動的に求められた丸め誤差限界値 E_R^l が歯止めになる仕組みである．

3. 任意精度 Jacobi 行列計算法

1 階微係数 $f'(x)$ を 3 点中心差分商を用いて近似すると，(2.1) 式は

$$
f'(x) = \frac{1}{h}\left\{\frac{1}{2} f(x + h) - \frac{1}{2} f(x - h)\right\} + E_T(h) \tag{3.1}
$$

である．本節では，これを Jacobi 行列の初期系列として使用し，補外計算により精度を高めていくアルゴリズムについて議論する．

前節で述べた一変数関数の数値微分のアルゴリズムを用いると，任意精度 Jacobi 行列計算が実現できる．しかし，Jacobi 行列の各 ij 成分ごとに $\mathbf{F}(\mathbf{Y})$ を呼び出して計算を行おうとすると, その回数は $O(n^2)$ になってしまう．これを少なく抑えるため, $J(\mathbf{Y}) = [\mathbf{J}_1\ \mathbf{J}_2\ ...\ \mathbf{J}_n]$ を列ベクトル単位で計算するようにすれば，呼び出し回数は $O(n)$ に減らすことが可能となる．しかし，各成分は全て異なる偏導関数であるため，収束判定はベクトルノルムを用いるのではなく，各成分ごとに個別に行う必要が出てくる．以下，そのアルゴリズムと収束判定法を述べる．

3.1 アルゴリズム

Jacobi 行列の各要素 $\partial F_i/\partial Y_j$ の近似計算には，前述の 3 点中点公式 (3 点公式) を初期系列とする補外法を使用し，ベクトル単位で全ての計算を行う．ここでは第 j 列目の近似ベクトル $\mathbf{J}_j \approx \partial\mathbf{F}/\partial Y_j$ の計算を対象として解説する．

初期系列 $\mathbf{J}_j^{l,1}$ の計算は，まず $\mathbf{Y} = [Y_1\ \cdots Y_n]^T$ の第 j 成分を基準に刻み幅 h_j を Romberg 数列に乗じて

$$\mathbf{Y}^{j\pm} = \left[\cdots\ Y_{j-1}\ Y_j \pm 2^{-(l-1)}h_j\ Y_{j+1}\ \cdots\right]^T$$

とし，中心差分を用いた近似式 (3.1) に基づいて

$$\mathbf{J}_j^{l,1} = (2^{l-1}h_j)^{-1}\left(2^{-1}\mathbf{F}(\mathbf{Y}^{j+}) - 2^{-1}\mathbf{F}(\mathbf{Y}^{j-})\right)$$

を計算する．最適な h_j は各要素ごとに異なるが，列単位で計算する本手法の場合は最も大きな値に揃える必要がある．今回は任意の j に対して $h_j = 1$ と設定した．

補外計算では以下の表の初期系列より右の下三角成分を求める．

初期系列				
$\mathbf{J}_j^{1,1}$				
$\mathbf{J}_j^{2,1}$	$\mathbf{J}_j^{2,2}$			
$\vdots$	$\vdots$	$\ddots$		
$\mathbf{J}_j^{L-1,1}$	$\mathbf{J}_j^{L-1,2}$	$\cdots$	$\mathbf{J}_j^{L-1,L-1}$	
$\mathbf{J}_j^{L,1}$	$\mathbf{J}_j^{L,2}$	$\cdots$	$\mathbf{J}_j^{L,L-1}$	$\mathbf{J}_j^{L,L}$

ここで各段の計算は,

$$\begin{aligned}\mathbf{J}_j^{l,k} &:= \mathbf{J}_j^{l,k-1} + (4^{k-1}-1)^{-1}\left(\mathbf{J}_j^{l,k-1} - \mathbf{J}_j^{l-1,k-1}\right)\\ &= \mathbf{J}_j^{l,k-1} + \mathbf{R}_j^{l,k}\end{aligned}$$

として行う．この際，行列の列ごとに補外計算を行うため，収束判定については $\mathbf{J}_j^{l,k} = [J_{1j}^{l,k} \cdots J_{ij}^{l,k} \cdots J_{nj}^{l,k}]^T$ の各要素 $J_{ij}^{l,k}$ ごとに行う必要がある．

3.2　収束判定

収束判定式には (2.5) を使用する．この際，$J_{ij}^{l,k}$ の収束判定に使用する補外法の丸め誤差限界 $E_{R,ij}^{l}$ は，(2.4) 式と (2.2) 式より，p 進 q 桁計算においては

$$E_{R,ij}^{l} = \frac{\max\left(\left|F_i(\mathbf{Y}^{j+})\right|, \left|F_i(\mathbf{Y}^{j-})\right|\right) \cdot c \cdot p^{-q}}{h_j / 2^{l-1}} \tag{3.2}$$

となる．

よってこの $E_{R,ij}^{l}$ と, ユーザが与えた $\varepsilon_r, \varepsilon_a$ を用いて，修正項 $\mathbf{R}_j^{l,k} = [R_{1j}^{l,k} \cdots R_{nj}^{l,k}]^T$ の各要素ごとに

$$\left|R_{ij}^{l,k}\right| \le \max\left(\varepsilon_r \left|J_{ij}^{l,k-1}\right| + \varepsilon_a, E_{R,ij}^{l}\right) \tag{3.3}$$

を判定し，これを満足していればその i 番目の要素は収束したと判断する．もしこの基準を満足したときには，次の段ではその i 番目の要素の計算を飛ばして，未収束要素の補外計算のみを続ける．従って，補外計算が停止するのは全ての列要素が収束した時となる．この収束過程を視覚的に表現すると Fig. 1 のようになる．

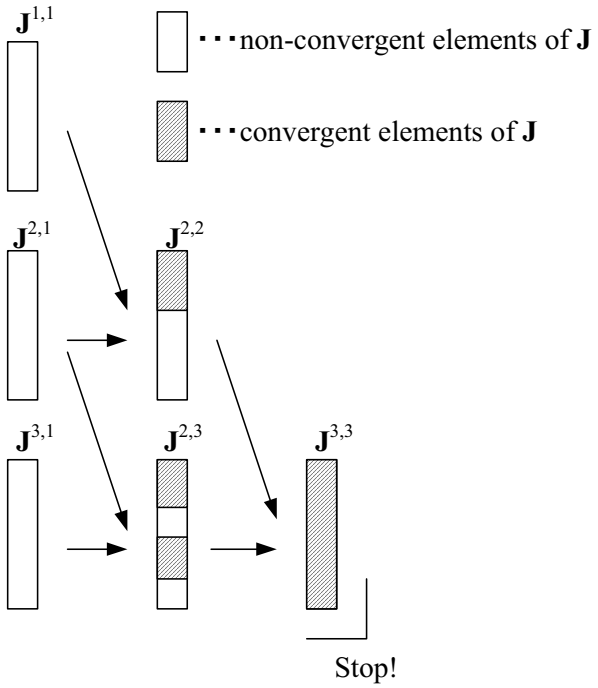

Fig. 1.　Convergent process.

この収束判定により，あくまで列単位の計算を行いつつ，成分ごとの決め細やかな収束判定を行うことができるため，最大 L 段までの補外を実施したとしても，全ての要素を計

算するためには最大 $2nL$ 回の関数 $\mathbf{F}(\mathbf{Y})$ 呼び出しで済む．もし，列単位ではなく，要素単位でこの補外計算を行おうとすると，関数評価回数は $2n^2L$ に激増する．計算時間の多くは初期系列における関数計算で占められているため，これによって全体の計算時間の大幅な縮減が期待できる．

4. 数値実験

数値実験には次のハードウェア・ソフトウェアを使用した．

ハードウェア　AMD Athlon64 X2 3800+, 4GB RAM

ソフトウェア　Fedora Core 4 x86_64, BNCpack [2], MPFR 2.1.2 [7]/GMP 4.1.4 [4], GCC 4.0.2

ここで使用する $\mathbf{F}(\mathbf{Y})$ は

$$F_i(\mathbf{Y}) = \begin{cases} \sin\left(\sum_{k=1}^{n} Y_k\right) & (i \bmod 3 = 0) \\ \cos\left(\sum_{k=1}^{n} Y_k\right) & (i \bmod 3 = 1) \\ \prod_{k=1}^{n} Y_k & (i \bmod 3 = 2) \end{cases} \tag{4.1}$$

である．この時，真の Jacobi 行列は

$$\frac{\partial F_i}{\partial Y_j} = \begin{cases} \cos\left(\sum_{k=1}^{n} Y_k\right) & (i \bmod 3 = 0) \\ -\sin\left(\sum_{k=1}^{n} Y_k\right) & (i \bmod 3 = 1) \\ \prod_{k=1,k\neq j}^{n} Y_k & (i \bmod 3 = 2) \end{cases}$$

となる．次元数 n が大きくなるにつれて，成分の絶対値の大きさが $n!$ 倍まで拡大するため，精度の良い値を求めにくくなる例である．

今回は $n = 30$ 及び $n = 1000$ とし，$\mathbf{Y} = [1\ 2\ \cdots n]^T$ における Jacobi 行列を評価した．また，あえて $\varepsilon_r = \varepsilon_a = 0$ と設定し，丸め誤差限界のみによる収束判定を行った．その結果を Table 1 と Table 2 に示す．左から，計算 bit 数 (10 進桁換算)，$J(\mathbf{Y})$ の相対誤差の最大値，計算時間 (秒)，最大段数である．比較のために，$n = 30$ の時の IEEE754 倍精度 (53bit) 計算の結果も挙げておく．

この結果，全ての計算桁数で (3.3) に基づいた停止則が有効に働いていることが確認できた．特に，$n = 1000$ の時は，$1000! \approx 4.0 \times 10^{2567}$ となるが，このような極端に成分の絶対値の大きさに違いがある場合でも，計算桁数が多ければ (この場合は 512bit 以上)，それに応じた精度が得られていることが分かる．従って，ユーザが小さすぎる $\varepsilon_r, \varepsilon_a$ を指定したとしても，丸め誤差限界値によって計算桁数に応じた精度の Jacobi 行列が得られることが示された．

次に $n = 30$ の時，$\varepsilon_a = 0$ とし，十分な計算桁数が確保されている時に相対許容度 ε_r によってどの程度の精度桁が得られるかを調べてみた．その結果を Table 3 に示す．

Table 1. Numerical results for (4.1): $n = 30$.

# bits (decimal)	max. relative errors	comp. time(sec)	max. #stages
IEEE754 double	1.02E – 5	0.01	6
128(38.5)	7.65E – 37	0.15	9
256(77.1)	2.80E – 74	0.35	13
512(154.1)	2.57E – 149	0.92	19
1024(308.3)	1.28E – 300	2.9	28
2048(616.5)	5.30E – 606	12.4	40
4096(1233)	1.76E – 1216	65.3	58
8192(2466)	2.06E – 2441	358.6	84

Table 2. Numerical results for (4.1): $n = 1000$.

# bits	max. relative errors	comp. time(sec)	max. #stages
128	3.97E – 8	226.4	9
256	2.43E – 12	466.6	13
512	7.73E – 145	1047.6	19
1024	3.80E – 296	3012.8	28
2048	7.17E – 601	11434.8	40

Table 3. Numerical results in 8192 bits computation.

$\log_{10} \varepsilon_r$	max. relative errors	comp. time(sec)	max. #stages
50	2.11E – 51	26.5	10
100	8.90E – 102	41.6	15
200	9.12E – 201	62.6	21
500	7.34E – 506	127.5	36
1000	3.16E – 1005	215.0	52
2000	6.56E – 2001	368.0	75

ε_r の値と，相対誤差がほぼ一致していることがわかる．これにより，十分な計算桁数が確保できている場合は，相対許容度 ε_r によって相対誤差の調整を行うことができる事が示された．

4.1 Testset 問題

常微分方程式の初期値問題を集めた Testset [9] 問題から，Hires 問題 (8 次元, (4.2) 式) と Medakzo 問題 (400 次元, (4.3) 式) に適用する．関数は以下の通り．

$$\mathbf{F}(\mathbf{Y}) = \begin{bmatrix} -1.71Y_1 + 0.43Y_2 + 8.32Y_3 + 0.0007 \\ 1.71Y_1 - 8.75Y_2 \\ -10.03Y_3 + 0.43Y_4 + 0.035Y_5 \\ 8.32Y_2 + 1.71Y_3 - 1.12Y_4 \\ -1.745Y_5 + 0.43Y_6 + 0.43Y_7 \\ -280Y_6Y_8 + 0.69Y_4 + 1.71Y_5 - 0.43Y_6 + 0.69Y_7 \\ 280Y_6Y_8 - 1.81Y_7 \\ -280Y_6Y_8 + 1.81Y_7 \end{bmatrix} \tag{4.2}$$

$$\begin{aligned} F_{2j-1}(t,\mathbf{Y}) &= \alpha_j \frac{Y_{2j+1} - Y_{2j-3}}{2\Delta\zeta} \\ &\quad + \beta_j \frac{Y_{2j-3} - 2Y_{2j-1} + Y_{2j+1}}{(\Delta\zeta)^2} - kY_{2j-1}Y_{2j} \\ F_{2j}(t,\mathbf{Y}) &= -kY_{2j}Y_{2j-1} \quad (j = 1, 2, ..., 199) \\ &\text{ここで} \\ k &= 100,\ c = 4,\ \Delta\zeta = 1/200 \\ \alpha_j &= 2(j\Delta\zeta - 1)^3/c^2, \\ \beta_j &= (j\Delta\zeta - 1)^4/c^2 \quad (j = 1, 2, ..., 200) \\ Y_{-1}(t) &= \begin{cases} 2 & (t \in (0, 5]), \\ 0 & (t \in (5, 20]) \end{cases} \\ Y_{400}(t) &= Y_{399}(t) \end{aligned} \tag{4.3}$$

使用した $\mathbf{Y} = [Y_1\ ...\ Y_n]^T$ は，いずれも $Y_i = i$ とした．また Medakzo 問題においては $t = 0$ とした．

前述の問題同様，IEEE754 倍精度計算も含めた計算結果を Table 4, 5 に示す．いずれも最初のテスト問題同様，ほぼ計算 bit 数に近い相対誤差になっていることが分かる．非線型性が弱いためか，段数も最小で済んでいる．但し，Medakzo 問題では，Hires 問題と異なり，IEEE754 倍精度計算の結果は，計算桁数に比して悪くなっている．

Table 4. Hires Problem.

# bits	max. relative errors	comp. time(sec)	max. #stages
IEEE754 double	3.37E – 12	0.0	2
128	7.65E – 37	0.0	2
256	3.17E – 73	0.0	2
512	4.11E – 150	0.01	2
1024	2.33E – 304	0.01	2
2048	4.87E – 613	0.03	2
4096	3.51E – 1229	0.04	2
8192	5.05E – 2462	0.13	2

Table 5. Medakzo Problem.

# bits	max. relative errors	comp. time(sec)	max. #stages
IEEE754 double	1.03E – 3	0.24	2
128	1.61E – 31	2.0	2
256	5.19E – 70	3.1	2
512	3.90E – 147	5.1	2
1024	3.27E – 301	7.4	2
2048	1.87E – 609	18.2	2
4096	4.58E – 1226	52.6	2
8192	5.98E – 2459	166.6	2

5. 結論

多倍長計算を用いた数値実験により，本手法の有効性が示された．一般的には AD が使用できる関数に対しては計算量の点で不利であると思われるが，AD が使用できない関数や計算環境において，高精度な Jacobi 行列が必要な時には一つの手段として使用できるものである．

謝辞 本論文を執筆するにあたり，永坂秀子先生より多くの助言を頂いた．このアルゴリズムの開発は，静岡理工科大学・大椙弘順助教授との共同研究 [6] が直接のきっかけとなり，研究にあたっては静岡理工科大学より特定研究費の補助を受けた．また，査読者より有益な助言も頂いた．関係者各位に感謝する．

参考文献

[1] Automatic Differentiation, http://www.autodiff.org/.

[2] BNCpack, http://na-inet.jp/na/bnc/.

[3] 福井義成, 数値微分における補外法の誤差, 日本応用数理学会論文誌 15(2005), 521-535.

[4] GNU Multiple Precision Arithmetic Library, http://swox.com/gmp/.

[5] E.Hairer, S.P.Nøsett, G.Wanner, Solving Ordinary Differential Equations I, Springer-Verlag, Berlin, 1991.

[6] 幸谷智紀・大椙弘順, 多倍長計算を用いた仮想化細胞間モデルの漸近安定性解析, 京都大学数理解析研究所講究録, 1499(2006), 202-207.

[7] MPFR Project, http://www.mpfr.org/.

[8] 永坂秀子・福井義成, 数値微分の誤差, 情報処理学会論文誌, 22(1981), 411-416.

[9] Test Set for Initial Value Problem Solvers, http://pitagora.dm.uniba.it/~testset/.

幸谷智紀 (正会員)　〒437-8555 静岡県袋井市豊沢 2200-2

1993 年日本大学大学院博士前期課程修了，同年石川職業能力開発短期大学校講師，1997 年日本大学大学院博士後期課程修了，1999 年静岡理工科大学講師，現在に至る．数値計算，誤差解析の研究に従事．SIAM，情報処理学会，日本数学会各会員．博士 (理学).