数理解析研究所講究録
1514巻 2006年 183-189

Taylor級数の逆関数計算の高速化とその応用

舘野 裕文
HIROFUMI TATENO
神奈川工科大学工学研究科
GRADUATE SCHOOL OF ENGINEERING, KANGAWA INSTITUTE OF TECHNOLOGY
平山 弘
HIROSHI HIRAYAMA
神奈川工科大学工学部
FACULTY OF ENGINEERING, KANGAWA INSTITUTE OF TECHNOLOGY*

1 はじめに

関数をTaylor展開することは、自動微分 [3] と呼ばれる方法と数学的には同じものであり、四則演算、微積分、初等関数などを定義することができる。Taylor級数の係数を浮動小数点で表現することで、高速にTaylor級数 [2] の演算を行うことができる。倍精度浮動小数点の数値計算で係数に倍精度浮動小数点を使った場合は、係数値に左右されるが一般的に20〜30次でのTaylor級数の演算が必要である。

Taylor級数の逆関数計算は応用上重要なものである。逆関数計算では関数を原点に合わせてから計算をする。[2] 任意の展開点での逆関数を求めるためには、その展開点での値が出るように初期の展開点を少しずつずらして繰り返し演算を行う。このため、逆関数計算は高速に計算できる必要がある。

本研究ではTaylor級数が与えられた時の逆関数計算の方法を検討する。逆関数の計算方法として逆関数を満たす微分方程式をPicardの逐次近似法で解く方法、古くから知られているLagrange inversion formula、形式的Newton法の高速化を行い、これらを比較して個々の長所短所を示す。

Taylor級数の逆関数計算の応用例として、Taylor級数は微分が簡単なことを利用してGamma関数の極大極小値が容易に求めることができることを示す。
さらに零点のTaylor展開として

$$\sin x = x\cos x \tag{1}$$

の方程式の解を求める。

Taylor級数の実装は関数及び演算子オーバーロードの可能なFortran90、C++言語、C#言語が有用である。演算子オーバーロードの使えない言語では、利用方法が煩雑となり、事実上利用不可能になってしまう。本研究では、C++言語で実装し、メンバ関数の中では演算子オーバーロードを使用せずに計算している。これは、余計な計算を省き、コンパイラの最適化がかかり易くするためである。またTaylor級数の係数や展開点は倍精度浮動小数点を使って表現する。

本研究の実験環境として、CPUはPentium4 2.20GHz、メモリを256Mで、Red Hat Linux9(Kernel 2.4.20-8)上において、コンパイラにGCC 3.2.2-5(-O3 -march=pentium4)を使用して実験を行う。

*hirayama@sd.kanagawa-it.ac.jp

2 逆関数計算法

2.1 Picard の逐次近似法

Picard の逐次近似法 [1] を使った逆関数の計算は、$y = f(x)$ の逆関数が $x = f(y)$ となるのを利用し、この両辺を微分して

$$\frac{dy}{dx} = \frac{1}{f'(y)} = v(y) \tag{2}$$

という微分方程式を得る。これを Picard の逐次近似法を使い、

$$y(x_0) = y_0,\ y_n(x) = y_0 + \int_{x_0}^{x} v\left(y_{n-1}(t)\right) dt \tag{3}$$

で繰り返し演算を行い、逆関数を求める。

展開点 f_p で n 次までの Taylor 級数 $f(x)$ は

$$f(x) = f_0 + f_1 (x - f_p) + f_2 (x - f_p)^2 + f_3 (x - f_p)^3 + \cdots + f_n (x - f_p)^n \tag{4}$$

となる。式 (3) を Taylor 級数で解くために、$f(x)$ を x 軸方向に $-f_p$、y 軸方向に $-f_0$ 平行移動した関数を $w(x)$ と定義し、初期条件が $w(0) = 0$ になるようにする。これを使って式 (3) を解く。$w(x)$ の零点で n 次までの Taylor 級数は次のようになる。

$$w(x) = f_1 x + f_2 x^2 + f_3 x^3 + \cdots + f_n x^n \tag{5}$$

この関数の逆関数 $w^{-1}(x)$ を求め、これを平行移動することで $f^{-1}(x)$ を求める。ただし、$f^{-1}(x)$ の Taylor 展開点を f_p^{-1} とする。これは

$$v(x) = \frac{1}{w'(x)} = \frac{1}{f_1 + 2f_2 x + 3f_3 x^2 + \cdots + nf_n x^{n-1}} \tag{6a}$$

$$w_1^{-1}(x) = \frac{1}{f_1} x,\quad w_k^{-1}(x) = \int_0^x v\left(w_{k-1}^{-1}(t)\right) dt \qquad (k = 2, \cdots, n) \tag{6b}$$

$$f_p^{-1} = f_0,\quad f^{-1}(x) = w_n^{-1}(x) + f_p \tag{6c}$$

となる。式 (6b) の $w^{-1}(x)$ の下添え字は繰り返し回数である。Picard の逐次近似法は一回の繰り返しで一つ高次の係数まで正確に求めることができる。逆関数は初期値として一次まで求められるので繰り返しは $n-1$ となる。k 回目の計算では k 次の係数までは正確に求まるので、k 次までしか計算しないことで計算量を削減できる。ここまでがほぼ公式通りの計算方法である。

w_k^{-1} は前ステップの計算において $k-1$ 次まで正確に求まっているので k 次の部分のみの計算となる。この最適化を行うと式 (6a)、(6b)、(6c) は

$$v(x) = \frac{1}{w'(x)} = \frac{1}{f_1 + 2f_2 x + 3f_3 x^2 + \cdots + nf_n x^{n-1}} \tag{7a}$$

$$w_1^{-1}(x) = \frac{1}{f_1} x,\quad h_k(t) = v\left(w_{k-1}^{-1}(t)\right),\quad w_k^{-1}(x) = w_{k-1}^{-1} + \frac{h_{k,k-1}}{k} x^k \qquad (k = 2, \cdots, n) \tag{7b}$$

$$f_p^{-1} = f_0,\quad f^{-1}(x) = w_n^{-1}(x) + f_p \tag{7c}$$

となる。ただし、$h_k(t)$ の下添え字 k は繰り返し回数であり、$h_{k,k-1}$ は Taylor 展開した $h_k(t)$ の $k-1$ 次の係数とする。式 (7b) の合成関数は $k-1$ 次までの計算になる。さらに式 (7b) の合成関数はべき乗計算が主で、この計算は k 回目の計算で $k-2$ 次まで前ステップで計算しているので、その結果を保持しておけば $k-1$ 次の部分だけの計算となり、計算量の大幅な削減ができる。

2.2 Lagrange inversion formula

Lagrange inversion formula [5] は $y = f(x)$ を

$$g(x) = f(x) - f(0) \tag{8a}$$

$$w = \frac{x}{g(x)} \tag{8b}$$

$$c_n = \frac{1}{n!}\left\{\left(\frac{d}{dx}\right)^{n-1} w^n\right\}_{x=0} \tag{8c}$$

$$f^{-1}(x) = \sum_{n=1}^{\infty} c_n \left(x - f(0)\right)^n \tag{8d}$$

として逆関数を求める。ただし、$f^{-1}(x)$ は $f(0)$ での Taylor 級数である。
これを Taylor 級数の形で展開、整理して、式 (8c) のべき乗の部分を乗算で解いていった場合は

$$w_0 = \frac{1}{f_1}, \quad w_j = -w_0 \sum_{i=0}^{j-1} w_i f_{j-i+1} \qquad (j = 1, \cdots, n) \tag{9a}$$

$$h_{0,0} = 1, \quad h_{0,k} = 0, \quad h_{j,k} = \sum_{i=1}^{k} w_i \, h_{j,k-i} \qquad (j = 1, \cdots, n \qquad k = 1, \cdots, n) \tag{9b}$$

$$f_p^{-1} = f_0, \quad {f^{-1}}_0 = f_p, {f^{-1}}_i = \frac{h_{i,i-1}}{i} \qquad (i = 1, \cdots, n) \tag{9c}$$

となり、べき乗で解いていった場合は

$$w_0 = 1, \quad w_j = -\frac{1}{f_1} \sum_{i=0}^{j-1} w_i f_{j-i+1} \qquad (j = 1, \cdots, n) \tag{10a}$$

$$h_{j,0} = \left(\frac{1}{f_1}\right)^j, \quad h_{j,k} = \frac{1}{k} \sum_{i=1}^{k} \{(j+1)\, i - k\}\, w_j \, h_{j,k-i} \qquad (j = 1, \cdots, n \qquad k = 1, \cdots, j-1) \tag{10b}$$

$$f_p^{-1} = f_0, \quad {f^{-1}}_0 = f_p, \quad {f^{-1}}_i = \frac{h_{i,i-1}}{i} \qquad (i = 1, \cdots, n) \tag{10c}$$

となる。

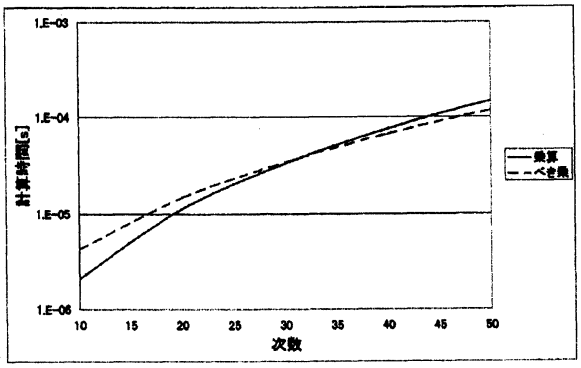

図 1: Lagrange inversion formula

乗算で解いた場合とべき乗で解いた場合を比較すると図1のようになる。

30次を超えた辺りから乗算よりもべき乗が速く計算できるのは、乗算の場合は計算結果を再利用するために最終的に必要な n 次までの計算を始めから行うのに対して、べき乗は計算コストは乗算より高いが $j-1$ 次までの計算ですむ為である。

この結果から20〜30次までの範囲では、乗算を使った方法が優れている。

2.3 形式的 Newton 法

$y=f(x)$ の逆関数は、$x=f(y)$ となるので、

$$g(y)=x-f(y)=0 \tag{11}$$

を Newton 法を使って解くことで、逆関数を求めることができる。Newton 法の一般式は

$$y_{i+1}=y_i-\frac{f(y_i)}{f'(y_i)} \tag{12}$$

なので、式 (11) を適応すると

$$g_{i+1}=g_i-\frac{f(g_i)-x}{f'(g_i)} \tag{13}$$

となる。これを Taylor 級数に適応すると

$$\begin{aligned} g_{i+1}&=g_i-\frac{f(g_i)-x}{f'(g_i)} \\ g(x)&=g_{lim}+f_p \end{aligned} \tag{14}$$

となる。ただし、$g(x)$ の展開点は0である。Newton 法の誤差は二乗の速度で収束するので n 次まで求めるならば

$$lim=\mathrm{floor}\left(\log_2(n+1)\right)+2 \tag{15}$$

まで繰り返せば求めることができる。

このままでも計算は可能であるが、問題によっては誤差が非常に大きくなってしまうため、関数を原点にずらして解く。これは、

$$h(x)=f(x)-f(f_p) \tag{16a}$$

$$g_0=\frac{1}{h'(0)}x,\quad g_{i+1}=g_i-\frac{h(g_i)-x}{h'(g_i)} \tag{16b}$$

$$g(x)=g_{lim}+f_p \tag{16c}$$

となり、初期値として一次まで求められるので繰り返し回数 lim は

$$lim=\mathrm{floor}\left(\log_2(n+1)\right) \tag{17}$$

になる。

2.4 比較実験

最適化した Picard の逐次近似法は、Lagrange inversion formula に比べて 20 次で約 2.6 倍、30 次で約 3.8 倍速く、ほぼ公式通りの Picard の逐次近似法とでは 20 次で約 113 倍、30 次で約 258 倍速い。形式的 Newton 法が波打っているのは次数が二の倍数のときに計算効率がよくなるからである。また、100 次まで実験行っても 20 次以降の性能順位は変わらなかった。

この結果から最適化した Picard の逐次近似法がもっとも優れており、次に低次の場合は Lagrange inversion formula、高次の場合は形式的 Newton 法が優れている。この計算速度の優位性は他のマシンを使ったとしても変わらないと考えられる。最適化した Picard の逐次近似法がもっとも優れているのは他の方法に比べ、計算結果の使い回しをできる箇所が多く、そのため計算回数が少なくなっていることのよるものだと考えられる。

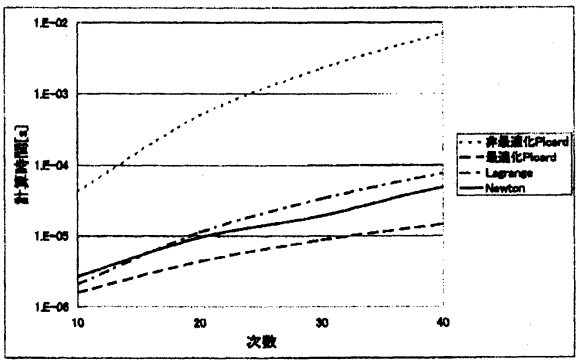

図 2: 逆関数計算の速度比較

3 逆関数の応用例

3.1 Gamma 関数の極値

Gamma 関数の極大値及び極小値を計算する。Gamma 関数として式 (18) を使う。

$$\Gamma(x) = \frac{\Gamma(N+x)}{\prod_{k=0}^{N-1}(k+x)} \tag{18}$$

式 (18) の分子は Stirling の公式 [4] を利用して

$$\log\Gamma(x) \approx \left(x-\frac{1}{2}\right)\log x - x + \frac{1}{2}\log(2\pi) + \sum_{m=1}^{\infty}\frac{B_{2m}}{2m(2m-1)x^{2m-1}} \tag{19}$$

から計算する。

まず式 (18) の x に $x+a$ を代入して展開点 a での Taylor 級数を求める。初期値を $a=1.5$ とし、$N=20$ で 19 次まで Taylor 級数を計算すると Gamma 関数は

$$0.8862 + 0.03234*(x-1.5) + 0.4148*(x-1.5)^2 - 0.1073*(x-1.5)^3 + 0.1446*(x-1.5)^4 - 0.07752(x-1.5)^5\cdots \tag{20}$$

表 1: Gamma 関数の極値とその位置

初期値	極値の位置	極大極小値	繰り返し
1.5	1.461632144968	8.856031944108e-001	1
-0.5	-0.504083008264	-3.544643611155e+000	1
-1.5	-1.573498473162	2.30240725833e+000	2
-2.5	-2.610720868444	-8.881363584012e-001	2
-3.5	-3.635293366436	2.451275398343e-001	2
-4.5	-4.653237761743	-5.277963958731e-002	2
-5.5	-5.667162441556	9.324594482614e-003	2
-6.5	-6.678418213073	-1.397396608949e-003	2
-7.5	-7.687788325031	1.818784449094e-004	2
-8.5	-8.695764163816	-2.09252904465e-005	2
-9.5	-9.702672540001	2.157416104522e-006	2

となり、これを微分すると

$$0.03234 + 0.8296 * (x - 1.5) - 0.3219 * (x - 1.5)^2 + 0.5786 * (x - 1.5)^3 - 0.3876 * (x - 1.5)^4 + 0.3517 * (x - 1.5)^5 \cdots \quad (21)$$

となる。これから Gamma 関数を微分した式の逆関数を計算すると

$$1.5 + 1.205 * (x - 0.03234) + 0.5637 * (x - 0.03234)^2 - 0.6941 * (x - 0.03234)^3 - 1.253 * (x - 0.03234)^4 + 0.5992 * (x - 0.03234)^5 \cdots \quad (22)$$

となる。この式に $x = 0.03234$ を代入し、零点を求めると

$$x = 1.461632144968362 \quad (23)$$

が得られる。よって極値の位置を求めることができ、これを式 (18) に代入することで

$$\Gamma(1.461632144968362) = 0.8856031944108 \quad (24)$$

となる。求まった極値の位置を初期値として繰り返し計算することで任意の精度で極値を求めることができる。また初期値を変えて計算することで任意の位置での極値を求めることができる。初期値を変えて誤差が 10^{-15} 以下になるまで計算した結果を表 1 に示す。

これによって微積分を含んだ式も容易に扱えることがわかり、変曲点なども求めることができることがわかる。

3.2 零点の Taylor 展開

次の方程式を考える。

$$\sin x = x \cos x \quad (25)$$

この方程式の解は

$$x_n = n\pi + \frac{\pi}{2} - a_n \tag{26}$$

である。式 (25) を

$$\tan x = x \tag{27}$$

と変形し、式 (26) を代入すると

$$\cot a_n = n\pi + \frac{\pi}{2} - a_n \tag{28}$$

となり、$t = \frac{1}{n\pi}$ として t について整理すると

$$t = \frac{\sin a_n}{\cos a_n - \frac{\pi}{2}\sin a_n + a_n \sin a_n} \tag{29}$$

となる。この式は $t = 0$ のとき、$a_n = 0$ である。このことから式 (29) の逆関数を求めることで a_n を求めることができ、この結果を式 (26) に代入することで方程式の解を求めることができる。

これを計算すると

$$x_n = n\pi + \frac{\pi}{2} - \frac{1}{n\pi} + \frac{1.5708}{(n\pi)^2} - \frac{3.13407}{(n\pi)^3} + \frac{7.01738}{(n\pi)^4} - \frac{16.8243}{(n\pi)^5} + \frac{42.2085}{(n\pi)^6} - \frac{109.369}{(n\pi)^7} \cdots \tag{30}$$

という結果を得ることができる。

これによって無限遠点における Taylor 級数を求めることが可能であることがわかる。

4 おわりに

倍精度浮動小数点での数値計算における Taylor 級数の逆関数計算で Taylor 級数が与えられた場合は、最適化を行った Picard の逐次近似法がもっとも優れている。

Taylor 級数の逆関数計算は微積分を含んだ式に対して有用であり、零点や変曲点などを容易に求めることができる。また無限遠点における Taylor 級数を求める計算を行うことも可能である。

[1] Erwin Kreyszig.: Advanced Engineering Mathematics, John Wiley & Sons Inc, 1983.

[2] 平山弘, 小宮聖司, 佐藤創太郎: Taylor 級数法による常微分方程式の解法, 日本応用数理, Vol. 12, No. 1, pp. 1–8, 2002.

[3] Louis B. Rall.: Automatic Differentiation: Techniques and Applications, Springer-Verlag, 1981.

[4] Milton Abramowitz and Irene A.Stegun.: Handbook of Mathematical Functions, With Formulas, Graphs, and Mathematical Tables, Dover Pubns, 1972.

[5] N. De Bruijn.: Asymptotic Methods in Analysis, Dover Pubns, 1981.