Vol.2021-HPC-178 No.4
2021/3/15

数学関数の数値的級数展開法

平山 弘[1,a)]　小宮 聖司[1,b)]

概要：有限項で打ち切った Taylor 級数の四則演算および関数計算は、プログラミング言語を使えば容易に行うことができる。C++言語や Fortran 等のオペラレーター・オーバーロード機能を使えば非常に使いやすくなる。通常の数値のように扱うことができる。
このプログラムを使うと、これらのプログラミング言語で記述された関数は容易に Taylor 展開できる。ループや条件文を含む関数でも容易に Taylor 展開できることを示す。逆関数は、微分方程式で定義できるので、微分方程式をピカールの逐次近似法 (Picard iteration) を使えば容易に Taylor 展開できる。
この方法を使って、ガンマ関数の逆数の数値 Taylor 展開を、100 桁の精度で 100 次まで計算した。従来の展開式は、16 桁の精度で 26 次まであったので大幅に精度および次数を拡張した。これによって、30 桁精度や 60 桁精度以上のガンマ関数も容易に計算することができる。

キーワード：数値的 Taylor 展開，高精度高次 Taylor 級数，微分方程式の級数展開法

Numerical Series Expansion Method for Mathematical Functions

Hiroshi Hirayama[1,a)]　Komiya Seiji[1,b)]

Abstract: The arithmetic operations and function calculations of Taylor series truncated by finite terms can be easily defined using programming languages. Using an operator overload function such as C ++ or Fortran makes it very easy to use. It can be treated like a numerical calculation.
Using this program, functions written in these programming languages can easily be expanded into Taylor series.
It is shown that Taylor expansion can be easily performed even with functions including loops and conditional statements. Since the inverse function can be defined by a differential equation, the differential equation can be easily Taylor-expanded by using Picard iteration.
Using this method, the reciprocal Taylor expansion of the gamma function was calculated to the 100th order with 100-digit precision. The conventional expansion formula has 16-digit precision up to 26th order, so the precision and order have been greatly expanded. This makes it possible to easily calculate gamma functions with 30-digit precision or 60-digit precision or higher.

Keywords: Numerical Taylor series, High-precision high-order Taylor series, series expansion method for differential equations

1. はじめに

Taylor 級数の係数を浮動小数点数の配列で表現し、有限項で打ち切った Taylor 級数をここでは Taylor 級数と呼ぶことにする。オペレータ・オーバロード機能を持つ C++言語、Fortran 等を使うと、これらの Taylor 級数間の四則演算や関数計算を通常の数値計算のように扱うことができる。

このように計算された Taylor 級数の係数は、関数値、1 階および高階微分係数を階乗で割った数値になる。微分係数の計算に差分近似による計算を行っていないので、打ち

[1] 神奈川工科大学創造工学部自動車システム開発工学科
Department of Vehicle System Engineering, Faculty of Creative Engneering, Kanagawa Institute of Technology, Shimo-Ogino 1030, Atsugi, Kanagawa, 243-0292, Japan
[a)] hirayama@kanagawa-it.ac.jp
[b)] kom@eng.kanagawa-it.ac.jp

切り誤差が入らないため通常の数値計算と同様に高精度で計算出来る。

本論文では、この Taylor 展開法を簡単に説明し、いろいろな関数の展開例を示す。特に通常の Taylor 展開法を適用するのが困難な例としてガンマ関数を示す。

以下の 2 節で述べるように微分方程式を満たす関数は、微分方程式が Taylor 展開式の係数の漸化式となるため、効率的に計算ができる。

本論文ではガンマ関数の Taylor 展開を行う。ガンマ関数の数値的 Taylor 展開式は、Abramowitz 等の公式集 [1] に載るような Taylor 展開式で、最初に計算されたのが 1883 年で、一部修正されているが現在もガンマ関数の計算に使われているものである。計算は 16 桁の精度で 26 次まで計算されている。ここでは、ガンマ関数の逆数関数を原点について、100 桁精度で、100 次まで計算した。これによって、ガンマ関数の 4 倍精度や 8 倍精度の関数が容易に計算できるようになった。さらにガンマ関数の関連関数の Psi 関数や polygamma 関数の Taylor 展開式もこれから容易に計算できる。

2. Taylor 級数の計算法

ここでは、関数を Taylor 級数に展開する方法を簡単に説明する。詳しくは Rall[8] および 平山等 [6] を参照せよ。

展開位置が同じ Taylor 級数間の演算は、一般性を失うことなく、原点で展開された Taylor 級数であると仮定することができる。Taylor 級数の展開位置は、平行移動により任意の位置に移動できるため、原点で展開された Taylor 級数のみを考慮すれば十分である。n 次の Taylor 級数を次のように定義する。

$$f(x) = f_0 + f_1 x + f_2 x^2 + f_3 x^3 + f_4 x^4 \cdots + f_n x^n \quad (1)$$

$$g(x) = g_0 + g_1 x + g_2 x^2 + g_3 x^3 + g_4 x^4 \cdots + g_n x^n \quad (2)$$

$$h(x) = h_0 + h_1 x + h_2 x^2 + h_3 x^3 + h_4 x^4 \cdots + h_n x^n \quad (3)$$

2.1 Taylor 級数の四則演算

n 次の Taylor 級数の四則演算プログラムは容易に作成できる。以下の公式は、原点で展開された級数だけでなく、任意の点で展開された Taylor 級数でも有効である。

(1) 加減算 $h(x) = f(x) \pm g(x)$

$h(x) = f(x) \pm g(x)$ となるので，この式に (1) の $f(x)$，(2) の $g(x)$ および (3) の $h(x)$ を代入し、同じ次数の係数を等しいとすると次の関係式が得られる。

$$h_j = f_j \pm g_j (j = 0, \cdots, n)$$

(2) 乗算 $h(x) = f(x)g(x)$

$h(x) = f(x)g(x)$ となるので，この式に (1) の $f(x)$，(2) の $g(x)$ および (3) の $h(x)$ を代入し、同じ次数の係数を等しいとすると次の関係式が得られる。

$$h_j = \sum_{k=0}^{j} f_i g_{j-i} (j = 0, \cdots, n)$$

(3) 除算 $h(x) = \dfrac{f(x)}{g(x)}$

$h(x) = \dfrac{f(x)}{g(x)}$ であるから，この式に (1) の $f(x)$，(2) の $g(x)$ および (3) の $h(x)$ を代入し、同じ次数の係数を等しいと置く。$g_0 \neq 0$ ならば、次のように $h(x)$ の係数が計算できる。

$$h_0 = \frac{f_0}{g_0}, \qquad h_j = \frac{1}{g_0}\left(f_j - \sum_{k=0}^{j-1} h_k g_{j-k}\right) \quad (j \geq 1)$$

もし $f_i = g_i = 0$ $(i = 0, \cdots, j)$ ならば、分子と分母を x^{j+1} で割り、その式を上の計算式で除算を行うことができる。

(4) 逆数 $h(x) = \dfrac{1}{f(x)} = rcp(f(x))$

逆数を $h(x)$ とすると $h(x) = \dfrac{1}{f(x)}$ である。これから、$h(x)f(x) = 1$ となる。この式に (1) と (3) を代入し展開し、同じ次数の係数を等しいと置く。$g_0 \neq 0$ ならば、逆数の係数は次の式によって計算することができる。

$$h_0 = \frac{1}{f_0}, \qquad h_n = -\frac{1}{f_0}\sum_{k=0}^{n-1} h_k f_{n-k} \qquad (k = 1, \cdots, m)$$

2.2 Taylor 級数の数学関数計算

多くの基本関数は、単純な微分方程式を満たす。この微分方程式を利用することにより、Taylor 級数の関数を容易に計算できる。

(1) 指数関数 $h(x) = e^{f(x)}$

もし $h(x) = e^{f(x)}$ ならば、次の微分方程式を満たす。

$$\frac{dh(x)}{dx} = h(x)\frac{df(x)}{dx}$$

(1) と (3) をこの微分方程式に代入し、同じ次数の係数を比較すると次の関係式が得られる。

$$h_0 = e^{f_0}, \qquad h_j = \frac{1}{j}\sum_{k=1}^{j} k h_{j-k} f_k \quad (j \geq 1)$$

$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots$ の x に $f(x)$ を代入しても、計算できるが上の漸化式を計算する方法は、効率的に計算できる。定数項だけを計算することを考える。

$$h_0 = 1 + f_0 + + \frac{f_0^2}{2!} + \frac{f_0^3}{3!} + \cdots$$

f_0 が小さい数値ならば効率的に計算できるが、5 とか 10 程度以上の数値だとすると値が収束するまで計算しなければならないなど非効率的である。また、-5 とか-10 程度以下の数値だとすると桁落ちが生じ、計算効率だけでなく計算精度も悪くなる。この定数項はプログラミング言語に準備されている指数関数を使って $\exp(f_0)$ と計算すべきである。1 次の項の係数は、定数項と同じ方法でも計算できる

が、指数関数は微分方程式を満たすため、係数間には漸化式が成り立つ。$h_1 = h_0 f_1$ の関係式から容易に計算できる。

n 次の項の係数も (n-1) 次以下の項の係数から、係数間に成り立つ漸化式を利用して容易に計算できる。この性質は、以下の多くの関数で成り立つ。

(2) 対数関数 $h(x) = \log f(x)$

もし $h(x) = \log f(x)$ ならば、次の微分方程式を満たす。

$$f(x)\frac{dh(x)}{dx} = \frac{df(x)}{dx}$$

(1) と (3) をこの微分方程式に代入し、同じ次数の係数を比較すると次の関係式が得られる。この式から、次のような関係式が得られる。

$$\begin{aligned} h_0 &= \log f_0, \\ h_j &= \frac{1}{jf_0}\left(nf_j - \sum_{k=1}^{j-1} kh_k f_(j-k)\right), \quad (j = 1, \cdots, m) \end{aligned}$$

(3) べき乗関数 $h(x) = f(x)^\alpha (\alpha$ は定数)

もし $h(x) = f(x)^\alpha (\alpha$ は定数) ならば、次の微分方程式を満たす。

$$f(x)\frac{dh(x)}{dx} = \alpha\frac{df(x)}{dx}h(x)$$

(1) と (3) をこの微分方程式に代入し、同じ次数の係数を比較すると次の関係式が得られる。

$$h_0 = f_0^\alpha, \qquad h_j = \frac{1}{jf_0}\sum_{k=1}^{j}(k(\alpha+1) - j)f_k h_{j-k} \quad (j \geq 1)$$

$\alpha = \frac{1}{2}$ とすると平方根を計算できる。

(4) 三角関数 $g(x) = sinf(x), \quad h(x) = \cos f(x)$

もし $g(x) = sinf(x), \quad h(x) = \cos f(x)$ ならば、次の連立微分方程式を満たす。

$$\begin{aligned} \frac{dg(x)}{dx} &= h(x)\frac{df(x)}{dx} \\ \frac{dh(x)}{dx} &= -g(x)\frac{df(x)}{dx} \end{aligned}$$

(1)、(2) と (3) をこの微分方程式に代入し、同じ次数の係数を比較すると次の関係式が得られる。

$$\begin{aligned} g_0 &= \sin f_0, \qquad g_j = \frac{1}{j}\sum_{k=1}^{j} kh_{j-k}f_k \\ h_0 &= \cos f_0, \qquad h_j = -\frac{1}{j}\sum_{k=1}^{j} kg_{j-k}f_k \qquad (j = 1, \cdots, n) \end{aligned}$$

三角関数は、このように sin と cos を同時に計算すると、計算式が単純で見易い公式となる。sin と cos を同時に計算する関数 sin_cos(x,s,c) を準備すると便利である。tan はこのようにして得られた sin と cos の Taylor 級数をわり算することによって得る。この事情は、以下の双曲線関数 sinh と cosh の場合も同様である。

(5) 微分 $h(x) = \frac{df(x)}{dx}$

$h(x) = \frac{df(x)}{dx}$ のとき、次のような関係式が得られる。

$$h_n = 0, \quad h_j = (j+1)f_{j+1} \qquad (j = 0, \cdots, n) \qquad (4)$$

このように、最高次数の係数 h_n は、0 とする。

(6) 積分 $h(x) = \int_0^x f(t)dt$

$h(x) = \int_0^x f(t)dt$ のとき次のような関係式が得られる。

$$h_0 = 0, \quad h_j = \frac{1}{j}f_{j-1} \qquad (j = 1, \cdots, n) \qquad (5)$$

定数項は、積分定数なので、任意で良いが、ここで作成したプログラムでは、0 とする。

(7) 逆三角関数

逆三角関数には次の 3 関数がある。

$$\begin{aligned} h(x) = \sin^{-1} f(x) &= \text{asin}(f(x)) \\ h(x) = \cos^{-1} f(x) &= \text{acos}(f(x)) \\ h(x) = \tan^{-1} f(x) &= \text{atan}(f(x)) \end{aligned}$$

これらの関数を微分すると、次の式が成り立つ。

$$\begin{aligned} \frac{d}{dx}\sin^{-1} f(x) &= \frac{f'(x)}{\sqrt{1-f(x)^2}} \\ \frac{d}{dx}\cos^{-1} f(x) &= -\frac{f'(x)}{\sqrt{1-f(x)^2}} \\ \frac{d}{dx}\tan^{-1} f(x) &= \frac{f'(x)}{1+f(x)^2} \end{aligned}$$

定数項を考慮して、積分すると次の式が成り立つ。

$$\begin{aligned} \sin^{-1} f(x) &= \sin^{-1} f_0 + \int_0^x \frac{f'(t)}{\sqrt{1-f(t)^2}}dt \\ \cos^{-1} f(x) &= \cos^{-1} f_0 - \int_0^x \frac{f'(t)}{\sqrt{1-f(t)^2}}dt \\ \tan^{-1} f(x) &= \tan^{-1} f_0 + \int_0^x \frac{f'(t)}{1+f(t)^2}dt \end{aligned}$$

他の数学関数についても、同様な微分方程式が得られる場合がある。得られた微分方程式から、Taylor 級数の数学関数を計算することができる。

これらの計算式から分かるように、n 次の Taylor 級数が与えられれば、それらの級数の加減乗算、指数対数関数、三角関数等が n 次まで正確に計算できる。

3. Taylor 級数の具体的な計算例

関数を Taylor 級数展開するには、次の公式を使う。

$$f(a+x) = f(a) + f'(a)x + \frac{f''(a)}{2!}x^2 + \cdots + \frac{f^{(n)}}{n!}x^n + \cdots (6)$$

関数 $f(x)$ を $x = a$ で Taylo 展開するには、次の公式によって行うことができる。

$$\begin{aligned} f(x) &= f(a + (x-a)) = f(a) + f'(a)(x-a) \\ &+ \frac{f''(a)}{2!}(x-a)^2 + \cdots + \frac{f^{(n)}}{n!}(x-a)^n + \cdots \quad (7) \end{aligned}$$

この公式は式 (6) で x を $(x-a)$ に変更することによって、得ることができる。$(x-a)$ を t と置いて、x に $a+t$ を代入して、t の Taylor 級数を計算する。その後 t を $(x-a)$ と置き換えると (7) の Taylor 級数が得られる。

3.1 逆関数の計算

ここでは，逆関数の Taylor 展開法について述べる．関数の逆関数の Taylor 展開法については Lagrange inversion theorem[3], [7] などいろいろな方法が知られている。ここでは、微分方程式の級数展開法 [5] によって求める。関数 $y=f(x)$ の逆関数は $y=f^{-1}(x)$ と書くことにする。

関数 $y=f(x)$ の逆関数の微分方程式は、$f(y)=x$ を微分することによって、次のように微分方程式が得られる。

$$y'(x)=\frac{1}{f'(y)} \qquad \text{初期条件} \quad y_0=f^{-1}(x_0)$$

これを、Taylor 級数を使ったピカールの逐次近似法 (Picard iteration) で求める。

ピカールの逐次近似法とは、微分方程式

$$y'(x)=f(x,y) \qquad \text{初期条件} \quad y_0=f(x_0) \tag{8}$$

を反復公式

$$y_n(x)=y_0+\int_{x_0}^{x} f(t,y_{n-1}(t))dt \qquad (n=1,\cdots) \tag{9}$$

を繰り返し計算し、解 $y(x)$ の近似値を求める方法である。

最初の計算は、$n=1$ として (9) の右辺の被積分関数に初期値を代入する。初期値 $y_0(x)$ は定数であり、t の定数部分は 0 であるから、(9) の右辺の被積分関数は $f(0,y_0(t))$ も定数になる。この結果を (9) に代入し、積分すると次数が 1 次増加し、1 次の Taylor 展開式が得られる。$y_1(x)$ は、$y(x)$ の 1 次まで正しい Taylor 展開式になる。

これを再度、$n=2$ として (3.1) の右辺の被積分関数に $y_1(t)$ および t を代入する。$y_1(t)$ および t は 1 次の Taylor 展開式であるから、$f(t,y_1(t))$ は、1 次まで正しい Taylor 展開式が得られる。

一般に k 次の Taylor 展開式と k 次の Taylor 展開式の演算は、k 次まで正しい Taylor 展開式が得られるからである。

これを積分すれば、$y_2(x)$ は 2 次まで正しい Taylor 展開式が得られる。このような操作を n 回繰り返せば、$y_n(x)$ は n 次まで正しい Taylor 展開式が得られる。

逆関数の計算例

$f(x)=e^{-x}-x$ の逆関数を Taylor 展開式を計算する。逆関数が満たす微分方程式 (3.1) は次のようになる。初期値は、$x=0$ のとき $f(0)=1$ であるから、逆にして、$x=1$ のとき $y=0$ となる。したがって方程式は次のようになる。

$$\frac{dy}{dx}=-\frac{1}{e^{-y(x)}+1} \qquad y(1)=0 \tag{10}$$

$y(x)$ の $x=1$ に置ける微分係数は、(10) の右辺の被積分関数に $x=1$ を代入すると $y'(1)=-\frac{1}{2}$ となる。この結果を (9) に代入すると

$$y_1(x)=\int_1^x -\frac{1}{2}dt=-\frac{1}{2}(x-1) \tag{11}$$

が得られる。この 1 次式の結果を再度 (9) の右辺の被積分関数に代入する。1 次までは正しい式を代入するの結果も 1 次までは正しい結果を得ることができる。(10) の分母を 1 次まで計算する。指数関数の taylor 展開 2.2 から

$$-e^{-y_1(x)}-1=-1+\frac{1}{2}(x-1)-1=-2+\frac{1}{2}(x-1)$$

この式の逆数を 2.1 にしたがって計算する。

$$rcp(-2+\frac{1}{2}(x-1))=\frac{1}{-2+\frac{1}{2}(x-1)}=-\frac{1}{2}+\frac{1}{8}(x-1)$$

この結果を (9) に代入すると次の結果が得られる。

$$y_2(x)=\int_1^x -\frac{1}{2}+\frac{1}{8}(x-1)dt=-\frac{1}{2}(x-1)+\frac{1}{16}(x-1)^2$$

この操作を繰り返し 7 次まで計算すると次のようになる。

$$y_7(x)=-\frac{(x-1)}{2}+\frac{(x-1)^2}{16}-\frac{(x-1)^3}{192}-\frac{(x-1)^4}{3072}+\frac{13\,(x-1)^5}{61440}-\frac{47\,(x-1)^6}{1474560}-\frac{73\,(x-1)^7}{41287680}$$

Taylor 展開式の係数は通常浮動小数点数で計算するが、計算は厳密に計算出来ることを強調するために、分数を使って計算してある。$y_7(x)$ は $f(x)=e^{-x}-x$ の逆関数の 7 次までの Taylor 展開式なので、$x=0$ とおいて零点を計算すると 0.56714111812531 が得られる。この値を元の式に代入すると 3.40428×10^{-6} となりこの関数が逆関数の展開式であることがわかる。

3.2 ゼロ割を含む計算

次の関数を $x=0$ で Taylor 展開する。

$$f(x)=\frac{x}{e^x-1}=\sum_{n=0}^{\infty}\frac{B_n}{n!}x^n \tag{12}$$

ここで、B_n はベルヌ-イ数 (Bernoulli number) である。単純に $f(0)$ とすると、分子分母共にゼロとなるため、ゼロ割が生じ Taylor 展開式の定数項は計算出来ない。このような場合にも Taylor 級数として計算すれば、容易に計算出来る。(12) は、次のように、分子と分母に分けて Taylor 展開すると次のようになる。

$$f(x)=\frac{x}{e^x-1}=\frac{x}{x+\frac{x^2}{2!}+\frac{x^3}{3!}+\cdots} \tag{13}$$

分子と分母を x で割ると次のようになる。

$$f(x)=\frac{x}{e^x-1}=\frac{1}{1+\frac{x}{2!}+\frac{x^2}{3!}+\cdots} \tag{14}$$

この式から、$f(0)=1$ が得られる。Taylor 級数の割り算を

行うことによって、$f(x)$ の Taylor 展開式が得られる。

C++言語用のプログラムは以下のようになる。

```
 1 :  #include "taylor_template.h"
 2 :  typedef taylor_template<double> taylor ;
 3 :  int main()
 4 :  {
 5 :      taylor x, y ;
 6 :      x = taylor( 0.0, 0.0, 1.0);
 7 :      y = x/(exp(x)-1) ;
 8 :      cout << y << endl ;
 9 :  }
```

ここで、コンストラクター taylor(a, b, c) は $b+c(x-a)$ を定義する。この計算では、除算演算で分子と分母の Taylor 級数の定数項がゼロになるので、分子と分母を x で割ってから除算を行っている。(12) の $f(x)$ を $x=0$ で Taylor 展開し 10 次まで表示すると以下のようになる。係数の絶対値が 10^{-10} 以下のものはゼロとして、表示しないと以下のようになる。

```
1-0.5*x+0.0833333*x^2-0.00138889*x^4
 +3.30688e-05*x^6-8.2672e-07*x^8+2.08768e-08*x^10
```

この計算は、分数を使って厳密に計算できる。以下にその結果を示す。

$$1-\frac{x}{2}+\frac{x^2}{12}-\frac{x^4}{720}+\frac{x^6}{30240}-\frac{x^8}{1209600}+\frac{x^{10}}{47900160}$$

これから、次のベルヌーイ数が得られる。$B_0=1$, $B_1=-\frac{1}{2}$, $B_2=\frac{1}{6}$, $B_3=0$, $B_4=-\frac{1}{30}$, $B_5=0$, $B_6=\frac{1}{42}$, $B_7=0$, $B_8=-\frac{1}{30}$, $B_9=0$, $B_{10}=\frac{5}{66}$

4. 微分方程式満たさない関数の Taylor 展開

C 言語の数学関数で最初に標準関数に採用されるなどからわかるようにガンマ関数は、数学的なコンピュータの計算分野では非常に重要な関数である。しかしながらガンマ関数 ($\Gamma(x)$) が満たす微分方程式は、公式集を見ても存在しない。このため、これまで挙げた例題のように簡単に Taylor 展開をすることはできない。

ここでは、この関数の逆数 $\frac{1}{\Gamma(x)}$ を原点で Taylor 展開する。$\Gamma(x)$ は x が 0 および負の整数値のとき極となり、関数値が発散する。ガンマ関数の逆数にすると、極は零点になり、複素平面全面で正則な関数となり、収束が非常に速い級数になると期待できるためである。この計算は 1883 年に Bourguet[2] が計算した。その後一部修正され、Davis によって、公式集 [1] に表 1 のように掲載されている。

しかしながら、この展開式はよく使われているがその展開式の計算法については、ほとんど知られていない。この展開式の計算法については、数値計算ソフトウエア [9] の中で森正武氏が議論しているが、今回はこの方法は試していない。

ここでは、次の公式を使って、4 倍精度や 8 倍精度のガンマ関数の計算に利用できる様に 100 次まで 100 桁以上の精度で計算する。

表 1 Series Exapansion for $\frac{1}{\Gamma(x)}=\sum_{k=1}^{\infty}c_k x^k$

k	c_k
1	1.00000 00000 000000
2	0.57721 56649 015329
3	-0.65587 80715 202538
4	-0.04200 26350 340952
5	0.16653 86113 822915
6	-0.04219 77345 555443
7	-0.00962 19715 278770
8	0.00721 89432 466630
9	-0.00116 51675 918591
10	-0.00021 52416 741149
11	0.00012 80502 823882
12	-0.00002 01348 547807
13	-0.00000 12504 934821
14	0.00000 11330 272320
15	-0.00000 02056 338417
16	0.00000 00061 160950
17	0.00000 00050 020075
18	-0.00000 00011 812746
19	0.00000 00001 043427
20	0.00000 00000 077823
21	-0.00000 00000 036968
22	0.00000 00000 005100
23	-0.00000 00000 000206
24	-0.00000 00000 000054
25	0.00000 00000 000014
26	-0.00000 00000 000001

$$\log\Gamma(x) = (x-\frac{1}{2})\log x - x + \frac{1}{2}\log(2\pi) + \sum_{m=1}^{\infty}\frac{B_{2m}}{2m(2m-1)x^{2m-1}} \tag{15}$$

級数の形から、x の値が大きいと収束は速くなると推定できる。ガンマ関数の次の性質から大きな数数値で、(15) の式を展開する。ガンマ関数は $\Gamma(x+1)=x\Gamma(x)$ の性質を持つので、n が正の整数の時

$$\Gamma(x)=\frac{\Gamma(x+N)}{\prod_{k=0}^{N-1}(x+k)} \tag{16}$$

が成り立つので、N を大きくとり、収束を速くする。

(15) の式の級数は、$m=35$ まで使う。この計算を行うにはベルヌ-イ数は B_{70} まで必要とする。

ベルヌ-イ数は次の漸化式から容易に計算することができる。

$$B_0=1,\quad B_n=-\frac{1}{n+1}\sum_{k=0}^{n-1}\left(\begin{array}{c} n+1 \\ k \end{array}\right)B_k \tag{17}$$

この計算は、浮動小数点数で計算したとき、桁落ちが生じ極めて計算しにくい計算として知られている。最近のよく使われる Python 言語では、桁数の多い整数が容易に使えるので、それを利用して分数を実装し、それを利用すれば容易に計算出来る。

計算は高精度計算プログラム MP_PACK[4] を使って、10^8 進数 32 桁 (10 進数で 256 桁) の精度で計算した。

$N = 280$、$N = 290$、$N = 300$ の 3 つのケースを計算し、この計算結果の絶対誤差、相対誤差から 120 桁以上一致することを確認している。1 次の係数は、理論的に 1 であること、2 次の係数はオイラー定数 γ であることが知られている。1 次の係数が 1 であることは、計算結果を見ることによって容易に確認出来る。2 次の係数は、Wolfram Alpha のウェブサイトでオイラー定数を計算し、出力した 100 桁については完全に一致していることを確認した。

計算時間は、Intel i7-4790K 4.0GHz で 1 回の計算に約 0.22 秒必要であった。この計算では、ベルヌーイ数は、文字列として表現された有理数として、与えている。このため、この計算にはベルヌーイ数を計算をするための時間は含まれていない。C++言語コンパイラーとして、Visual Studio 2017 の C++言語を使用した。

この計算結果のガンマ関数の逆数の原点における Taylor 級数の 1 次から 96 次までを Abramowitz の公式集 [1] と同様な形式で表 2 に示す。出力は小数点以下 100 桁を 10 桁毎に空白を入れた形式にした。97 次以降の係数は、 $c_{97} = -2.54 \times 10^{-102}$、$c_{98} = 3.77 \times 10^{-103}$、$c_{99} = -2.01 \times 10^{-104}$、$c_{100} = 6.62 \times 10^{-106}$ となり、小数点以下 100 桁はすべて 0 となるため省略した。

ここで得られた Taylor 級数は、原点における Taylor 展開式であるが、

$$\frac{1}{\Gamma(x+1)} = \frac{1}{x}\frac{1}{\Gamma(x)}$$

から、x で割ることによって容易に $\frac{1}{\Gamma(x+1)}$ の Taylor 級数が得られる。

一般に Taylor 級数 $f(x)$ の x に $x + a$ を代入して計算すると関数 $f(x)$ の $x = a$ における Taylor 級数になる。この計算は、関数論で言う解析接続にあたるので、a を大きな数値にすると精度の良い Taylor 級数は得られない。得るためには、$f(x)$ は高精度で高次の Taylor 級数である必要がある。

5. 終わりに

Taylor 展開を数値的に行う方法を簡単に説明した。この方法を利用すれば、解析的には非常に難しいと思われる関数も容易に Taylor 展開ができる。このような関数例として、実用上非常に重要なガンマ関数の数値的 Taylor 展開を行った。10 進数 100 桁精度で 100 次まで、ガンマ関数の逆数を原点について Taylor 展開を行った。その結果を表 2 にその結果を示した。これを使えば、4 倍精度や 8 倍精度のガンマ関数を容易に作成できる。

ここでは、ガンマ関数について述べたが、この方法は他の多くの関数に適用できる。

得られる高精度で高次の Taylor 級数は、四則演算、関数計算、微分および積分が容易にできるので、いろいろな問題に適用できる可能性がある。ガンマ関数についてならば、ガンマ関数の対数を計算し微分すれば、Psi(Digamma) 関数、複数回微分すれば、Polygamma functions の Taylor 展開式が得られる。

参考文献

[1] Abramowitz M. and Stegun I.A., Handbook of Mathmatical Functions, Dover, New York, (1970)
[2] Bourguet L., sur les intégrales eulériennes et quelques autres fonctions uniformes, Acta Math, **2** (1883), 261–295
[3] Bruijn, N.D., Asymptotic Methods in Analysis, Dover, New York,1981
[4] 平山 弘, C++ 言語による高精度計算パッケージの開発、日本応用数理学会論文誌，5 (1995),307–318
[5] 平山、小宮、佐藤, Taylor 級数法による常微分方程式の解法, 日本応用数理学会論文誌, **12** (2002), 1–8.
[6] 平山, 舘野, 浅野, 川口, Taylor 級数演算ライブラリの使用法, 東北大学情報シナジーセンター大規模科学計算システム広報 SENAC, 40(2007) 29–68
[7] Wikipedia，Lagrange inversion theorem, https://en.wikipedia.org/wiki/Lagrange_inversion_theorem
[8] Rall,L.B., Automatic Differentiation-Technique and Applications, Lecture Notes in Computer Science, Vol.120, Springer-Verlag, New York(1981)
[9] 渡部力, 名取亮, 小国力, Fortran77 による数値計算ソフトウエア, 丸善, 東京 (1989)

Table 2: Series Exapansion for $\frac{1}{\Gamma(x)} = \sum_{k=1}^{\infty} c_k x^k$

k	c_k
1	1.0000000000 0000000000 0000000000 0000000000 0000000000 0000000000 0000000000 0000000000 0000000000 0000000000
2	0.5772156649 0153286060 6512090082 4024310421 5933593992 3598805767 2348848677 2677766467 0936947063 2917467495
3	-0.6558780715 2025388107 7019515145 3904812797 6638047858 4347292362 4456838708 3835372210 2086182815 9940213640
4	-0.0420026350 3409523552 9003934875 4298187113 9450040110 6093522065 8129761800 9687597598 8547107701 2947877132
5	0.1665386113 8229148950 1700795102 1052357177 8150224717 4340570468 9031789938 6605647424 8319471914 6580416266
6	-0.0421977345 5554433674 8208301289 1873913016 5268418982 2486376918 8732754590 1118558899 6067347284 2937553150
7	-0.0096219715 2787697356 2114921672 3481989753 6294225211 3002105138 8626273116 7351446073 9536466888 0156588200
8	0.0072189432 4666309954 2395010340 4465727099 0480088023 8318001094 7811736225 9497415854 2714089090 1208498888
9	-0.0011651675 9185906511 2113971084 0183886668 0933379538 4057443407 5052756200 2584816653 0722148800 1091721885
10	-0.0002152416 7411495097 2815729963 0536478064 7824192337 8338750350 2674890856 3946371678 4739186185 1194625624
11	0.0001280502 8238811618 6153198626 3281643233 9489209969 3677214900 5458380412 0355204346 8758738922 3798955199
12	-0.0000201348 5478078823 8655689391 4210218183 8229483329 7979115261 1626709082 2918618897 0477623308 1238365030
13	-0.0000012504 9348214267 0657345359 4738330922 4232265562 1153959815 3499231574 9121245561 0792211796 2287578923
14	0.0000011330 2723198169 5882374129 6203307449 4332400483 8621075654 2955053954 6040842730 3676342059 0995993415
15	-0.0000002056 3384169776 0710345015 4130020572 8365125790 2629337945 3468317253 3245680371 0999187118 8900376836
16	0.0000000061 1609510448 1415817862 4986828553 4286727586 5719712320 8673240292 7723507435 2503861972 8262116887
17	0.0000000050 0200764446 9222930055 6650480599 9130304461 2742494481 7189533788 7737472131 7711613245 1381492840
18	-0.0000000011 8127457048 7020144588 1265654365 0557773875 9504932587 5909618926 3169643391 4362731439 5179682881
19	0.0000000001 0434267116 9110051049 1540332312 2501914007 0982312581 2121087107 3927347588 0712482098 8847060670
20	0.0000000000 0778226343 9905071254 0499373113 6077722606 8086181392 9388194355 0732692986 9575078978 0388273795
21	-0.0000000000 0369680561 8642205708 1878158780 8576623657 0963451360 9951364845 4655442999 7433930129 6044610189
22	0.0000000000 0051003702 8745447597 9015481322 8632318027 2688606970 7632117350 1048565735 2447899581 7069832630
23	-0.0000000000 0002058326 0535665067 8322242954 4855237419 7460910808 1014718805 8196444349 1534742482 9558791661
24	-0.0000000000 0000534812 2539423017 9823700173 1872793994 8989715478 1206821116 8095493210 9314539143 4840956648
25	0.0000000000 0000122677 8628238260 7901588938 4662242242 8165455750 4563213660 1135999606 3732793980 3489061753
26	-0.0000000000 0000011812 5930169745 8769513764 5868422978 3121155729 1804847879 8375081233 3473331503 8984505739
27	0.0000000000 0000000118 6692254751 6003325797 7724292867 4071088494 0796648271 1074006108 7642308631 6231444621
28	0.0000000000 0000000141 2380655318 0317815558 0394756670 9037086350 7503345256 2564122262 8600714911 7138456955
29	-0.0000000000 0000000022 9874568443 5370206592 4785806336 9926028450 5931419036 7014889830 4900770840 7409248302
30	0.0000000000 0000000001 7144063219 2733743338 3963370267 2570668126 5606251743 3174649858 1922023399 5833771638
31	0.0000000000 0000000000 0133735173 0493693114 8647813951 2226802287 5059471761 8947898582 8552542182 1683205030
32	-0.0000000000 0000000000 0205423355 1766672789 3250253513 5573379668 2037935238 7364127300 5599486628 7097450154
33	0.0000000000 0000000000 0027360300 4860799984 4831509904 3309820148 6531169583 6363370165 4294535181 9627740172
34	-0.0000000000 0000000000 0001732356 4459105166 3905742845 1564779799 0697491087 9499841377 2556229948 7072534376
35	-0.0000000000 0000000000 0000023606 1902449928 7287343450 7354275310 0792641355 2145370486 3263852154 2638701891
36	0.0000000000 0000000000 0000018649 8294171729 4430718413 1618786668 9894586842 9073668232 1745253705 4797777933
37	-0.0000000000 0000000000 0000002218 0956242071 9720439971 6913626860 3797317795 0067567580 0408431165 9563560264
38	0.0000000000 0000000000 0000000129 7781974947 9936688244 1448633059 4165619499 8646391331 9683205656 3724197497
39	0.0000000000 0000000000 0000000001 1806974749 6652840622 2745415509 9715185596 8463784158 3240354324 0365478494
40	-0.0000000000 0000000000 0000000001 1245843492 7708809029 3654674261 4395121194 1179558301 0816733382 7883581516
41	0.0000000000 0000000000 0000000000 1277085175 1408662039 9020667775 1124647748 7720656004 7924470389 8962028069
42	-0.0000000000 0000000000 0000000000 0073914511 6961514082 3461289330 1085528237 1056899245 1526766631 5906588427
43	0.0000000000 0000000000 0000000000 0000113475 0257554215 7609541652 5946930639 3008612195 9263334745 7404746508
44	0.0000000000 0000000000 0000000000 0000463913 4641058722 0299448049 0795222846 3057968679 7271496895 8217318643
45	-0.0000000000 0000000000 0000000000 0000053473 3681843919 8875077418 1967098933 2090488590 5773560162 4741918274
46	0.0000000000 0000000000 0000000000 0000003207 9959236133 5262286123 7279082794 3910901463 5972615518 7021700674

47 -0.0000000000 0000000000 0000000000 0000000044 4582973655 0756882101 5903521246 4363740143 6685748718 2886701604
48 -0.0000000000 0000000000 0000000000 0000000013 1117451888 1988712901 0584943899 2219023662 5449557426 0409965770
49 0.0000000000 0000000000 0000000000 0000000001 6470333525 4381388681 8259327906 3941453996 4472120222 7866183871
50 -0.0000000000 0000000000 0000000000 0000000000 1056233178 5035812186 0056107153 8285049997 0350802925 4127123868
51 0.0000000000 0000000000 0000000000 0000000000 0026784429 8264304947 8354963071 8908519484 9401875232 1278522023
52 0.0000000000 0000000000 0000000000 0000000000 0002424715 4948517826 8967303293 8370921240 7368104586 4775876041
53 -0.0000000000 0000000000 0000000000 0000000000 0000373658 7834535612 5540345591 2127031637 0587670646 6066764057
54 0.0000000000 0000000000 0000000000 0000000000 0000026283 3298094019 5449089037 6118736393 5422384941 7809111652
55 -0.0000000000 0000000000 0000000000 0000000000 0000000929 8175995376 8862996016 6899151817 2726444661 1851487776
56 -0.0000000000 0000000000 0000000000 0000000000 0000000023 2794241869 9470598604 2620556222 9184566877 7409599255
57 0.0000000000 0000000000 0000000000 0000000000 0000000006 1696208352 4438742035 4431773150 5479945891 0052114624
58 -0.0000000000 0000000000 0000000000 0000000000 0000000000 4928295586 7709899305 0445868221 2532594157 8457644432
59 0.0000000000 0000000000 0000000000 0000000000 0000000000 0218351318 3414510697 2778284986 9135599057 0140645048
60 -0.0000000000 0000000000 0000000000 0000000000 0000000000 0001218722 1891475165 5525045260 5074758145 0109055103
61 -0.0000000000 0000000000 0000000000 0000000000 0000000000 0000711710 8841662874 6319456527 3384830201 9494248526
62 0.0000000000 0000000000 0000000000 0000000000 0000000000 0000069205 0405432868 9253528422 2290662700 4443346172
63 -0.0000000000 0000000000 0000000000 0000000000 0000000000 0000003676 4384683566 7632767974 8481822479 4505781352
64 0.0000000000 0000000000 0000000000 0000000000 0000000000 0000000085 6309805627 5654327981 7124547431 5656002468
65 0.0000000000 0000000000 0000000000 0000000000 0000000000 0000000004 9630454283 6684438484 3070623712 9480600881
66 -0.0000000000 0000000000 0000000000 0000000000 0000000000 0000000000 7154294577 0816152181 9727983371 3681554340
67 0.0000000000 0000000000 0000000000 0000000000 0000000000 0000000000 0455172768 9088504117 7416935027 4857188167
68 -0.0000000000 0000000000 0000000000 0000000000 0000000000 0000000000 0016183993 0532029443 4385001529 9377648603
69 -0.0000000000 0000000000 0000000000 0000000000 0000000000 0000000000 0000038180 4342439995 0246432820 3775563378
70 0.0000000000 0000000000 0000000000 0000000000 0000000000 0000000000 0000051850 5241190584 8229547486 7404192329
71 -0.0000000000 0000000000 0000000000 0000000000 0000000000 0000000000 0000004167 1368092239 2088614527 1246428654
72 0.0000000000 0000000000 0000000000 0000000000 0000000000 0000000000 0000000191 6290692937 3887193002 4617437731
73 -0.0000000000 0000000000 0000000000 0000000000 0000000000 0000000000 0000000003 8089281324 6836587334 7899305159
74 -0.0000000000 0000000000 0000000000 0000000000 0000000000 0000000000 0000000000 2206386105 5924121015 7996790336
75 0.0000000000 0000000000 0000000000 0000000000 0000000000 0000000000 0000000000 0277223109 6009895416 4769454219
76 -0.0000000000 0000000000 0000000000 0000000000 0000000000 0000000000 0000000000 0015987660 4781001810 5748331347
77 0.0000000000 0000000000 0000000000 0000000000 0000000000 0000000000 0000000000 0000531973 0780417403 4027556256
78 -0.0000000000 0000000000 0000000000 0000000000 0000000000 0000000000 0000000000 0000000805 1746141684 2390431668
79 -0.0000000000 0000000000 0000000000 0000000000 0000000000 0000000000 0000000000 0000001248 4629810263 7951133539
80 0.0000000000 0000000000 0000000000 0000000000 0000000000 0000000000 0000000000 0000000096 4318876839 9223842755
81 -0.0000000000 0000000000 0000000000 0000000000 0000000000 0000000000 0000000000 0000000004 2827980483 0174792127
82 0.0000000000 0000000000 0000000000 0000000000 0000000000 0000000000 0000000000 0000000000 0950871423 6903044186
83 0.0000000000 0000000000 0000000000 0000000000 0000000000 0000000000 0000000000 0000000000 0027131392 1386943834
84 -0.0000000000 0000000000 0000000000 0000000000 0000000000 0000000000 0000000000 0000000000 0004096877 9415069156
85 0.0000000000 0000000000 0000000000 0000000000 0000000000 0000000000 0000000000 0000000000 0000237429 8001974016
86 -0.0000000000 0000000000 0000000000 0000000000 0000000000 0000000000 0000000000 0000000000 0000008277 0890210072
87 0.0000000000 0000000000 0000000000 0000000000 0000000000 0000000000 0000000000 0000000000 0000000090 7249760942
88 0.0000000000 0000000000 0000000000 0000000000 0000000000 0000000000 0000000000 0000000000 0000000010 6455581950
89 -0.0000000000 0000000000 0000000000 0000000000 0000000000 0000000000 0000000000 0000000000 0000000000 9285335619
90 0.0000000000 0000000000 0000000000 0000000000 0000000000 0000000000 0000000000 0000000000 0000000000 0433331359
91 -0.0000000000 0000000000 0000000000 0000000000 0000000000 0000000000 0000000000 0000000000 0000000000 0011745606
92 -0.0000000000 0000000000 0000000000 0000000000 0000000000 0000000000 0000000000 0000000000 0000000000 0000026908
93 0.0000000000 0000000000 0000000000 0000000000 0000000000 0000000000 0000000000 0000000000 0000000000 0000023899
94 -0.0000000000 0000000000 0000000000 0000000000 0000000000 0000000000 0000000000 0000000000 0000000000 0000001556
95 0.0000000000 0000000000 0000000000 0000000000 0000000000 0000000000 0000000000 0000000000 0000000000 0000000060
96 -0.0000000000 0000000000 0000000000 0000000000 0000000000 0000000000 0000000000 0000000000 0000000000 0000000001