最良近似式の計算

平　山　　弘*

Calculation of mini-max Approximation

Hiroshi HIRAYAMA

Abstract

It is shown that the mini-max approximation of the function can be given easily by using the arithmetic package for the multiple-precision floating point number. And the C++ program for mini-max approximation is given. The numerical example is shown how to use this program.

1. はじめに

最良近似式とは、次のように定義される。区間$[a, b]$において，与えられた関数$f(x)$を，n次の以下の多項式$p_n(x)$で近似することを考える。このとき，$f(x)$と$p_x(x)$の距離Lを定義する。$p_n(x)$の次数nを一定に保つとき，$L(f, p_n)$を最小にする関数を，最良近似式という。

最良近似式の計算は，関数近似式を研究する研究者の間ではよく使用されてきたが，通常の計算機利用者は，それを使うことはほとんどなかった。最良近似式を求めるには，近似式以上の精度で計算する必要があるが，通常このような手段はなかったからである。また，そのような計算手段があったとしても，計算は性質の悪い連立非線形方程式を解かなければならない。このような理由から，最良近似式の計算は，数値解析の専門書以外で紹介されることもないため，さらに使われることはなかった。

今まで関数近似の専門家の作った数学関数の最良近似式以外使うことが出来なかったが，最良近似式が簡単に作ることができるならば，利用者の分野でよく使われる式の最良近似式を作り計算の高速化をはかることができる可能性があり，大いに役立つと期待できる。

本論文では，最良近似式作成上の問題点を，高精度で計算するプログラム提供[1][2]することによって解決した。ここでは，ガンマ関数の倍精度の最良近似式を具体的に計算して，その有効性を示す。

1994年9月22日受理

* 機械システム工学科

2. 最良近似式の計算方法

ある関数の計算を何回も計算するような場合，その関数の定義通り計算するより，多項式または有理関数でその関数を近似し，その近似式を計算した方が高速に精度よく計算できる場合がある。このような関数として，計算機に組み込まれている三角関数や指数関数，ライブラリーになっているいろいろな特殊関数などがある。

連続関数の多項式近似については，Weierstrass[4]が，「閉区間において連続な関数は，多項式によっていくらでも高精度で近似できる」ことを証明している。この定理の証明方法と同じ方法で多項式を作ると，非常に高次の多項式となり，実際の計算には使いものにならない。実際の応用には，低次で精度の高い近似式を求める必要がある。

区間$[a, b]$において，与えられた関数$f(x)$を，n次の多項式$p_n(x)$で近似することを考える。このとき，$f(x)$と$p_n(x)$の距離Lを

$$L(f, p_n)\equiv\sup\{|f(x)-p_n(x)|;\ a\leq x\leq b \qquad (1)$$

と定義する。$p_n(x)$の次数nを一定に保つとき，$L(f, p_n)$を最小にする関数を，最良近似関数という。

関数$f(x)$を近似する多項式$p_n(x)$の差を$e_n(x)$とする。すなわち

$$e_n(x)=f(x)-p_n(x) \tag{2}$$

と定義する。$p_n(x)$ が最良近似式であるとき，$e_n(x)$ は次の性質[5]をもつことが知られている。

1) $e_n(x)$ の相隣る極大極小値は符号が異なる。
2) $e_n(x)$の極値は，区間内で少なくとも（$n+2$）個ある。
3) $e_n(x)$の極大極小値の絶対値はすべて等しい。

の性質がある。この性質から，$p_n(x)$ を求めることができる。このような性質は，多項式の最良近似式だけでなく，有理関数の最良近似式も同様な性質をもつ。

区間$[a,\ b]$において，与えられた関数 $f(x)$ を，分母が m 次の多項式 $q_m(x)$，分子が 1 次の多項式 $p_l(x)$ である有理関数 $\frac{p_l(x)}{q_m(x)}$ で近似することを考える。分数は分子と分母に定数を掛けても同じものになるので，$q_m(0)=1$ として不定部分をなくすものとする。
$f(x)$ とこの有理関数の差 $e(x)$ を

$$e(x)=w(x)\left\{f(x)-\frac{p_l(x)}{q_m(x)}\right\} \tag{3}$$

と定義する。$W(x)$ は，重み関数で区間$[a,\ b]$でゼロにならない関数である。このとき，$e(x)$ は上の$e_n(x)$と同様な性質をもっている。ただし，

$$n=l+m \tag{4}$$

である。多項式と同様に，有理関数の場合も同様に求めることができる。

最良近似は一般に解析的には求められないので，逐次近似法で求める。以下で示すプログラムは，山下[3]による方法をC++に変更したものである。

逐次近似計算を始めるには，初期値を設定しなければならない。まず反復計算で使う極値の位置を求める。誤差関数 $e(x)$ は，Chebyshev 多項式と似ている関数と仮定できるので，極値の位置を Chebyshev 多項式の極値と同じと仮定して，

$$x_j=\left(\frac{b-a}{2}\right)\cos\left(\frac{2j}{n+1}\ \frac{\pi}{2}\right)+\left(\frac{b+a}{2}\right) \tag{5}$$

と置く。同様に誤差関数 $e(x)$ の零点 x_i も計算できる。Chebyshev 関数の零点と同じと仮定すると，

$$x_i=\left(\frac{b-a}{2}\right)\cos\left(\frac{2i+1}{n+1}\ \frac{\pi}{2}\right)+\left(\frac{b+a}{2}\right) \tag{6}$$

が得られる。$p_l(x)$, $q_m(x)$ を以下のような多項式と仮定する。

$$p_l(x)=\sum_{k=1}^{l}a_kx^k \tag{7}$$

$$q_m(x)=\sum_{k=0}^{m}b_kx^k \tag{8}$$

(6) で示される誤差関数 $e(x)$ の零点で

$$f(x)_i-\frac{p_l(x_i)}{q_m(x_i)}=0 \tag{9}$$

が成り立つ。この式を変形すると

$$p_l(x_i)-f(x_i)\{q_m(x_i)-1\}=f(x_i) \tag{10}$$

となる。この式に，(7)，(8) を (10) に代入すると

$$\sum_{k=0}^{l}(x_i{}^k)a_k\sum_{k=0}^{m}(-f(x_i)x_i{}^k)b_k=f(x_i) \tag{11}$$

が得られる。(11) を解くことによって，a_k と b_k の初期値が得られる。

何回かの補正を行って得られた近似有理関数を $\frac{p_l(x)}{q_m(x)}$ とし，最良近似式を $\frac{p_l{}^*(x)}{q_m{}^*(x)}$ とする。これらの誤算関数をそれぞれ$e(x)$, $e^*(x)$ とする。これらは

$$e(x)=w(x)\left\{f(x)-\frac{p_l(x)}{q_m(x)}\right\} \tag{12}$$

$$e^*(x)=w(x)\left\{f(x)-\left\{\frac{p_l{}^*(x)}{q_m{}^*(x)}\right\}\right. \tag{13}$$

と定義する。$\Delta p_l(x)$, $\Delta q_m(x)$ をそれぞれ $p_l(x)$, $q_m(x)$ の補正として，

$$p_l{}^*(x)=p_l(x)+\Delta p_l(x) \tag{14}$$

$$q_m{}^*(x)=q_m(x)+\Delta q_m(x) \tag{15}$$

が成り立つ。また，3) の条件から最良近似の誤差関数 $e^*(x)$ の極大または極小は絶対値が同じになるから，その絶対値の値を ρ とする。x_j を $e^*(x)$ の極値をとる位置だとすると

$$e^*(x_j)=(-1)^j\rho \qquad j=0,2,\cdots,n+1 \tag{16}$$

が成り立つ。このとき，

$$\begin{aligned}e(x_j)-e^*(x_j)&=\left\{w(x_j)f(x_j)-\frac{p_l(x_j)}{q_m(x_j)}\right\}\\&\quad-w(x_i)f(x_j)-\frac{p_l{}^*(x_j)}{q_m{}^*(x_j)}\end{aligned} \tag{17}$$

$$=w(x_j)\left\{\frac{p_l{}^*(x)}{q_m{}^*(x)}-\frac{p_l(x)}{q_m(x)}\right\} \tag{18}$$

$$=w(x_j)\frac{\Delta p_l{}^*(x_j)q_m(x_j)-\Delta q_m(x_g)p_l(x_j)}{q_m(x_j)\{q_m(x_j)+\Delta q_m(x_j)\}} \tag{19}$$

$$=e(x_j)-(-1)^j\rho \tag{20}$$

$\Delta q_m(x)$ が $q_m(x)$ に比べて，非常に小さいとすると，(19)，(20) から

$$w(x_j)\frac{\Delta p_l^*(x_j)q_m(x_j)-\Delta q_m(x_j)p_l(x_j)}{q_m(x_j)^2}=e(x_j)-(-1)^j\rho \tag{21}$$

が得られる。相隣る極値での式 (21) を加えると

$$\begin{aligned}&w(x_j)\frac{\Delta p_l^*(x_j)q_m(x_j)-\Delta q_m(x_j)p_l(x_j)}{q_m(x_j)^2}\\&+w(x_{j+1})\frac{\Delta p_l^*(x_{j+1})q_m(x_{j+1})-\Delta q_m(x_{j+1})p_l(x_{j+1})}{q_m(x_{j+1})^2}\\&=e(x_j)+e(x_{j+1})\end{aligned} \tag{22}$$

$\Delta p_l(x)$, $\Delta q_m(x)$ は多項式であるから

$$\Delta p_l(x)=\sum_{k=0}^{m}\Delta b_k x^k \tag{23}$$

$$\Delta q_m(x)=\sum_{k=0}^{m}\Delta b_k x^k \tag{24}$$

と置ける。これを (22) に代入して，

$$\begin{aligned}&\sum_{k=0}^{l}\left\{\frac{w(x_j)x_j^k}{q_m(x_j)}+\frac{w(x_{j+1})x_{j+1}{}^k}{q_m(x_{j+1})}\right\}\Delta a_k\\&-\sum_{k=0}^{m}\left\{\frac{w(x_j)x_j{}^kp_l(x_j)}{q_m(x_j)^2}+\frac{w(x_{j+1})x_{j+1}{}^kp_l(x_{j+1})}{q_m(x_{j+1})^2}\right\}\Delta b_k\\&=e(x_j)+e(x_{j+1})\qquad j=0,1,\cdots,n\end{aligned} \tag{25}$$

この (25) の $(n+1)$ 元の方程式を解くことに，Δa_k および Δb_k を求める。この結果を (23)，(24) に代入することによって，$\Delta p_l(x)$, $\Delta q_m(x)$ を計算し，それぞれ $p_l(x)$, $q_m(x)$ を補正する。

得られた $p_l(x)$, $q_m(x)$ から，極値と極値の位置を山登り法[3]によって求める。極値の値の絶対値の最大値と最小値の比が小さいならば，最良近似式が求まったと判断するそうでない場合には，(25) の式をもう一度計算し有理式を補正する。

3. プログラム及び数値例

以下のプログラムでは，ガンマ関数の倍精度実数用の近似式を計算する。そこで最初にガンマ関数の計算方法を説明する。

ガンマ（gamma）関数は，階乗を拡張したもので，

$$\Gamma(s)=\int_0^\infty x^{s-1}e^{-x}dx\quad(\mathrm{Re}\ s>0) \tag{1}$$

と定義される。ガンマ関数は階乗の拡張なので，s が整数値のとき

$$\Gamma(s)=(s-1)! \tag{2}$$

となる。このことは，部分積分によって容易に証明できる。この関数は，次のように漸近展開[6]できる。

$$\begin{aligned}\log(\Gamma(s))=&\left(s-\frac{1}{2}\right)\log s-z+\frac{1}{2}\log(2\pi)\\&+\sum_{m=1}^{\infty}\frac{B_{2m}}{2m(2m-1)s^{2m-1}}\end{aligned} \tag{3}$$

この級数は，発散級数なので，級数の値を計算するには，何項かまで計算し，そこで計算を打ち切らなければならない。この級数の値と計算値の違いは，打ち切った項の大きさ程度となるから，級数の項が十分に地先うなるかまたは出来るだけ小さくなったところで計算を打ち切るようにしなければならない。級数の項を小さくするには，s を大きくする。s は，ガンマ関数の性質

$$\Gamma(s-1)=s\Gamma(s) \tag{4}$$

を何回か使って，n を整数とすると

$$\Gamma(s)=\frac{\Gamma(s+n)}{s(s+1)(s+2)\cdots(s+n-1)} \tag{5}$$

を得る。この公式によって，計算するガンマ関数の s の値をいくらでも大きくすることができる。この s を使って，(3) の式からガンマ関数の値を求め，(5) の式から元の s のガンマ関数の値を計算する。s を十分に大きいならば，(3) の級数は十分に速く収束し非常に計算し易いが，(5) によって元の s の値の関数値に戻すのが大変になる。この兼ね合いから，(3) の式を計算する最適な s の値を決めることができる。この値は，計算精度に依存する。

(3) の級数の計算には，ベルヌイ数が必要なので，実際の計算に使えるベルヌイ数の個数などの制限が付く。

上で示したアルゴリズムで計算するプログラムを以下に示す。このプログラムでは，(3) の級数を計算する時の s の値を計算精度（10 進数で）の 1.5 倍として計算している。計算精度を 10 進数で 100 桁ならば，s として 150 を選ぶことになる。計算精度 200 桁以内ならば，この方法で，十分計算できる。

```
1: //::::::::::::::::::::::::::::::::::::::::::::::::::::
2: //          ガンマ関数
3: //::::::::::::::::::::::::::::::::::::::::::::::::::::
```

```
 4: long_float gamma (const long_float& x )
 5: {
 6:  long lim, p ;
 7:  long_float s, t, w, z;
 8:  p=prec() ;
 9:  lim=p*6 ;
10:  s=x ;
11:  t=1 ;
12:  while( s < lim )
13:  {
14:       t*= s ;
15:       s+= 1 ;
16:  }
17:  w=(2*s-1)*log(s)/2-s+log(2*pi())/2-log
     (t) ;
18:  t=1 ;
19:  t/=s ;
20:  s=1/square(s) ;
21:  for( int k=1 ; k<=50 ; k++ )
22:  {
23:       int k2 = 2*k ;
24:       z=to_long_float(bern(k2))/(k2*(k2
     -1))*t ;
25:       t*=s ;
26:       w+=z ;
27:       if(exp_part(w)-exp_part(z)>p)
     break ;
28:  }
29:  z=exp(w) ;
30:  return z ;
31: }
```

以下で，最良近似式の計算を行う。計算内容は以下に示す。

近似区間

$$-\frac{1}{2} \leq x \leq \frac{1}{2}$$

最良近似を求める関数

$$\Gamma(x+2)$$

最良近似式の形

$$\frac{a_0+a_1x+a_2x^2+a_3x^3+a_4x^4+a_5x^5+a_6x^6+a_7x^7}{1+b_1x+b_2x^2+b_3x^3+b_4x^4+b_5x^5+b_6x^6+b_7x^7}$$

このプログラムで使うガンマ関数は，上で述べた方法で計算する。

この例題を計算するプログラムは，以下のようになる。

プログラム：

```
 1: # include"  long—num.h"
 2: //::::::::::::::::::::::::::::::::::::::::
 3: //  MINI MAX APPROXMATION
 4: //       PROGRAM APPROX
 5: //::::::::::::::::::::::::::::::::::::::::
 6: long_float ffx(const long_float& x ) ;
 7: void yama(const long_float& eps, long_float&
    x,
 8:              const long_float& xa, const long
    float& xb ) ;
 9: long_float pq(const long_float& x) ;
10: long_float ffx(const long_float& x) ;
11: long_float wwx(const long_float& x) ;
12: long_float eex(const long_float& x) ;
13: long_float pol(const long_float * a, const long
    n,
14:                   const long_float& x ) ;
15: # define  NS   20
16: # define  NS2  22
17: # define  NSZ  420
18: long       l, m, n ;
19: long_float  p[NS],  q[NS],  z[NSZ] ;
20: long_float  a, b, eps1, eps2;
21: long_float gamma( const long_float& x ) ;
22: void main()
23: {
24:  long_float  x[NS2], ex[NS2] ;
25:  long_float  u, fu, xa, xb, fp, fq ;
26:  long_float  emax, emin, ee, wu ;
27:  long_float  xpi = pi() ;
28:  long       i, j, k ;
29:  u=1 ;
30:  a=-u/2 ; b= u/2 ;
31:  eps1=1.0e-5 ; eps2=1.0e-5 ;
32:  //
33:  l=7 ; m=7 ;
34:  n=l+m ;
35:  //
36:  long_float a1=(b-a)/2 ;
37:  long_float a2=(b+a)/2 ;
```

```
38:  long_float pn=xpi/(2*(n+1));
39: // (A)
40:  for(i=0;i<=l;i++)
41:  {
42:       p[i]=0;
43:  }
44:  for( i=0;i<=m;i++ )
45:  {
46:       q[i]=0;
47:  }
48:  q[0]=1;
49: // 極値点
50:  x[0]=b;
51:  for(i=1;i<=n;i++)
52:  {
53:       x[i]=a1*cos(2*i*pn)+a2;
54:  }
55:  x[n+1]=a;
56: // (B)
57:  for(i=1;i<=n+1;i++)
58:  {
59:       u=a1*cos((2*i-1)*pn)+a2;
60:       fu=ffx(u);
61:       z[i-1]=1;
62:       for(j=1;j<=l;j++)
63:       {
64:            z[i-1+NS*j]=z[i-1+NS*(j
    -1)]*u;
65:       }
66:       if(m!=0)
67:       {
68:            z[i-1+NS*(l+1)]=-fu*u;
69:       }
70:       if(m!= 1)
71:       {
72:            for(j=2;j<=m;j++)
73:              {
74:                 z[i-1+NS *(l+j)]=z[i-1+
    NS*(l+j-1)]*u;
75:            }
76:       }
77:       z[i-1+NS*(n+1)]=fu;
78:  }
79: //  (C)
80:  while(1)
81:  {
82:       for(k=1;k<=n+1;k++)
83:       {
84:            for(i=k+1;i<=n+2;i++)
85:            {
86:                 z[k-1+NS*(i-1)]/=z[k
    -1+NS*(k-1)];
87:            }
88:            for(i=1;i<=n+1;i++)
89:            {
90:                 if(k!=i)
91:                 {
92:                      long_float tmp=z[i-1+
    NS*(k-1)];
93:                      for(j=k ; j<=n+1 ;
    j++)
94:                      {
95:                           z[i-1+NS*j]
    --tmp*z[k-1+NS*j];
96:                      }
97:                 }
98:            }
99:       }
100: // (D)
101:      for( i=1;i<=m;i++ )
102:      {
103:           q[i] += z[1+i+NS*(n+1)];
104:      }
105:      for(i=1;i<=l+1;i++)
106:      {
107:           p[i-1] += z[i-1+NS*(n+
    1)];
108:      }
109: // (E)
110:      for(i=1; i<=n; i++)
111:      {
112:           xa =(x[i+1]+x[i])/2;
113:           xb =(x[i]+x[i-1])/2;
114:           yama(eps1, x[i], xa, xb);
115:      }
116: // (F)
117:      emax=0;
118:      emin=1.0e+50;
```

```
119:        for(i=1 ; i<=n+2 ; i++)
120:        {
121:              ee=eex(x[i-1]) ;
122:              ex[i-1]=ee ;
123:               if(abs(ee)>emax)emax=abs
    (ee) ;
124:              if(abs(ee)<emin)emin=abs(ee) ;
125:        }
126:        if( emax-emin<eps2*emax)break ;
127: // (G)
128:        u=x[0] ;
129:        fp=pol(p, l, u) ;
130:        fq=pol(q, m, u) ;
131:        wu=wwx(u) ;
132:        z[0]=wu/fq ;
133:        for(j=1 ; j<=1; j++)
134:        {
135:              z[NS*j]=z[NS*(j-1)]*u ;
136:        }
137:        if(m!=0)
138:        {
139:               z[NS*(1+1)]=-wu*fp*u/
    square(fq);
140:              if(m!=1)
141:              {
142:                    for(j=2 ; j<=m ; j++)
143:                    {
144:                          z[NS*(1+j)]=z[NS*
    (1+j-1)]*u ;
145:                    }
146:              }
147:        }
148:        z[NS*(n+1)]=eex(u) ;
149:        for( i=1 ; i<=n+1 ; i++ )
150:        {
151:              u=x[i] ;
152:              fp=pol(p, l, u) ;
153:              fq=pol(q, m, u) ;
154:              wu=wwx(u) ;
155:              z[i]=wu/fq ;
156:              for(j=1 ; j<=l ; j++ )
157:              {
158:                    z[i+NS*j]=z[i+NS*(j
    -1)]*u ;
159:              }
160:              if(m!=0)
161:              {
162:                    z[i+NS*(l+1)]=-wu*fp*
    u/square(fq) ;
163:                    for(j=2 ; j<=m ; j++)
164:                    {
165:                          z[i+NS*(l+j)]=z[i+
    NS*(l+j-1)]*u ;
166:                    }
167:              }
168:              z[i+NS*(n+1)]=eex(u) ;
169:              for(j=1 ; j<=n+2 ; j++)
170:              {
171:                    z[i-1+NS*(j-1)]+=z[i+
    NS*(j-1)] ;
172:              }
173:        }
174:   }
175: // (H)
176:  cout << "          *** 計算結果 ***"
    << "¥n";
177:  cout << "  分子の係数  A=" << "¥n" ;
178:  for(i=0 ; i<=l ; i++)
179:  {
180:       cout << i << " " << p[i] << "¥n" ;
181:   }
182:   cout << "  分母の係数  B=" << "¥n" ;
183:   for(i=0 ; i<=m ; i++)
184:   {
185:        cout << i << " " << q[i] << "¥n" ;
186:   }
187:   cout << "  極値点  X=" << "¥n" ;
188:   for(i=1 ; i<=n+2 ; i++)
189:   {
190:        cout << i << " " << x[i-1] << "¥n";
191:   }
192:   cout << "  極値  EX=" << "¥n" ;
193:   for(i=1 ; i<=n+2 ; i++)
194:   {
195:        cout << i << " " << ex[i-1] << "¥n";
196:   }
197: }
198: //::::::::::::::::::::::::::::::::::::::::::::
```

```
long_float pol(const long_float * a, const long n,
                    const long_float& x)
{
    long_float ret;
    ret=a[n];
    for(long i=n-1; i>=0; i--)
    {
        ret=ret * x+a[i];
    }
    return ret;
}
//::::::::::::::::::::::::::::::::::::::::::::::
void yama(const long_float& eps, long_float& x,
          const long_float& xa, const long float& xb )
{
    long      key;
    long_float h, f0, f1, f2, x1, x2;

    key=9;
    h=(xb-xa)/4;
    f0=abs(eex(x));
    while(1)
    {
        x1=x+h;
        if(x1>xb)x1=xb;
        x2=x-h;
        if(x2<xa)x2=xa;
        f1=abs(eex(x1));
        f2=abs(eex(x2));
        if((f2<=f0) && (f0>=f1))
        {
            h/=2;
            if(abs(f0-f1)+abs(f0-f2)<eps *f0 ) return;
        }
        else if(f2<f1)
        {
            x=x1;
            f0=f1;
            if(key==2)h/=2;
            key=1;
        }
        else if(f2>f1)
        {
            x=x2;
            f0=f2;
            if(key==1)h/=2;
            key=2;
        }
    }
}
//::::::::::::::::::::::::::::::::::::::::::::::
long_float pq( const long_float& x )
{
    long_float px, qx;
//
    px=p[l];
    for(long i=l-1; i>=0; i--)
    {
        px=px*x+p[i];
    }
    qx=q[m];
    for(i=m-1; i>=0; i--)
    {
        qx=qx*x+q[i];
    }
    return  px/qx;
}
//::::::::::::::::::::::::::::::::::::::::::::::
long_float ffx(const long_float& x)
{
    long_float p=gamma(x+2);
    return p;
}
//::::::::::::::::::::::::::::::::::::::::::::::
long_float wwx(const long_float& x)
{
    return 1;
}
//::::::::::::::::::::::::::::::::::::::::::::::
long_float eex( const long_float& x )
{
    return wwx(x)*(ffx(x)-pq(x));
}
```

実行結果は以下のようになる。

＊＊＊ 計算結果 ＊＊＊

分子の係数 A＝

0 1.00000000000000000055621491053605825465149523
1 8.87739364132926796030972682246349015708 68e-01
2 4.14325944493523124264794711643188166770 07e-01
3 1.33620185777182017282576756154657525060 16e-01
4 3.06916617382709265946963497426879751045 01e-02
5 5.41990644416282297021381721558038709662 88e-03
6 6.45639092888369872669666964419622242478 00e-04
7 5.50186802413990693873076811915229342058 02e-05

分母の係数 B＝

0 1.00000000000000000000000000000000000000000000
1 4.64955029034459615456525490167704690584 45e-01
2 -1.94090088733938835164233779553033531230 75e-01
3 -5.73857171463040359038787465329580028293 39e-02
4 2.27089606722020718258083553619311725779 92e-02
5 1.03046892999271743251553976490191614517 99e-03
6 -1.08005685846898719613588587058993994937 67e-03
7 1.10264416883499012614566812132854444903 24e-04

極値点 X＝

1 5.00000000000000000000000000000000000000000000e-01
2 4.88691591678872786717209262611170772200 53e-01
3 -4.55267652151313061554334345115380423566 01e-01
4 4.01238227726183768073807391185886634675 02e-01
5 3.29001275848012799586092792276000810254 10e-01
6 2.41947653001330129718298752303723328478 57e-01
7 1.44070962703610344374824776060060099363 62e-01
8 3.98891445792505662526552950648658989836 07e-02
9 -6.54792859784370929448760141649576128981 18e-02
10 -1.67387977960206667234332352579016513876 31e-01
11 -2.61301790855326937540316501602691763346 56e-01
12 -3.43291689107650850759963283946303606083 57e-01
13 -4.10197488856124023722745713998136207164 67e-01
14 -4.59542958356463631228545625507250328692 27e-01
15 -4.89816484603440746697529261199265163082 66e-01
16 -5.00000000000000000000000000000000000000000000e-01

極値 EX＝

1 1.48522655041203432011296693518999012969 00e-18
2 -1.48522655042113221386106859145213463633 00e-18
3 1.48522655043705940029953241648925328683 00e-18
4 -1.48522655046157221671833529353593434653 00e-18
5 1.48522655048131115329353908070174240546 00e-18
6 -1.48522655049007826802583432782477522890 00e-18
7 1.48522655047953624501739729655619670373 00e-18
8 -1.48522655046452261504318321986882042055 00e-18
9 1.48522655046063822444959072814890670069 79e-18
10 -1.48522655047696088466051305345197265744 48e-18
11 1.48522655049271463872595447909945439938 21e-18
12 -1.48522655047235764093303702759129863510 17e-18
13 1.48522655043741738151020049971873004155 22e-18
14 -1.48522665234325934864966445832036129706 60e-18
15 1.48522655049035422072304464060808527456 31e-18
16 -1.48522655053104023066080554372443322601 77e-18

極値は，すべて絶対値が約 1.485×10^{-18} でほぼ同じで，隣の極値とは符号が異なっており，最良近似の条件を満たしている。従って，最大誤差は，となるので，倍精度実数の近似公式としては，十分な精度を持っていることになる。

計算機による出力は，40 桁以上の精度で出力されているが，近似式の係数としては，最大誤差より小さい係数は，不要である。

4. ま と め

最良近似式を求めるプログラムを与えることができた。これによって，数学関数だけでなくいろいろな分野の公式の最良近似式を求めることができる。最良近似式を求めるには条件の悪い行列をもつ連立方程式を解かなければならないが，この数値的な問題点は十分な高精度計算することによって解決することができた。

参 考 文 献

[1] 平山 弘："C ++言語のための多倍長浮動小数点パッケージ"，神奈川工科大学研究報告B理工編，Vol. 17 (1993)，pp. 145-150.

[2] 平山 弘："多倍長数演算システムの開発"，神奈川工科大学研究報告B理工編，Vol. 18 (1994)，pp. 145-150.

[3] 山下眞一郎："最良近似式を求めるプログラム"，情報処理，Vol. 10，6 (1969)，pp. 442-446.

[4] R. クーラン，D. ヒルベルト(斉藤監訳)："数理物理学の方法"，東京図書，(1959).

[5] 宇野利雄："数値計算"，朝倉書店，(1963).

[6] M. Abramowitz and I. A. Stegun："Handbook of Mathmatical Functions"，Dover (1972).