8. 数値積分

任意の区間[a,b]における f(x)の定積分

$$(1)\quad I=\int_a^b f(x)\ dx$$

の値は，つぎのように n 点の関数値の和により近似的に与えられる。

$$(2)\quad I_n=\sum_{k=1}^{n} A_k f(x_k)$$

このとき，x_k を分点，A_k を重みという。

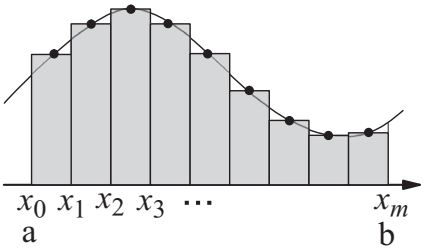
8.1 ニュートン・コーツ（複合型）積分公式

積分区間[a,b]を等分割して n 個の分点をとり，被積分関数 f(x)を n-1 次ラグランジュ補間多項式で近似して得られる積分公式を n 点ニュートン・コーツ公式という。ただし高次多項式による補間は凹凸の激しい不自然な形状に陥ることがあるので，実際の適用には積分区間[a,b]をまず m 個の小区間に等分し，各小区間ごとに比較的低次の補間公式を用いることが多い。これを複合型積分といい，代表的なものを以下に記す。

複合中点則：分点として各小区間の中点をとり，各小区間ごとに 0 次式補間を行うもの

$$M_m=h\sum_{i=1}^{m} f\left(\frac{x_{i-1}+x_i}{2}\right)$$

$$\begin{cases} h=\dfrac{b-a}{m} \\ x_i=a+ih \qquad i=0,1,\cdots,m \end{cases}$$

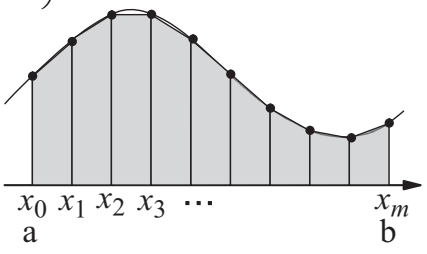


であり，誤差は $0(h^2)$ となる。

複合台形則：分点として小区間の端点をとり，各小区間ごとに 1 次式補間（面積の台形近似）を行うもの

$$T_m=\sum_{i=1}^{m}\frac{h}{2}\left(f\left(x_{i-1}\right)+f(x_i)\right)=h\left(\frac{1}{2}f(x_0)+\sum_{i=1}^{m-1}f(x_i)+\frac{1}{2}f(x_m)\right)$$

$$\begin{cases} h=\dfrac{b-a}{m} \\ x_i=a+ih \qquad i=0,1,\cdots,m \end{cases}$$

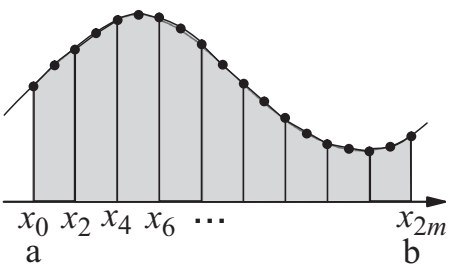


であり，誤差は $0(h^2)$ となる。

複合シンプソン則：分点として小区間の端点と中点をとり，各小区間ごとに 2 次式補間を行うもの

$$S_m=\sum_{i=1}^{m}\frac{h}{3}\left(f\left(x_{2i-2}\right)+f(x_{2i-1})+f(x_{2i})\right)=\frac{h}{3}\left(f(x_0)+f(x_{2m})+2\sum_{i=1}^{m-1}f(x_{2i})+4\sum_{i=1}^{m}f(x_{2i-1})\right)$$

$$\begin{cases} h=\dfrac{b-a}{m} \\ x_i=a+ih \qquad i=0,1,\cdots,m \end{cases}$$



であり，誤差は $0(h^4)$ となる。
また複合シンプソン則は以下のように表現できる。

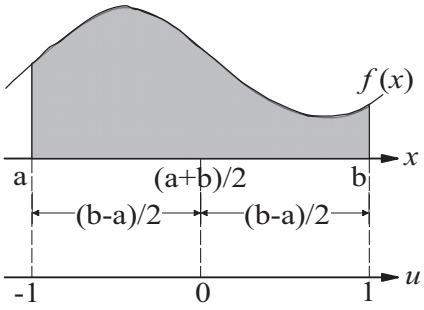
$$S_m=(2/3)M_m+(1/3)T_m$$

8.2 ガウス型積分

(1)式の定積分の値を求めるための n 点ガウス・ルジャンドル積分公式は

$$G_n = \frac{b-a}{2}\sum_{i=1}^{n} w_i\, f(x_i)$$

$$x_i = \frac{a+b}{2} + \frac{b-a}{2}u_i$$



となる。ここに

- u_i は分点（ $-1 < u_i < 1$; n 次ルジャンドル多項式の零点）
- w_i は重み（ $0 < w_i <= 1$ ）

であるが，それらの値は通常，数表に掲載されている。分点 $u_1, u_2, \cdots, u_n$ は原点に対称に配置されており（n が奇数のときは $u_{(n+1)/2}=0$），対称な点に対する重みは等しい。
n 点ガウス型公式は，被積分関数 f(x)が 2n-1 次以下の多項式であれば厳密な積分値を与える。

【アルゴリズム 8.2.1】

n 点ガウス公式の分点 $u_1, u_2, \cdots, u_n$ と重み $w_1, w_2, \cdots, w_n$ を数表からセットする

```
s:=0
for i=1 to n
    xi:=(a+b)/2+{(b-a)/2}ui
    s:=s+wif(xi)
end
s:= {(b-a)/2}s
```

【問題】

つぎの定積分（8.4 節参照）

$$\int_0^1 e^x \cos x\, dx = 1.3780246\cdots$$

の値を n 点ガウス公式（n=3,4,5）により求める。

【プログラム 8.2.1】

```
 1 /* Gaussian quadrature */
 2
 3 #include <stdio.h>
 4 #include <math.h>
 5
 6 float gu[]={
 7   0.5773503,
 8   0.0       , 0.7745967,
 9   0.3399810, 0.8611363,
10   0.0       , 0.5384693, 0.9061798,
11   0.2386192, 0.6612094, 0.9324695,
12   0.0       , 0.4058452, 0.7415312, 0.9491079,
13   0.1834346, 0.5255324, 0.7966665, 0.9602899,
14   0.0       , 0.3242534, 0.6133714, 0.8360311, 0.9681602,
15   0.1488743, 0.4333954, 0.6794096, 0.8650634, 0.9739065
```

```
};
float gw[]={
   1.0,
   0.8888889, 0.5555556,
   0.6521452, 0.3478548,
   0.5688889, 0.4786287, 0.2369269,
   0.4679139, 0.3607616, 0.1713245,
   0.4179592, 0.3818301, 0.2797054, 0.1294850,
   0.3626838, 0.3137066, 0.2223810, 0.1012285,
   0.3302394, 0.3123471, 0.2606107, 0.1806482, 0.0812744,
   0.2955242, 0.2692667, 0.2190864, 0.1494513, 0.0666713
};
int index[]={ 0, 0, 0, 1, 3, 5, 8, 11,15,19,24};

float f(float x)
{
    return( exp(x)*cos(x) );
}

void main(void)
{
     int n,m,i,k,n20,n21;
     float a,b,abm,ab2,s,wi,xi,fx;
     float u[10], w[10];

     printf("order of Gaussian quadrature: n ? ");
     scanf("%d", &n);
     if((n<2)||(n>10)){                              /* error */
            printf("n should be 2 =< n =< 10\n");
            exit(1);
     }else{              /* setup Gaussian points and weights */
         n20= n/2;
         n21=(n+1)/2;
         for(i=0;i<n21;i++){
             u[n20+i]= gu[index[n]+i];
             w[n20+i]= gw[index[n]+i];
         }
         for(i=0;i<n20;i++){
             u[i]=-u[n-1-i];
             w[i]= w[n-1-i];
         }
     }
     printf("interval of quadrature: a, b ? ");
     scanf("%g%g", &a, &b);

     printf("\n\t u \t\t weight \t x \t\t f(x)\n");
     abm=(a+b)/2;  ab2=(b-a)/2;
     s = 0;
     for(i=0;i<n;i++){
            xi = abm+ab2*u[i];
            fx = f(xi);
```

```
67              s += w[i]*fx;
68              printf("%15.6e %15.6e %15.6e %15.6e\n", u[i], w[i], xi, fx);
69      }
70      s *=ab2;
71      printf("\nintegral=%13.7g\n", s);
72 }
```

【実行例】

```
-------------------------実行開始-----------------------
order of Gaussian quadrature: n ? 3                              ……  3点ガウス公式
interval of quadrature: a, b ? 0 1.0

         u               weight   x               f(x)
  -7.745967e-01    5.555556e-01    1.127017e-01    1.112197e+00
   0.000000e+00    8.888889e-01    5.000000e-01    1.446889e+00
   7.745967e-01    5.555556e-01    8.872983e-01    1.533658e+00

integral=     1.378021
-------------------------おしまい-----------------------

-------------------------実行開始-----------------------
order of Gaussian quadrature: n ? 4                              ……  4点ガウス公式
interval of quadrature: a, b ? 0 1.0

         u               weight   x               f(x)
  -8.611363e-01    3.478548e-01    6.943184e-02    1.069316e+00
  -3.399810e-01    6.521452e-01    3.300095e-01    1.315923e+00
   3.399810e-01    6.521452e-01    6.699905e-01    1.531770e+00
   8.611363e-01    3.478548e-01    9.305682e-01    1.514922e+00

integral=     1.378025
-------------------------おしまい-----------------------

-------------------------実行開始-----------------------
order of Gaussian quadrature: n ? 5                              ……  5点ガウス公式
interval of quadrature: a, b ? 0 1.0

         u               weight   x               f(x)
  -9.061798e-01    2.369269e-01    4.691011e-02    1.046875e+00
  -5.384693e-01    4.786287e-01    2.307653e-01    1.226175e+00
   0.000000e+00    5.688889e-01    5.000000e-01    1.446889e+00
   5.384693e-01    4.786287e-01    7.692347e-01    1.550482e+00
   9.061798e-01    2.369269e-01    9.530899e-01    1.502192e+00

integral=     1.378025
-------------------------おしまい-----------------------
```

8.3 ロンバーグ積分

区間[a,b]における f(x)の定積分(1)の値を求めるためのロンバーグ積分のアルゴリズムは，以下のとおり。

小区間数を $n_1=1$（小区間幅：(b-a)）から始めて倍々にして $n_k=2^{k-1}$（小区間幅：$h_k=(1/2)^{k-1}(b-a)$）としていくとき，k=1,2,……に対し，収束するまでつぎの Step1，Step2 を繰り返す。

Step1）k 段目（小区間数：n_k，小区間幅：h_k）の初期近似として，複合台形則 $T(h_k)$ により $T_{k,1}$ を求める。

k＝1 のとき　$$T_{1,1}=T(h_1)=\frac{h_1}{2}\left(f(a)+f(b)\right)$$

k≧2 のとき　$$T_{k,1}=T(h_k)=h_k\left(\frac{1}{2}f(a)+\sum_{i=1}^{n_k-1}f(a+ih_k)+\frac{1}{2}f(b)\right)$$

これは，k-1 段目の複合台形則 $T(h_{k-1})$ を用いて効率的に計算することができる。

$$T_{k,1}=T(h_k)=\frac{1}{2}T(h_{k-1})+h_k\sum_{i=1}^{n_{k-1}}f\left(a+(2i-1)h_k\right)$$

Step2）k≧2 では，線形補外の反復により $x=h^2=0$ における多項式補外値を求める。

$$T_{k,j+1}=\frac{4^jT_{k,j}-T_{k-1,j}}{4^j-1},\qquad j=1,2,\cdots,k-1$$

収束判定条件は下式とする。

$$\left|\frac{T_{k,k}-T_{k-1,k-1}}{T_{k,k}}\right|<\varepsilon$$

⇒⇒　j：補間多項式の次数を増して精度を上げる方向

k：区間幅 h_k を 0 に近づける方向（⇓）

$$\begin{array}{lllll}
T_{1,1} & & & & \\
\searrow & & & & \\
T_{2,1} & \rightarrow & T_{2,2} & & \\
\searrow & & \searrow & & \\
T_{3,1} & \rightarrow & T_{3,2} & \rightarrow & T_{3,3} \\
\searrow & & \searrow & & \searrow \\
T_{4,1} & \rightarrow & T_{4,2} & \rightarrow & T_{4,3} \rightarrow T_{4,4} \\
\vdots & & & & \ddots
\end{array}$$

【アルゴリズム 8.3.1】

```
n:=1                              {k=1}
h:=b-a
t1:=(h/2)(f(a)+f(b))

for k=2,3,……                     {k=2,3,…：収束するまで繰り返す}
    for j=1 to k-1
        t^old_j:=t_j
    end
    n:=2n              |          {Step1}
    h:=h/2             |          {区間数を倍（区間幅を1/2）にして複合台形則T(h)により初期近似を行う}
    t1:=T(h)           |          {T(h)の計算は，上記のような効率的な計算に置き換えてもよい}
    r:=1
    for j=1 to k-1        {Step2}     {線形補外の反復により，h=0における多項式補外値を求める}
        r:=4r                         {r=4^j}
        tj+1:=(r tj-t^old_j)/(r-1)    {tj=Tk,j，t^old_j=Tk-1,j}
    end
    if |(tk-t^old_k-1)/tk|<ε  then break   {収束したらループから出る}
end
```

【問題】

8.2 節の問題と同じく，つぎの定積分（8.4 節参照）

$$\int_0^1 e^x \cos x \, dx = 1.3780246\cdots$$

の値をロンバーグ積分により求める。

【プログラム 8.3.1】

```
 1 /*             Romberg's quadrature            */
 2 /* (trapezoid rule + Richardson extrapolation) */
 3
 4 #include <stdio.h>
 5 #include <math.h>
 6
 7 #define KMAX 20
 8
 9 float a,b,h;
10 int   n;
11
12 float f(float x)
13 {
14     return( exp(x)*cos(x) );
15 }
16
17 float trapezoid2(float tk1)
18 {
19       float sum;
20       int i;
21
22       h /=2 ;
23       sum=0;
24       for(i=0;i<n;i++)
25          sum += f(a+h*(2*i+1));
26       n *=2;
27       return(tk1/2+h*sum);
28 }
29
30 void main(void)
31 {
32      float eps,r,t[KMAX],t1[KMAX];
33      int   k,j;
34
35      printf("torelence of Romberg's quadrature: eps ? ");
36      scanf("%g", &eps);
37      printf("interval of quadrature: a, b ? ");
38      scanf("%g%g", &a, &b);
39
40      printf("\n n   trapezoid \t\t*** extrapolation ***\n");
41      n=1;
42      h=b-a;
```

```
43      t[0]=h*(f(a)+f(b))/2;
44      printf("%2d %13.6e\n", n, t[0]);
45
46      for(k=1;k<KMAX;k++){
47          for(j=0;j<k;j++)
48              t1[j]=t[j];
49          t[0] = trapezoid2(t1[0]);
50          r    =1;
51          for(j=0;j<k;j++){
52                  r     *= 4;
53                  t[j+1] = (r*t[j]-t1[j])/(r-1);
54          }
55          printf("%2d", n);
56          for(j=0;j<=k;j++)
57               printf(" %13.6e", t[j]);
58          printf("\n");
59          if( fabs((t[k]-t1[k-1])/t[k])<eps )
60                   break;
61      }
62      if(k>=KMAX)
63               printf("divergent ?\n");
64      printf("\nintegral=%13.7g\n", t[k]);
65 }
```

【実行例】

```
-------------------------実行開始-----------------------
torelence of Romberg's quadrature: eps ? 1.0e-6
interval of quadrature: a, b ? 0 1.0

 n   trapezoid           *** extrapolation ***
 1  1.234347e+00
 2  1.340618e+00  1.376042e+00
 4  1.368582e+00  1.377904e+00  1.378028e+00
 8  1.375658e+00  1.378017e+00  1.378025e+00  1.378025e+00
16  1.377433e+00  1.378024e+00  1.378025e+00  1.378025e+00  1.378025e+00

integral=     1.378025
-------------------------おしまい-----------------------
```

8.4 精度の比較検証

定積分

$$I = \int_0^1 e^x \cos x \, dx$$

を数値積分公式により計算して，その誤差を調べてみよう。被積分関数

$$f(x) = e^x \cos x$$

の原始関数は

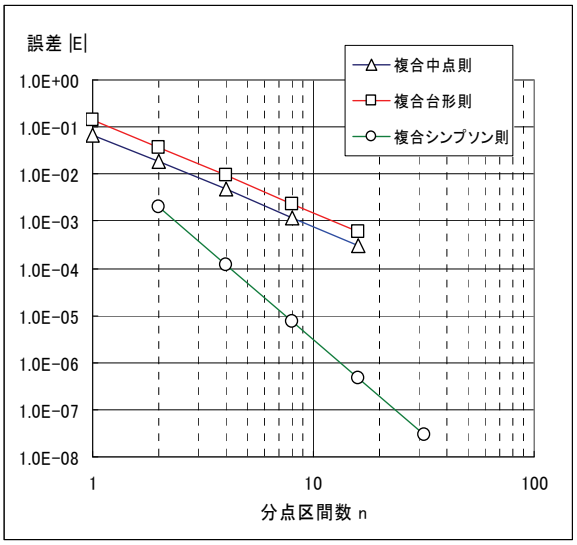
$$F(x) = e^x (\cos x + \sin x)/2$$

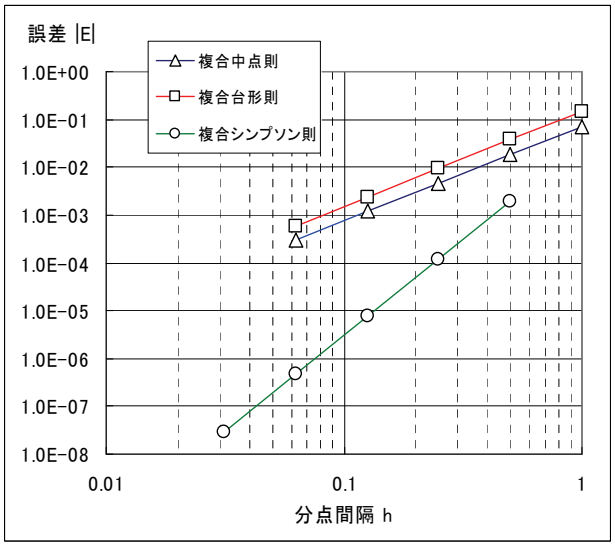
であるから，定積分の値（最も近い浮動小数点数に丸められた 16 桁の値）は，以下のようになる。

$$I = \int_0^1 e^x \cos x \, dx = 1.37802\ 46135\ 47364$$

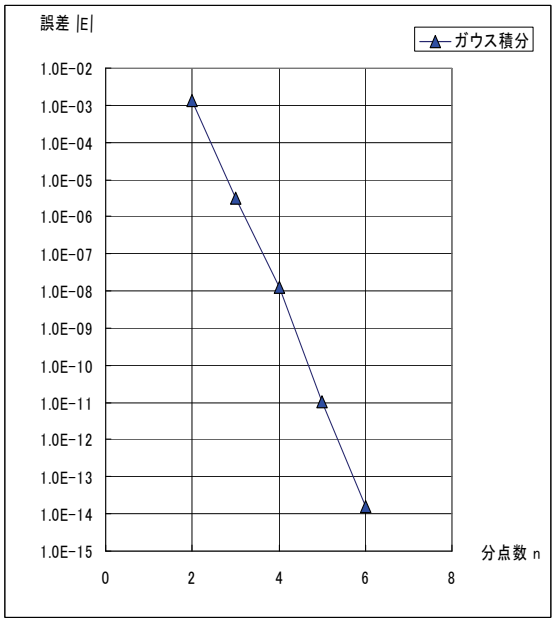
いくつかの数値積分公式により上記の定積分の値を算出し，誤差を比較検討する。なお，ここで示す計算結果はエクセル（精度 15 桁?）を用いたものである。

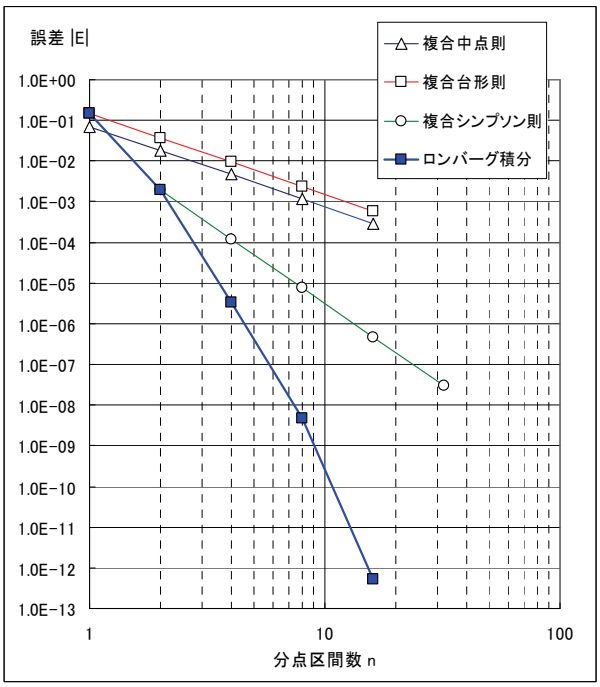
複合中点則，複合台形則，複合シンプソン則により数値積分を行ったときの誤差 E と分点区間数 n との関係を左図に，誤差 E と分点間隔 h との関係を右図に両対数グラフで示す。主要な誤差項が $E=O(h^p)\approx Ch^p$ である場合，log をとると $\log|E|\approx\log|C|+p(\log h)$ となり，誤差|E|は勾配 p の直線上にほぼ並ぶはずである。
複合中点則と複合台形則の誤差は $O(h^2)$，シンプソン則の誤差は $O(h^4)$ であり，右図中で計算値の誤差はそれぞれ勾配 2 と勾配 4 の直線上にほぼ並んでいることが確認される。また複合シンプソン則の誤差は他の 2 者に比べて小さいことがグラフに現れている。





n 点ガウス・ルジャンドル積分公式の誤差を分点数 n を横軸にとって左図に示す。また，ロンバーグ積分 $T_{11}, T_{22}, T_{33}, T_{44}, T_{55}$ の誤差（それぞれ $O(h_1^2)$，$O(h_2^4)$，$O(h_3^6)$，$O(h_4^8)$，$O(h_5^{10})$，$h_k=(1/2)^{k-1}$）を分点区間数 n を横軸にとって上述の複合積分則とともに右図に示す。T_{11} は 1 区間の台形則，T_{22} は 1 区間（分点区間数 2）のシンプソン則と同じであるので，誤差も同様であることが確認できる。
以上の図より，ガウス積分の精度が一番良いことが確認される。





参考文献

高倉葉子：数値計算の基礎―解法と誤差―，コロナ社（2007）
森口繁一，伊理正夫，武市正人 編：C による算法痛論，東京大学出版会（2000）
森口繁一：数値計算工学，岩波書店（1989）
Heath, Michael T.: Scientific Computing, An Introductory Survey, McGraw-Hill (2002)