数値積分における台形公式の高精度化

数値積分におけるシンプソンの公式やブールの公式（ニュートン・コーツの公式）を，台形公式の様に変形した，台形公式より収束の速いアルゴリズムを考えました。
特徴は，分割数による場合分けが不要であり，台形公式の端点の処理を変えただけで，滑らかな関数に対してシンプソンの公式やブールの公式と同程度に高速に収束することです。特に，端点の処理だけで収束が速くなるのが不思議であったので，高次の場合にも同様であるか試してみたところ，10 次式での近似まで実際に成り立つことを確かめました。
ただし，高次式での近似が常に速く収束するとは限りません。f(x)=1/SQR(1-(k*SIN(x))^2) (k2≪1)を 0 から π/2 まで積分するのような場合は，変形版は元の台形公式やシンプソンの公式より遅く収束します。また，例えば f(x)=SQR(1-x^2) を-1 から 1 まで積分して π/2 を求めるような場合は，端点が微分可能でないため，シンプソンの公式やブールの公式を用いても，高次収束することはありません。
個々の関数に対して最適なアルゴリズムを用いれば良いのですが，高次の公式は低次の公式より，平均すると速く収束します。
さらに，和を求めるのに，順番に足していくのではなく，二分木構造により和を求めることにより，丸め誤差の集積が起こらないようにしました。

台形公式

$h=\frac{(b-a)}{n}$，$x_i=a+i\times h$ $(i=0,1,2,\cdots\cdots,n)$ として $y_i=f(x_i)$ とすると

$$\int_{x_0}^{x_n} f(x)dx \approx h\left\{\frac{y_0+y_n}{2}+(y_1+y_2+y_3+\cdots\cdots+y_{n-1})\right\}$$

シンプソンの公式やブールの公式を，この様な形に変形する。

シンプソンの公式の変形

シンプソンの公式を，台形公式の様な形に変える。
シンプソンの公式による数値積分は $h=\frac{(b-a)}{2}$ として

$$\int_a^b f(x)dx \approx \frac{h}{3}\left\{f(a)+4f\left(\frac{a+b}{2}\right)+f(b)\right\}$$

と表される。この公式を合成する。

$h=\frac{(b-a)}{n}$，$x_i=a+i\times h$ $(i=0,1,2,\cdots\cdots,n)$ として $y_i=f(x_i)$ とすると

$$\int_{x_0}^{x_2} f(x)dx+\int_{x_1}^{x_3} f(x)dx+\int_{x_2}^{x_4} f(x)dx+\cdots\cdots+\int_{x_{n-2}}^{x_n} f(x)dx$$

$$\approx\frac{h}{3}\{(y_0+4y_1+y_2)+(y_1+4y_2+y_3)+(y_2+4y_3+y_4)+\cdots\cdots+(y_{n-2}+4y_{n-1}+y_n)\}$$

となる。この等式を変形すると

$$\int_{x_0}^{x_1} f(x)dx+2\int_{x_1}^{x_{n-1}} f(x)dx+\int_{x_{n-1}}^{x_n} f(x)dx\approx\frac{h}{3}(y_0+5y_1+6y_2+\cdots\cdots+6y_{n-2}+5y_{n-1}+y_n)\ \cdots①$$

ここで

$$x_{\frac{1}{2}}=\frac{x_1+x_2}{2}, x_{n-\frac{1}{2}}=\frac{x_{n-1}+x_n}{2} \quad , \quad y_{\frac{1}{2}}=f\left(x_{\frac{1}{2}}\right), y_{n-\frac{1}{2}}=f\left(x_{n-\frac{1}{2}}\right)$$

とすると，シンプソンの公式により

$$\int_{x_0}^{x_1} f(x)dx+\int_{x_{n-1}}^{x_n} f(x)dx \approx \frac{h}{6}\left(y_0+4y_{\frac{1}{2}}+y_1+y_{n-1}+4y_{n-\frac{1}{2}}+y_n\right) \quad \cdots②$$

であるから，（①＋②）÷2　より

$$\int_{x_0}^{x_n} f(x)dx \approx \frac{h}{12}\left(3y_0+4y_{\frac{1}{2}}+11y_1+12y_2+12y_3+\cdots\cdots+12y_{n-2}+11y_{n-1}+4y_{n-\frac{1}{2}}+3y_n\right)$$

$$=\frac{h}{12}\left\{3y_0+4y_{\frac{1}{2}}+11y_1+11y_{n-1}+4y_{n-\frac{1}{2}}+3y_n+12\left(y_2+y_3+\cdots\cdots+y_{n-2}\right)\right\}$$

$$=\frac{h}{12}\left\{3\left(y_0+y_n\right)+4\left(y_{\frac{1}{2}}+y_{n-\frac{1}{2}}\right)+11\left(y_1+y_{n-1}\right)\right\}+h\left(y_2+y_3+\cdots\cdots+y_{n-2}\right)$$

となる。n は3以上の自然数で，偶数でも奇数でもよい。

シンプソン 3/8 公式の変形

次に，シンプソン 3/8 公式を台形公式の様な形に変える。

ンプソン 3/8 公式による数値積分は　$h=\frac{(b-a)}{3}$　として

$$\int_a^b f(x)dx \approx \frac{3}{8}h\left\{f(a)+3f(a+h)+3f(a+2h)+f(b)\right\}$$

と表される。この公式を合成する。

$h=\frac{(b-a)}{n}$，$x_i=a+i\times h$ $(i=0,1,2,\cdots\cdots,n)$ として　$y_i=f(x_i)$　とすると

$$\int_{x_0}^{x_3} f(x)dx+\int_{x_1}^{x_4} f(x)dx+\int_{x_2}^{x_5} f(x)dx+\cdots\cdots+\int_{x_{n-3}}^{x_n} f(x)dx$$

$$\approx\frac{3}{8}h\left\{\left(y_0+3y_1+3y_2+y_3\right)+\left(y_1+3y_2+3y_3+y_4\right)+\cdots\cdots\right.$$

$$\left.\cdots\cdots+\left(y_{n-4}+3y_{n-3}+3y_{n-2}+y_{n-1}\right)+\left(y_{n-3}+3y_{n-2}+3y_{n-1}+y_n\right)\right\}$$

となる。この等式を変形すると

$$\int_{x_0}^{x_1} f(x)dx+2\int_{x_1}^{x_2} f(x)dx+3\int_{x_3}^{x_{n-2}} f(x)dx+2\int_{x_{n-2}}^{x_{n-1}} f(x)dx+\int_{x_{n-1}}^{x_n} f(x)dx$$

$$\approx\frac{3}{8}h\left(y_0+4y_1+7y_2+8y_3+8y_4+\cdots\cdots+8y_{n-3}+7y_{n-2}+4y_{n-1}+y_n\right)$$

両辺に

$$\int_{x_0}^{x_1} f(x)dx+\int_{x_0}^{x_2} f(x)dx+\int_{x_{n-2}}^{x_n} f(x)dx+\int_{x_{n-1}}^{x_n} f(x)dx$$

をたして

$$3\int_{x_0}^{x_n} f(x)dx \approx \frac{3}{8}h\left(y_0+4y_1+7y_2+8y_3+8y_4+\cdots\cdots+8y_{n-3}+7y_{n-2}+4y_{n-1}+y_n\right)$$

$$+\int_{x_0}^{x_1} f(x)dx+\int_{x_0}^{x_2} f(x)dx+\int_{x_{n-2}}^{x_n} f(x)dx+\int_{x_{n-1}}^{x_n} f(x)dx$$

ここで，シンプソン 3/8 の公式により

$$\int_{x_0}^{x_1} f(x)dx\approx\frac{1}{8}h\left(y_0+3y_{\frac{1}{3}}+3y_{\frac{2}{3}}+y_1\right)$$

$$\int_{x_0}^{x_2} f(x)dx\approx\frac{2}{8}h\left(y_0+3y_{\frac{2}{3}}+3y_{\frac{4}{3}}+y_2\right)$$

であるから，

$$\int_{x_0}^{x_1} f(x)dx+\int_{x_0}^{x_2} f(x)dx\approx\frac{h}{8}\left(3y_0+3y_{\frac{1}{3}}+9y_{\frac{2}{3}}+y_1+6y_{\frac{4}{3}}+2y_2\right)$$

$\int_{x_{n-2}}^{x_n} f(x)dx+\int_{x_{n-1}}^{x_n} f(x)dx$ も同様である。よって，次の式を得る。

$$3\int_{x_0}^{x_n} f(x)dx\approx 3h(y_3+y_4+\cdots\cdots+y_{n-3})$$

$$+\frac{h}{8}\left\{6(y_0+y_n)+3\left(y_{\frac{1}{3}}+y_{n-\frac{1}{3}}\right)+9\left(y_{\frac{2}{3}}+y_{n-\frac{2}{3}}\right)+13(y_1+y_{n-1})+6\left(y_{\frac{4}{3}}+y_{n-\frac{4}{3}}\right)+23(y_2+y_{n-2})\right\}$$

3 で割って

$$\int_{x_0}^{x_n} f(x)dx\approx h(y_3+y_4+\cdots\cdots+y_{n-3})$$

$$+\frac{h}{24}\left\{6(y_0+y_n)+3\left(y_{\frac{1}{3}}+y_{n-\frac{1}{3}}\right)+9\left(y_{\frac{2}{3}}+y_{n-\frac{2}{3}}\right)+13(y_1+y_{n-1})+6\left(y_{\frac{4}{3}}+y_{n-\frac{4}{3}}\right)+23(y_2+y_{n-2})\right\}$$

となる。n は 5 以上の任意の自然数である。

ブールの公式の変形

次に，ブールの公式を台形公式の様な形に変える。

ブールの公式による数値積分は $h=\frac{(b-a)}{4}$ として

$$\int_a^b f(x)dx\approx\frac{2}{45}h\{7f(a)+32f(a+h)+12f(a+2h)+32f(a+3h)+7f(b)\}$$

と表される。この公式を合成する。

$h=\frac{(b-a)}{n}$，$x_i=a+i\times h$ $(i=0,1,2,\cdots\cdots,n)$ として $y_i=f(x_i)$ とすると

$$\int_{x_0}^{x_4} f(x)dx+\int_{x_1}^{x_5} f(x)dx+\int_{x_2}^{x_6} f(x)dx+\cdots\cdots+\int_{x_{n-4}}^{x_n} f(x)dx$$

$$\approx\frac{2}{45}h\{(7y_0+32y_1+12y_2+32y_3+7y_4)+(7y_1+32y_2+12y_3+32y_4+7y_5)+\cdots\cdots$$

$$\cdots\cdots+(7y_{n-5}+32y_{n-4}+12y_{n-3}+32y_{n-2}+7y_{n-1})+(7y_{n-4}+32y_{n-3}+12y_{n-2}+32y_{n-1}+7y_n)\}$$

となる。この等式を変形すると

$$\int_{x_0}^{x_1} f(x)dx+2\int_{x_1}^{x_2} f(x)dx+3\int_{x_2}^{x_3} f(x)dx+4\int_{x_3}^{x_{n-3}} f(x)dx+3\int_{x_{n-3}}^{x_{n-2}} f(x)dx+2\int_{x_{n-2}}^{x_{n-1}} f(x)dx+\int_{x_{n-1}}^{x_n} f(x)dx$$

$$\approx\frac{2}{45}h(7y_0+39y_1+51y_2+83y_3+90y_4+90y_5+\cdots\cdots+90y_{n-4}+83y_{n-3}+51y_{n-2}+39y_{n-1}+7y_n)$$

両辺に

$$\int_{x_0}^{x_1} f(x)dx+\int_{x_0}^{x_2} f(x)dx+\int_{x_0}^{x_3} f(x)dx+\int_{x_{n-3}}^{x_n} f(x)dx+\int_{x_{n-2}}^{x_n} f(x)dx+\int_{x_{n-1}}^{x_n} f(x)dx$$

をたして

$$4\int_{x_0}^{x_n} f(x)dx\approx\frac{2}{45}h(7y_0+39y_1+51y_2+83y_3+90y_4+90y_5+\cdots\cdots+90y_{n-4}+83y_{n-3}+51y_{n-2}+39y_{n-1}+7y_n)$$

$$+\int_{x_0}^{x_1} f(x)dx+\int_{x_0}^{x_2} f(x)dx+\int_{x_0}^{x_3} f(x)dx+\int_{x_{n-3}}^{x_n} f(x)dx+\int_{x_{n-2}}^{x_n} f(x)dx+\int_{x_{n-1}}^{x_n} f(x)dx$$

ここで，ブールの公式により

$$\int_{x_0}^{x_1} f(x)dx\approx\frac{h}{90}\left(7y_0+32y_{\frac{1}{4}}+12y_{\frac{1}{2}}+32y_{\frac{3}{4}}+7y_1\right)$$

$$\int_{x_0}^{x_2} f(x)dx\approx\frac{2}{90}h\left(7y_0+32y_{\frac{1}{2}}+12y_1+32y_{\frac{3}{2}}+7y_2\right)$$

$$\int_{x_0}^{x_3} f(x)dx\approx\frac{3}{90}h\left(7y_0+32y_{\frac{3}{4}}+12y_{\frac{3}{2}}+32y_{\frac{9}{4}}+7y_3\right)$$

であるから，

$$\int_{x_0}^{x_1} f(x)dx+\int_{x_0}^{x_2} f(x)dx+\int_{x_0}^{x_3} f(x)dx$$

$$\approx\frac{h}{90}\left(42y_0+32y_{\frac{1}{4}}+76y_{\frac{1}{2}}+128y_{\frac{3}{4}}+31y_1+100y_{\frac{3}{2}}+14y_2+96y_{\frac{9}{4}}+21y_3\right)$$

$\int_{x_{n-3}}^{x_n} f(x)dx+\int_{x_{n-2}}^{x_n} f(x)dx+\int_{x_{n-1}}^{x_n} f(x)dx$ も同様である。よって，次の式を得る。

$$4\int_{x_0}^{x_n} f(x)dx\approx 4h(y_4+y_5+\cdots\cdots+y_{n-4})$$

$$+\frac{h}{90}\left(70y_0+32y_{\frac{1}{4}}+76y_{\frac{1}{2}}+128y_{\frac{3}{4}}+187y_1+100y_{\frac{3}{2}}+218y_2+96y_{\frac{9}{4}}+353y_3\right)$$

$$+\frac{h}{90}\left(70y_n+32y_{n-\frac{1}{4}}+76y_{n-\frac{1}{2}}+128y_{n-\frac{3}{4}}+187y_{n-1}+100y_{n-\frac{3}{2}}+218y_{n-2}+96y_{n-\frac{9}{4}}+353y_{n-3}\right)$$

4 で割って

$$\int_{x_0}^{x_n} f(x)dx \approx h(y_4+y_5+\cdots\cdots+y_{n-4})$$

$$+\frac{h}{360}\left(70y_0+32y_{\frac{1}{4}}+76y_{\frac{1}{2}}+128y_{\frac{3}{4}}+187y_1+100y_{\frac{3}{2}}+218y_2+96y_{\frac{9}{4}}+353y_3\right)$$

$$+\frac{h}{360}\left(70y_n+32y_{n-\frac{1}{4}}+76y_{n-\frac{1}{2}}+128y_{n-\frac{3}{4}}+187y_{n-1}+100y_{n-\frac{3}{2}}+218y_{n-2}+96y_{n-\frac{9}{4}}+353y_{n-3}\right)$$

となる。n は 7 以上の任意の自然数である。

6 次関数近似の公式

さらに高次の収束をするアルゴリズムを求める。6 次関数のグラフが 7 点を通るとき，　$h=\frac{(b-a)}{6}$　とすると

$$\int_a^b f(x)dx \approx \frac{1}{140}h\{41f(a)+216f(a+h)+27f(a+2h)+272f(a+3h)+27f(a+4h)+216f(a+5h)+41f(b)\}$$

と表される。この公式を合成すると次の様になる。

$$\int_{x_0}^{x_n} f(x)dx \approx h(y_6+y_7+\cdots\cdots+y_{n-6})$$

$$+\frac{h}{5040}(861y_0+216y_{\frac{1}{6}}+459y_{\frac{1}{3}}+920y_{\frac{1}{2}}+945y_{\frac{2}{3}}+1296y_{\frac{5}{6}}+2208y_1+162y_{\frac{4}{3}}+816y_{\frac{3}{2}}$$
$$+567y_{\frac{5}{3}}+2955y_2+2008y_{\frac{5}{2}}+108y_{\frac{8}{3}}+3459y_3+999y_{\frac{10}{3}}+3662y_4+1080y_{\frac{25}{6}}+4999y_5)$$

$$+\frac{h}{5040}(861y_n+216y_{n-\frac{1}{6}}+459y_{n-\frac{1}{3}}+920y_{n-\frac{1}{2}}+945y_{n-\frac{2}{3}}+1296y_{n-\frac{5}{6}}+2208y_{n-1}+162y_{\frac{4}{3}}+816y_{n-\frac{3}{2}}$$
$$+567y_{n-\frac{5}{3}}+2955y_{n-2}+2008y_{n-\frac{5}{2}}+108y_{n-\frac{8}{3}}+3459y_{n-3}+999y_{n-\frac{10}{3}}+3662y_{n-4}+1080y_{n-\frac{25}{6}}+4999y_{n-5})$$

n は 11 以上の任意の自然数である。

8 次関数近似の公式

さらに高次の収束をするアルゴリズムを求める。8 次関数のグラフが 9 点を通るとき，　$h=\frac{(b-a)}{8}$　として

$y_i=f(a+hi)$　$(i=0,1,2,\cdots\cdots,8)$　とすると

$$\int_a^b f(x)dx \approx \frac{4}{14175}h(989y_0+5888y_1-928y_2+10496y_3-4540y_4+10496y_5-928y_6+5888y_7+989y_8)$$

と表される。上記と同様にしてこの公式を合成した式は，n が 15 以上の任意の自然数で成り立つ。

次に，通常の公式による数値積分と，上記の変形判による数値積分の，十進 BASIC にプログラムと 2 進モードでの出力例を示す。

```
LET a=1
LET b=2
LET s=LOG(2)
FOR i=1 TO 10
    LET n=10*2^i
    PRINT USING "#####":n;
```

```
    PRINT daikei(a,b,n)-s;simpson(a,b,n)-s;simpson2(a,b,n)-s;simpson3(a,b,n)-s;boole(a,b,n)-s;int6(a,b,n)-s;int8(a,b,n)-s
NEXT I
END

EXTERNAL FUNCTION f(x)
LET f=1/x
end FUNCTION

EXTERNAL FUNCTION daikei(a,b,n)
LET h=(b-a)/n
LET s=(f(a)+f(b))/2
FOR i=1 TO n-1
    LET s=s+f(a+h*i)
NEXT i
LET daikei=s*h
END FUNCTION

EXTERNAL FUNCTION simpson(a,b,n)
LET h=(b-a)/n
LET s=(f(a)+f(b))
LET t=0
FOR i=1 TO n-1 STEP 2
    LET t=t+f(a+h*i)
NEXT i
LET s=s+4*t
LET t=0
FOR i=2 TO n-2 STEP 2
    LET t=t+f(a+h*i)
NEXT i
LET s=s+2*t
LET simpson=s*h/3
END FUNCTION

EXTERNAL FUNCTION simpson2(a,b,n)
LET h=(b-a)/n
LET s=(f(a)+f(b))*3
LET s=s+(f(a+h/2)+f(b-h/2))*4
LET s=s+(f(a+h)+f(b-h))*11
LET s=s/12
FOR i=2 TO n-2
    LET s=s+f(a+h*i)
NEXT i
LET simpson2=s*h
END FUNCTION

EXTERNAL FUNCTION simpson3(a,b,n)
LET h=(b-a)/n
```

```
LET s=(f(a)+f(b))*6
LET s=s+(f(a+h/3)+f(b-h/3))*3
LET s=s+(f(a+h*2/3)+f(b-h*2/3))*9
LET s=s+(f(a+h)+f(b-h))*13
LET s=s+(f(a+h*4/3)+f(b-h*4/3))*6
LET s=s+(f(a+h*2)+f(b-h*2))*23
LET s=s/24
FOR i=3 TO n-3
    LET s=s+f(a+h*i)
NEXT i
LET simpson3=s*h
END FUNCTION

EXTERNAL FUNCTION boole(a,b,n)
LET h=(b-a)/n
LET s=(f(a)+f(b))*70
LET s=s+(f(a+h/4)+f(b-h/4))*32
LET s=s+(f(a+h/2)+f(b-h/2))*76
LET s=s+(f(a+h*3/4)+f(b-h*3/4))*128
LET s=s+(f(a+h)+f(b-h))*187
LET s=s+(f(a+h*3/2)+f(b-h*3/2))*100
LET s=s+(f(a+h*2)+f(b-h*2))*218
LET s=s+(f(a+h*9/4)+f(b-h*9/4))*96
LET s=s+(f(a+h*3)+f(b-h*3))*353
LET s=s/360
FOR i=4 TO n-4
    LET s=s+f(a+h*i)
NEXT i
LET boole=s*h
END FUNCTION

EXTERNAL FUNCTION int6(a,b,n)
LET h=(b-a)/n
LET s=(f(a)+f(b))*861
LET s=s+(f(a+h/6)+f(b-h/6))*216
LET s=s+(f(a+h/3)+f(b-h/3))*459
LET s=s+(f(a+h/2)+f(b-h/2))*920
LET s=s+(f(a+h*2/3)+f(b-h*2/3))*945
LET s=s+(f(a+h*5/6)+f(b-h*5/6))*1296
LET s=s+(f(a+h)+f(b-h))*2208
LET s=s+(f(a+h*4/3)+f(b-h*4/3))*162
LET s=s+(f(a+h*3/2)+f(b-h*3/2))*816
LET s=s+(f(a+h*5/3)+f(b-h*5/3))*567
LET s=s+(f(a+h*2)+f(b-h*2))*2955
LET s=s+(f(a+h*5/2)+f(b-h*5/2))*2008
LET s=s+(f(a+h*8/3)+f(b-h*8/3))*108
LET s=s+(f(a+h*3)+f(b-h*3))*3459
```

```
LET s=s+(f(a+h*10/3)+f(b-h*10/3))*999
LET s=s+(f(a+h*4)+f(b-h*4))*3662
LET s=s+(f(a+h*25/6)+f(b-h*25/6))*1080
LET s=s+(f(a+h*5)+f(b-h*5))*4999
LET s=s/5040
FOR i=6 TO n-6
    LET s=s+f(a+h*i)
NEXT i
LET int6=s*h
END FUNCTION

EXTERNAL FUNCTION int8(a,b,n)
LET h=(b-a)/n
LET s=(f(a)+f(b))*35604
LET s=s+(f(a+h/8)+f(b-h/8))*5888
LET s=s+(f(a+h/4)+f(b-h/4))*10848
LET s=s+(f(a+h*3/8)+f(b-h*3/8))*28160
LET s=s+(f(a+h/2)+f(b-h/2))*17156
LET s=s+(f(a+h*5/8)+f(b-h*5/8))*39936
LET s=s+(f(a+h*3/4)+f(b-h*3/4))*52608
LET s=s+(f(a+h*7/8)+f(b-h*7/8))*47104
LET s=s+(f(a+h)+f(b-h))*43213
LET s=s+(f(a+h*9/8)+f(b-h*9/8))*31488
LET s=s+(f(a+h*5/4)+f(b-h*5/4))*16352
LET s=s+(f(a+h*3/2)+f(b-h*3/2))*20940
LET s=s+(f(a+h*7/4)+f(b-h*7/4))*5280
LET s=s+(f(a+h*15/8)+f(b-h*15/8))*83968
LET s=s+(f(a+h*2)+f(b-h*2))*31410
LET s=s+(f(a+h*9/4)+f(b-h*9/4))*60192
LET s=s+(f(a+h*5/2)+f(b-h*5/2))*19284
LET s=s+(f(a+h*21/8)+f(b-h*21/8))*91136
LET s=s+(f(a+h*3)+f(b-h*3))*103575
LET s=s+(f(a+h*25/8)+f(b-h*25/8))*52480
LET s=s+(f(a+h*7/2)+f(b-h*7/2))*(-8228)
LET s=s+(f(a+h*15/4)+f(b-h*15/4))*58336
LET s=s+(f(a+h*4)+f(b-h*4))*99196
LET s=s+(f(a+h*35/8)+f(b-h*35/8))*102912
LET s=s+(f(a+h*9/2)+f(b-h*9/2))*(-5568)
LET s=s+(f(a+h*5)+f(b-h*5))*184153
LET s=s+(f(a+h*21/4)+f(b-h*21/4))*28832
LET s=s+(f(a+h*6)+f(b-h*6))*177718
LET s=s+(f(a+h*49/8)+f(b-h*49/8))*41216
LET s=s+(f(a+h*7)+f(b-h*7))*225811
LET s=s/226800
FOR i=8 TO n-8
    LET s=s+f(a+h*i)
NEXT i
```

LET int8=s*h
END FUNCTION

20	1.56201232748732E-4	1.94105170825409E-7	1.7629476944947E-7	3.65357800458099E-7	2.62170829490316E-9	9.00871599540665E-11	5.27866639288277E-12
40	3.90594491954666E-5	1.21880106007666E-8	1.15956145796403E-8	2.50114172706262E-8	4.97863972270807E-11	5.29021271233887E-13	1.04360964314765E-14
80	9.76543427999577E-6	7.62641727547475E-10	7.43554995352724E-10	1.63754210191769E-9	8.60200799479571E-13	2.33146835171283E-15	0
160	2.44139432903356E-6	4.76791939263421E-11	4.70732341995017E-11	1.04778519194326E-10	1.38777878078145E-14	-2.22044604925031E-16	0
320	6.10350817553673E-7	2.98006064269885E-12	2.96129787358268E-12	6.62669918938263E-12	3.33066907387547E-16	1.11022302462516E-16	1.11022302462516E-16
640	1.52587844026719E-7	1.86517468137026E-13	1.85629289717326E-13	4.16444656536896E-13	-2.22044604925031E-16	-1.11022302462516E-16	0
1280	3.81469688059966E-8	1.17683640610267E-14	1.06581410364015E-14	2.52020626589911E-14	-8.88178419700125E-16	-8.88178419700125E-16	-8.88178419700125E-16
2560	9.53674417214501E-9	2.22044604925031E-16	2.1094237467878E-15	2.88657986402541E-15	1.33226762955019E-15	1.22124532708767E-15	1.11022302462516E-15
5120	2.38418507159111E-9	1.4432899320127E-15	-6.66133814775094E-16	-4.44089209850063E-16	-5.55111512312578E-16	-6.66133814775094E-16	-6.66133814775094E-16
10240	5.96050431234119E-10	-1.33226762955019E-15	3.99680288865056E-15	3.88578058618805E-15	3.77475828372553E-15	3.99680288865056E-15	4.10782519111308E-15

c 言語での 1 次，2 次，4 次，6 次，8 次，10 次関数近似公式のコード

```
#include <stdio.h>
#define f(x) 1/(x*x)

double daikei(double a, double b, int n){
	int i;
	double h,s,x;
	h=(b-a)/n;
	s=((f(a))+(f(b)))/2.0;
	for(i=1;i<=n-1;i++){
		x=a+h*i;
		s=s+(f(x));
	}
	return s*h;
}

double simpson(double a, double b, int n){//n は 3 以上
	int i;
	double h,s,x,y;
	h=(b-a)/n;
	s=((f(a))+(f(b)))/4.0;
	x=a+h*0.5;
	y=b-h*0.5;
	s=s+((f(x))+(f(y)))/3.0;
	x=a+h;
	y=b-h;
	s=s+((f(x))+(f(y)))*11.0/12;
	for(i=2;i<=n-2;i++){
		x=a+h*i;
		s=s+(f(x));
	}
	return s*h;
```

```
}

double boole(double a, double b, int n){//n は 7 以上
	int i;
	double h,s,dx,x,y;
	double iti[8]={1.0/4,1.0/2,3.0/4,1.0,3.0/2,2.0,9.0/4,3.0};
	double keisuu[8]={32,76,128,187,100,218,96,353};
	h=(b-a)/n;
	s=((f(a))+(f(b)))*70;
	for(i=0;i<=7;i++){
		dx=h*iti[i];
		x=a+dx;
		y=b-dx;
		s=s+((f(x))+(f(y)))*keisuu[i];
	}
	s=s/360;
	for(i=4;i<=n-4;i++){
		x=a+h*i;
		s=s+(f(x));
	}
	return s*h;
}

double int6(double a, double b, int n){//n は 11 以上
	int i;
	double h,s,dx,x,y;
	double iti[17]={1.0/6,1.0/3,1.0/2,2.0/3,5.0/6,1.0,4.0/3,3.0/2,5.0/3,2.0,5.0/2,8.0/3,3.0,10.0/3,4.0,25.0/6,5.0};
	double keisuu[17]={216,459,920,945,1296,2208,162,816,567,2955,2008,108,3459,999,3662,1080,4999};
	h=(b-a)/n;
	s=((f(a))+(f(b)))*861;
	for(i=0;i<=16;i++){
		dx=h*iti[i];
		x=a+dx;
		y=b-dx;
		s=s+((f(x))+(f(y)))*keisuu[i];
	}
	s=s/5040;
	for(i=6;i<=n-6;i++){
		x=a+h*i;
		s=s+(f(x));
	}
	return s*h;
}

double int8(double a, double b, int n){//n は 15 以上
	int i;
	double h,s,dx,x,y;
```

```
	double iti[29]={1.0/8,1.0/4,3.0/8,1.0/2,5.0/8,3.0/4,7.0/8,1.0,9.0/8,5.0/4,3.0/2,7.0/4,15.0/8,2.0
	,9.0/4,5.0/2,21.0/8,3.0,25.0/8,7.0/2,15.0/4,4.0,35.0/8,9.0/2,5.0,21.0/4,6.0,49.0/8,7.0};
	double keisuu[29]={5888,10848,28160,17156,39936,52608,47104,43213,31488,16352,20940,5280,83968,31410
	,60192,19284,91136,103575,52480,-8228,58336,99196,102912,-5568,184153,28832,177718,41216,225811};
	h=(b-a)/n;
	s=((f(a))+(f(b)))*35604;
	for(i=0;i<=28;i++){
		dx=h*iti[i];
		x=a+dx;
		y=b-dx;
		s=s+((f(x))+(f(y)))*keisuu[i];
	}
	s=s/226800;
	for(i=8;i<=n-8;i++){
		x=a+h*i;
		s=s+(f(x));
	}
	return s*h;
}

double int10(double a, double b, int n){//n は 19 以上
	int i;
	double h,s,dx,x,y;
	double iti[41]={1.0/10,1.0/5,3.0/10,2.0/5,1.0/2,3.0/5,7.0/10,4.0/5,9.0/10
	,1.0,6.0/5,7.0/5,3.0/2,8.0/5,9.0/5
	,2.0,21.0/10,12.0/5,5.0/2,27.0/10,14.0/5
	,3.0,16.0/5,7.0/2,18.0/5
	,4.0,21.0/5,9.0/2,24.0/5,49.0/10
	,5.0,27.0/5,28.0/5
	,6.0,63.0/10,32.0/5
	,7.0,36.0/5
	,8.0,81.0/10
	,9.0};
	double keisuu[41]={106300,164075,591300,67600,958868,776475,1016500,86675,1880200
	,1851848,-504300,205125,2644104,-1527450,628625
	,1177276,2724000,-571875,2136840,2770500,-734250
	,4772079,-2278500,4353576,-3483050
	,4097507,-189450,4377812,-2375550,1906800
	,5210935,-1707150,1839525
	,2621502,3195700,-388200
	,5361569,413675
	,4892386,956700
	,5971453};
	h=(b-a)/n;
	s=((f(a))+(f(b)))*883685;
	for(i=0;i<=40;i++){
		dx=h*iti[i];
```

```
            x=a+dx;
            y=b-dx;
            s=s+((f(x))+(f(y)))*keisuu[i];
        }
        s=s/5987520;
        for(i=10;i<=n-10;i++){
            x=a+h*i;
            s=s+(f(x));
        }
        return s*h;
}

int main(){
        int i,n=20;
        double a=0.5,b=1.5,s=4.0/3;
        for(i=0;i<=10;i++){
            printf("分割数=%5d 誤差=%24.20f %24.20f %24.20f %24.20f %24.20f %24.20f\n",n,daikei(a,b,n)-
s,simpson(a,b,n)-s,boole(a,b,n)-s,int6(a,b,n)-s,int8(a,b,n)-s,int10(a,b,n)-s);
            n=n*2;
        }
}
```

c 言語での 10 次関数近似公式の通常のコード

```
#include <stdio.h>
#define f(x) 1/(x*x)

double int10(double a, double b, int n){
        double h,s,t,x,y;
        int i;
        h=(b-a)/n;
        s=(f(a))+(f(b));
        t=0;
        for(i=10;i<n;i=i+10){
            x=a+h*i;
            t=t+(f(x));
        }
        s=(s+t*2)*16067;
        t=0;
        for(i=1;i<n;i=i+10){
            x=a+h*i;
            y=x+h*8;
            t=t+(f(x))+(f(y));
        }
        s=s+t*106300;
        t=0;
        for(i=2;i<n;i=i+10){
```

```
            x=a+h*i;
            y=x+h*6;
            t=t+(f(x))+(f(y));
        }
        s=s+t*(-48525);
        t=0;
        for(i=3;i<n;i=i+10){
            x=a+h*i;
            y=x+h*4;
            t=t+(f(x))+(f(y));
        }
        s=s+t*272400;
        t=0;
        for(i=4;i<n;i=i+10){
            x=a+h*i;
            y=x+h*2;
            t=t+(f(x))+(f(y));
        }
        s=s+t*(-260550);
        t=0;
        for(i=5;i<n;i=i+10){
            x=a+h*i;
            t=t+(f(x));
        }
        s=s+t*427368;
        return s*h*5.0/299376;
}

int main(){
        int i,n=20;//n は 10 の倍数
        for(i=1;i<=10;i++){
            printf("分割数=%5d 誤差=%24.20f\n",n,int10(0.5,1.5,n)-4.0/3);
            n=n*2;
        }
}
```

数値積分における丸め誤差の解消法

数値積分において，分割数 n を大きくすると，丸め誤差のために，かえって誤差が増加することが知られています。しかし，それは，数を次々に加算することによって起こることであり，和を取る順序を変えれれば丸め誤差は解消することができます。それを示すのが以下のプログラムであり，二分木構造により和を求めるものです。ただし，処理速度が遅いのが玉に瑕です。

ポイントは，次々に加算すると，巨大な数と小さな数を足すことになり，このとき，小さな数の有効数字が小さなものとして足され，丸め誤差が大きく作用します。これを解消するには，常に同じくらいの大きさの数同士の和を取れば，丸め誤差は小さくなります。それを実現するのが二分木構造により 2 つの数の和を求めることであり，このとき，和を取る 2 つの数の大きさは常に同じオーダーになります。

さらに，数値積分においては，最後に和を取った個数 n で割るため，丸め誤差の集積は起こらなくなります。

そのため，sin(x)/(cos(x*x)+1+1.0/1024)のような特異な関数を 0 から 10 まで積分するような場合でも，分割数を十分に大きくできるので，積分値が高精度に求まります。

```
#include <stdio.h>
#include <math.h>
#define f(x) 4/(1+x*x)

double daikei0(double a, double b, int n){
	int i;
	double h,s,x;
	h=(b-a)/n;
	s=((f(a))+(f(b)))/2.0;
	for(i=1;i<=n-1;i++){
		x=a+h*i;
		s=s+(f(x));
	}
	return s*h;
}

double wa(double a,double h,int k, int n){
	if (n>1){
		int m=(n+1)>>1;
		return wa(a,h,k,m)+wa(a,h,k+m,n>>1);
	}else{
		double x=a+h*k;
		return f(x);
	}
}

double daikei(double a,double b,int n){
	double s,h=(b-a)/n;
	s=(f(a)+f(b))/2;
	s=s+wa(a+h,h,0,n-1);
	return s*h;
}

int main(){
	double a=0,b=1,t;
	int n=1000000000;
	t=daikei0(a,b,n);
	printf("誤差=%23.20f\n",t-M_PI);
	t=daikei(a,b,n);
	printf("誤差=%23.20f\n",t-M_PI);
}
```

高精度に出力する実用コード

以下のコードは，sqrt(x)，sqrt(1-x*x)，1/sqrt(x)，1/sqrt(1-x*x))などの関数も積分できるように，置換積分を用いています。区間[0，1]における sin(x)/x，sqrt(sin(x))*cos(x)，log(sin(x))*cos(x) などの積分や，区間[0，10]における sin(x)/(cos(x*x)+1+1.0/1024) の積分も，高精度かつ高速に行います。
ただし，1/sqrt(1-x*x)の積分は，グラフを x 軸方向に 1 だけ平行移動して，1/sqrt(x*(2-x))とし，0 から 1 まで積分します。すなわち，±∞に発散する特異点は，積分の分割，平行移動により，積分区間の端点，かつ，x=0 の位置にしかないようにします。また，1/sqrt(x) より発散が速い関数には対応していません。

```
#include <stdio.h>
#include <math.h>

double g(double x){
	return  1/sqrt(x*(2-x));
}

double f(double a,double ab,double t){
	double y,tt;
	tt=t*t;
	y=a+(((t-70.0/20)*t+84.0/20)*t-35.0/20)*tt*tt*ab;
	tt=tt-t;
	return g(y)*tt*tt*tt;
}

double tanten(double a,double b,double ab,double h){
	int i;
	double s=0,t;
	static double iti[41]={1.0/10,1.0/5,3.0/10,2.0/5,1.0/2,3.0/5,7.0/10,4.0/5,9.0/10
	,1.0,6.0/5,7.0/5,3.0/2,8.0/5,9.0/5
	,2.0,21.0/10,12.0/5,5.0/2,27.0/10,14.0/5
	,3.0,16.0/5,7.0/2,18.0/5
	,4.0,21.0/5,9.0/2,24.0/5,49.0/10
	,5.0,27.0/5,28.0/5
	,6.0,63.0/10,32.0/5
	,7.0,36.0/5
	,8.0,81.0/10
	,9.0};
	static double keisuu[41]={106300,164075,591300,67600,958868,776475,1016500,86675,1880200
	,1851848,-504300,205125,2644104,-1527450,628625
	,1177276,2724000,-571875,2136840,2770500,-734250
	,4772079,-2278500,4353576,-3483050
	,4097507,-189450,4377812,-2375550,1906800
	,5210935,-1707150,1839525
	,2621502,3195700,-388200
	,5361569,413675
	,4892386,956700
	,5971453};
	//s=(f(a,ab,0)+f(b,-ab,0))*883685;
	for(i=0;i<=40;i++){
```

```
        t=h*iti[i];
        s=s+(f(a,ab,t)+f(b,-ab,t))*keisuu[i];
    }
    return s/5987520;
}

double wa(double a,double ab,double h,int k, int n){
    if (n>1){
        int m=(n+1)>>1;
        return wa(a,ab,h,k,m)+wa(a,ab,h,k+m,n>>1);
    }else{
        return f(a,ab,h*k);
    }
}

double int10(double a, double b, int n){
    double h=1.0/n,ab=(b-a)*(-20),s;
    int m=(n-19)/2;
    s=wa(a,ab,h,10,m)+wa(b,-ab,h,10,n-19-m)+tanten(a,b,ab,h);
    return s*(b-a)*(-140)*h;
}

double integral(double a,double b){
    int i,n;
    double s1,s2,r;
    n=80;
    s1=int10(a,b,n);
    for(i=1;i<=24;i++){
        //printf("n=%10d 積分値= %4.20f\n",n,s1);
        n=n*2;
        s2=int10(a,b,n);
        r=(s2-s1)/s1;
        if (r<0)r=-r;
        if (r<1e-14) break;
        s1=s2;
    }
    return s2;
}

int main(){
    double a,b,s;
    a=0;
    b=1;
    s=integral(a,b);
    printf("積分値=%.20f\n",s);
}
```

二重指数関数型数値積分公式(DE 公式)によるコード (有限区間)

比較のために書いてみました。発散が速いため上記のコードでは計算が難しい，1/cbrt(x*x)の 0 から 1 の積分も計算できます。しかし，sinh(x)，cosh (x)，tanh(x)を計算するため速度は遅くなります。

ただし，機能が単純であるためか http://www.kurims.kyoto-u.ac.jp/~ooura/intde-j.html にあるパッケージよりもかなり速いようです。

```
#include <stdio.h>
#include <math.h>
#define RANGE 4.6

double g(double x){
	return 1/sqrt(x*(2-x));
}

double f(double a,double ab,double t){
	double x,sh,ey;
	sh=sinh(t);
	ey=exp(M_PI*sh);
	x=a+ab/(ey+1);
	return g(x)*sqrt(1+sh*sh)/(ey+1/ey+2);
}

double wa(double a,double ab,double h,int k, int n){
	if (n>1){
		int m=(n+1)>>1;
		return wa(a,ab,h,k,m)+wa(a,ab,h,k-m,n>>1);
	}else{
		return f(a,ab,h*k);
	}
}

double int10(double a, double b, int n){
	double h=RANGE*2/n,ab=b-a,s;
	int m=n/2;
	s=wa(a,ab,h,m,m+1)+wa(b,-ab,h,m,n-m);
	return s*ab*M_PI*h;
}

double integral(double a,double b){
	int i,n;
	double s1,s2,r;
	n=40;
	s1=int10(a,b,n);
	for(i=1;i<=25;i++){
		//printf("n=%10d 積分値= %4.20f\n",n,s1);
		n=n*2;
		s2=int10(a,b,n);
```

```
        r=(s2-s1)/s1;
        if (r<0)r=-r;
        if (r<1e-14) break;
        s1=s2;
    }
    return s2;
}

int main(){
    double a,b,s;
    a=0;
    b=1;
    s=integral(a,b);
    printf("積分値=%.20f\n",s);
}
```

二重指数関数型数値積分公式によるコード (a から∞の区間)

```
#include <stdio.h>
#include <math.h>
#define RANGE 4.0

double g(double x){
    return 1/(sqrt(x)*(1+x));
}

double f(double a,double t){
    double x,sh,y;
    sh=sinh(t);
    y=exp(M_PI*sh);
    return g(a+y)*y*sqrt(1+sh*sh);
}

double wa(double a,double h,int k, int n){
    if (n>1){
        int m=(n+1)>>1;
        return wa(a,h,k,m)+wa(a,h,k+m,n>>1);
    }else{
        return f(a,h*k);
    }
}

double int10(double a,int n){
    double h=RANGE*2/n;
    return wa(a,h,-n/2,n)*h*M_PI;
}
```

```
double integral(double a){
	int i,n;
	double s1,s2,r;
	n=40;
	s1=int10(a,n);
	for(i=1;i<=25;i++){
		//printf("n=%10d 積分値= %4.20f\n",n,s1);
		n=n*2;
		s2=int10(a,n);
		r=(s2-s1)/s1;
		if (r<0)r=-r;
		if (r<1e-14) break;
		s1=s2;
	}
	return s2;
}

int main(){
	double a,s;
	a=0;
	s=integral(a);
	printf("積分値=%.20f\n",s);
}
```

二重指数関数型数値積分公式によるコード (－∞から∞の区間)

```
#include <stdio.h>
#include <math.h>
#define RANGE 4.6

double g(double x){
	return 1/(1+x*x);
}

double f(double t){
	double x,sh,y;
	sh=sinh(t);
	y=sinh(M_PI_2*sh);
	return g(y)*sqrt(1+y*y)*sqrt(1+sh*sh);
}

double wa(double h,int k, int n){
	if (n>1){
		int m=(n+1)>>1;
		return wa(h,k,m)+wa(h,k+m,n>>1);
	}else{
		return f(h*k);
```

```
    }
}

double int10(int n){
    double h=RANGE*2/n;
    return wa(h,-n/2,n)*M_PI_2*h;
}

double integral(){
    int i,n;
    double s1,s2,r;
    n=40;
    s1=int10(n);
    for(i=1;i<=25;i++){
        //printf("n=%10d 積分値= %4.20f\n",n,s1);
        n=n*2;
        s2=int10(n);
        r=(s2-s1)/s1;
        if (r<0)r=-r;
        if (r<1e-14) break;
        s1=s2;
    }
    return s2;
}

int main(){
    double s;
    s=integral();
    printf("積分値=%.20f\n",s);
}
```

シンプソンの公式と高次の公式の証明

命題　$f(x)$ を２次関数とし，　$\frac{b-a}{2}=h$　とおくと

$$\int_a^b f(x)dx=\frac{h}{3}\{f(a)+4f(a+h)+f(b)\}$$

証明

ラグランジュ型の補間多項式を用いると

$$f(x)=f(-1)\cdot\frac{(x-0)(x-1)}{(-1-0)(-1-1)}+f(0)\cdot\frac{(x+1)(x-1)}{(0+1)(0-1)}+f(1)\cdot\frac{(x+1)(x-0)}{(1+1)(1-0)}$$

と表される。変形して

$$f(x)=\frac{1}{2}f(-1)\cdot(x^2-x)-f(0)\cdot(x^2-1)+\frac{1}{2}f(1)\cdot(x^2+x)$$

よって

$$\int_{-1}^{1} f(x)dx=\int_{-1}^{1}\left\{\frac{1}{2}f(-1)\cdot(x^2-x)-f(0)\cdot(x^2-1)+\frac{1}{2}f(1)\cdot(x^2+x)\right\}dx$$

$$=\int_{0}^{1}\left\{f(-1)\cdot x^2-2f(0)\cdot(x^2-1)+f(1)\cdot x^2\right\}dx$$

$$=\frac{1}{3}f(-1)+\frac{4}{3}f(0)+\frac{1}{3}f(1)\quad\cdots①$$

一般に $\int_{a}^{b} f(x)dx$ を考える。

$t=\frac{1}{h}(x-a)-1$ とおき，$f(x)=g(t)$ とおくと，$dx=hdt$ であるから

$$\int_{a}^{b} f(x)dx=h\int_{-1}^{1} g(t)dt$$

$g(x)$ は2次関数であるから，①により

$$\int_{-1}^{1} g(t)dt=\frac{1}{3}g(-1)+\frac{4}{3}g(0)+\frac{1}{3}g(1)$$

よって

$$\int_{a}^{b} f(x)dx=h\left\{\frac{1}{3}g(-1)+\frac{4}{3}g(0)+\frac{1}{3}g(1)\right\}$$

ここで，$g(-1)=f(a),g(0)=f(a+h),g(1)=(b)$ であるから

$$\int_{a}^{b} f(x)dx=\frac{h}{3}\{f(a)+4f(a+h)+f(b)\}$$

□

次に，高次の多項式の場合を示す。

命題　$f(x)$ を6次関数とし，$\frac{b-a}{6}=h$ とおき，$y_i=f(a+hi),(i=-3,-2,-1,0,1,2,3)$ とおくと

$$\int_{a}^{b} f(x)dx=\frac{1}{140}h\left(41y_{-3}+216y_{-2}+27y_{-1}+272y_0+27y_1+216y_2+41y_3\right)$$

証明　$a=-3,b=3,h=1$ のときに示せば十分である。

ラグランジュ型の補間多項式を用いると

$$f(x)=y_{-3}\cdot\frac{(x+2)(x+1)(x-0)(x-1)(x-2)(x-3)}{(-3+2)(-3+1)(-3-0)(-3-1)(-3-2)(-3-3)}$$
$$+y_{-2}\cdot\frac{(x+3)(x+1)(x-0)(x-1)(x-2)(x-3)}{(-2+3)(-2+1)(-2-0)(-2-1)(-2-2)(-2-3)}$$
$$+\cdots\cdots+$$
$$+y_{3}\cdot\frac{(x+3)(x+2)(x+1)(x-0)(x-1)(x-2)}{(3+3)(3+2)(3+1)(3-0)(3-1)(3-2)}$$

と表される。これを-3から3まで積分すると

$$\int_{-3}^{3} f(x)\,dx=\frac{1}{140}\left(41\,y_{-3}+216\,y_{-2}+27\,y_{-1}+272\,y_0+27\,y_1+216\,y_2+41\,y_3\right)$$

□

置換積分用関数の求め方

閉区間[0，1]で定義された関数で，端点で微分可能でない関数を，閉区間[0，1]に置換積分する関数の求め方の例を示す。

$f(x)$ は，　$f(0)=0, f(1)=1$ かつ $f'(0)=f^{(2)}(0)=f^{(3)}(0)=0, f'(1)=f^{(2)}(1)=f^{(3)}(1)=0$ を満たす7次関数とする。

$$\int x^3(1-x)^3\,dx=\int\left(-x^6+3x^5-3x^4+x^3\right)dx=-\frac{1}{7}x^7+\frac{1}{2}x^5-\frac{3}{5}x^3+\frac{1}{4}x+C$$

また，部分積分を用いると

$$\int_0^1 x^3(1-x)^3\,dx=\frac{3\cdot2\cdot1}{4\cdot5\cdot6}\int_0^1 x^6\,dx=\frac{3\cdot2\cdot1}{4\cdot5\cdot6\cdot7}\left[x^7\right]_0^1=\frac{3\cdot2\cdot1}{4\cdot5\cdot6\cdot7}=\frac{1}{140}$$

よって

$$f(x)=140\left(-\frac{1}{7}x^7+\frac{1}{2}x^5-\frac{3}{5}x^3+\frac{1}{4}x\right)=-20x^7+70x^6-84x^5+35x^4$$