Chapter 5 Numerical Differentiation and Integration

5-1 Numerical Differentiation

2-point formulae: $f'(x_0) \approx \dfrac{f(x_0+h)-f(x_0)}{h} + O(h) \approx \dfrac{f(x_0)-f(x_0-h)}{h} + O(h)$

3-point formulae: $f'(x_0) \approx \dfrac{1}{2h}[f(x_0+h)-f(x_0-h)] - \dfrac{h^2}{6} f^{(3)}(\xi_0)$

$$\approx \frac{1}{2h}[-3f(x_0)+4f(x_0+h)-f(x_0+2h)] + \frac{h^3}{3} f^{(3)}(\xi_1)$$

$$\approx \frac{1}{2h}[f(x_0-2h)-4f(x_0-h)+3f(x_0)] + \frac{h^3}{3} f^{(3)}(\xi_2)$$

$$f''(x_0) \approx \frac{1}{h^2}[f(x_0-h)-2f(x_0)+f(x_0+h)] - \frac{h^2}{24}[f^{(4)}(\xi_1)+f^{(4)}(\xi_{-1})]$$

$$\approx \frac{1}{h^2}[-f(x_0+3h)+4f(x_0+2h)-5f(x_0+h)+2f(x_0)] + O(h^2)$$

$$\approx \frac{1}{12h^2}[-f(x_0+2h)+16f(x_0+h)-30f(x_0)+16f(x_0-h) - f(x_0-2h)] + O(h^4)$$

$$f'''(x_0) \approx \frac{1}{h^3}[f(x_0+3h)-3f(x_0+2h)+3f(x_0+h)-f(x_0)] + O(h)$$

$$\approx \frac{1}{2h^3}[f(x_0+2h)-2f(x_0+h)+2f(x_0-h)-f(x_0-2h)] + O(h^2)$$

Eg. Find $f'(2.0)$ for $f(x)=xe^x$ by numerical differentiation methods.

(Sol.) $f(x)=xe^x$, $f'(x)=(x+1)e^x$ $\Rightarrow$ exact $f'(2.0)=22.167168$

For $f'(x_0) \approx \dfrac{1}{2h}[f(x_0+h)-f(x_0-h)]$

Choose h=0.1, $f'(2.0) \approx \dfrac{1}{0.2}[f(2.1)-f(1.9)] = 22.28790$

h=0.2, $f'(2.0) \approx \dfrac{1}{0.4}[f(2.2)-f(1.8)] = 22.414163$

For $f'(x_0) \approx \dfrac{1}{2h}[-3f(x_0)+4f(x_0+h)-f(x_0+2h)]$

Choose h=0.1, $f'(2.0) \approx \dfrac{1}{0.2}[-3f(2.0)+4f(2.1)-f(2.2)]$=22.03231

Eg. Find $f''(2.0)$ for $f(x)=xe^x$ by $f''(x_0) \approx \dfrac{1}{h^2}[f(x_0-h)-2f(x_0)+f(x_0+h)]$.

(Sol.) Choose h=0.1, $f''(2.0) \approx \dfrac{1}{0.01}[f(1.9)-2f(2.0)+f(2.1)]$=29.5932

5-2 Numerical Integration by Trapezoidal & Simpson's Rules

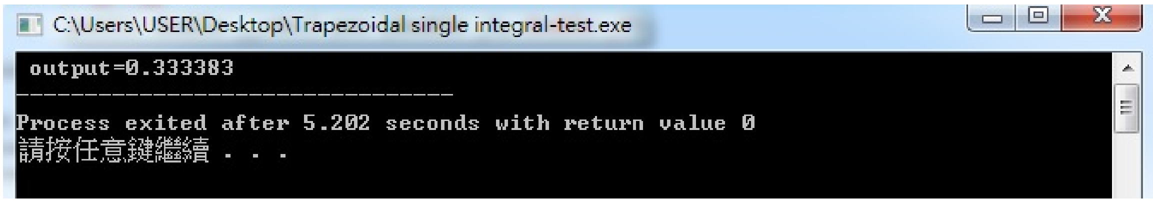
Composite trapezoidal integration method: *h*=(*b*-*a*)/*n*

$$\int_a^b f(x)dx \approx \frac{h}{2}\left[f(a)+f(b)+2\sum_{i=1}^{n-1} f(a+ih)\right]-\frac{(b-a)h^2}{12}f''(\mu)$$

Eg. Use trapezoidal integration method to calculate $\int_0^1 x^2 dx$.

C/C++ program:

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

float f1(float x)
{
return pow(x,2);
}

void SingleIntegral()
{
int m=10000;
float x,xl=0.0, xu=1.0;
float hx, aint, s, output;

hx=(xu-xl)/m;
aint=(f1(xl)+ f1(xu))*hx;
for   (int j=1; j<=m-1; j++)
          {
            x=xl+j*hx;
            s=f1(x);
           aint=aint+s*hx;
          }
output=aint;
printf(" output=%f ", output);
}

main ()
{
SingleIntegral();
}
```



Eg. Compute single integral $\int_0^1 (1+x+x^2)dx$ **.**

C/C++ program:

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

float f1(float x)
{   return 1+x+pow(x,2);   }

void SingleIntegral()
{
int m=10000;
float x,xl=0.0, xu=1.0; float hx, aint, s, output;
hx=(xu-xl)/m;
aint=(f1(xl)+ f1(xu))*hx;
for   (int j=1; j<=m-1; j++)
        {
        x=xl+j*hx; s=f1(x); aint=aint+s*hx;
        }
output=aint; printf(" output=%f ", output);
}

main ()
{   SingleIntegral();   }
```

Eg. Compute double integral $\int_0^1 \int_0^1 xydxdy$ by the trapezoidal rule.

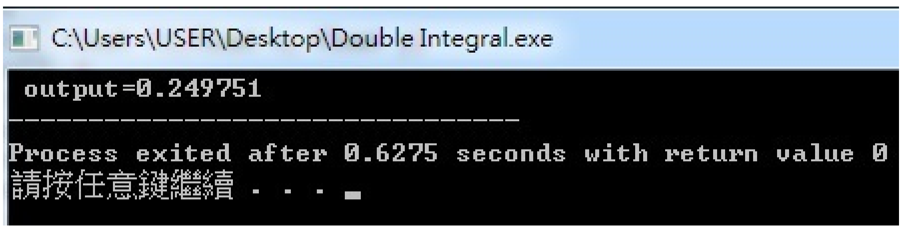
Fortran program:

```
            real f
            write (*,*) 'xl=, xu=, yl=, yu=, m=, n='
            read (*,*) xl, xu, yl, yu, m,n
            hx=(xu-xl)/m
            hy=(yu-yl)/n
            aint=(f(xl,yl)+ f(xl,yu)+ f(xu,yl)+ f(xu,yu))*hy
              do 5 j=1,m-1
                x=xl+j*hx
                s=(f(x,yl)+f(x,yu))/2
                do 10 jj=1,n-1
                  y=yl+jj*hy
                  s=s+f(x,y)
10              continue
                aint=aint+s*hy
5             continue
              aint=aint*hx
            output=aint
            write (*,*) output
            stop
            end


      real function f(x,y)
        f=x*y
        return
      end
```

```
 xl=, xu=, yl=, yu=, m=, n=
0 1 0 1 100 100
  0.2476000
Press any key to continue
```

C/C++ program:

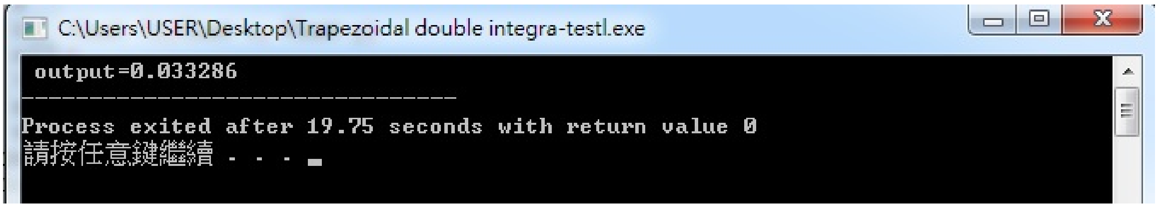
```
#include<stdio.h>
#include<stdlib.h>
#include<math.h>

float f2(float x, float y)
{
Return x*y;
}

void DoubleIntegral()
{
int m=1000,n=1000;
float x,y,xl=0.0, xu=1.0, yl=0.0, yu=1.0;
float hx, hy, aint, s, output;

hx=(xu-xl)/m;
hy=(yu-yl)/n;
aint=(f2(xl,yl)+ f2(xl,yu)+ f2(xu,yl)+ f2(xu,yu))*hy;
for (int j=1; j<=m-1; j++)
            {
                x=xl+j*hx;
                s=(f2(x,yl)+f2(x,yu))/2;
              for ( int i=1; i<=n-1; i++)

                  {
                  y=yl+i*hy;
                  s=s+f2(x,y);
                }
                aint=aint+s*hy;
            }
             aint=aint*hx;
output=aint;
printf(" output=%f ", output);
}


main ()
{
DoubleIntegral();
}
```



Eg. Use trapezoidal integration method to calculate $\int_{0.1}^{0.5}\int_{x^3}^{x^2} e^{y/x}\,dydx$ **.**

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

float f2(float x, float y)
{
  return exp(y/x) ;
}

float yl(float x)
{
  return pow(x,3) ;
}

float yu(float x)
{
  return pow(x,2) ;
}

void DoubleIntegral()
{
int m=2000,n=2000;
float x,y,xl=0.1, xu=0.5;
float hx, hy, hyl, hyu, aint, s, output;
hx=(xu-xl)/m; hyl=(yu(xl)-yl(xl))/n; hyu=(yu(xu)-yl(xu))/n;
aint=(f2(xl,yl(xl))+f2(xl,yu(xl)))*hyl/2.+ (f2(xu,yl(xu))+f2(xu,yu(xu)))*hyu/2.;
for    (int j=1; j<=m-1; j++)
              {
                x=xl+j*hx; hy=(yu(x)-yl(x))/n;
                s=(f2(x,yl(x))+f2(x,yu(x)))/2;
               for ( int i=1; i<=n-1; i++)
                    {
                    y=yl(x)+i*hy;
                    s=s+f2(x,y);
                    }
                aint=aint+s*hy;
              }
```

```
        aint=aint*hx;
output=aint;
printf(" output=%f ", output);
}

main ()
{
DoubleIntegral();
}
```



Composite Simpson's 1/3 integration method: *h*=(*b*-*a*)/2*m*

$$\int_a^b f(x)dx \approx \frac{h}{3}\left[f(a) + 2\sum_{i=1}^{m-1} f(a+2ih) + 4\sum_{i=1}^{m} f(a+(2i-1)h) + f(b) \right] - \frac{(b-a)}{180} h^4 f^{(4)}(\mu)$$

Eg. Use composite Simpson's 1/3 algorithm to calculate $\int_0^{\pi} \sin x dx$ **with *m*=10.**

(Sol.) $h=\frac{\pi-0}{2\times 10}=\frac{\pi}{20}$, $\frac{h}{3}=\frac{\pi}{60}$, $f(0)=f(\pi)=0$

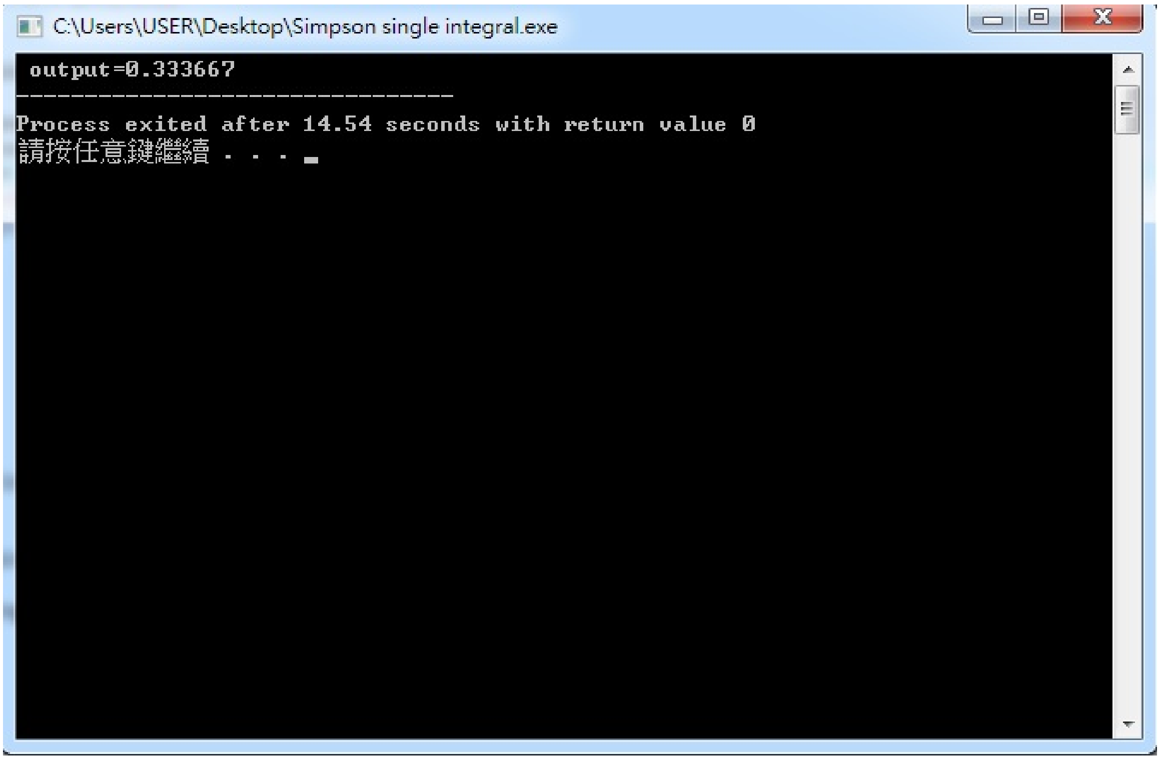
$$\int_0^{\pi} \sin x dx \approx \frac{\pi}{60}\left[2\sum_{j=1}^{9} \sin\left(0+\frac{2j\pi}{20}\right) + 4\sum_{j=1}^{10} \sin\left[0+\frac{(2j-1)\pi}{20}\right] + f(0) + f(\pi) \right] = 2.000006$$

Eg. Use composite Simpson's 1/3 rule to calculate $\int_0^1 x^2 dx$.

C/C++ program

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

float f(float x)
{
  return pow(x,2);
}

void SingleIntegral()
{
float x,y,xl=0.0, xu=1.0;
float hx, aj1, aj2, aj3, aint, output;
int m=1000;
hx=(xu-xl)/2./m;
aj1=0.; aj2=0.; aj3=0.;
for (int i=0; i<=2*m; i++)
        { x=xl+i*hx;
          if (i==0) aj1= aj1+f(x);
          if (i==2*m) aj1= aj1+f(x);
          if (i%2==0) aj2= aj2+f(x);
          if (i%2==1) aj3= aj3+f(x);
          }
aint=(aj1+2*aj2+4*aj3)*hx/3.;
output=aint;
printf(" output=%f ", output);
}
```

```
main ()
{
SingleIntegral();
}
```

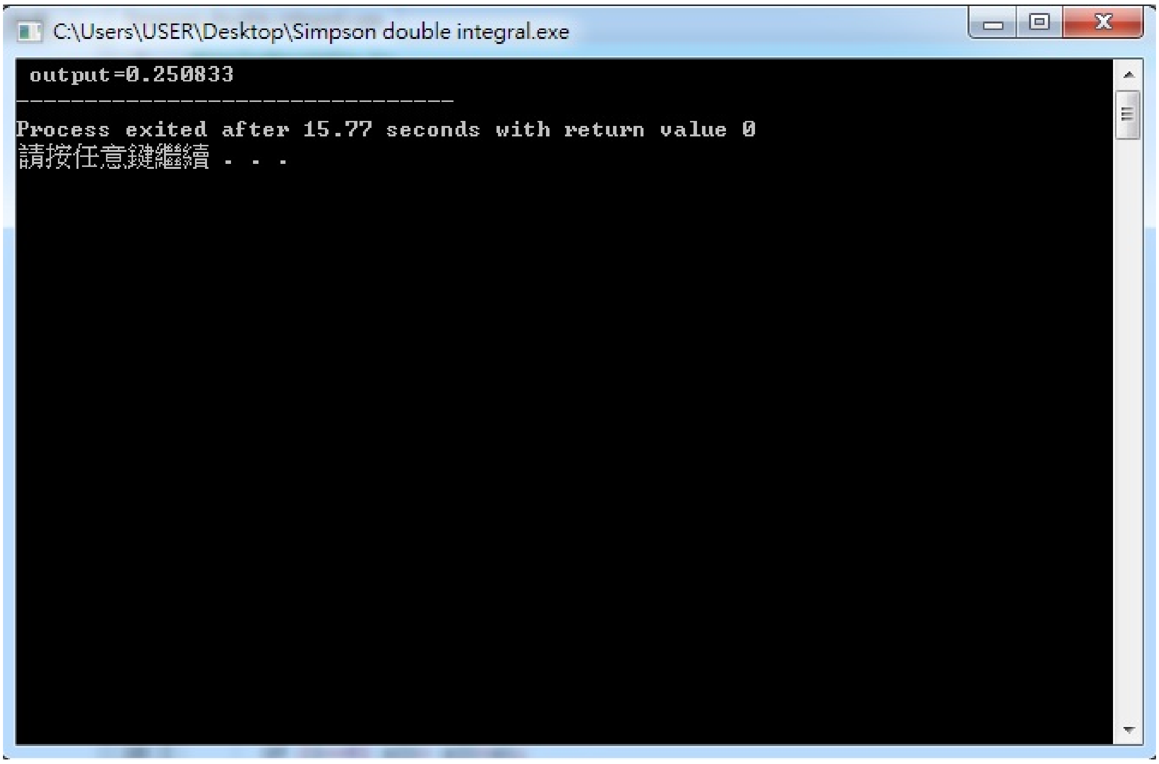


Eg. Use composite Simpson's 1/3 rule to calculate $\int_0^1\int_0^1 xydxdy$ **.**

C/C++ program:

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

float f2(float x, float y)
{    return x*y; }

void DoubleIntegral()
{
float x,y,xl=0.0, xu=1.0, yl=0.0, yu=1.0;
float hx, hy, ak1, ak2, ak3, aj1, aj2, aj3, a1, aint, output;
int m=200, n=200;
```

```
hx=(xu-xl)/2./m;    hy=(yu-yl)/2./n;
aj1=0.; aj2=0. ; aj3=0.;
           for (int i=0; i<=2*n; i++)
             { x=xl+i*hx;
             ak1=f2(x,yl)+f2(x,yu);
             ak2=0.; ak3=0.;
             for (int j=1; j<=2*m-1; j++)
               { y=yl+j*hy;
               if (j%2==0) ak2= ak2+f2(x,y);
               if (j%2==1) ak3= ak3+f2(x,y);
               }
             a1=(ak1+2*ak2+4*ak3)*hy/3. ;
             if (i==0) aj1= aj1+a1;
             if (i==2*n) aj1= aj1+a1;
             if (i%2==0) aj2= aj2+a1;
             if (i%2==1) aj3= aj3+a1;
             }
           aint=(aj1+2*aj2+4*aj3)*hx/3.;
           output=aint;
printf(" output=%f ", output);
}

main ()
{
DoubleIntegral();
}
```



Eg. Use composite Simpson's 1/3 rule to calculate $\int_{0.1}^{0.5}\int_{x^3}^{x^2} e^{y/x}\,dydx$ **.**

(Sol.) $I=\int_a^b\int_{y_\ell(x)}^{y_u(x)} f(x,y)dydx$ Now, a=0.1, b=0.5, $y_l(x)=x^3$, $y_u(x)=x^2$

We can write a Fortran Program to approximate it.

```
external f,yl,yu
write (*,*) 'xl=, xu=, m=, n='
read (*,*) a,b,m,n
h=(b-a)/2./n
  aj1=0.
  aj2=0.
  aj3=0.
  do 1 i=0, 2*n
    x=a+i*h
    hy=(yu(x)-yl(x))/2./m
    ak1=f(x,yl(x))+f(x,yu(x))
    ak2=0.
    ak3=0.
    do 2 j=1, 2*m-1
      y=yl(x)+j*hy
      jr=mod(j,2)
      if (jr.eq.0) ak2= ak2+f(x,y)
      if (jr.eq.1) ak3= ak3+f(x,y)
```

```
2       continue
        a1=(ak1+2*ak2+4*ak3)*hy/3.
        ir=mod(i,2)
        if ((i.eq.0).or.(i.eq.(2*n))) aj1= aj1+a1
        if (ir.eq.0) aj2= aj2+a1
        if (ir.eq.1) aj3= aj3+a1
1      continue
       dint=(aj1+2*aj2+4*aj3)*h/3.
               output=dint
      write (*,*) output
      stop
      end

function f(x,y)
       f=exp(y/x)
       return
      end

function yl(x)
       yl=x**3
       return
      end

      function yu(x)
       yu=x**2
       return
      end
```

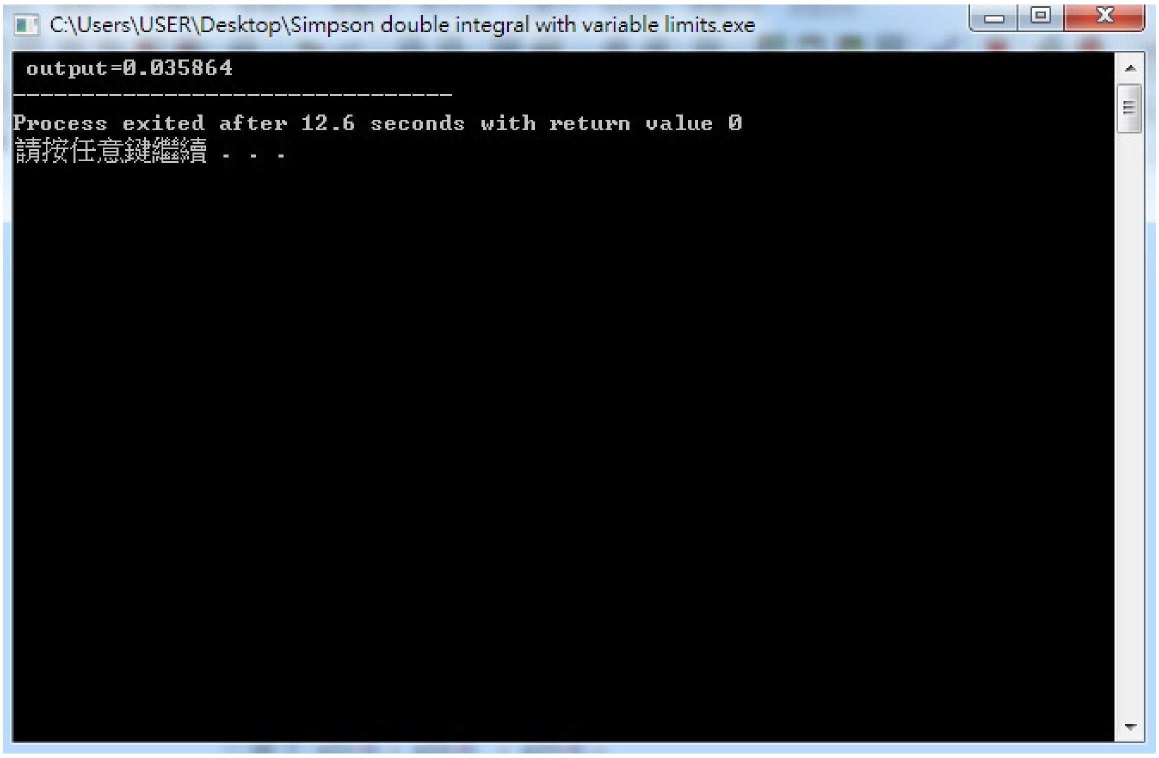
```
 xl=, xu=, m=, n=
0.1 0.5 10 10
  3.5863694E-02
Press any key to continue
```

C/C++ program

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

float f2(float x, float y)
{
 return exp(y/x) ;
```

```
}

float yl(float x)
{
 return pow(x,3) ;
}

float yu(float x)
{
 return pow(x,2) ;
}

void DoubleIntegral()
{
float x,y,xl=0.1, xu=0.5;
float hx, hy, ak1, ak2, ak3, aj1, aj2, aj3, a1, aint, output;
int m=10, n=10;
hx=(xu-xl)/2./m;
aj1=0.; aj2=0. ; aj3=0.;
      for (int i=0; i<=2*n; i++)
       {x=xl+i*hx; hy=(yu(x)-yl(x))/2./n;
       ak1=f2(x,yl(x))+f2(x,yu(x));
       ak2=0.;
       ak3=0.;
       for (int j=1; j<=2*m-1; j++)
         { y=yl(x)+j*hy;
         if (j%2==0) ak2= ak2+f2(x,y);
         if (j%2==1) ak3= ak3+f2(x,y);
                        }
       a1=(ak1+2*ak2+4*ak3)*hy/3. ;
       if (i==0) aj1= aj1+a1;
       if (i==2*n) aj1= aj1+a1;
       if (i%2==0) aj2= aj2+a1;
       if (i%2==1) aj3= aj3+a1;
       }
      aint=(aj1+2*aj2+4*aj3)*hx/3.;
              output=aint;
printf(" output=%f ", output);
```

```
}

main ()
{
DoubleIntegral();
}
```



In **Matlab** language, we can use the following instructions to execute the numerical integral of a function:

```
>>S = 'sqrt(x)';
>>numeric(int(S,0.5,0.6))      % Calculate
```

$\int_{0.5}^{0.6} \sqrt{x}dx$

ans=

0.0741

5-3 Romberg Integration and Gaussian Quadrature method

Romberg integration method:

1. Define $R_{1,1}=\frac{h_1}{2}[f(a)+f(b)]=\frac{b-a}{2}[f(a)+f(b)]$, $h_1=b-a$

$$R_{2,1}=\frac{h_2}{2}[f(a)+2f(a+h_2)+f(b)]=\frac{(b-a)}{4}\left[f(a)+f(b)+2f\left(a+\frac{b-a}{2}\right)\right]$$

$$=\frac{1}{2}\left[R_{1,1}+h_1 f\left(a+\frac{1}{2}h_1\right)\right],\quad \mathrm{h}_2=\frac{\mathrm{b-a}}{2}$$

$$\vdots$$

$$R_{k,1}=\frac{1}{2}\left[R_{k-1,1}+h_{k-1}\cdot\sum_{i=1}^{2^{k-2}} f\left(a+\left(i-\frac{1}{2}\right)h_{k-1}\right)\right],\ k=2,3,\ldots,n$$

$$\Rightarrow \int_a^b f(x)dx=\frac{4R_{k,1}-R_{k-1,1}}{3}+O(h_k^4),\quad h_k=\frac{b-a}{2^{k-1}}$$

2. Define $R_{k,2}=\frac{4R_{k,1}-R_{k-1,1}}{3}$, $k=2, 3, \ldots, n$

3. $R_{i,j}=\frac{4^{j-1}R_{i,j-1}-R_{i-1,j-1}}{4^{j-1}-1}$, $\begin{matrix} i=2,3,4,\cdots,n \\ j=2,3,\cdots,i \\ (2\le j\le i\le n)\end{matrix}$ $\Rightarrow \int_a^b f(x)dx=R_{n,n}$.

Eg. Use Romberg algorithm to calculate $\int_0^\pi \sin x dx$ with n=6.

(Sol.) $R_{1,1}=\frac{\pi}{2}[\sin 0+\sin\pi]=0$, $R_{2,1}=\frac{1}{2}\left[R_{1,1}+\pi\sin\frac{\pi}{2}\right]=1.57079633$

$$R_{3,1}=\frac{1}{2}\left[R_{2,1}+\frac{\pi}{2}\left(\sin\frac{\pi}{4}+\sin\frac{3\pi}{4}\right)\right]=1.89611890$$

$$R_{4,1}=\frac{1}{2}\left[R_{3,1}+\frac{\pi}{4}\left(\sin\frac{\pi}{8}+\sin\frac{3\pi}{8}+\sin\frac{5\pi}{8}+\sin\frac{7\pi}{8}\right)\right]=1.9742160$$

$$R_{5,1}=1.99357034,\quad R_{6,1}=1.99839336$$

$$R_{ij}=\frac{4^{j-1}R_{i,j-1}-R_{i-1,j-1}}{4^{j-1}-1},\quad i=2,3,\cdots,6 \text{ and } j=2,\cdots,i$$

0					
1.57079633	2.09439511				
1.89611890	2.00455976	1.99857073			
1.97423160	2.0026917	1.99998313	2.00000555		
1.99357034	2.00001659	1.99999975	2.00000001	1.99999999	
1.99839336	2.00000103	2.0000000	2.00000000	2.00000000	2.00000000

$$\Rightarrow \int_0^\pi \sin x dx \approx R_{6,6}=2$$

Gaussian Quadrature method:

Set

$t=(2x-a-b)/(b-a)$ $\Rightarrow \int_a^b f(x)dx = \int_{-1}^{1} f\left(\frac{(b-a)t+b+a}{2}\right)\frac{b-a}{2}dt \equiv \int_{-1}^{1} g(t)dt = \sum_{i=0}^{n-1} c_i g(x_i)$,

where $c_i = \int_{-1}^{1}\left[\prod_{\substack{j=0\\ j\neq i}}^{n-1}\frac{(x-x_j)}{(x_i-x_j)}\right]dx$, **and $x_0, x_1, x_2, \ldots, x_{n-1}$ are the zeros of the n^{th}-order Legendre's polynomial.**

Eg. Compute $\int_1^{1.5} e^{-x^2}dx$.

(Sol.) $\int_1^{1.5} e^{-x^2}dx = \frac{1}{4}\int_{-1}^{1} e^{-\frac{(t+5)^2}{16}}dt$

Table for the Gaussian Quadrature method:

As n=2, x_0=0.5773502692, x_1=-x_0, and c_0=1, c_1=1

$$\int_1^{1.5} e^{-x^2}dx \approx \frac{1}{4}[1\cdot e^{-\frac{(x_0+5)^2}{16}} + 1\cdot e^{-\frac{(x_1+5)^2}{16}}] = 0.1094003$$

As n=3, x_0=0.7745966692, x_1=0, x_2=-x_0 and c_0=0.55555555556, c_1=0.8888888889, c_2=c_0

$$\int_1^{1.5} e^{-x^2}dx \approx \frac{1}{4}[c_0\cdot e^{-\frac{(x_0+5)^2}{16}} + c_1\cdot e^{-\frac{(x_1+5)^2}{16}} + c_2\cdot e^{-\frac{(x_2+5)^2}{16}}] = 0.1093642$$

$\int_1^{1.5} e^{-x^2}dx \approx R_{4,4} = 0.1093643$ is obtained by the Romberg method.

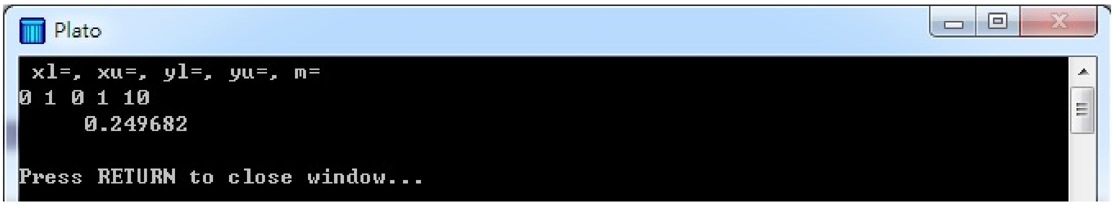
5-4 Hwa's Integral Method (華羅庚積分法)

For computing an n-dimensional multiple integral, it needs n nested loops for most methods. And then more CPU time consumes. But Hwa's method needs only one loop.

Eg. Hwa's method to compute a double integral $\int_0^1 \int_0^1 xy\,dx\,dy$ **.**

```
external QUADR,F
write (*,*) 'xl=, xu=, yl=, yu=, m='
read (*,*) a,b,c,d,m
output=QUADR(F,a,b,c,d,m)
write (*,*) output
stop
end

 Function F(x,y)
  F=x*y
  return
end

FUNCTION QUADR(F,X1,X2,Y1,Y2,M)
   EXTERNAL F
   INTEGER FIBON(20)
   DATA
 FIBON/13,21,34,55,89,144,233,377,610,987,1597,2584,4181,6765,10946,
 17711,28657,46368,75025,121393/
   NH=FIBON(M-1)
   N=FIBON(M)
   XINT=X2-X1
   YINT=Y2-Y1
   KK=0
   QUADR =0.
   DO 10 K=1,N
     X=(XINT*K)/N+X1
     KK=KK+NH
     IF (KK.GE.N) KK=KK-N
     Y=(YINT*KK)/N+Y1
10    QUADR=QUADR+F(X,Y)
   QUADR=QUADR*XINT*YINT/N
   RETURN
 END
```

For xl=0, xu=1, yl=0, yu=1, m=10, $QUADR \approx 0.2496822$. And the exact value of the integral is about 0.25.



Eg. Hwa's method to compute a quadruple integral $\int_0^1\int_0^1\int_0^1\int_0^1 wxyzdwdxdydz$ **.**

```
        external QUADPR,F
        write (*,*) 'wl=, wu=, xl=, xu=, yl=, yu=, zl=, zu=, m='
        read (*,*) a,b,c,d,e,ff,g,h,m
        output=QUADPR(F,a,b,c,d,e,ff,g,h,m)
        write (*,*) output
        stop
        end

    function F(w,x,y,z)
         F=w*x*y*z
         return
        end

    FUNCTION QUADPR(F,W1,W2,X1,X2,Y1,Y2,Z1,Z2,M)
          EXTERNAL F
    INTEGER FN(24),F2(24),F3(24),F4(24)
          INTEGER H2,H3,H4
    DATA
        FN/60,118,180,286,440,562,732,932,1142,1354,2129,3001,4001,5003,600
        7,8191,10007,20039,28117,39029,57091,82001,100063,147312/
          DATA
        F2/8,18,8,16,21,53,248,116,150,492,766,174,113,792,1351,2488,1206,196
        68,17549,30699,52590,57270,92313,136641/
    DATA
        F3/18,40,46,94,136,89,294,288,187,550,1281,266,766,1889,5080,5939,34
        21,17407,1900,34367,48787,58903,24700,116072/
          DATA
        F4/22,52,74,138,216,221,324,314,274,658,1906,1269,2537,191,3086,7859
        ,2842,14600,24455,605,38790,17672,95582,76424/
          N=FN(M)
          H2=F2(M)
          H3=F3(M)
          H4=F4(M)
          WINT=W2-W1
          XINT=X2-X1
          YINT=Y2-Y1
          ZINT=Z2-Z1
```

```
          K2=0.
          K3=0.
          K4=0.
          QUADR=0.
          DO 10 K=1,N
                W=(WINT*K)/N+W1
                K2=K2+H2
             IF (K2.GE. N) K2=K2-N
                X=(XINT*K2)/N+X1
                K3=K3+H3
             IF (K3.GE.N) K3=K3-N
                Y=(YINT*K3)/N+Y1
                K4=K4+H4
             IF (K4.GE.N) K4=K4-N
                Z=(ZINT*K4)/N+Z1
10           QUADPR=QUADPR+F(W,X,Y,Z)
             QUADPR=QUADPR*WINT*XINT*YINT*ZINT/N
          RETURN
       END
```

For wl=0, wu=1, xl=0, xu=1, yl=0, yu=1, zl=0, zu=1, m=10, $QUADPR \approx$ 0.0615789. And the exact value of the integral is about 0.0625.