6

NUMERICAL INTEGRATION

Numerical integration methods are classified by taking into account both the integrand function's type and the values for the integration's limits of integration.

If the integration function is a function with only one argument (*one dimensional integrals*) then we deal with **quadrature formulae**. If the function to be integrated has two arguments (*two dimensional integrals*), then we deal with **cubature formulae**.

I. The first group of methods refers to the continuous functions and with finite limits of integration. These methods are divided in two subgroups according to the dividing module of the interval of integration:

a) Methods that divide the integration interval into subintervals of the same length, the number of the subintervals being imposed by the operator. Among these methods we will study: the rectangular rule, the trapezoidal rule, Simpson's method and Richardson's method;

b) Methods that divide the integration interval so that the computational error is minimal. Among these methods we will study the Gaussian quadrature formula.

II. A second group of methods refers to unfit integrals, namely the integration of functions with discontinuities of first and second kind, the integration of continuous functions on infinite integration intervals.

III. A third group of numerical methods for integrating deals with the dual integration of functions of two variables. These are also called *cubature* integration formulas. We will deal with the trapezoidal rule and Simpson rule.

6.1. INTEGRATION OF SINGLE INTEGRALS. CONSTANT DIVISION METHODS

These methods divide the integration interval in a number of *n* subintervals of equal length. Obviously, the number *n* affects the precision of the outcome of integration, as follows: the greater the *n*, the higher the precision. These two sizes are directly proportional. *n* is a number chosen by the designer (programmer itself).

6.1.1. THE RECTANGLE METHOD (the *rectangular rule*)

This method of calculation has big errors for functions that are not constant. We can use this technique in cases in which there are no precision requirements in evaluating an integral. It is common sense to say that, if the number of subintervals of the integration interval increases then the error of calculation decreases.

This increase in the number of subintervals is done taking in consideration that the time of calculation will be worse. It is considered:

$$I = \int_a^b f(x)\mathrm{d}x \tag{6.1}$$

where $f(x)$ is a continuous function on $[a,b]$ and a, b are finite values.

In figure 6.1 ***I*** represents the area below the graph of the function *f(x)* and limited by the two vertical dotted lines (as *f(a)* and *f(b)*).

The numerical computation of this integration is carried out by dividing (*partitioning*) the interval $[a,b]$ in ***n*** equal parts, using:

$$\Delta x_i = \frac{b-a}{n} = x_{i+1} - x_i = h,\ \boldsymbol{i} = 0,1,...,\boldsymbol{n}-1. \tag{6.2}$$

The *area* of each such small rectangle is approximately given by:

$$s_i = \frac{b-a}{n} \cdot f(x_i) \tag{6.3}$$

and it is gathered

$$\sum_{i=0}^{n-1} s_i = h \cdot \sum_{i=0}^{n-1} f(x_i) \tag{6.4}$$

where $x_i = a + h \cdot i$

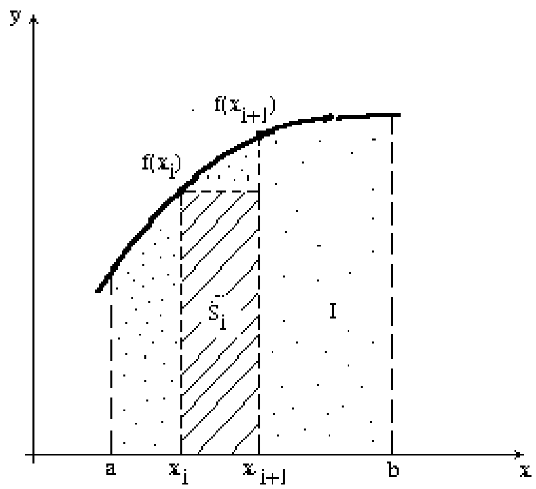

*Figure 6.1 - Graphical representation of the integral, **I**.*

Formula (6.4) represents the ***integration formula*** based on ***rectangular small areas***.

If the integrand is a constant function (a polynomial of degree 0) then this method gives no errors, because the area formed from Ox axis and the verticals from the heads of the interval is equal to the integral of the function on the given interval.

This method has no practical applications due to its large computational error.

6.1.1.1. Algorithm 6.1. The Rectangular rule.

```
Variables
(
ls: left limit of the integration interval, real;
ld: right limit of the integration interval, real;
n: number of subintervals, integer;
sum: value of the integral, real;
)
{
  sum=0;
  compute h=(ld-ls)/nrpas;
  for i=0,…,n-1  compute sum =sum +h* f(ls+i*h);
  print sum as the integration value;
}
```

6.1.1.2. Implementation of the Algorithm 6.1. The rectangular rule.

```
/* Function that implements the rectangular formula
 * for integration.
 * This function returns the value of the integral.
 */
double RectangleF
(
  double (*f)(double),
  double ls,
  double ld,
  int nrpas
)
{
   int i;
   double suma=0,h;
   h=(ld-ls)/nrpas;
   for(i=0;i<=nrpas-1;i++) suma += h*f(ls+i*h);
   return suma;
}
```

6.1.2. THE TRAPEZOIDAL METHOD (the *trapezoidal rule*)

$I = \int_a^b f(x)\mathrm{d}x$, where *f(x)* is continuous on $[a,b]$ and a, b are finite.

The function is represented through the area of the dotted line in figure 6.2, and the integral ***I*** represents the area delimited by the graphic of function *f(x)*, the Ox axis and by the two dotted vertical lines (in *f(a)* and *f(b)*).

The $[a,b]$ interval is divided into ***n*** equal parts, having a *spacing* of:

$$h_i = (x_n - x_0)/n, \text{ for } i=0,1,\ldots,n-1,\ x_0=a \text{ and } x_n=b$$

Every small area $\boldsymbol{I_i}$ is approximated with the area of some trapezoid having for its vertex the following coordinates: $(x_i, x_{i+1}, f(x_i), f(x_{i+1}))$. Such a small area is denoted by S_i in figure 6.2.

That is why every such small area is:

$$I_i = h_i \cdot (f(x_{i+1}) + f(x_i))/2 \tag{6.5}$$

Therefore, the entire area - that is our integral I - will be:

$$I = \sum_{i=0}^{n-1} I_i = \frac{1}{2}\sum_{i=0}^{n-1} h_i \cdot (f(x_{i+1}) + f(x_i)) \tag{6.6}$$

This expression shows the formula of numeric integration through *the trapezoidal rule* and has zero cropping error for functions approximated by polynomials with degree at most equal to 1 (any straight line).

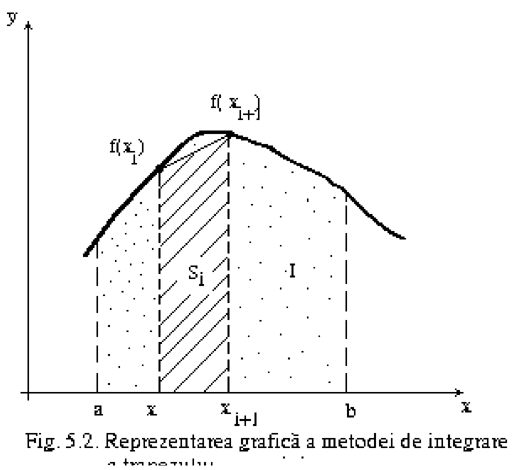

Figure 6.2 - Graphical representation of the trapezoidal rule.

The trapezoidal method is superior in terms of cropping errors compared to the rectangular formula, without major gain regarding the calculation time for the same number of steps of integration.

Its simplicity makes it usable in many cases, its accuracy depends on the chosen number of subintervals. The higher this number is the accuracy is better, but the calculation time increases. This integral represents the fundamental for the Richardson's method.

6.1.2.1. Truncation error for trapezoidal formula

We will calculate the truncation error for $\boldsymbol{I_i}$ (6.5). For this we will expand the function $f(x)$ around the points x_i and x_{i+1}:

$$f(x) = (x_i) + \frac{(x - x_i)}{1!} \cdot f'(x_i) + \frac{(x - x_i)^2}{2!} \cdot f''(x_i) + \dots \tag{6.7}$$

$$f(x) = f(x_{i+1}) + \frac{(x - x_{i+1})}{1!} \cdot f'(x_{i+1}) + \frac{(x - x_{i+1})^2}{2!} f''(x_{i+1}) + \dots \tag{6.8}$$

With these two expansions (6.7) and (6.8) we will build a new function, the average of these functions that best approximates *f(x)* in the range (x_i, x_{i+1}). Considering $x_{i+1} = x_i + h$ we can write the new function like this:

$$f(x)=\frac{f(x_{i+1})+f(x_i)}{2}+(x-x_i)\cdot\frac{f'(x_i)+f'(x_{i+1})}{2}-h\cdot f'(x_{i+1})+$$
$$+(x-x_i)^2\cdot\frac{f''(x_i)+f''(x_{i+1})}{4}-h\cdot(x-x_i)\frac{f''(x_{i+1})}{2}+\frac{h^2}{4}\cdot f''(x_{i+1})+\ldots \tag{6.9}$$

For the integration of the function in (6.9) from x_i to x_{i+1} the following outcome is obtained:

$$\int_{x_i}^{x_{i+1}} f(x)\mathrm{d}x=\frac{f(x_{i+1})+f(x_i)}{2}\cdot h+\frac{f'(x_{i+1})+f'(x_i)}{4}\cdot h^2-\frac{f'(x_{i+1})}{2}\cdot h^2+$$
$$+\frac{f''(x_{i+1})+f''(x_i)}{12}\cdot h^3-\frac{f''(x_{i+1})}{4}\cdot h^3+\frac{f''(x_{i+1})}{4}\cdot h^3+\ldots=$$
$$=\frac{f(x_{i+1})+f(x_i)}{2}\cdot h-\frac{f'(x_{i+1})-f'(x_i)}{4}\cdot h^2-\frac{f''(x_{i+1})+f'(x_i)}{12}\cdot h^3+\ldots \tag{6.10}$$

We see that the *truncation error* is:

$$e_{T_i}=-\frac{f(x_{i+1})-f(x_i)}{4}\cdot h^2-\frac{f''(x_{i+1})+f'(x_i)}{12}\cdot h^3+\ldots \tag{6.11}$$

We consider the truncation error like:

$$e_{T_i}\approx k\cdot h^2\cdot\left[f'(x_{i+1})-f'(x_i)\right] \tag{6.12}$$

where ***k*** can be determined in such a way that the formulas (6.11) and (6.12) being equal. We consider a function for which we have the truncation error through trapezoidal formula. This is $f(x)=x^2$.

Considering $x_{i+1}=x_i+h$

we have:

$$\int_{x_i}^{x_{i+1}} x^2\mathrm{d}x=\frac{x^3}{3}\Big|_{x_i}^{x_{i+1}}=\frac{x_{i+3}^3-x_i^3}{3}=x_i^2\cdot h+h^2\cdot x_i+\frac{h^3}{3} \tag{6.13}$$

By applying the trapezoidal formula of iteration to the same function we have:

$$\int_{x_i}^{x_{i+1}} x^2\mathrm{d}x=\frac{(x_{i+1}^2+x_i^2)}{2}\cdot h+e_{T_i}=x_i^2\cdot h+x_i\cdot h^2+\frac{h^2}{2}+e_{T_i} \tag{6.14}$$

From (6.13) and (6.14) we have: $e_{T_i}=-h^3/6$ (6.15)

By applying (6.12) for $f(x)=x^2$ we have:

$$e_{T_i}=k\cdot h^2\cdot(2x_{i+1}-2x_i)=2\cdot k\cdot h^3 \tag{6.16}$$

From relations (6.16) and (6.15) we have: $k=-\frac{1}{12}$

The truncation error for the small trapezoidal area limited by $(x_i,x_{i+1},f(x_i),f(x_{i+1}))$ is:

$$e_{T_i} \approx -\frac{1}{12} \cdot h^2 \cdot \left[f'(x_{i+1}) - f'(x_i) \right] \tag{6.17}$$

Consequently, for the whole integral on the interval $[a,b]$ we have the following approximate truncation error:

$$e_T \approx -(1/2) \cdot h^2 \cdot \left[f^{'}(b) - f^{'}(a) \right] \tag{6.18}$$

This error represents the approximate sum of the areas between the curve and the spring through the points $(x_i, f(x_i)),(x_{i+1}, f(x_{i+1}))$, $i = 0,1,...,n-1$.

6.1.2.2. Rounding error for the trapezoidal formula

The formula for calculating the numeric integral through the trapezoidal method is given in the expression (6.6). We build a procedural graph for that mathematical relation, considering that, for every node in which we do the operations r_i, $i = 1,2,...,n+3$, $f(x_i)$ has the relative errors ε_i, for $i = 0,1,...,n$ respectively:

$$E_I = (((...(((\varepsilon_1 \cdot \frac{f(x_1)}{f(x_1)+f(x_2)} + \varepsilon_2 \cdot \frac{f(x_2)}{f(x_1)+f(x_2)} + r_1)\frac{f(x_1)+f(x_2)}{f(x_1)+f(x_2)+f(x_3)} +$$

$$+\varepsilon_3 \cdot \frac{f(x_3)}{f(x_1)+f(x_2)+f(x_3)} + r_2)...+r_{n-3})\frac{f(x_1)+f(x_2)+...+f(x_{n-2})}{f(x_1)+f(x_2)+...+f(x_{n-2})+f(x_{n-1})} +$$

$$+\varepsilon_{n-1}\frac{f(x_{n-1})}{f(x_1)+f(x_2)+...+f(x_{n-1})} + r_{n-2} + r_{n-1})\frac{2f(x_1)+2f(x_2)+...+2(x_{n-1})}{f(x_0)+2f(x_1)+...+2f(x_{n-1})+f(x_n)} +$$

$$+(\varepsilon_0\frac{f(x_0)}{f(x_0)+f(x_1)} + \varepsilon_n\frac{f(x_n)}{f(x_0)+f(x_n)} + r_n)\frac{f(x_0)+f(x_n)}{f(x_0)+2f(x_1)+...+2f(x_{n-1})+f(x_n)} +$$

$$+r_{n+1} + \varepsilon_h + r_{n+2} = r_{n+1} + \varepsilon_h + r_{n+2} + r_n\frac{f(x_0)+f(x_n)}{f(x_0)+2f(x_1)+...+2f(x_{n-1})+f(x_n)} +$$

$$+\varepsilon_0\frac{f(x_0)}{f(x_0)+2f(x_1)+...+2f(x_{n-1})+f(x_n)} + \varepsilon_n\frac{f(x_n)}{f(x_0)+2f(x_1)+...+2f(x_{n-1})+f(x_n)} +$$

$$+(r_{n-2}+r_{n-1})\frac{2f(x_1)+2f(x_2)+...+2f(x_{n-1})}{f(x_0)+2f(x_1)+...+2f(x_{n-1})+f(x_n)} + \varepsilon_{n-1}\frac{2f(x_{n-1})}{f(x_0)+2f(x_1)+...+2f(x_{n-1})+f(x_n)} +$$

$$+r_{n-3}\frac{2[f(x_1)+f(x_2)+...+f(x_{n-2})]}{f(x_0)+2f(x_1)+...+2f(x_{n-1})+f(x_n)} + \varepsilon_{n-2}\frac{2f(x_{n-2})}{f(x_0)+2f(x_1)+...+2f(x_{n-1})+f(x_n)} +$$

$$+r_{n-4}\frac{2[f(x_1)+f(x_2)+...+f(x_{n-3})]}{f(x_0)+2f(x_1)+...+2f(x_{n-1})+f(x_n)} + ... + \varepsilon_2\frac{2f(x_2)}{f(x_0)+2f(x_1)+...+2f(x_{n-1})+f(x_n)} + \tag{6.19}$$

Considering that some relative error is $\varepsilon_I = \frac{e_I}{I}$, where e_I is the absolute error, then we can devise the absolute error:

$$e_I = (r_{n+1} + \varepsilon_h + \varepsilon_{n+2}) \cdot I + r_n \cdot \frac{h \cdot [f(x_0) + f(x_n)]}{2} + \varepsilon_0 \cdot \frac{h \cdot f(x_0)}{2} +$$

$$+ \varepsilon_{n-1} \cdot h \cdot f(x_{n-1}) + (r_{n-1} + r_{n-2}) \cdot h \cdot [f(x_1) + f(x_2) + ... + f(x_{n-1})] +$$

$$+ \varepsilon_{n-2} \cdot h \cdot f(x_{n-2}) + r_{n-3} \cdot h \cdot [f(x_1) + f(x_2) + ... + f(x_{n-2})] +$$

$$+ \varepsilon_{n-3} \cdot h \cdot f(x_{n-3}) + r_{n-4} \cdot h \cdot [f(x_1) + f(x_2) + ... + f(x_{n-3})] + ... +$$
$$+ \varepsilon_2 \cdot h \cdot f(x_2) + \varepsilon_1 \cdot h \cdot f(x_1) + r_1 \cdot h \cdot [f(x_1) + f(x_2)] \qquad (6.20)$$

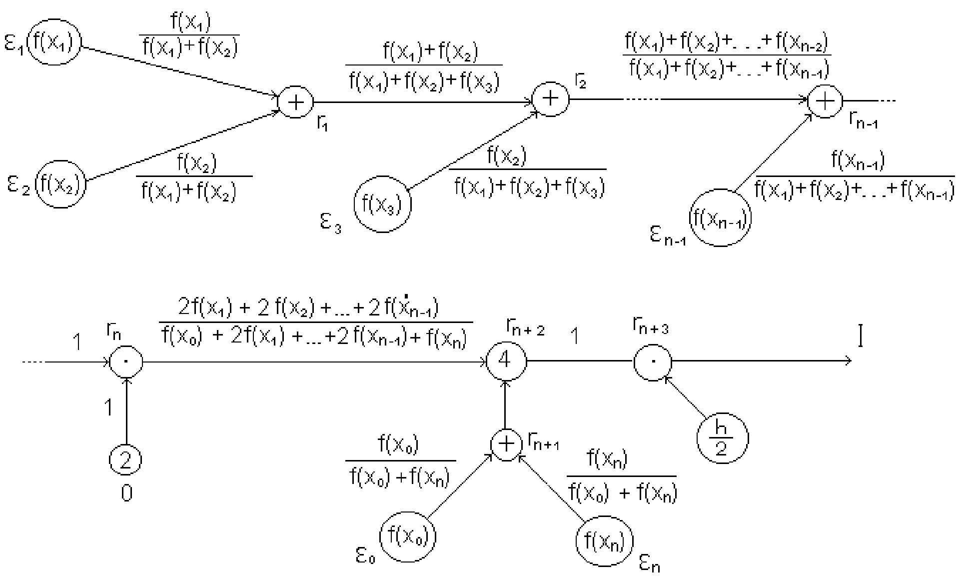

Fig. 6.3 - The procedural graph for trapezoidal rule.

If we consider that all the relative errors that occur in this computation are obtained by rounding-off as well and are lower then $5 \cdot 10^{-t}$, where ***t*** is the number of digits required by mantissa representation on the personal computer, and $|f(x_i)| < \alpha$, for *i = 0, 1, 2,...., n*, then we have the following upper limit of the absolute error:

$$e_I \le 5 \cdot 10^{-t} \cdot [3 \cdot I + h \cdot (\alpha + \frac{\alpha}{2} + \alpha + 2 \cdot (n-1) \cdot \alpha + \alpha + (n-2) \cdot \alpha +$$
$$\alpha + (n-3) \cdot \alpha + \alpha + (n-4) \cdot \alpha + ... + \alpha + 2 \cdot \alpha + \alpha)] =$$

$$= 5 \cdot 10^{-t} \cdot [3nh\alpha + 5 \cdot h \cdot \alpha / 2 + h \cdot \alpha(n^2 + 3 \cdot n - 8)/2] =$$

$$= 5 \cdot 10^{-t} h.\alpha(n^2 + 9 \cdot n - 3)/2$$

$$e_I \le 5 \cdot 10^{-t} \cdot h \cdot \alpha(n^2 + 9 \cdot n - 3)/2 \quad (6.21)$$

It can be seen that the rounding error depends proportionally on the value of *n* (number of divisions on *[a, b]*). Therefore, when *n* increases, the rounding error is increased due to increased number of computations (in fact an increased number of operations).

6.1.2.3. Algorithm 6.2. Trapezoidal method

```
Variables
(  ls: left limit of the integration interval, real;
   ld: right limit of the integration interval, real;
   n: number of subintervals, integer;
   h: the length of each sub-interval, real;
)
{
     sum: value of the integral, real;

       compute  h = (b-a)/n ;
       compute  sum = (f(x0)+f(xn))/2 h ;
       for i=1 to n-1
           compute sum = sum + h*f(ls+i*h);
       print "The value of the integral is:", sum;
}
```

6.1.2.4. Implementation for Algorithm 6.2

```
/* Function that implements the trapezoidal integration
 * method.
 * This function returns the value of the integral.
 */
double TrapezoidalF
(
   double (*f)(double),
   double ls,
   double ld,
   int nrpas)
{
  int i;
  double suma, h;

  h = (ld-ls)/nrpas;
  suma = 0.5*h*(f(ls)+f(ld));
  for(i=1; i<=nrpas-1; i++) suma += h*f(ls+i*h);
  return suma;
}
```

6.1.3. RICHARDSON's METHOD

This method gives a more accurate computation of the numeric integral than the trapezoidal method and was obtained by modifying the *trapezoidal method*. It uses a double partitioning schema for the integration interval. The length of every such small division is also the same among all the divisions. So, it is classified as a method using equally spaced partitioning.

It departs from the truncation error of the trapezoidal method (6.18):

$$e_T = Ch^2$$

for the division:

$$h = (b-a)/n$$

For another division $k = (b-a)/m$ f(b+Fthe following truncation error is obtained

$$e_T=Ck^2 \qquad (6.22)$$

As a result

$$I=I_h+Ch^2$$
$$I=I_k+Ck^2 \qquad (6.23)$$

By subtracting one equation in (6.23) from the other the constant C can be computed as: $C=\frac{I_h\text{-}I_k}{k^2-h^2}$

Replacing C either in (6.22) or (6.23) the formula of the integral I will be:

$$I = I_h + \frac{I_h - I_k}{\left(\frac{k}{h}\right)^2 - 1} \qquad (6.24)$$

expression that bears the name of RICHARDSON's formula. It has greater accuracy than the trapezoidal formula.

6.1.3.1. Algorithm 6.3. RICHARDSON's method

```
{ Variables
   ls:left  limit of the integration interval, real;
   ld: right  limit of the integration interval, real;
  n: number of subintervals,  integer;
  m: number of subintervals,  integer;
  h: value of the length of a sub interval of division n, real;
  k: value of the length of a sub interval of division m, real;
  sumh: value of the  integral of division h, real;
  sumk: value of the  integral of division k, real;
  sum: value of the  integral, real;
)
{  compute h = (b-a)/n;
   compute k = (b-a)/m;
   compute sumh = h*(f(a) + f(b))/2;
   compute sumk = k*(f(a) + f(b))/2;
 for i=1 to n-1
      compute sumh = sumh + h*f(ls+i*h);
 for i=1 to m-1
      compute sumk = sumk + k*f(ls+i*k);
```

```
 compute sum = sumh + (sumh-sumk)/((k/h)*(k/h)-1);
 print sum as the value of the integral;
}
```

6.1.3.2. Implementation of the algorithm 6.3

```
/* Function that implements the RICHARDSON'S integration
 * method.
 * The function returns the value of the integral.
 */
double RichardsonF
(
   double (*f)(double),
   double ls,
   double ld,
   int nrpash,
   int nrpask
)
{
int i;
double sum, sumh, sumk, h, k;

h = (ld-ls)/nrStep_h;
k = (ld-ls)/nrStep_k;
sumh = 0.5*h*(f(ls)+f(ld));
sumk = 0.5*k*(f(ls)+f(ld));

for(i=1; i<=nrpash-1; i++) sumh += h*f(ls+i*h);
for(i=1; i<=nrpask-1; i++) sumk += k*f(ls+i*k);

sum = sumh + (sumh-sumk)/(((k*k)/(h*h)-1));
return sum;
}
```

6.1.4. SIMPSON's METHOD

SIMPSON's method uses the whole process of dividing the integration interval in equal subintervals, but the approximation is done here with the area under a parabola, for two adjacent intervals. Parabola passes through three consecutive points of the division.

The computation formula for SIMPSON's method may be inferred more easily by using the formula of RICHARDSON. This formula is used for two divisions which have the following relations:

$$k = 2h,\ k = \frac{b-a}{m},\ h = \frac{b-a}{n} \qquad (6.25)$$

We write the formula for *trapezoidal method*, for each division:

$$I_h = \frac{h}{2}\left[f(x_0) + 2f(x_1) + 2f(x_2) + 2f(x_3) + \ldots + 2f(x_{n-1}) + f(x_n)\right] \qquad (6.26)$$

$$I_k = h\left[f(x_0)+2f(x_2)+2f(x_4)+2f(x_6)+...+2f(x_{n-2})+f(x_n)\right] \quad (6.27)$$

Then we apply RICHARDSON's formula (6.23):

$$I = h\left[\frac{f(x_0)}{2}+f(x_1)+f(x_2)+f(x_3)+f(x_4)+...+f(x_{n-1})+\frac{f(x_n)}{2}\right]+$$
$$+h\left[\frac{f(x_0)}{6}+\frac{1}{3}f(x_1)+\frac{1}{3}f(x_2)+\frac{1}{3}f(x_3)+\frac{1}{3}f(x_4)+...+\frac{1}{3}f(x_{n-1})+\frac{f(x_n)}{6}\right]+$$
$$+h\left[-\frac{1}{3}f(x_0)-\frac{2}{3}f(x_2)-\frac{2}{3}f(x_4)-...-\frac{2}{3}f(x_{n-2})-\frac{1}{3}f(x_n)\right] \quad (6.28)$$

The expression above represents the numeric calculation formula of the integral according to SIMPSON's method.

6.1.4.1. Algorithm 6.4. SIMPSON's method

```
Variables
(  ls: left limit of the integration interval, real;
   ld: right limit of the integration interval, real;
   n: number of subintervals, integer;
)
{
  h: value of the length of a sub interval, real;
  sum: value of the  integral, real;
  compute h = (ld - ls)/n ;
  compute h*(f(ls)+f(ld))/3.0;
  for i=1 to n-1
        if 'i is even' then
          compute sum =sum +(2./3)* h* f(ls+i*h);
        else compute sum=sum+(4./3)*h* f(ls+i*h);
  print sum as the value of the integral;
 }
```

6.1.4.2. Implementation of the algorithm 6.4

```
/* Function that implements the integration method of
 * RICHARDSON.
 * Function that returns the value of the integral.
 */
double SimpsonF
(
    double (*f)(double),
    double ls,
    double ld,
    int nrpas
)
{
  int i;
```

```
    double suma,h;
    h = (ld-ls)/nrpas;
    suma = h*(f(ls)+f(ld))/3.0;
    for(i=0; i<=nrpas-1; i++)
              suma += 2*(1+i%2)*h*f(ls+i*h)/3.0;
    return suma;
}
```

6.2. INTEGRATION OF SINGLE-VARIABLE FUNCTIONS USING VARIABLE-PARTITIONING SCHEMA

Among these methods we will talk about the GAUSS' quadrature formula. This method determines the points of dividing of the interval of integration so that the calculation error of the integral gives minimal errors.

6.2.1. GAUSS' QUADRATURE FORMULA

This method reduces any range of integration *[a, b]* at the range *[-1. 1]* with the substitution formula:

$$y = \frac{2x - (b + a)}{b - a} \tag{6.29}$$

This substitution is necessary because of the fact that this method uses the LEGENDRE's special polynomials.

Among these polynomials' characteristics we are interested in the following:

- they are defined on [-1,1]
- on [-1,1] they have *n* unique real roots, that we are going to consider the points of division for the Ox-axis.

For $x = a$ we have $y = -1$ and for $x = b$ we have $y = 1$. Substitution is given by the formula:

$$x = \frac{1}{2}(b - a)y + \frac{1}{2}(b + a) \tag{6.30}$$

and

$$\mathrm{d}x = \frac{1}{2}(b - a)\mathrm{d}y \tag{6.31}$$

In the end, integral $I = \int_a^b f(x)\mathrm{d}x$ will become a new integral, in variable *y*, as follows:

$$I = \int_{-1}^{+1} f\left[\frac{1}{2}(b - a)y + \frac{1}{2}(b + a)\right]\frac{1}{2}(b - a)\mathrm{d}y = \int_{-1}^{+1} \varphi(y)\mathrm{d}y \tag{6.32}$$

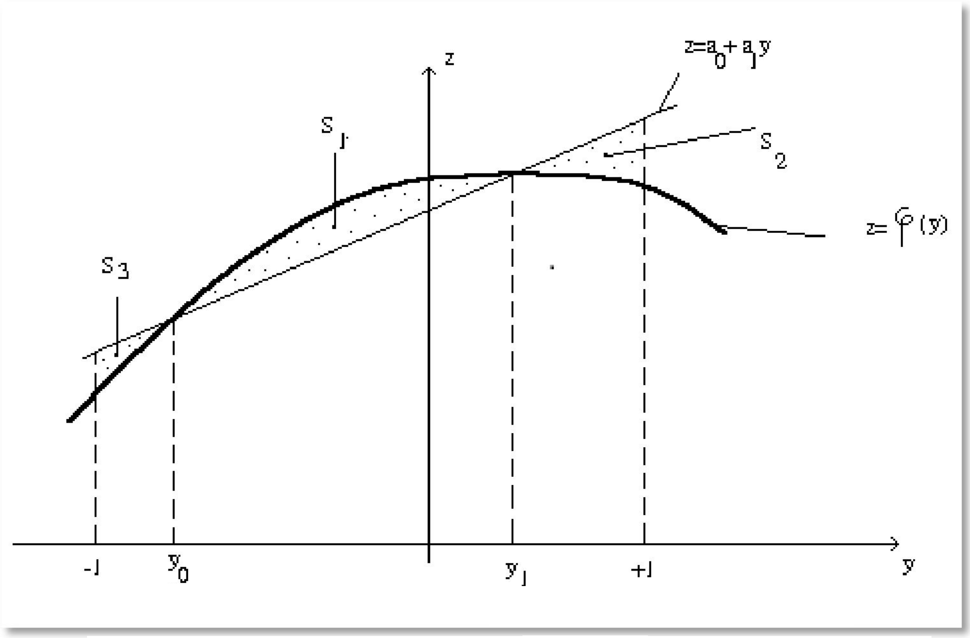

Fig.6.4 - Graphical draw for both $z = \varphi(y)$ *and division points.*

to give zero error by - at most - a 3rd degree polynomial. The choice of the dividing points within [-1.1] is done in such a way so that between the areas S1, S2, S3 the following relationship holds:

$$S1 = S2 + S3 \tag{6.33}$$

The formula for calculating the integral is proven for the case of two points in the integration interval, points that are determined in such a way for the integral $I = \int_{-1}^{+1} \varphi(y)\mathrm{d}y$ to give zero error by a polynomial cubic inclusive. Choosing points of dividing the range [-1.1] is done in such a way that between areas S1, S2, S3 to have the relationship

$$S_1=S_2+S_3 \tag{6.33}$$

until a function $\varphi(y)$ of third degree.

The integral through the quadrature method is computed using the following formula: $I = k_0\varphi(y_0) + k_1\varphi(y_1)$

where y_0 şi y_1 are points for dividing the interval, and k_0, k_1 are some weights (constant values).

We will now compute the exact value of the integral $I = \int_{-1}^{1} \varphi(y)\mathrm{d}y$ taking into account formula (6.33):

$$I=\int_{-1}^{+1} \varphi(y)\mathrm{d}y=\int_{-1}^{+1} (a_0 + a_1 y)\mathrm{d}y \tag{6.34}$$

where $z = a_0 + a_1 \cdot y$ represents the equation of a straight-line which passes through the points $(y_0, \varphi(y_0))$ and $(y_1, \varphi(y_1))$.

We consider the function $\varphi(y)$ of degree at most three, for which the integral is computed with no error:

$$\varphi(y) = b_0 + b_1 y + b_2 y^2 + b_3 y^3 \tag{6.35}$$

This function can be written as follows:

$$\varphi(y) = a_0 + a_1 y + (y - y_0)(y - y_1)(\alpha_0 + \alpha_1 y) \tag{6.36}$$

underlining the passing of the curve $\varphi(u)$ through the points $(y_0, \varphi(y_0))$ şi $(y_1, \varphi(y1))$ of the straight-line: $z = a_0 + a_1 y$.

The equality (6.34) is written as follows:

$$\int_{-1}^{+1} (a_0 + a_1 y)\mathrm{d}y = \int_{-1}^{+1} \left[a_0 + a_1 y + (y - y_0)(y - y_1)(\alpha_0 + \alpha_1 y)\right]\mathrm{d}y \tag{6.37}$$

and, for this equality to be satisfied for any α_0 and α_1, the following must happen

$$\int_{-1}^{+1} (y - y_0)(y - y_1)\mathrm{d}y = 0 \tag{6.38}$$

and

$$\int_{-1}^{+1} y(y - y_0)(y - y_1)\mathrm{d}y = 0 \tag{6.39}$$

From these two equations we have the system:

$$\begin{cases} \int_{-1}^{+1} \left[y^2 - (y_0 + y_1)y + y_0 y_1\right]\mathrm{d}y = 0 \\ \int_{-1}^{+1} \left[y^3 - (y_0 + y_1)y^2 + y_0 y_1 y\right]\mathrm{d}y = 0 \end{cases} \tag{6.40}$$

Or:

$$\begin{cases} y_0 y_1 + \dfrac{1}{3} = 0 \\ y_0 + y_1 = 0 \end{cases} \tag{6.41}$$

with the solutions $\mathrm{y}_0 = -\dfrac{1}{\sqrt{3}}$ and $\mathrm{y}_1 = \dfrac{1}{\sqrt{3}}$ (6.42)

To calculate k_0, k_1 we use the equality:

$$\mathrm{I} = \int_{-1}^{+1} \varphi(y)\mathrm{d}y = \int_{-1}^{+1} (a_0 + a_1 y)\mathrm{d}y = k_0 \varphi(-\frac{1}{\sqrt{3}}) + k_1 \varphi(\frac{1}{\sqrt{3}}) \tag{6.43}$$

$$\int_{-1}^{+1} (a_0 + a_1 y)\mathrm{d}y = (a_0 y + \frac{a_1}{2} y^2)\Big|_{-1}^{+1} = 2a_0 \tag{6.44}$$

Replacing (6.44) in (6.43) we have:

$$k_0(a_0 - \frac{1}{\sqrt{3}}a_1) + k_1(a_0 + \frac{1}{\sqrt{3}}a_1) = 2a_0 \quad (6.45)$$

Using identification in the both terms of the assignment we have the system:

$$\begin{cases} k_0 + k_1 = 2 \\ k_0 - k_1 = 0 \end{cases} \quad (6.46)$$

with the solutions: $k_0 = 1$, $k_1 = 0$ (6.47)

The formula used to compute the integral by the Gaussian quadrature method is:

$$I = \int_a^b f(x)\mathrm{d}x = \int_{-1}^{+1}(y)dy = \varphi(-\frac{1}{\sqrt{3}}) + \varphi(\frac{1}{\sqrt{3}}) \quad (6.48)$$

where: $$\varphi(y) = \frac{1}{2}(b-a)f(y) \quad (6.49)$$

6.2.1.1. The truncation error of the GAUSS' quadrature formula using two points

The integral from polynomials with degree at most three gives no truncation errors, since we used LEGENDRE's polynomials of degree 3. For such a case, assuming the given functions are polynomials with a degree higher than three, we have some truncation error, as follows:

$$e_T = k\psi^{(\mathrm{IV})}(\xi), \text{ -1<ξ< 1} \quad (6.50)$$

To determine k we consider $\varphi(y) = y^4$

$$\int_{-1}^{+1}\varphi(y)\mathrm{d}y = \int_{-1}^{+1} y^4\mathrm{d}y = \frac{y^5}{5}\Big|_{-1}^{+1} = \frac{2}{5} \quad (6.51)$$

$$\int_{-1}^{+1}\varphi(y)\mathrm{d}y = \int_{-1}^{+1} y^4\mathrm{d}y = \varphi(-\frac{1}{\sqrt{3}}) + \varphi(-\frac{1}{\sqrt{3}}) + e_T = \frac{2}{9} + e_T \quad (6.52)$$

From the equalities (6.51) and (6.52) we have:

$$e_T = \frac{8}{45}$$

By applying the formula (6.55) for $\varphi(y) = y^4$ where $\varphi^{(IV)}(y) = 24$ we have:

$$\frac{8}{45} = k \cdot 24 \qquad \text{that is: } k = \frac{1}{135}$$

The truncation error for the formula of the Gaussian quadrature method when using two division points is:

$$e_T = \frac{1}{135}\varphi^{(\mathrm{IV})}(\xi), \quad \text{-1<ξ<1} \quad (6.53)$$

6.2.2. GAUSS' quadrature formula using more than two division points

In this case

$$\int_{a}^{b} f(x)\mathrm{d}x = \int_{-1}^{+1} \varphi(u)\mathrm{d}u = \sum_{i=0}^{n-1} k_i \varphi(y_i) \tag{6.54}$$

by using ***n*** points of division and ***n*** weights.

The values of the points of division in the interval [-1,1] are the roots of the LEGENDRE's polynomials which are defined through the relationship of recurrence:

$$P_0(y)=1,\ P_1(y)=y$$

$$P_n(y)=\frac{1}{n}\left[(2n-1)yP_{n-1}(y)-(n-1)P_{n-2}(y)\right] \tag{6.55}$$

and the weights are given by the formula:

$$k_i = \frac{2}{(1-y_i^2)\left[P_n'(y_i)\right]^2} \tag{656}$$

For LEGENDRE's polynomials with a degree up to 16, the roots and weights are tabulated along with the implementation of this method.

In the case of *n* division points of the integration interval [-1, 1] the truncation error is zero for all polynomials with grade lower than *2n-1*, including *2n-1*.

6.2.2.1. The truncation error of the Gauss' quadrature using more than two points

We consider a polynomial with degree of 2n for which the Gaussian integral has the truncation error:

$$e_T = k\varphi^{(2n)}(\xi), \quad -1<\xi< 1 \tag{6.57}$$

We calculate the integral for $\varphi(y) = y^{2n}$

$$\int_{-1}^{+1} \varphi(y)\mathrm{d}y = \int_{-1}^{+1} y^{2n}\mathrm{d}y = \left.\frac{y^{2n+1}}{(2n+1)}\right|_{-1}^{+1} = \frac{2}{2n+1} \tag{6.58}$$

$$\int_{-1}^{+1} \varphi(y)\mathrm{d}y = \int_{-1}^{+1} y^{2n}\mathrm{d}y = \sum_{i=0}^{n-1} k_i {y_i}^{2n} + e_T \tag{6.59}$$

From the equalities (6.58) and (6.59) we have:

$$e_T = \frac{2}{2n+1} - \sum_{i=0}^{n-1} k_i y_i^{2n} \tag{6.60}$$

By applying the formula (6.57), where $\varphi^{(2n)}(\xi)=(2n)!$ and equal to (6.60) we have:

$$k = \frac{1}{(2n)!}\left(\frac{2}{2n+1} - \sum_{i=0}^{n-1} k_i u_i^{2n}\right) \quad (6.61)$$

and the truncation error:

$$e_T = \frac{\varphi^{(2n)}(\xi)}{(2n)!}\left(\frac{2}{2n+1} - \sum_{i=0}^{n-1} k_i u_i^{2n}\right) \quad (6.62)$$

6.2.2.2. Algorithm 6.5. The Gauss' quadrature formula

```
Variables
(
        ls: left limit of the integration interval, real;
        ld: right limit of the integration interval, real;
        A[n,i]: matrix of weights, real;
        U[n,i]: matrix of the solutions of Legendre polynomials,
        real;
        n: desired degree of Legendre polynomials;
)
{
sum: value of the integral, real;
Build the A matrix;
Build the U matrix;

sum=0;
for i=1 to n compute
      sum = sum +    A[n,i]*(ld-ls)/2*
                     f((ld-ls)/2*Y[n,i]+(ls+ld)/2);
print sum as the integral;
}
```

6.2.2.3. Implementation of the algorithm 6.5

```
/* Function that implements the integration method
 * using Gaussian quadrature.
 * Function returns the value of the integral.
 */
double GaussianQuadrature
(
    double (*f)(double),
    double ls,
    double ld,
    int ord_pol
)
{
    static double A[17][17]=
    {
    /* n=0 */
    {  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
    /* n=1 */
    {  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
    /* n=2 */
    {  1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
```

```
    /* n=3 */
    {5.0/9.0,8.0/9.0,5.0/9.0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},

/* n=4 */
     {0.347854854137454          0.652145145862564
      0.652145145862564          0.347854854137454
     0                           0
     0                           0
     0                           0
     0                           0
     0                           0
     0                           0}
/* n=5 */
     {0.236926885056189,         0.478628670499366
      0.568888888888889,         0.47862867049936,
      0.236926885056189          0,
     0,                          0,
     0,                          0,
     0,                          0,
     0,                          0,
     0,                          0}
/* n=6 */
     { 0.17132449237917,         0.360761573048139,
     0.467913934572691,          0.467913934572691,
     0.360761573048139,          0.17132449237917,
     0,                          0,
     0,                          0,
     0,                          0,
     0.                          0,
     0,                          0}
/* n=7 */
     {0.12948496616887,          0.279705391489277,
     0.381830050505119,          0.417959183673469,
     0.381830050505119,          0.279705391489277,
     0.12948496616887,           0,
     0,                          0,
     0,                          0,
     0,                          0,
     0,                          0}
/* n=8 */
     {0.101228536290376,         0.222381034453374,
     0.313706645877887,          0.362684783378362,
     0.362684783378362,          0.313706645877887,
     0.222381034453374,          0.101228536290376,
     0,                          0,
     0,                          0,
     0,                          0,
     0,                          0},
/* n=9 */
         {   0.08127438361574,   0.180648160694857
     0.260610696402935           0.312347077040003,
         0.33023935500126        0.312347077040003,
     0.260610696402935,          0.180648160694857,
     0.08127438361574,           0,
     0,                          0,
     0,                          0,
```

```
        0,                          0},
/* n=10 */
        {0.066671344308688,         0.14945134915058
        0.219086362515982,          0.269266719309996,
        0.295524224714753,          0.295524224714753,
        0.269266719309996,          0.219086362515982,
        0.14945134915058,           0.066671344308688,
        0,                          0,
        0,                          0,
        0,                          0},
/* n=11 */
        {0.055668567116,            0.125580369465,
        0.186290210928,             0.233193764592,
        0.26280454451,              0.272925086778,
        0.26280454451,              0.233193764592,
        0.186290210928,             0.125580369465,
        0.055668567116,             0,
        0,                          0,
        0,                          0},
/* n=12 */
        {0.047175336387,            0.106939325995,
        0.160078328542,             0.203167426723,
        0.233492536538,             0.249147045813,
        0.249147045813,             0.233492536538,
        0.203167426723,             0.160078328542,
        0.106939325995,             0.047175336387,
        0,                          0,
        0,                          0},
/* n=13 */
        {0.040484004765,            0.092121499838,
        0.13887351022,              0.178145980762,
        0.207816047537,             0.226283180263
        0.232551553231,             0.226283180263,
        0.207816047537              0.178145980762,
        0.13887351022,              0.092121499838
        0.040484004765,             0,
        0,                          0},
        0,
/* n=14 */
        {0.035119460332,            0.08015808716,
        0.121518570688,             0.157203167158,
        0.185538397478,             0.205198463721,
        0.215263853463,             0.215263853463,
        0.205198463721,             0.185538397478,
        0.157203167158,             0.121518570688,
        0.08015808716,              0.035119460332,
        0,                          0},
/* n=15 */
        {0.030753241996,            0.070366047488,
        0.107159220467,             0.139570677926,
        0.166269205817,             0.186161000016,
        0.198431485327,             0.202578241926,
        0.198431485327,             0.186161000016,
        0.166269205817,             0.139570677926,
        0.107159220467,             0.070366047488,
        0.030753241996,             0},
```

```
/* n=16 */
     {0.027152459412,               0.062253523939,
     0.095158511682,                0.124628971256,
     0.149595988817,                0.169156519395,
     0.182603415045,                0.189450610455,
     0.189450610455,                0.182603415045,
     0.169156519395,                0.149595988817,
     0.124628971256,                0.095158511682,
     0.062253523939,                0.027152459412};

   static double U[17][17]=
   {
/* n=0 */
   {  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
/* n=1 */
   {  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
/* n=2 */
     {0.5773502692,                -0.5773502692,
     0,                            0,
     0,                            0,
     0,                            0,
     0,                            0,
     0,                            0,
     0,                            0,
     0,                            0},
/* n=3 */
     {0.7745966692,                0,
     -0.7745966692                 0,
     0,                            0,
     0,                            0,
     0,                            0,
     0,                            0,
     0,                            0,
     0,                            0},
/* n=4 */
     {0.861136311594053,           0.339981043584856,
     -0.339981043584856,           -0.861136311594053,
     0,                            0,
     0,                            0,
     0,                            0,
     0,                            0,
     0,                            0,
     0,                            0}
/* n=5 */
     {0.906179845938664,           0.538469310105683,
     0,                            -0.538469310105683,
     -0.906179845938664,           0,
     0,                            0,
     0,                            0,
     0,                            0,
     0,                            0,
     0,                            0},
/* n=6 */
     {0.932469514203152,           0.661209386466265,
     0.238619186083197,            -0.238619186083197,
     -0.661209386466265,           -0.932469514203152,
```

```
        0,                          0,
        0,                          0,
        0,                          0,
        0,                          0,
        0,                          0},
/* n=7 */
        {0.949107912342759,         0.741531185599394,
        0.405845151377397,          0,
        -0.405845151377397,         -0.741531185599394,
        -0.949107912342759,          0,
         0,                          0,
         0,                          0,
         0,                          0,
         0,                          0},
/* n=8 */
        {0.96269856497336,           0.796666477413627,
         0.525532409916329,          0.18343464249565,
        -0.18343464249565,          -0.525532409916329,
        -0.796666477413627,         -0.96269856497336,
        0,                          0,
        0,                          0,
        0,                          0,
        0,                          0},
/* n=9 */
        {0.968160239507626,          0.836031107326636,
        0.613371432700591,           0.324253423403809,
        0,                          -0.324253423403809,
        -0.613371432700591,         -0.836031107326636,
        -0.968160239507626,         0,
        0,                          0,
        0,                          0,
        0,                          0},
/* n=10 */
        {0.973906528517172,          0.865063366688985,
        0.679409568299024,           0.433395394129247,
        0.148874338981631,          -0.148874338981631,
        -0.433395394129247,         -0.679409568299024,
        -0.865063366688985,         -0.973906528517172,
        0,                          0,
        0,                          0,
        0,                          0},
/* n=11 */
        {0.978228658146,            0.887062599768,
         0.730152005574,            0.519096129207,
         0.269543155952,            0,
        -0.269543155952,            -0.519096129207,
        -0.730152005574,            -0.887062599768,
        -0.978228658146,            0,
        0,                          0,
        0,                          0},
/* n=12 */
        {0.981560634247,           .1725637,
         0.769902671494,           17954287,
         0.367831498998,           33408511,
        -0.125233408511,           '831498998,
        -0.587317954287,           902671494,
        -0.90411725637,            .560634247,
```

```
        0,
        0,
  /* n=13 */
       {0.984183054719,              0.917598399223,
        0.801578090733,              0.64234933944,
        0.448492751036,              0.230458315955,
        0,                          -0.230458315955,
       -0.448492751036,             -0.64234933944,
       -0.801578090733,             -0.917598399223,
       -0.984183054719,              0,
       0,                            0},
  /* n=14 */
       {0.986283808697,              0.928434883664,
        0.82720131507,               0.687292904812,
        0.515248636358,              0.319112368928,
        0.108054948707,             -0.108054948707,
       -0.319112368928,             -0.515248636358,
       -0.687292904812,             -0.82720131507,
       -0.928434883664,             -0.986283808697,
        0,                           0},
  /* n=15 */
       {0.98799251802,               0.937273392401,
        0.84820658341,               0.72441773136,
        0.570972172609,              0.394151347078,
        0.201194093997,              0,
       -0.201194093997,             -0.394151347078,
       -0.570972172609,             -0.72441773136,
       -0.84820658341,              -0.937273392401,
       -0.98799251802,               0},
  /* n=16 */
       {0.989400934992,              0.944575023075,
        0.865631202388,              0.755404408355,
        0.617876244403,              0.458016777657,
        0.281603550779,              0.095012509838,
       -0.095012509838,             -0.281603550779,
       -0.458016777657,             -0.617876244403,
       -0.755404408355,             -0.865631202388,
       -0.944575023075,             -0.989400934992}
                                     };
     int i;
     double sum=0;
     for(i=1; i<=ord_pol; i++)
        sum += 0.5*(ld-ls)*A[ord_pol][i-1]*f(0.5*(ld-ls)
               *U[ord_pol][i-1]+0.5*(ls+ld));
     return sum;}
```

6.3. COMPARISON AMONG ONE-DIMENSIONAL INTEGRATION METHODS

Out of all the methods for one dimensional integration, *Gaussian quadrature method* is the most accurate, having the same precision as the method of Simpson using twice as much division points than it usually requires, or the same as the trapezoidal rule using four times the number of division points usually needed.

For the same precision for computing the integral, either the efficiency increases or the required computational time decreases, depending on the method we choose. Thus, in ascending order from the computational error point of view we have: the trapezoidal method, the RICHARDSON's method, the SIMPSON's method and eventually the GAUSSIAN quadrature method.

6.4. NUMERICAL COMPUTATION OF IMPROPER INTEGRALS

Definition 6.1: An *improper* integral is the integral for which:

1. At least one of the limits of integration is infinite but the function is continuous on the integration interval, or
2. The function has points of discontinuity of either first or second kind within the integration domain but the integration limits are finite.

Improper integrals having either one of the following form:

$$\int_{-\infty}^{+\infty} f(x)dx\,,\ \int_{a}^{+\infty} f(x)dx\,,\ \int_{-\infty}^{a} f(x)dx \tag{6.63}$$

can be reduced to:

$$\int_{a}^{+\infty} f(x)dx \tag{6.64}$$

As such, we will study the improper integral in (6.64).

If the function to be integrated - defined in the interval $[a,\infty]$ - can be integrated on the whole interval and the following limit exists:

$$\lim_{A\to\infty}\int_{a}^{A} f(x)dx = k \tag{6.65}$$

then:

$$\int_{a}^{+\infty} f(x)dx = k \tag{6.66}$$

In this case, such improper integrals are *convergent*. When the limit does not exist or is infinite, the improper integral is *divergent*. The value of A may be large, since:

$$\left|\int_{A}^{\infty} f(x)dx\right| < \varepsilon \tag{6.67}$$

where ε is a positive constant, chosen arbitrarily small.

In this case the improper integral:

$$\int_{a}^{\infty} f(x)dx \cong \int_{a}^{A} f(x)dx \tag{6.68}$$

can be computed using either one of the methods presented in section 6.1.

Functions with a discontinuity point of the first kind within the integration interval $[a,b]$, be this point $c \in [a,b]$, have the following mathematical property:

$$f(c-0)=\lim_{\substack{x\to c\\ x<c}} f(x) \qquad \text{and} \qquad f(c+0)=\lim_{\substack{x\to c\\ x>c}} f(x) \tag{6.69}$$

and $f(c)\neq f(c-0)$ or $f(c)=f(c-0)$ and $f(c)\neq f(c+0)$ or $f(c)=f(c+0)$.

In such situations we can say that:

$$\int_a^b f(x)dx=\int_a^c f_1(x)dx+\int_c^b f_2(x)dx \tag{6.70}$$

where:

$$f_1(x)=\begin{cases} f(x), \text{ for } a<x<c \\ f(c-0), \text{ for } x=c \end{cases}$$

$$f_2(x)=\begin{cases} f(x), \text{ for } c<x<b \\ f(x+c), \text{ for } x=c \end{cases}$$

If the integral (6.70) exists, we can say that the improper integral is *convergent* and therefore its value can be calculated with either method devised in section 6.1.

The integrand $f(x)$ has a point of discontinuity of second kind, $c\in[a,b]$, if at least one of the integration limits has infinite value. In this case:

$$\int_a^b f(x)dx=\int_a^{c-e} f(x)dx+\int_{c+e}^b f(x)dx \tag{6.71}$$

where ε can be taken arbitrarily small so that the following condition is fulfilled:

$$\left|\int_{c-\varepsilon}^{c+\varepsilon} f(x)dx\right|<\varepsilon_1, \varepsilon_1>0$$

The integral in (6.71) can be computed with one of the methods studied in section 6.1: rectangular method, the trapezoidal method, RICHARDSON'S method, the SIMPSON 's method or the GAUSS's quadrature formula.

6.5. NUMERICAL COMPUTATION OF TWO DIMENSIONAL INTEGRALS

For simplicity we consider the domain of integration of the function of two variables a rectangle (figure 6.5)

$$\iint_D f(x,y)dx=\int_a^b\int_c^d f(x,y)dxdy \tag{6.72}$$

represents the *double integral* for functions of two variables $f(x,y)$.

To calculate the value of such integrals we will use the ***trapezoidal***

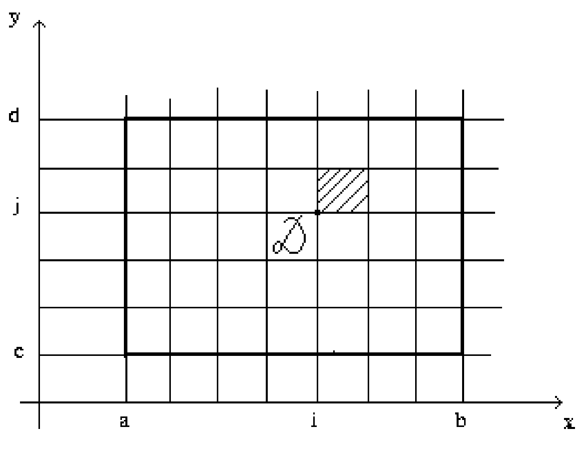

Figure 6.5 - Graphical representation of the two dimensional integration domain, here a rectangle.

cubature and ***Simpson's cubature formulae***, presented in the following.

6.5.1. Trapezoidal cubature method

The intervals $[a,b]$ and $[c,d]$ can be split into equal width subintervals of lengths:

$$h = \frac{b-a}{n} \quad , \qquad k = \frac{d-c}{m} \tag{6.73}$$

And we have the rectangle with the vertices (peaks) $[x_i, y_i], [x_{i+1}, y_i], [x_{i+1}, y_{i+1}]$, $[x_i, y_{i+1}]$, where $x_i = a + i \cdot h$, $y_i = c + j \cdot k$.

For the rectangle that contains the vertices $[x_i, y_i]$ the integral I_{ij} is calculated by applying the trapezoidal formula:

$$I_{ij} = \int_{x_i}^{x_{i+1}} dx \int_{y_j}^{y_{j+1}} f(x,y)dxdy \approx \int_{x_i}^{x_{i+1}} \left\{ \frac{k}{2}\left[f(x,y_j) + f(x,y_{j+1}) \right] \right\} dx =$$

$$= \frac{k}{2}\left[\int_{x_i}^{x_{i+1}} f(x,y_j)dx + \int_{x_i}^{x_{i+1}} f(x,y_{j+1})dx \right] =$$

$$= \frac{k \cdot h}{4}\left[f(x_i,y_j) + f(x_i,y_{j+1}) + f(x_{i+1},y_j) + f(x_{i+1},y_{j+1}) \right] \tag{6.74}$$

Integral on the entire domain $[a,b,c,d]$ is:

$$I \cong \sum_{i=0}^{n-1}\sum_{j=0}^{m-1} I_{ij} = \frac{kh}{4}\sum_{i=0}^{n-1}\sum_{j=0}^{m-1}\left[f(x_i,y_j) + f(x_i,y_{j+1}) + f(x_{i+1},y_j) + f(x_{i+1},y_{j+1}) \right] \tag{6.75}$$

(6.75) is known as the *cubature trapezoidal rule.*

6.5.1.1. Algorithm 6.6. The trapezoidal cubature formula

```
Variables
(
   a: left limit of the integration interval on Ox, real;
   b: right limit of the integration interval on Ox, real;
   c: left limit of the integration interval on Oy, real;
   d: right limit of the integration interval on Oy, real;
   n: number of subintervals on ax Ox, integer;
   m: number of subintervals on ax Oy, integer;
)
{
   h: value of the length of a subinterval with n division, real;
   k: value of the length of a subinterval with m division, real;
   sum: value of the  integral, real;
   {
     compute h = (b-a)/n ; compute k = (d-c)/m ;
```

```
    sum=0;
    for i=1 to n-1
    for j=1 to m-1

    compute
    sum = sum + ((h*k)/4)*(f(a+i*h,c+j*k)+f(a+i*h,c+(j+1)*k) +
            f(a+(i+1)h,c+j*k)+f(a+(i+1)h,(j+1)*k))
    print sum as the value of the integral;
}
```

6.5.1.2. Implementation of Algorithm 6.6

```
/* Function that implements Simpson method of cubature */
 * Function returns the value of a function's integral of
 * two variables function.
 */
double TrapezoidalCubature
(
   double (*f)(double,double),
   double sx,
   double dx,
   double sy,
   double dy,
   int nx,
   int ny
)
{  double sum=0,h,k;
   int i,j;
   h=(dx-sx)/nx;
   k=(dy-sy)/ny;
   for(i=0; i<=nx-1; i++)
      for(j=0; j<=ny-1; j++)
         sum +=
         0.25*h*k*( f(sx+i*h,sy+j*k)+f(sx+i*h,sy+(j+1)*k)
         + f(sx+(i+1)*h,sy+j*k)+f(sx+(i+1)*h,sy+(j+1)*k));
   return sum;
}
```

6.5.2. SIMPSON'S CUBATURE FORMULA

For the same rectangle $[a,b,c,d]$ depicted in fig. 6.5 we will now apply the SIMPSON's cubature formula.

We will consider the rectangle of integration with the vertices (peaks) $[x_i, y_i]$, $[x_{i+1}, y_i], [x_{i+1}, y_{i+1}], [x_i, y_{i+1}]$ and a central point, $[x_i, y_i]$.

Then the fundamental integral I_{ij} is:

$$I_{ij} = \frac{kh}{9}\{f(x_{i-1},y_{j-1}) + f(x_{i+1},y_{j-1}) + f(x_{i-1},y_{j+1}) + f(x_{i+1},y_{j+1}) + $$
$$+4[f(x_i,y_{j+1}) + f(x_i,y_{j-1}) + f(x_{i-1},y_j) + f(x_{i+1},y_j)]\} + 16f(x_i,y_i) \quad (6.76)$$

Integral on the entire rectangle $[a,b,c,d]$ is given by the formula called the cubature of Simpson***:***

$$I=\sum_{i=0}^{n-1}\sum_{j=0}^{m-1}I_{ij}=\frac{kh}{9}\sum_{i=0}^{n-1}\sum_{j=0}^{m-1}\Big\{f(x_{i-1},y_{j-1})+f(x_{i+1},y_{j-1})+f(x_{i-1},y_{j+1})+f(x_{i+1},y_{j+1})+$$

$$+4\big[f(x_i,y_{j+1})+f(x_i,y_{j-1})+f(x_{i-1},y_j)+f(x_{i+1},y_j)\big]+16f(x_i,y_j)\Big\} \quad (6.77)$$

6.5.2.1. Algorithm 6.7. SIMPSON's cubature method

```
Variables
(
  a: left limit of the integration interval on Ox, real;
  b: right limit of the integration interval on Ox, real;
  c: left limit of the integration interval on Oy, real;
  d: right limit of the integration interval on yx, real;
  n: number of subintervals on ax Ox, integer;
  m: number of subintervals on ax Oy, integer;
  h: value of the length of a subinterval with n division, real
)
{
k: value of the length of a subinterval with m division, real;
sum: value of the  integral, real;
  compute h=(b-a)/n;
  compute k=(d-c)/m;
  sum=0; i=1;
  repeat
   j=1;
   repeat
    sum = sum + (
       (h*k)/9)*(f(a+(i-1)*h, c+(j-1)*k)
       + f(a+(i+1)*h, c+(j-1)*k)+f(a+(i-1)*h, c+(j+1)*k)+
     4*(f(a+i*h,(j+1)*k))+f(a+i*h, c+(j-1)*k)+
         f(a+(i-1)*h,c+j*k)+f(a+(i+1)*h,c+j*k
         )
      + 16*f(a+i*h, c+j*k))
      j=j+2;
     until j>m-1;
    i=i+2;
    until i>n-1;
 print the value of sum as the computed value of the integral;
}
```

6.5.2.2. Implementation of the Algorithm 6.7

```
/* Function that implements cubature Simpson's method.
 * It returns the value of the 2-dimensional integral.
 */
```

```
double SimpsonCubature
(
    double (*f)(double,double),
    double sx,
    double dx,
    double sy,
    double dy,
    int nx,
    int ny
)
{
  double suma=0,h,k;
  int i,j;

  h=(dx-sx)/nx;
  k=(dy-sy)/ny;

  for(i=1;i<=nx-1;i+=2)
  {
    for(j=1;j<=ny-1;j+=2)
    suma +=
      h*k*(f(sx+(i-1)*h,sy+(j-1)*k)+f(sx+(i+1)*h,sy+(j-1)*k)+
       f(sx+(i-1)*h,sy+(j+1)*k)+f(sx+(i+1)*h,sy+(j+1)*k)+
       4*f(sx+i*h,sy+(j+1)*k)+4*f(sx+i*h,sy+(j-1)*k)+
       4*f(sx+(i-1)*h,sy+j*k)+4*f(sx+(i+1)*h,sy+j*k)+
       16*f(sx+i*h,sy+j*k)
       )/9;
  }
  return suma;
}
```

6.6. EXERCISES

1. We have the function $f(x) = \dfrac{x^3}{1+\cos(1+x)}.e^{x^2}.(1+\sin x^2)$

   for which we want the integral from 0 to 3.
   R: The value of the integral is given in the table 6.1.

Table 6.1

Method	*Number of subintervals*	*Value of the integral*
Rectangular rule	10000	1394.642843
Trapezoidal rule	10000	1395.704297
Richardson	5000 and 10000	1395.703984
Simpson	10000	1395.703984
Gauss quadrature	Legendre polynomial of degree n=15	1395.7031

2. We have the following function of two variables:

$$f(x,y) = \frac{x^2 + y^2}{1 + 2.x.y}.\exp(1 + x).\sin(x + y + 2)$$

The value of the integral is required from the given function on the domain $x \in [0,2]$; $y \in [0,2]$.

Value of the integral is given in the table 6.2.

Table 6.2

Method	***Number of points on Ox***	***Number of points on Oy***	***Value of the integral***
Cubature trapezoidal rule	100	100	-24.730047
Simpson's cubature	100	100	-24.733155