Computer Oriented Programs on Multi-Dimensional Numerical Integration

D.A.GISMALLA & Dr. F. M. DAWALBAIT

***Abstract*— Computer Oriented Programs are SOFTWARE BULIT-IN FUNCTIONS OR WRITTEN SOFTWARE in a different COMPUTER LANGUAGES. In the past only NUMERICAL SOFTWARE ROUTINES available are NAG ROUTINES written in FORTRAN and ALGOLW which both are POCESSING LANGUAGES and then later C++ languages .The FORTRAN LANGUAGE can compute to a very high accuracy but require and need MAIN-FRAMES COMPUTERS . Instead people now a days are inclined to use PERSONEL COMPUTERS for the money cost is few in such away any person can get one .**

In this paper We FIRST list some of the FUNCTIONS BULIT IN MATHEMATICA OR MATLAB programming languages that deals and compute numerically or (analytically in closed forms for 1-dimentional) and higher dimensional integrals.

SECOND, We will design a MATLAB SOTWARE PROGRAMS for 1-dimentional problems or higher multi-dimensional. For 1-dimentional Integrals METHODS We use are Trapezoidal, Simpson's and Gaussian Quadrature Rules. For 2-dimentional Integrals We apply Radon's approach and Gismlla's approach with regions are the SQUARE REGION and RECTANGLE REGION WITH WEGIHT FUNCTION THE SQUARE ROOT of X, respectively .THIRD, for 3-dimentional or higher Integrals We apply Levin's Transform with our expectation to evaluate Integrals to a high accuracy ((as We have dealt with before in[4] but using FORTRAN languages on main frame COMPUTERS)).

Further, the reader can be acquainted with some symbolic languages and distinguishes them from the usual processing languages. Furthermore, the reader will know that LEVIN'S TRANSFORM can be used as one of the best method to sum a large class of series to a high accuracy.

***Index Terms*— Symbolic Languages and Built-in Functions .Simpson's, Trapezoidal, Gaussian, Radon's, Gismalla's and Levin's Rules. Full Machines Accuracy and Main Frame Computers.**

I. BACKGROUND AND REVIEWS

The problem of numerical integration is one of the oldest problem in mathematics that gives a beautiful and an insight light to a wide range of theoretical and computational Methods in Mathematics and Numerical Analysis. This can be seen when literature retrieval for theory and algorithms is seek, especially for one-dimensional Quadrature Formulae such as Lobbotta, Newton's, Trapezoidal, Simpson and Gaussian Quadrature Rules or for two-dimensional Cubature Rules as in the references [6] ,[8] ,[12],[17],[18], [19] and [20].

The theory of Cubature is burst out when the Product Rules generated from the one-dimensional Rules such as Simpson Rules having the disadvantages of accumulating rounding errors and do not attain the required minimum number of points for functions evaluations in the formula. Stroud [17] has developing many cubature formulae on symmetric regions such as square and the cubic. He creates by theoretical approaches to formulate a matrix system of equations and then solve this system by choosing and selecting a special paten of points to the required cubature formula. The problem of selecting such paten of points in such away to solve the matrix system is difficult unless a particular suitable paten is chosen . This can be seen as a disadvantage. However, if these suitable paten of points can be chosen in advance in such away to solve the system of equations , then many cubature formulas can be obtained.

Cohen & Gismalla [6] has selected a certain paten of points on the square and the cubic to construct some types of cubature formulas known as symmetric cubature formulas as in [6]. Selecting in advance the suitable paten of points can construct good formulas , not for symmetric regions only but on any region with a weight function for that region as in Gismalla[7].

The Germanium (or the Russian) Radon [14] investigates and develop a large information theories on polynomials , orthogonal polynomials and zeros polynomials to establish his fifth degree cubature formula on the square region. If someone reads the translation for this work, he will certainly acknowledge the valuable efforts done to connect these theories in such away to establish Radon Rule. One of the neatness and beauty of Radon approach is that Gismalla [7] has summarized it in steps as an algorithm in such away to construct cubature formulas not on a symmetric region as a square but on any region with its weight function . This can be seen in Gismalla[7] for deriving the same fifth degree formula on rectangle region $\Omega = \{0 \le x \le 1 \; and \; -1 \le x \le 1\}$ with weight function $\sqrt{x}$.

The theories of polynomials as investigated in Radon Rule as algorithms to construct cubature formulas was carried out further in Moller [23] and Shimds [24]. Moller seeks formulas that attain the minimum odd number of points while Shimd's work[24] to attain the minimum even number of points . Gismalla[8] apply Schimd's approach to construct a forth degree formula and two sixth degree formula on the rectangle region $\Omega = \{0 \le x \le 1 \; and \; -1 \le x \le 1\}$ with

Manuscript received Feb. 06, 2014.

D.A.GISMALLA, Dept. of Mathematics , College of Arts and Sciences, Ranyah Branch, TAIF University ,Ranyah(Zip Code :21975) ,TAIF , KINDOM OF SAUDI ARAIBA **Moblie No. 00966 551593472**

Dr. F. M. DAWALBAIT, Dept. of Mathematics , College of Arts and Sciences, Ranyah Branch, TAIF University ,Ranyah(Zip Code :21975) ,TAIF , KINDOM OF SAUDI ARAIBA **Moblie No. 00966 551593472**

weight function $\sqrt{x}$. Both these two approaches can be computerized to generate many cubature formulas but the cost of complexity is too great and high . Even if the complexity is revealed one can expect to obtain formulas with some points outside the region in consideration.

Nevertheless, Stroud [16] has written a great deal of MATLAB SOFTWARE for many regions such as the circle, square, cubic and triangle. The useful of formulas on a triangle can be dealt with for problems on finite element methods especially when it is symmetric. NAG ROUTINES in FORTRAN languages are available but with the restriction that most of them on main frames computers. QUADPACK for numerical Integration can found on FORTRAN or C++ languages or elsewhere.

II. THE BULIT-IN MATHEMATICA INTEGRATE FUNCTIONS

Here , We discuss some of the built_in symbolic Mathematica functions for integration . These functions are Integrate , NIntegrate and Manipulate with Integrate.

A. Integrate

The Integrate function is used to evaluate the indefinite integral $\int \frac{1}{x^3+1}$ dx in Mathematica Mode environment . Typing just the command Integrate in the following line below and then press the two keys SHIFT+ENTER simultaneously to get the result for required indefinite integral as in Example 2.1. The reader must know a little knowledge about how to execute a Mathematica function command . Once the command Integrate[1/(x^3+1),x] is typed and the two keys SHIFT+ENTER are pressed simultaneously , the Mathematica will execute it as input automatically with in[1]:= to indicate that this the first input as

in[1]:= Integrate[1/(x^3+1),x]

The output result will be preceded automatically with out[1]:= to indicate that this the first output result as an answer

out[1]:= $\frac{\mathrm{ArcTan}[\frac{(-1+2\,x)}{\sqrt{3}}]}{\sqrt{3}}+\frac{1}{3}$ Log[1+x]- $\frac{1}{6}$ Log[1-x+x^2]

Similarly , any further input or output will be preceded by its number in sequence automatically for second input or output

in[2]:= or out[2]:= ,.... etc.

Example 2.1

The following Integrate function gives the indefinite integrals

Integrate[*f*,*x*] , gives the indefinite integral $\int f\, d\, x$

In[1]:= Integrate[1/(x^3+1),x]

Out[1]= $\frac{\mathrm{ArcTan}[\frac{(-1+2\,x)}{\sqrt{3}}]}{\sqrt{3}}+\frac{1}{3}$ Log[1+x]- $\frac{1}{6}$ Log[1-x+x^2]

---

In[2]:= Integrate[x^n,x]

Out[2]= $\frac{x^{n+1}}{n+1}$

---

In[3]:= Integrate[1/(x^4-a^4),x]

Out[3]= $-\frac{\mathrm{ArcTan}[\frac{x}{a}]}{2a^3}+\frac{\mathrm{Log}[a-x]}{4a^3}-\frac{\mathrm{Log}[a+x]}{4a^3}$

---

In[4]:= Integrate[Log[1-x^2],x]

Out[4]= -2x – Log[-1+x] + Log[1+x] + x Log[1-x^2]

---

The following Integrate function gives the definite integral

Integrate[*f*,{*x*,*x*$_{min}$,*x*$_{max}$}] ,the definite integral $\int_{x_{min}}^{x_{max}} f(x)\,dx$

Here are Integrate functions for the definite integral $\int_a^b f(x)\,dx$.

In[5]:= Integrate[Sin[x]^2,{x,a,b}]

Out[5]= $\frac{1}{2}$(- a + b + Cos[a]Sin[a] – Cos[b]Sin[b])

---

In[6]:= Integrate[Exp[-x^2],{x,0,Infinity}]

Out[6]= $\frac{\sqrt{\pi}}{2}$

---

In[7]:= Integrate[1/(x^3+1),{x,0,1}]

Out[7]= $\frac{1}{18}$ (2$\sqrt{3}$ π +Log[64])

---

In[8]:= $\int_0^{\omega} Lox[x]Exp[-x^2]dx$

Out[8]= $-\frac{1}{4}\sqrt{\pi}$(EulerGamma+Log[4])

---

***Mathematica* cannot give you a formula for this definite integral $\int_0^1 x^x\,dx$ but instead We can get a numerical result**

In[9]:= Integrate[x^x],{x,0,1}]

Out[9]= $\int_0^1 x^x\,dx$

---

Here below N and % indicates the numerical for the current input % Integrate Command

In[10]:= N[%]

Out[10]= 0.783431

---

Here is the Integrate for the multiple integral

$\int_{x_{min}}^{x_{max}} f(x)\,dx \int_{y_{min}}^{y\ max} f(y)\,dy$ ....

Integrate[*f*,{*x*,*x*$_{min}$,*x*$_{max}$},{*y*,*y*$_{min}$,*y*$_{max}$}

For Multiple integral with x integration outermost:

In[11]:= Integrate[Sin[x y],{x,0,1},{y,0,x}]

Out[11]= $\frac{1}{2}$ (EulerGamma-CosIntegral[1])

---

In[12]:= $\int_0^1 \int_0^x Sin[xy]dy\,dx$

Out[12]= $\frac{1}{2}$ (EulerGamma-CosIntegral[1])

---

Integrals over Regions

This does an integral over the interior of the unit circle.

In[13]:= Integrate[If[x^2+y^2<1,1,0],{x,-1,1},{y,-1,1}]

Out[13]= π

Here is an equivalent form.

In[14]:= Integrate[Boole[x^2+y^2<1],{x,-1,1},{y,-1,1}]

Out[14]= π

Even though an integral may be straightforward over a simple rectangular region, it can be significantly more complicated even over a circular region.

This gives a Bessel function.

In[15]:= Integrate[Exp[x] Boole[x^2+y^2<1],{x,-1,1},{y,-1,1}]
Out[15]= 2 π BesselI[1,1]

B. NIntegrate

NIntegrate[*f*,{*x*,*x*$_{min}$,*x*$_{max}$}]
gives a numerical approximation to the integral
$$\int_{x_{min}}^{x_{max}} f(x)\,dx$$
NIntegrate[*f*,{*x*,*x*$_{min}$,*x*$_{max}$},{*y*,*y*$_{min}$,*y*$_{max}$},…]
gives a numerical approximation to the multiple integral
$$\int_{x_{min}}^{x_{max}} f(x)\,dx \int_{y_{min}}^{y_{max}} f(y)\,dy \ldots f$$

This implies that the built-in NIntegrate Mathematica function evaluates integrals in one or multi-dimensional integrals numerically to a good limit of accuracy as can be shown here in Example 2.2

Example2.2 Compute a numerical integral:
In[16]:= NIntegrate[Sin[Sin[x]],{x,0,2}]
Out[16]= 1.24706

This finds a numerical approximation to the integral
$$\int_0^\infty e^{-x^3}\,dx.$$
In[17]:= NIntegrate[Exp[-x^3],{x,0,Infinity}]
Out[17]= 0.89298

Compute a multi dimensional integral (with singularity at the origin):
In[18]:=
NIntegrate[$\frac{1}{\sqrt{\frac{x1}{3}+\frac{x2}{2}+\frac{x3}{2}+\frac{x4}{10}}}$,{x1,0,1},{x2,0,1},{x3,0,1},{x4,0,1 }]
Out[18]= 1.2391

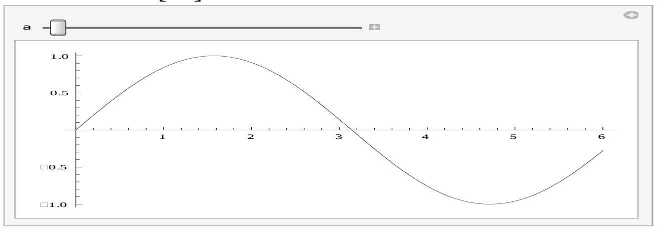
Here is the numerical value of the double integral
$$\int_{-1}^{1}\int_{-1}^{1}(x^2+y^2)\,dy\,dx$$
In[19]:= NIntegrate[x^2+y^2,{x,-1,1},{y,-1,1}]
Out[19]= 2.66667

C. Manipulate with Integrate

The Command Maipulate means that parameters can be manipulate continuously or in discrete steps. Also , it is possible to manipulate two parameters . The manipulation can be done for many results such as expansion or plotting a function. Here , We give an Example 2.3 for Manipulate with Plot and Integrate
Example 2.3 ((See In[20] –In[22]))

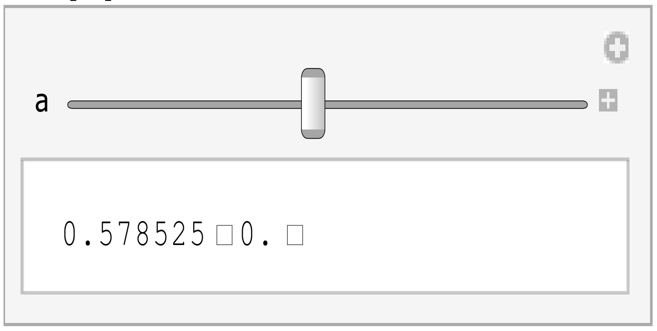
In [20]:= Manipulate[Plot[Sin[x (1+ax)],{x,0,6}],{a,0,2}]

Out[20]



In[21]:= Manipulate[Integrate[Sin[x(1+a x)],{x,0,6}],{a,0,2}]

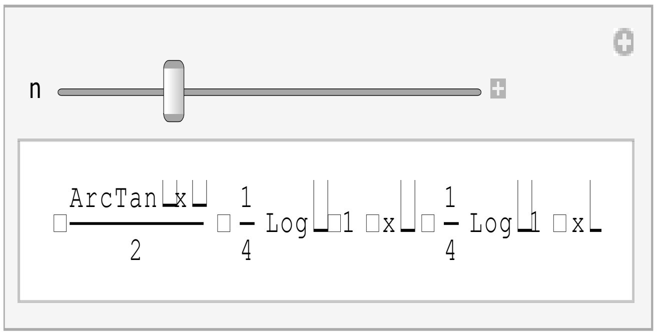
Out[21]=



```
0.578525 □0. □
```

In[2 2]:= Manipulate[Integrate[1/(x^n-1),x],{n,2,10,1}]

Out[22]=



```
  ArcTan⌊x⌋     1              1
□ ─────────── □ ─ Log⌊⌊□1 □x⌋⌋ □ ─ Log⌊⌊1 □x⌋
      2         4              4
```

III. THE BULIT-IN MATLAB QUAD FUNCTIONS

Here We shall describe the four Built-in Matlab QUAD functions to evaluate numerically an integrand over One-dimensional accurately to about six decimal places. These functions are QUAD, QUADL, DBLQUAD and TRIPLEQUAD as can be seen in sections *3.1, 3.2, 3.3 and 3.4, respectively.*

A. QUAD

QUAD Numerically evaluate integral, adaptive Simpson quadrature
Q = QUAD(FUN,A,B)

Q = QUAD(FUN,A,B) tries to approximate the integral of function FUN from A to B to within an error of 1.e-6 using recursive adaptive Simpson quadrature. The function Y =

FUN(X) should accept a vector argument X and return a vector result Y, the integrand evaluated at each element of X.

Q = QUAD(FUN,A,B,TOL) uses an absolute error tolerance of TOL instead of the default, which is 1.e-6. Larger values of TOL result in fewer function evaluations and faster computation, but less accurate results. The **QUAD** function in MATLAB 5.3 used a less reliable algorithm and a default tolerance of 1.e-3.

Use array operators.***, ./ and .^ in** in the definition of FUN so that it can be evaluated with a vector argument.

Example 2.4 Approximate the integral $\int_0^2 \frac{1}{x^3-2x-5}\,dx$ by using **quad** function in Matlab. Observe that the **FUN** in Q=QUAD(FUN , A,B) can be sent as an argument to the function quad using two approaches i.e. FUN can be **specified** as:

```
An inline object:
 F = inline('1./(x.^3-2*x-5)');
 Q = quad(F,0,2);

 A function handle:
 Q = quad(@myfun,0,2);
where myfun.m is an M-file:
 function y = myfun(x)
 y = 1./(x.^3-2*x-5);
```

Type in the Command window to demonstrate an inline object :

```
>> F = inline('1./(x.^3-2*x-5)');
>> Q = quad(F,0,2)
Q =  -0.4605
```

Or Type in the Command window to demonstrate a function handle:

```
>> Q = quad(@myfun,0,2)
Q =  -0.4605
```

B. QUADL

The differences between the built-in function **QUADL** and recursive adaptive **QUAD** is that it uses high order Lobatto quadrature and the function QUAD may be more efficient with low accuracies or nonsmooth integrands . Further **QUADL** can be executed similary as QUAD for FUN **can be specified as inline or as a function handle**.

Example 2.5 Approximate the integral $\int_0^2 \frac{1}{x^3-2x-5}\,dx$ by using quadL function in Matlab

----

Type in the Command window to demonstrate an inline object :

```
>> F = inline('1./(x.^3-2*x-5)');
>> Q = quadl(F,0,2)
Q =  -0.4605
```

Or Type in the Command window to demonstrate a function handle:

```
>> Q = quadl(@myfun,0,2)
Q =  -0.4605
```

C. DBLQUAD

DBLQUAD Numerically evaluate double integral. DBLQUAD(FUN,XMIN,XMAX,YMIN,YMAX) evaluates the double integral of FUN(X,Y) over the rectangle XMIN <= X <= XMAX, YMIN <= Y <= YMAX **FUN(X,Y)** should accept a vector X and a scalar Y and return a vector of values of the integrand .

DBLQUAD(FUN,XMIN,XMAX,YMIN,YMAX,TOL) uses a tolerance TOL instead of the default, which **is 1.e-6 . DBLQUAD(FUN,XMIN,XMAX,YMIN,YMAX,TOL,@QUADL)**
uses quadrature function QUADL instead of the default QUAD FUN can be an inline object or a function handle

Example 2.6 Approximate the integral

$\int_\pi^{2\pi}\{\int_0^\pi (y*sin(x)+x*cos(y))dy\}dx$ by using dblquad function in Matlab

----

Type in the Command window to demonstrate an inline object :

```
>> F = inline('y*sin(x)+x*cos(y)');
>> Q =   dblquad(F, pi, 2*pi, 0, pi)
Q =  -9.8696
```

Or Type in the Command window to demonstrate a function handle:

```
>> Q =   dblquad(@integrnd, pi, 2*pi, 0, pi
Q =-  -9.8696
```

Observe that integrnd.m is an M-file:

```
function f = integrnd(x, y )
f = y*sin(x)+x*cos(x);
```

D. TRIPLEQUAD

TRIPLEQUAD Numerically evaluate triple integral .

TRIPLEQUAD(FUN,XMIN,XMAX,YMIN,YMAX,ZMIN, ZMAX) evaluates the triple integral of FUN(X,Y,Z) over the three dimensional rectangular region **XMIN <= X <= XMAX, YMIN <= Y <=** YMAX, ZMIN <= Z <= ZMAX

FUN(X,Y,Z) should accept a vector X and scalar Y and Z and return a vector of values of the integrand.
TRIPLEQUAD(FUN,XMIN,XMAX,YMIN,YMAX,ZMIN, ZMAX,TOL) uses a tolerance TOL
instead of the default, which is 1.e-6

Example 2.7
Approximate the integral
$\int_0^\pi[\int_0^1\{\int_{-1}^1(y*sin(x)+z*cos(y))dz\}dy]dx$
by using triplequad function in Matlab

Type in the Command window to demonstrate an inline object :

```
>> F = inline('y*sin(x)+z*cos(y)');
>> Q = triplequad( F, 0, pi, 0, 1, -1, 1)
Q =  2.0000
```

Or Type in the Command window to demonstrate a function handle:

```
>> Q = triplequad(@integrnd, 0, pi, 0, 1, -1, 1)
```

```
Q =-  2.0000
Observe that  integrnd.m is an M-file:
function f = integrnd(x, y, z)
f = y*sin(x)+z*cos(x);
-------------------------------------------------
```

IV. MALAB SOFTWARE METHODS & ALGORITHMS FOR NUMERICAL INTEGRATION

These techniques are a generalization of a small low-order degree formula. The main interval is piece wisely divided into small sub-intervals and the required rule is applied on each ones. If we apply Simpson's rule on each subinterval it is called Simpson's **composite rule** as shown by Theorem (4.2) while **Trapezoidal composite rule** by the following Theorem (4.1)

A. Trapezoidal Rule

The composite Trapezoidal rule is given without proof in Theorem 4.1

Theorem 4.1 Trapezoidal composite rule

If $f \in C^2[a,b]$, there exists a $\mu \in [a,b]$ for which **Trapezoidal composite** rule over n subintervals of $[a,b]$ can be expressed with the error term *as*

$$\int_a^b f(x)dx = \frac{h}{2}[f(a)+ 2\sum_{j=1}^{n-1} f(x_j) + f(b)] - \frac{(b-a)h^2}{12} f''(\mu) \qquad (4.1)$$

Where a= $x_0 < x_1 < x_2 \ldots < x_{n-1} < x_n$ =b , h=(b-a)/n, and For each j=0,1,2,….,n

Example 2.7 Here , We write a Matlab program to approximate $\int_{1.1}^{1.6} e^x dx$ using the Trapezoidal composite rule given by Eqn.(4.1).The program is in Fig(4.1) gives the exact value , the approximated value and the approximated Error by printing values of **Ie, Ir and Error** respectively

B. Simpson's Rule

Theorem 4.2 Simpson's composite rule If $f \in C^4[a,b]$, there exists a $\mu \in [a,b]$ for which **Simpson's composite** rule over n=2m **subintervals** of $[a,b]$ can be expressed with the error term *as*

$$\int_a^b f(x)dx = \frac{h}{3}[f(a)+2\sum_{j=1}^{m-1} f(x_{2j}) + 4\sum_{j=1}^{m} f(x_{2j-1}) + f(b)] - \frac{(a-b)h^4}{180} f^{(4)}(\mu) \qquad (4.2)$$

Where a= $x_0 < x_1 < x_2 \ldots < x_{2m-1} < x_{2m}$ =b , h=(b-a)/2m, and j=0,1,2,3,….,2m

Example 2.8

We compute $\int_{1.1}^{1.6} e^x dx$ using the Simpson's composite rule given by Eqn.(4.2).The program in Fig(4.2) gives the exact value , the approximated value and the approximated Error by printing the values of **Ie, Ir and Error** respectively. Observe that Simpson's rule in Example 2.8 uses seven points because We must have the number of points to be even and one must be cautious about the factors 2&4 in theEqn.(4.2). This why some people prefer to apply the trapezoidal rule in Eqn.(4.1) instead of Simpson's rule.

C. Gaussian quadrature

All the Newton-Cotes formulas or the formulas that we have been given so farrequire that the values of the function whose integral is to be approximated be known at evenly spaced points, which might be the expected situation if tabulated data for the function was being used. If the function is given explicitly, however, the points of evaluating the function could be chosen in another manner, which leads to increase the accuracy of approximation.

Gaussian quadrature it chooses the points values for $x_1, x_2 \ldots\ldots\ldots, x_{n-1}, x_n$ in the interval [a ,b]and the constants $c_1, c_2 \ldots\ldots\ldots, c_{n-1}, c_n$ in an optimal manner to minimize the error obtained in performing the approximation given by Eqn.(4.3)

$$\int_a^b f(x)dx \approx \sum_{j=1}^{n} c_j f(x_j) \qquad (4.3)$$

Using the Legendre polynomials and their corresponding roots , it can be shown quadrature rule is exact for any polynomial P(x) (and has degree of precision at least 2n-1)given by Eqn.(4.4) .

$$\int_{-1}^{1} p(x)dx = \sum_{j=1}^{n} c_j p(x_j) \qquad (4.4)$$

The roots of Legendre polynomials are the abscissa points x_1 , x_2 , …., x_{n-1} , x_n and the coefficients c_1, c_2 , .. , c_{n-1}, c_n, both are given in Table 4.1 below

Table 4.1 lists the values for these coefficients and abscissa points for n=2,3,4. Other can be found in Stroud and Secrest[18] or elsewhere .

Since the simple linear transformation t=[1/(b-a)](2x-a-b) will translate any interval [a, b] into [-1, 1] provided b>a , the Legendre polynomials Eqn.(4.4) can be used to approximate

$$\int_a^b f(x)dx = \sum_{j=1}^{n} c_j \; f\left(\frac{(b-a)tb+a}{2}\right)\frac{(b-a)}{2}\,dt \qquad (4.5)$$

for any function that can be evaluated at the required points

Example 2.9 Approximate the integral

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

using Gaussian Quatrature rules with n=2 and n=3 .The exact value to seven decimal places is 0.1093643

Table 4.1 shows the coefficients and points for Gaussian rules for n=2,3&4

n	Roots	Coefficients
2	0.5773502692	1.0000000000
	-0.5773502692	1.0000000000
3	0.7745966692	0.5555555556
	0.0000000000	0.8888888889
	- 0.7745966692	0.5555555556
4	0.8611363116	0.3478548541
	0.3399810436	0.6521451549
	- 0.3399810436	0.6521451549
	0.8611363116	0.3478548541

```
% Example 2.7 Integrate using the
% Composite Trapezoidal Rule
% In Matlab command window
% a=1.1;
% b=1.6;
% n =5 ;
% syms x;
% f=inline('exp(x)') ;
% Ie=compTrapezoidal(f,a,b,n)

 function [Ie ,Ir ,Error]=compTrapezoidal(f,a,b,n)
 h=(b-a)/n;
 x=a:h:b;
 sum=0;
  for i=2:n
    sum=sum+feval(f,x(i));
  end

  % Ie the approximate Integral
  Ie=(h/2)*(feval(f,a)+feval(f,b)+2*sum);
  % Ir the exact value  of Integral
  Ir=int('exp(x)' ,a,b);
  % The absolute Actual
  % Error = abs(Appoximation - Exact)
   Error = abs(Ie-Ir);
```

```
 >> a=1.1;
 >> b=1.6;
 >> n =5 ;
>>  syms x;
>>  f=inline('exp(x)') ;
>>
Ie=compTrapezoidal(f,a,b,n)
 Ie =  1.9505
  Ir = exp(8/5)-exp(11/10)
```

Fig.(4.1) The Composite Trapezoidal Rule in file compTrapezoidal.m & its command window for running it.

```
% 4.3 Integrate using the Commposite Simpson Rule
%  In Matlab command window
%  a=1.1;
%  b=1.7;
%  n = 3 ; % n is even i.e. n=2*m divisions in the formula
%  syms x;
%  f=inline('exp(x)');
%  [Ie,Ir,Error]=compSimpson(f,a,b,n)

function [Ie,Ir,Error]=compSimpson(f,a,b,n)
h=(b-a)/(2*n);
x=a:h:b;
   sum=feval(f,a);
for i=2:2:2*n
   sum=sum+4*feval(f,x(i))+2*feval(f,x(i+1));
end
  sum=sum-feval(f,b);
% Ie he approximate Integral Ie
Ie = (h/3)*sum;
% Ir the exact value  of Integral
Ir=int('exp(x)',a,b);
% The absolute Actual Error =  abs(Appoximation - Exact)
Error = abs(Ie-Ir);
```

Fig.(4.2) (a) The Composite Simpson Rule in file compSimpson.m

```
% 4.3 Integrate using the Commposite Simpson Rule
%  In Matlab command window
%  a=1.1;
%  b=1.7;
%  n = 3 ; % n is even i.e. n=2*m divisions in the formula
%  syms x;
%  f=inline('exp(x)');
%  [Ie,Ir,Error]=compSimpson(f,a,b,n)

function [Ie,Ir,Error]=compSimpson(f,a,b,n)
h=(b-a)/(2*n);
x=a:h:b;
   sum=feval(f,a);
for i=2:2:2*n
   sum=sum+4*feval(f,x(i))+2*feval(f,x(i+1));
end
  sum=sum-feval(f,b);
% Ie he approximate Integral Ie
Ie = (h/3)*sum;
% Ir the exact value  of Integral
Ir=int('exp(x)',a,b);
% The absolute Actual Error =  abs(Appoximation - Exact)
Error = abs(Ie-Ir);
```

Fig.(4.2) (a) The Composite Simpson Rule in file compSimpson.m

```
% Example 4.3(a) Integration using
% Guassian Quatrature rules
% In Matlab command window
% syms x;
% f=inline('exp(-x^2)');
% c=[1.0000000000  1.0000000000];
% x=[0.5773502692 -0.5773502692];
% a=1; b=1.5 ;
% n=2 ;
% Ie=quassian(f,c,x,a,b,n)

function Ie=quassian(f,c,x,a,b,n)
sum=0;
for j=1:n
 t=((b-a)*x(j)+a+b)/2;
  sum=sum+c(j)*feval(f,t)*(b-a)/2;
end
Ie=sum;
```

```
>> syms x;
>>  f=inline('exp(-x^2)');
>> c=[1.0000000000  1.0000000000]
>> x=[0.5773502692 -0.5773502692];
 >>  a=1; b=1.5 ;
 >>  n=2 ;
  >>  Ie=quassian(f,c,x,a,b,n)
        Ie =  0.1094
```

Fig.(4.3(a)) Integration using Guassian Quatrature rules in the file Quassian.m

```
 >>  syms x;
>>  f=inline('exp(-x^2)');
>>  a=1; b=1.5 ;
>>  n=2 ;
>>  [Ie,Ir,Error]=quassianTable(f,a,b,n)
        Ie  =   0.1094
        Ir   =  1/2*erf(3/2)*pi^(1/2)-1/2*erf(1)*pi^(1/2)
     Error =  3941559804514209/36028797018963968
              - 1/2*erf(3/2)*pi^(1/2)+1/2*erf(1)*pi^(1/2)
```

Fig.(4.3(b)) The command window for the file quassianTable.m with the Gauss Table not passed as parameters as in Fig.(4.3(b)) below

```
 % Example 4.3(b)Integration using Guassian Quatrature rules
  % In Matlab command window
  % syms x;
  % f=inline('exp(-x^2)');
  % a=1; b=1.5 ;
  % n=2 ; % Guassian-Table is given and n runs from 2 to 5
  % [Ie,Ir,Error]=quassianTable(f,a,b,n)

function [Ie,Ir,Error]=quassianTable(f,a,b,n)

 c=[1.0000000000   0.5555555556   0.3478548451    0.2369268850
    1.0000000000   0.8888888889   0.6521451549    0.4786286705
    0.0000000000   0.5555555556   0.6521451549    0.5688888889
    0.0000000000   0.0000000000   0.3478548451    0.4786286705
    0.0000000000   0.0000000000   0.0000000000    0.2369268850];
x=[ 0.5773502692   0.7745966692   0.8611363116    0.9061798459
   -0.5773502692   0.0000000000   0.3399810436    0.5384693101
    0.0000000000  -0.7745966692  -0.3399810436    0.0000000000
    0.0000000000   0.0000000000  -0.8611363116   -0.5384693101
    0.0000000000   0.0000000000   0.0000000000   -0.9061798459];

  sum=0;
 for j=1:n
  t=((b-a)*x(j,n-1)+a+b)/2;
   sum=sum+c(j,n-1)*feval(f,t)*(b-a)/2;
 end
 Ie=sum;
 Ir=int('exp(-x^2)',a,b);
 Error = abs(Ie-Ir);
```

Fig.(4.3(b)) Guassian Quatrature rules in the file quassianTable.m

V. Cubature's Rules for 2-Dimentional Integration

Cubature formulae are quadrature formulae that having the minimum points to attain the required degree of accuracy when evaluating a polynomial as an integrand .In a series of papers rules for constructing cubature formulae have been established as in STRUOD [4], SCHIMD [17],RADON[14] andGISMALLA[6-8].

A. Gismalla's Rules

Here , We are cited two rules from Gismalla[7] , where the first one is a fourth degree using seven points to integrate an integrand f(x,y) with weight function $\sqrt{x}$ on the rectangle region of integration
$\Omega = \{ (x, y): 0 \le x \le 1 \; and \; -1 \le y \le 1\}$.
The Rule is given by Eqn.(4.6) , where the abscissas x(j) & y(j) and the coefficients c(j) for j=1(1)7 are given inTable 4.2 . Similarly the second is a fifth degree rule using seven points . The Rule is by Eqn.(4.6), where the abscissas x(j) & y(j) and the coefficients c(j) for j=1(1)7 are given in Table 4.3 .The Matlab Program for Gismalla fourth degree Rule is given in by the file GismallaRule4.m in Fig.(4.4) & for the fifth drgree Rule is by GismallaRule5.m in Fig.(4.5)

$$\int_0^1 \int_{-1}^1 \sqrt{x} \;\; f(x,y)dxdy \approx \sum_{j=1}^{7} c(j) f\big(x(j), y(j)\big) \qquad (4.6)$$

B. Radon's Rule

Radon[14] has established a fifth degree formula where the region of the integration is the square with the unity weight function w(x)=1 and the formula is given by Eqn.(4.7)

```
% Integration using  GismallaRule for Double Integration on a rectangle .
% This Quatrature rule of fourth degree to evaluate an intgrand
% x^0.5*f(x,y) on the regin R={(x,y}: 0=< x =<1 and -1=< y =<1}
% The integrand must be called without the weight function sqrt(x)
% as an inline or function handle object of the form f(x,y) only.
% In Matlab command window
% syms s;
% syms t;
% f=inline(' exp(-s*t)');
% a=0; b=1 ;
% c=-1; d=1 ;
% n=7; % Gismalla-Table is given for n = 7
% [Ie]= GismallaRule4(f,a,b, c, d ,n)

function [Ie]= GismallaRule4(f,a,b,c,d,n)
 x=[ 0.0607124836   0.0000000000   0.0000000000
     0.3323846207   0.5329336163   0.0000000000
     0.1994954883   0.8943391109   0.0000000000
     0.1851851852   0.8618614683   0.7745966692
     0.1851851852   0.8618614683   -.7745966692
     0.1851851852   0.3381385317   0.7745966692
     0.1851851852   0.3381385317   -.7745966692  ];
 TotalSum=0;
  for j=1:n
     s(j)=((b-a)*x(j,2)+a ) ;
      t(j)=((d-c)*x(j,3)+c+d)/2;
     sum(j)= x(j , 1)*feval(f , s(j) , t(j) ) ;
     TotalSum=TotalSum + sum(j);
End
Ie=(d-c)*0.5*TotalSum ;
```

```
>> syms s;
>>  syms t;
>> f=inline(' exp(-s*t)');
>> a=0; b=1 ;
>> c=-1; d=1 ;
>> n=7;
>> [Ie]= GismallaRule4(f,a,b, c, d ,n)
>>  Ie= 1.4318
```

Fig(4.4) The fourth degree GismallaRule4 with Matlab Program GismallaRule4.m & its Command Window

Table 4.2 The abscissas x(j) & y(j) and the coefficients c(j) for j=1(1)7 for the fourth degree

j	c(j)	x(j)	y(j)
1	0.0607124836	0.0000000000	0.0000000000
2	0.3323846207	0.5329336163	0.0000000000
3	0.1994954883	0.8943391109	0.0000000000
4	0.1851851852	0.8618614683	0.7745966692
5	0.1851851852	0.8618614683	-0.7745966692
6	0.1851851852	0.3381385317	0.7745966692
7	0.1851851852	0.3381385317	-0.7745966692

Table 4.3 The abscissas x(j) & y(j) and the coefficients c(j) for j=1(1)7 for the fifth degree

j	*c(j)*	*x(j)*	*y(j)*
1	*0.3911751538*	*0.5686185251*	*0.0000000000*
2	*0.1305837632*	*0.9871086266*	*0.0000000000*
3	*0.0708336756*	*0.0596574637*	*0.0000000000*
4	*0.1541977768*	*0.2899491979*	*0.7745966692*
5	*0.1541977768*	*0.2899491979*	*-0.7745966692*
6	*0.2161725936*	*0.8211619132*	*0.7745966692*
7	*0.2161725936*	*0.8211619132*	*-0.7745966692*

```
%  Integration using  Gismalla Rule for Double Integration on a rectangle .
%  This Quatrature rule of fifth degree to evaluate an intgrand
%  x^0.5*f(x,y) on the regin R={(x,y}: 0=< x =<1 and -1=< y =<1}
%  The integrand must be called without the weight function sqrt(x)
%  as an inline or function handle object of the form f(x,y) only.
%  In Matlab command window
%  syms s;
%  syms t;
%  f=inline(' exp(-s*t)');
%  a=0; b=1 ;
%  c=-1; d=1 ;
%  n=7; % Gismalla-Table  is given for  n = 7
%  [Ie]= GismallaRule5(f,a,b, c, d ,n)

function [Ie]= GismallaRule5(f,a,b,c,d,n)
 x=[ 0.3911751538  0.5686185251   0.0000000000
      0.1305837632  0.9871086266   0.0000000000
      0.0708336756  0.0596574637   0.0000000000
      0.1541977768  0.2899491979   0.7745966692
      0.1541977768  0.2899491979   -.7745966692
      0.2161725936  0.8211619132   0.7745966692
       0.2161725936  0.8211619132   -.7745966692   ];
 TotalSum=0;
  for j=1:n
      s(j)=((b-a)*x(j,2)+a ) ;
       t(j)=((d-c)*x(j,3)+c+d)/2;
      sum(j)= x(j , 1)*feval(f , s(j) , t(j) ) ;
      TotalSum=TotalSum  + sum(j);
End
Ie=(d-c)*0.5*TotalSum  ;
```

```
>> syms s;
>>  syms t;
>> f=inline(' cos(0.5*pi*(s+t))');
>> a=0; b=1 ;
>> c=-1; d=1 ;
>> n=7;
>> [Ie]= GismallaRule5(f,a,b, c, d ,n)
>>  Ie= 1.4316
```

Fig(4.5) The fifth degree GismallaRule5 with Matlab Program GismallaRule5.m & its Command Window

$$\int_{-1}^{1}\int_{-1}^{1} f(x,y)\,dxdy \approx \sum_{j=1}^{7} c(j) f(x(j),y(j)) \qquad (4.7)$$

Where the abscissa and weight coefficient are in Table 4.4 and the Matlab Program for RadonRule.m is in the Fig.(4.6). As in Gismalla[12] , the reader can test these the three programs GismallaRule4.m, GismallaRule5.m and Radon Rule 5.m for the three given integrals below with their exact values

$$I_1 = \int_0^1\int_{-1}^1 \sqrt{x}\cos\left(\frac{\pi}{2}(x+y)\right)dxdy \approx 0.45533 \qquad (4.8)$$

$$I_2 = \int_0^1\int_{-1}^1 \sqrt{x}\, e^{-xy}dxdy \approx 1.42196496 \quad (4.9)$$

$$\text{and } I_3 = \int_0^1\int_{-1}^1 \sqrt{x}\,\frac{1}{4+x+y}dxdy \approx \qquad 0.2956196 \qquad (4.10$$

```
% Integration using  RadonRule for Double Integration on a square .
% This Quatrature rule of fifth degree to evaluate an intgrand
% f(x,y) on the regin  R={(x,y}: -1=< x =<1  and -1=< y =<1}
% The integrand must be called with the  unity weight function
% as an inline or function handle object of the form f(x,y)  .
% In Matlab command window
% syms s;
% syms t;
% f=inline('sqrt(s)*exp(-s*t)');
% a=0; b=1 ;
% c=-1; d=1 ;
% n=7; % Radon-Table is given for  n = 7
% [Ie]= RadonRule5(f,a,b, c, d ,n)

 function [Ie]= RadonRule5(f,a,b,c,d,n)
  x=[  8/7      0            0
        20/63    0          (14/15)^0.5
        20/63    0         -(14/15)^0.5
          5/9    (3/5)^0.5       (1/3)^0.5
          5/9   -(3/5)^0.5       (1/3)^0.5
          5/9    (3/5)^0.5      -(1/3)^0.5
          5/9   -(3/5)^0.5      -(1/3)^0.5   ];
  TotalSum=0;
 for j=1:n
     s(j)=((b-a)*x(j,2)+a+b)/2 ;
     t(j)=((d-c)*x(j,3)+c+d)/2;
     sum(j)=x(j,1)*feval(f,s(j),t(j));
    TotalSum=TotalSum+sum(j);
 end
  TotalSum=(b-a)*(d-c)*TotalSum/4;
  Ie=TotalSum;
```

```
>> syms s;
>> syms t;
>>  f=inline('sqrt(s)*exp(-s*t)');
>>  a=0; b=1 ;
>>  c=-1; d=1 ;
>>  n=7;
>> [Ie]= RadonRule5(f,a,b, c, d ,n)
>>   Ie=1.43563886222718
```

Fig(4.6) The fifth degree RadonRule5 with Matlab Program

Table 4.4 The abscissas x(j) & y(j) and the coefficients c(j) for j=1(1)7 for RadonRule5 in Eqn.(4.7) with Matlab Program RadonRule5.m in Fig(4.6)

j	c(j)	x(j)	y(j)
1	8/7	0	0
2	20/63	0	(14/15)^0.5
3	20/63	0	- (14/15)^0.5
4	5/9	(3/5)^0.5	(1/3)^0.5
5	5/9	-(3/5)^0.5	- (1/3)^0.5
6	5/9	(3/5)^0.5	(1/3)^0.5
7	5/9	-(3/5)^0.5	- (1/3)^0.5

VI. Levin's Transform Rule for higher dimensional Integration

Levin's U-transform Method [12] has been shown and used to efficiently compute certain types of multiple integrals to a high decimal places of accuracy using FORTARN program languages . The Computing Machine was the main FRAME computer applied with a FULL machine accuracy to get high decimal places of accuracy. The Levin's Method has a drawback; it fails to compute certain types of series.

Instead of FORTARN program language written for Levin's Method which cannot be available now or after , We write it here instead using Matlab program. In contrast for using FULL machine accuracy when using FORTAN language, the command ***format long*** can be used instead in the Matlab program to get high decimal places of accuracy . Levin's U-transform is defined in Eqn.(4.11) as

$$U_{2*k+1}(s)= \frac{\sum_{j=0}^{2k+1}(-1)^j\binom{2k+1}{j}(j+1)^{2k-1}\left(\frac{S_{j+1}}{a_{j+1}}\right)}{\sum_{j=0}^{2k+1}(-1)^j\binom{2k+1}{j}(j+1)^{2k-1}\left(\frac{1}{a_{j+1}}\right)} \qquad (4.11)$$

where S_n is the partial sum of n terms of the convergent series of positive terms as in Eqn.(4.12).

$$S=\sum_{1}^{\infty} a_n \qquad (4.12)$$

Here , our aim is to apply Levin's U-transform Matlab program to evaluate the three multiple integrals considered in [12] using FORTARAN language . The three Lattice Green functions are

$$G(000)=\frac{1}{\pi^3}\int_0^{\pi}\int_0^{\pi}\int_0^{\pi}\frac{dx\,dy\,dz}{1-\cos x\cos y\cos z} \qquad (4.13)$$

$$G(100)=\frac{1}{\pi^3}\int_0^{\pi}\int_0^{\pi}\int_0^{\pi}\frac{\cos 2x\ \ dx\,dy\,dz}{1-\cos x\cos y\cos z} \qquad (4.14)$$

$$G(110)=\frac{1}{\pi^3}\int_0^{\pi}\int_0^{\pi}\int_0^{\pi}\frac{\cos 2x\cos 2y\ dx\,dy\,dz}{1-\cos x\cos y\cos z}=\sum_{n=0}^{\infty}a_n^3 \qquad (4.15)$$

Since $\cos(2x)=\cos^2(x)-1$ and

$\cos(2y)=\cos^2(y)-1$, it follows that

$$G(100)=2I_4-G(000) \qquad (4.16)$$

$$G(110)=4I_5-4I_4+G(000) \qquad (4.17)$$

where

$$I_4=\frac{1}{\pi^3}\int_0^{\pi}\int_0^{\pi}\int_0^{\pi}\frac{\cos^2 x\,dx\,dy\,dz}{1-\cos x\cos y\cos z} \qquad (4.18)$$

$$I_5=\frac{1}{\pi^3}\int_0^{\pi}\int_0^{\pi}\int_0^{\pi}\frac{\cos^2 x\ \cos^2 x\,dx\,dy\,dz}{1-\cos x\cos y\cos z} \qquad (4.19)$$

Now by expanding the integrand in G(000) , I_4 and I_5 in Eqn.(4.13) ,Eqn.(4.18) and Eqn.(1.19) respectively and applying Wallis' formula , We get the integrals as

$$G(000)=\sum_{n=0}^{\infty}a_n^3 \qquad (4.20)$$

$$I_4=\sum_{n=0}^{\infty}a_n^2\,a_{n+1} \qquad (4.21)$$

$$I_5=\sum_{n=0}^{\infty}a_n\,a_{n+1}^2 \qquad (4.22)$$

Table 4.5 LEVIN TRANSFORM SUM FOR G(000)

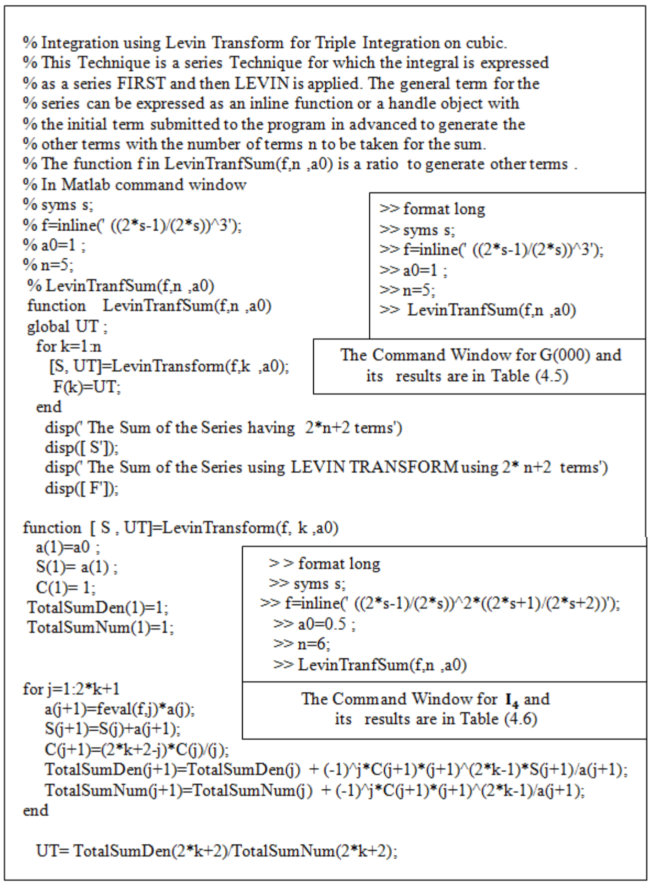
k	The Sum S_k	LEVIN TRANSFORM U_{2*k+1}
0	1.00000000	1.4000720
1	1.12500000	1.3931052
2	1.17773437	1.39320476
3	1.20825195	1.3932039225
4	1.22869634	1.39320392968
5	1.24360030	
6	1.25508015	
7	1.26427156	
8	1.27184504	
9	1.27822511	G(000)=1.39320392968
10	1.28369522	
11	1.28845279	

Table 4.6 LEVIN TRANSFORM SUM FOR I_4

k	The Sum S_k	LEVIN TRANSFORM U_{2*k+1}
0	0.500000000	0.740888952
1	0.593750000	0.832874813
2	0.637695312	0.841755935
3	0.664398193	0.842047393
4	0.682798147	0.842052522
5	0.696460112	0.842052578
6	0.707119970	
7	0.715736914	
8	0.722889651	I_4 =0.842052578
9	0.728950713	
10	0.734172180	G(100)=0.290901226
11	0.738731524	
12	0.742757787	
13	0.746347348	

Table 4.7 LEVIN TRANSFORM SUM FOR I_5

k	The Sum S_k	LEVIN TRANSFORM U_{2*k+1}
0	0.250000000	0.380674380
1	0.320312500	0.533267522
2	0.356933593	0.550563530
3	0.380298614	0.551141114
4	0.396858572	0.551151238
5	0.409382041	0.551151349
6	0.419280480	
7	0.427358865	
8	0.43411428	I_5= 0.551151349
9	0.439872237	
10	0.444856364	G(100)=0.229599014
11	0.449225735	
12	0.453097143	
13	0.456558504	



```
% Integration using Levin Transform for Triple Integration on cubic.
% This Technique is a series Technique for which the integral is expressed
% as a series FIRST and then LEVIN is applied. The general term for the
% series can be expressed as an inline function or a handle object with
% the initial term submitted to the program in advanced to generate the
% other terms with the number of terms n to be taken for the sum.
% The function f in LevinTranfSum(f,n ,a0) is a ratio to generate other terms .
% In Matlab command window
% syms s;
% f=inline(' ((2*s-1)/(2*s))^3');
% a0=1 ;
% n=5;
 % LevinTranfSum(f,n ,a0)
 function   LevinTranfSum(f,n ,a0)
 global UT ;
  for k=1:n
    [S, UT]=LevinTransform(f,k  ,a0);
     F(k)=UT;
  end
   disp(' The Sum of the Series having  2*n+2 terms')
   disp([ S']);
   disp(' The Sum of the Series using LEVIN TRANSFORM using 2* n+2  terms')
   disp([ F']);

function  [ S , UT]=LevinTransform(f, k ,a0)
  a(1)=a0 ;
  S(1)= a(1) ;
  C(1)= 1;
 TotalSumDen(1)=1;
 TotalSumNum(1)=1;

for j=1:2*k+1
   a(j+1)=feval(f,j)*a(j);
   S(j+1)=S(j)+a(j+1);
   C(j+1)=(2*k+2-j)*C(j)/(j);
   TotalSumDen(j+1)=TotalSumDen(j)  + (-1)^j*C(j+1)*(j+1)^(2*k-1)*S(j+1)/a(j+1);
   TotalSumNum(j+1)=TotalSumNum(j)  + (-1)^j*C(j+1)*(j+1)^(2*k-1)/a(j+1);
end

  UT= TotalSumDen(2*k+2)/TotalSumNum(2*k+2);
```

```
>> format long
>> syms s;
>> f=inline(' ((2*s-1)/(2*s))^3');
>> a0=1 ;
>> n=5;
>>  LevinTranfSum(f,n ,a0)
```

The Command Window for G(000) and its results are in Table (4.5)

```
> > format long
>> syms s;
>> f=inline(' ((2*s-1)/(2*s))^2*((2*s+1)/(2*s+2))');
 >> a0=0.5 ;
 >> n=6;
 >> LevinTranfSum(f,n ,a0)
```

The Command Window for $\mathbf{I_4}$ and its results are in Table (4.6)

Fig(4.7) Matlab Program LevinTranfSum.m with its Two Command Window for G(000) and $\mathbf{I_4}$. The Output are in Table 4.5 &Table 4.6

where the coefficients terms a's are given by

$$a_n = \frac{1.3.5 \ldots (2n-1)}{2.4.6 \ldots (2n)} \quad , \quad a_0 = 1 \qquad (4.23)$$

Now the Matlab Program to integrate G(000) , I_4 and I_5 in Eqn.(4.20) ,Eqn.(4.21) and Eqn.(4.22) respectively is given in Fig(4.7). The output results from the command window program for G(000) , I_4 and I_5 are given in Table 4.5 , Table 4.6 and Table 4.7 respectively..

Observe that to compute the integral $\mathbf{I_5}$,
We need to replace the generating ratio function and the initial term a0 by
f=inline(' ((2*s-1)/(2*s))*((2*s+1)/(2*s+2))^2');
and a0=0.25 ; in the command window program for LevinTranfSum.m given in Fig.(4.7) .

Hence the values for the integral G(100) in Eqn.(4.16) and G(110) in Eqn.(4.17) can be evaluated up to eight decimal places of accuracy as in Eqn.(4.24) where
G(110)=0.229599014
and G(100)=0.29091226 (4.24)

VII. Conclusion

The reader must know all types of Programs Built-in as ROUTINES SOFTWARE are inflexible programs to suit and solve any particular types of problems while the others are flexible programs. This can be seen if someone attempts to run the routine DBLQUAD on MALAB for the integral
I_1 in Eqn. (4.18) , I_2 in Eqn. (4.19)
or I_3 in Eqn (4.20) an ERROR will be occurred. This because DBLQUAD on MATLAB for interval [0,1] which runs on variable x will be transformed for a different one by the inside QUAD quadrature used by DBLQUAD . Even if one doesn't submit the parameters a=0 and b=1 in the MATLAB program GismallRule4.m or GismaleRule5.m an ERROR will be occurred due to the fact that any transformation to the interval [0,1] will effect to translate the corresponding weight function $\sqrt{x}$ given in Eqn. (4.6) to a different weight function and so the abscissas' and coefficients will be different from their corresponding in the given rule in Eqn.(4.6)

However, the program RadonRule5.m can run these integrals efficiently without any errors provided the integrand function must be submitted as
$\sqrt{x}$ f(x, y) exactly as with these
integrands given by Eqn.(4.18) , Eqn.(19) and Eqn.(20).

Levin's Transform in Eqn.(4.11) written as functions or subroutines using FORTRAN languages , even on main-frames computers will compute the integrals G(000) , I_4 and I_5 in Eqn.(4.20) ,Eqn.(4.21) and Eqn.(4.22) respectively only to five or sixth decimal places at most. In Gismalla[12] these integrals were computed to a very high decimal places up to 12 decimals places using the same Levin's Transform in Eqn.(4.11) . The cost is very very too expensive , the technique Levin's Transform must be phrased and unexpressed as function or procedures each times when it is called with SOME SPECIAL COMMAND WITTEN ON THE FIRST TOP LINES to generate the FULL MACHINE ACCRACCY which are unavailable unless given by the advisory of the main-frame
computers. The computational remarks with Levin's Transform as a function written in FORTRAN language can found in gismalla[12].

On the contrary, the LevinTrasfSum.m given in Fig.(4.7) computes these integrals neatly
up to eight digits of accuracy exactly without a great complexity and efforts using the symbolic languages MATLAB and even the program is SHORT.

Acknowledgment

First, I would like to thank the department of Mathematics, Tabuk University Saudi Arabia to allow me teach the Course of Mathematica 2010 that give me the KNOWLEDGE of what the symbolic languages are .

Second, I would like to thank the department of athematics, Gezira University Sudan to allow me teach the Course of Numerical Analysis applied with Matlab Language during the years 2011 & 2012 that give me the KNOWLEDGE to differentiate between the symbolic languages and the processing languages.

Further , Taif ,University , SUDAI ARABIA for the encouragement to Staff Member to establish research work .

REFERENCES

[1] Richard L. Burden, J. Douglas Faires, Numerical Analysis Prindle, Weber & Schmidt, Boston 1985

[2] Milton Abramowitz and Irene A. Stegun,Handbook of Mathematical of Mathematical Functions, Dover Publications, Inc., New York, USA 1972

[3] Carl-Erik Froberg, Numerical Mathematics Theory and Computer Applications, the Benjamin/Cummings Company, Inc., California USA 1985

[4] D.A.Gismalla , Lynne D. Jenkins and A.M.Cohen Acceleration of Convergence of Series for Certain Multiple Integrals , I.J.C.M, Vol. 24, pp 55-68, 1987.

[5] D.A.Gismalla, Summation Method for Some Special Series Exactly The International Journal of Mathematics, Science, Technology and Management (ISSN : 2319-8125) , Vol. 1 Issue 2, India, 2012

[6] A.M.Cohen and D.A.Gismalla, Integration Formulae for symmetric Functions of Two Variables. I.J .C.M, Vol. 19, 1986, pp. 37-48.

[7] D.A.Gismalla, Quadrature Formulae with the weight Function $\sqrt{X}$. I.J.C.M. Vol. 26, pp 57-67, 1987.

[8] D.A.Gismalla, Schmidt's Approach on Cubature Formulae. I.J.C.M. Vol. 32, pp 75-85, 1988.

[9] D.A.Gismalla, Chebyshev Approximation for COS ($\frac{1}{2}\pi X^{4)}$ &SIN ($\frac{1}{2}\pi X^{4)}$ Proceedings of the International Conference on Computing, pp.37, ICC 2010, INDIA

[10] D. A. Gismalla, MATLAB programs for some Numerical Methods and Algorithms, International Journal of Algorithms, Computing and Mathematics, Vol. 5 Number 1, pp.58, Feb. 2012

[11] D. A. Gismalla, Inversion Methods for Toeplitz Matrices, (a dissertation for the M.Sc. , the Software was written using Aglow & FORTRAN Languages), University of Wales, U.K. 1982.

[12] D. A. Gismalla, Methods for theEvaluations of N-dimensional Integrals, (a Thesis for the Ph.D. , Software was written using FORTRAN Language with Full Machine Accuracy) , University of Wales, U.K. 1984

[13] T. Hahn , Cuba – a library for multidimensional numerical integration http://www.feynarts.de/cuba Max-Planck-Institut fur Physik Fohringer Ring 6, D–80805 Munich, Germany January 26, 2005

[14] J.Radon, Zur Mechanische Kubature, Monatsh, Math.52(19480, 268-300.

[15] A.H.Stroud, Approximate Calculations of Multiple Integrals , Prentice-Hall ,Inc, Englewood Cliff, N.J.(1971)

[16] STROUD Numerical Integration in M Dimension file:///C:/Documents%20and%20Settings/math/Desktop/STROUD%20-%20Numerical%20Integration.htm

[18] Arthur Stroud , Approximate Calculation of Multiple Integrals, Prentice Hall, 1971, ISBN: 0130438936, LC: QA311.S85.

[19] Arthur Stroud, Don Secrest, Gaussian Quadrature Formulas, Prentice Hall, 1966, LC: QA299.4G3S7.

[20] Stephen Wandzura, Hong Xia ,Symmetric Quadrature Rules on a Triangle,Computers and Mathematics with Applications,Volume 45, 2003, pages 1829-1840.

[21] Stephen Wolfram, The Mathematica Book, Fourth Edition, Cambridge University Press, 1999, ISBN: 0-521-64314-7,

[22] James Lyness, Dennis Jespersen, Moderate Degree Symmetric Quadrature Rules for the Triangle, Journal of the Institute of Mathematics and its Applications, Volume 15, Number 1, February 1975, pages 19-32.

[23] Hermann Engels, Numerical Quadrature and Cubature, Academic Press, 1980, ISBN: 012238850X, LC: QA299.3E5.

[24] H.M.Moller , Kubatureformulen mit minimaler kontenazhl, Numer.Math.,25(1976),185-200

[25] Hans Joachim Schmid ,on cubature formulae with a minimal number of knots. Number.Math.31(1978) ,28-297

D.A.GISMALLA, Dept. of Mathematics , College of Arts and Sciences, Ranyah Branch, TAIF University ,Ranyah(Zip Code :21975) ,TAIF , KINDOM OF SAUDI ARAIBA **Moblie No. 00966 551593472**

Dr. F. M. DAWALBAIT, Dept. of Mathematics , College of Arts and Sciences, Ranyah Branch, TAIF University ,Ranyah(Zip Code :21975) ,TAIF , KINDOM OF SAUDI ARAIBA **Moblie No. 00966 551593472**