

* n = 1

En este caso la integral a calcular es simplemente :



* + Monte -Carlo de aciertos y fallos

El programa se puede implementar con fortan de la siguiente forma , los pasos salen comentados :

**Program** MONTECARLO\_HIT\_AND\_MISS\_1var

**Implicit Double precision** (A-H,o-z)

**DOUBLE PRECISION** g ,a,b,c,p,r,s ,u,v,pi

**integer** n , na

*! Definimos el itervalo de integracion a,b*

*! y la altura "c" maxima que alcanza la curva*

*! a integrar.*

a=0d0

b=1d0

c=1d0

na=0

*! Ponemos un valor inicial para n*

n=10

*! Creamos un archivo en Excel sobre el que escribiremos*

*! los resultados .*

**open**(60,File="ejerc10aciertfallos1var.xls")

*! Empezamos a contar el tiempo*

**call** cpu\_time(start)

*! Iniciamos un bule para j , que va a hacer el calculo*

*! para diferentes valores de n ( el programa se detendra*

*! con el valor que introducimos en la sentencia "if" de mas abajo)*

*! el valor de la derecha tiene que ser mayor que el que pongamos en el ! bloque if*

**do** j=n,10000000000

*! para este valor de "n" calculamos las variables aleatorias ,*

*! A partir de aqui copiamos el listado de programa de los apuntes ,*

*! excepto para el generador de num aleatorios que ahora es rand() .*

**do** 1 i=1,n

u=rand()

v=rand()

**if** (g(a+(b-a)\*u)**.gt.**c\*v) na=na+1

1 **continue**

p= **real** (na)/n

r=(b-a)\*c\*p

s=sqrt(p\*(1.-p)/n)\*c\*(b-a)

*! Hasta aqui llega el copiado*

*! Escribimos el numero de iteraciones (N) ,*

*! el resultado (r) y el error (s)*

**write**(60,20)n,r,s

*! ponemos a la derecha de .gt. el número de iteraciones deseadas*

**if** (n**.gt.**1d7) **go to** 30

*! Seguimos aumentando "n" si es suficientemente pequeña*

n=n\*10

na=0

**enddo**

*! Acabamos de contar el tiempo*

30 **call** cpu\_time(finish)

**write**(60,\*)"Temps = ",finish-start ,"s"

*! cerramos el archivo Excel*

**close**(60)

*! Introducimos el formato que queremos para nuestros resultados*

20 **Format**(I20,4X,F20.14,4X,E20.14)

**end**

*! Definimos la funcion a integrar*

**Function** g(x1)

**Implicit Double Precision** (A-H,O-Z)

g=exp(-x1\*\*2)\*cos(x1\*x1)

**end**

----------------------------------------------------------------------------------------------

Así , obtenemos un fichero con una salida :

N r s=error

|  |
| --- |
| 10 0.90000000000000 0.94868329805051E-01  100 0.68000000000000 0.46647615158762E-01  1000 0.71000000000000 0.14349216006458E-01  10000 0.69820000000000 0.45903895259553E-02  100000 0.69676000000000 0.14535663122128E-02  1000000 0.69868100000000 0.45883097131624E-03  10000000 0.69878650000000 0.14508064220210E-03  100000000 0.69875110000000 0.45880061055843E-04  Temps = 24.8977596 s  Donde tenemos para diferentes valores de n , el resultado de la integración y el error cometido .  Algo que podemos hacer con esto es intentar encontrar una forma de estimar el tiempo que tardaríamos para obtener una precisión determinada . Graficando log(error) vs logN :    Como vemos obtenemos una ecuación :    Si suponemos que N(t) es lineal , al hacer 10+100+1000+10000+…= 111111111 iteraciones en t=24.9s , Podemos escribir :    Luego tenemos una ecuación :  Que podemos resolver : |
|  |

* + Monte-Carlo de muestreo uniforme

De forma análoga completamos el programa con lo que tenemos en los apuntes. El listado de programa es el siguiente:

**Program** MONTECARLO\_MUESTREO\_UNIFORME\_1var

**Implicit Double precision** (A-H,o-z)

**DOUBLE PRECISION** g,g0,a,b,r1,s1,r,s,pi,u

**real** finish ,start

**integer** n

a=0d0

b=1d0

n=10

**open**(60,File="Montmostrunif1var.xls")

**call** cpu\_time(start)

**do** j=n,10000000000

*! copiamos lo que hay en los apuntes*

r1=0.

s1=0.

**do** 1 i=1,n

u=rand()

g0=g(a+(b-a)\*u)

r1=r1+g0

s1=s1+g0\*g0

1 **continue**

r1=r1/n

s1=sqrt((s1/n-r1\*r1)/n)

r=(b-a)\*r1

s=(b-a)\*s1

*! hasta aqui hemos copiado*

**write**(60,20)n,r,s

**if** (n**.gt.**1d7) **go to** 30

n=n\*10

**enddo**

30 **call** cpu\_time(finish)

**write**(60,\*)"Temps = ",finish-start ,"s"

**close**(60)

20 **Format**(I20,4X,F20.14,4X,E20.14)

**end**

**Function** g(x1)

**Implicit Double Precision** (A-H,O-Z)

g=exp(-x1\*\*2)\*cos(x1\*x1)

**end**

Los resultados obtenidos son los siguientes:

N r s

|  |
| --- |
| 10 0.83969783804519 0.50489014029151E-01 |
| 100 0.68200356867976 0.24666606107393E-01 |
| 1000 0.69590843388417 0.81300450775772E-02 |
| 10000 0.69715211810233 0.25332334644440E-02 |
| 100000 0.69639492735482 0.80329471666808E-03 |
| 1000000 0.69865817754566 0.25293416090562E-03 |
| 10000000 0.69867718519326 0.80025028083491E04 |
| 100000000 0.69874578045475 0.25295788531252E-04 |
| Temps = 24.89776 s |

Observemos que en este caso resulta ser ligeramente más lento este programa que el de aciertos y fallos , aunque el error cometido es bastante menor.

* + Método de Simpson simple

Este método consiste en hacer la siguiente aproximación a la integral de una función f(x) :

 \int_{a}^{b} f(x) \, dx \approx \frac{b-a}{6}\left[f(a) + 4f\left(\frac{a+b}{2}\right)+f(b)\right].

Implementarlo para una variable es muy sencillo :

**Program** SIMPSON\_1VAR

**IMPLICIT DOUBLE PRECISION** (A-H,O-Z)

*! definimos el intervalo de integracion*

a=0d0

b=1d0

*! hacemos la aproximacion a la integral de la*

*! funcion g(x1) definida mas abajo*

r=(b-a)/6d0\*(g(a)+4d0\*g((a+b)/2d0)+g(b))

*! abrimos un fichero excel que nos ense¤e el resultado*

**open**(20,File="ejerc10simp1var.xls")

*! escribimos el resultado en el fichero*

**write**(20,\*)r

*! cerramos el fichero*

**close**(20)

**END**

*! definimos la funcion que queremos integrar*

**FUNCTION** g(x1)

**Implicit Double Precision** (A-H,O-Z)

g=exp(-(x1\*x1))\*cos(x1\*x1)

**END**

* + Método de Simpson compuesto

Este método es el mismo que el simpson simple pero haciendo N subdivisiones . La interpretación del mismo se ha hallado en <http://es.wikipedia.org/wiki/Regla_de_simpson> :

*Se divide el intervalo [a,b] en n subintervalos iguales (con n par), de manera que*

*xi = a + ih, donde h = (b − a) / n para i = 0,1,...,n.*

*Aplicando la Regla de Simpson a cada subintervalo ,*

* [x_{j-1},x_{j+1}],\  j=1,3,5, ..., n-1, *

Tenemos : \int_a^b f(x) \, dx\approx 
\frac{h}{3}\bigg[f(x_0)+2\sum_{j=1}^{n/2-1}f(x_{2j})+
4\sum_{j=1}^{n/2}f(x_{2j-1})+f(x_n)
\bigg],

El programa fortran que nos permite hacer esto es el siguiente :

**Program** SIMPSON\_COMPUESTO\_1VAR

**IMPLICIT DOUBLE PRECISION** (A-H,O-Z)

**double precision** x(10000)

*! Definimos el intervalo de integracion a,b y el n£mero de*

*! iteraciones que vamos a hacer*

a=0d0

b=1d0

N=1000

*! calculamos h*

h=(b-a)/N

*! empezamos a contar el tiempo*

**call** cpu\_time(start)

*! construimos nuestro vector x(i)*

**do** i=1,N

x(i)=a+i\*h

**enddo**

*! ponemos los contadores de la integral a 0*

sum1=0d0

sum2=0d0

*! definimos hasta que termino hacemos la suma par y*

*! la impar*

nimp= n/2-1

np = n/2

*! calculamos la contribuci¢n del sumatorio impar*

**do** i=1,nimp

sum1=sum1+2\*g(x(2\*i))

**enddo**

*! calculamos la contribucion del sumatorio par*

**do** i=1,np

sum2=sum2+4\*g(x(2\*i-1))

**enddo**

*! Calculamos el resultado a partir de la suma de las diferentes*

*! contribuciones*

r=h/3d0\*(g(a)+sum1+sum2+g(b))

*! abrimos un fichero para escribir el resultado y lo escribimos*

**open**(20,File="ejerc10simp\_comp\_1var.xls")

**write**(20,\*)r

*! terminamos de contar el tiempo y lo escribimos*

**call** cpu\_time(finish)

**write**(20,30)finish-start

30 **Format**(e18.8)

**close**(20)

**END**

*! definimos la funcion que queremos integrar*

**FUNCTION** g(x1)

**Implicit Double Precision** (A-H,O-Z)

g=exp(-(x1\*x1))\*cos(x1\*x1)

**END**

* + Mathematica

Para hacer el cálculo con mathematica es mas rápido utilizar NIntegrate[..] :

NIntegrate[Cos[x1\*x1]\*E^(-x1^2),{x1,0,1}]

O bien : 

* n = 2

Para este caso tenemos que hacer la siguiente integral :



* + Monte -Carlo de aciertos y fallos :

El listado de programa es el siguiente:

**Program** MONTECARLO\_HIT\_AND\_MISS\_2var

**Implicit Double precision** (A-H,o-z)

**DOUBLE PRECISION** g ,a,b,c,p,r,s ,u,v,pi

**integer** n , na

a=0d0

b=1d0

c=1d0

na=0

n=10

**open**(60,File="ejerc10aciertfallos2var.xls")

**call** cpu\_time(start)

**do** j=n,10000000000

**do** 1 i=1,n

u=rand()

v=rand()

*! Introducimos una nueva variable aleatoria "z"*

z=rand()

*! vemos que g ahora es funcion de 2 variables*

**if** (g(a+(b-a)\*u,a+(b-a)\*z)**.gt.**c\*v) na=na+1

1 **continue**

p= **real** (na)/n

r=(b-a)\*c\*p

s=sqrt(p\*(1.-p)/n)\*c\*(b-a)

**write**(60,20)n,r,s

**if** (n**.gt.**1d7) **go to** 30

n=n\*10

na=0

**enddo**

30 **call** cpu\_time(finish)

**write**(60,\*)"Temps = ",finish-start ,"s"

**close**(60)

20 **Format**(I20,4X,F20.14,4X,E20.14)

**end**

*! definimos la funcion de 2 variables a integrar*

**Function** g(x1,x2)

**Implicit Double Precision** (A-H,O-Z)

g=exp(-x1\*\*2-x2\*\*2)\*cos(x2\*x1+x1\*x2)

**end**

Como podemos observar los cambios no son demasiados respecto al cálculo con 1 variable . La salida del programa es la siguiente :

|  |
| --- |
| N r s  10 0.60000000000000 0.15491933384830E+00 |
| 100 0.49000000000000 0.49989998999800E-01 |
| 1000 0.48100000000000 0.15799968354399E-01 |
| 10000 0.48790000000000 0.49985356855783E-02 |
| 100000 0.49031000000000 0.15808418766594E-02 |
| 1000000 0.49271000000000 0.49994685307540E-03 |
| 10000000 0.49297870000000 0.15809829263667E-03 |
| 100000000 0.49288799000000 0.49994941675510E-04 |
| Temps = 27.9397791 s |

* + Monte -Carlo de muestreo uniforme:

El listado de programa es el siguiente :

**Program** MONTECARLO\_HIT\_AND\_MISS\_2var

**Implicit Double precision** (A-H,o-z)

**DOUBLE PRECISION** g ,a,b,c,p,r,s ,u,v,pi

**integer** n , na

a=0d0

b=1d0

c=1d0

na=0

n=10

**open**(60,File="ejerc10aciertfallos2var.xls")

**call** cpu\_time(start)

**do** j=n,10000000000

**do** 1 i=1,n

u=rand()

v=rand()

*! Introducimos una nueva variable aleatoria "z"*

z=rand()

*! vemos que g ahora es funcion de 2 variables*

**if** (g(a+(b-a)\*u,a+(b-a)\*z)**.gt.**c\*v) na=na+1

1 **continue**

p= **real** (na)/n

r=(b-a)\*c\*p

s=sqrt(p\*(1.-p)/n)\*c\*(b-a)

**write**(60,20)n,r,s

**if** (n**.gt.**1d7) **go to** 30

n=n\*10

na=0

**enddo**

30 **call** cpu\_time(finish)

**write**(60,\*)"Temps = ",finish-start ,"s"

**close**(60)

20 **Format**(I20,4X,F20.14,4X,E20.14)

**end**

*! definimos la funcion de 2 variables a integrar*

**Function** g(x1,x2)

**Implicit Double Precision** (A-H,O-Z)

g=exp(-x1\*\*2-x2\*\*2)\*cos(x2\*x1+x1\*x2)

**end**

La salida del programa es la siguiente :

N r s

|  |
| --- |
| 10 0.62706857970012 0.73199262694129E-01 |
| 100 0.47509789711395 0.25319643468459E-01 |
| 1000 0.49552863250579 0.86737930572375E-02 |
| 10000 0.49089163964587 0.27127465102122E-02 |
| 100000 0.49038664235021 0.86195063417819E-03 |
| 1000000 0.49230574707721 0.27265572365960E-03 |
| 10000000 0.49230803554183 0.86162665514332E-04 |
| 100000000 0.49228623283936 0.27249583317120E-04 |
| Temps = 27.331375 s |

* + Simpson simple :

**Program** SIMPSON\_2VAR

**IMPLICIT DOUBLE PRECISION** (A-H,O-Z)

*! Empezamos a contar el tiempo*

**call** cpu\_time(start)

*! 1. definimos el intervalo de integracion*

a=0d0

b=1d0

c=(a+b)/2d0

*! 4. calculamos la segunda integral a partir de la funcion*

*! externa definida*

r=(b-a)/6d0\*(g2(a)+4d0\*g2(c)+g2(b))

*! 5. Abrimos un fichero excel*

**open**(20,File="ejerc10simp2var.xls")

*! 6. escribimos resultados , paramos el tiempo y lo escribimos*

**write**(20,\*)r

**call** cpu\_time(finish)

**write**(20,\*)"Temps = ",finish-start ,"s"

**close**(20)

**END**

*! 2. definimos la funcion de dos variables*

**FUNCTION** g1(x1,x2)

**Implicit Double Precision** (A-H,O-Z)

*! parameter (a,b)*

g1=exp(-(x1\*x1)-(x2\*x2))\*cos(2d0\*x1\*x2)

**END**

*! 3. Funcion que nos hace la primera integral y nos queda*

*! una integral de una variable*

**Function** g2(x2)

**IMPLICIT DOUBLE PRECISION** (A-H,O-Z)

*! parameter (a,b)*

a=0d0

b=1d0

c=(a+b)/2d0

g2=(b-a)/6d0\*(g1(a,x2)+4d0\*g1(c,x2)+g1(b,x2))

**END**

* + Simpson compuesto :

La filosofía de este programa es la misma seguida para el Simpson simple a nivel de trabajar con funciones externas e ir reduciendo variables a medida que integramos , pero haciendo las “N” subdivisiones en cada paso . Se consigue de la siguiente forma :

**Program** SIMPSON\_2VAR

**IMPLICIT DOUBLE PRECISION** (A-H,O-Z)

**DOUBLE PRECISION** X2(1000000)

a=0d0

b=1d0

N=10000

h=(b-a)/N

**do** i=1,N

*! ahora llamamos a las abcissas finales x2*

x2(i)=a+i\*h

**enddo**

sum1=0d0

sum2=0d0

nimp= n/2-1

np = n/2

*! El cambio aqui es la utilizacion de la funcion g2*

**do** i=1,nimp

sum1=sum1+2\*g2(x2(2\*i))

**enddo**

**do** i=1,np

sum2=sum2+4\*g2(x2(2\*i-1))

**enddo**

*! 3. Calculamos el resultado final en el programa principal*

r=h/3d0\*(g2(a)+sum1+sum2+g2(b))

**open**(20,File="ejerc10simp\_comp\_2var.xls")

**write**(20,\*)r

**call** cpu\_time(finish)

**write**(20,30)finish-start

30 **Format**(e18.8)

**close**(20)

**end**

*! 1. definimos la funcion de dos variables a integrar*

**FUNCTION** g1(x1,x2)

**Implicit Double Precision** (A-H,O-Z)

*! parameter (a,b)*

g1=exp(-(x1\*x1)-(x2\*x2))\*cos(2d0\*x1\*x2)

**END**

*! 2. Hacemos el primer paso de integracion reduciendo una*

*! variable , de modo analogo al programa simpson compuesto*

*! para una variable*

**Function** g2(x2)

**IMPLICIT DOUBLE PRECISION** (A-H,O-Z)

**DOUBLE PRECISION** x1(1000000)

N=10000

sum1=0d0

sum2=0d0

nimp= n/2-1

np = n/2

a=0d0

b=1d0

h=(b-a)/N

**do** i=1,N

x1(i)=a+i\*h

**enddo**

**do** i=1,nimp

sum1=sum1+2\*g1(x1(2\*i),x2)

**enddo**

**do** i=1,np

sum2=sum2+4\*g1(x1(2\*i-1),x2)

**enddo**

*! asignacion de g2*

g2=h/3d0\*(g1(a,x2)+sum1+sum2+g1(b,x2))

**END**

* + Mathematica :

El cálculo es muy directo :

NIntegrate[NIntegrate[Cos[x1\*x2\*2]\*E^(-x1^2-x2^2),{x1,0,1}],{x2,0,1}]

Si queremos graficar la funcion en 3 Dimensiones obtenemos :

Plot3D[Cos[x1\*x2\*2]\*E^(-x1^2-x2^2),{x1,0,1},{x2,0,1},AxesLabel{x,y,z}]



También podemos visualizar el gráfico despues de integrar la primera variable :

Plot[NIntegrate[Cos[x1\*x2\*2]\*E^(-x1^2-x2^2),{x1,0,1}],{x2,0,1}]



Observando que el valor del área bajo la curva es el resultado de nuestra integral.

* n = 3

En este caso debemos resolver :



* + Monte -Carlo de aciertos y fallos :
* **Program** MONTECARLO\_HIT\_AND\_MISS\_3var
* **Implicit Double precision** (A-H,o-z)
* **DOUBLE PRECISION** g ,a,b,c,p,r,s ,u,v,pi
* **integer** n , na
* *!*
* a=0d0
* b=1d0
* c=1d0
* na=0
* n=10
* **open**(60,File="ejerc10aciertfallos3var.xls")
* **call** cpu\_time(start)
* **do** j=n,10000000000
* **do** 1 i=1,n
* u=rand()
* v=rand()
* z=rand()
* *! definimos una nueva variable aleatoria*
* w=rand()
* *! ahora g es funcion de 3 variables*
* **if** (g(a+(b-a)\*u,a+(b-a)\*z,a+(b-a)\*w)**.gt.**c\*v) na=na+1
* 1 **continue**
* p= **real** (na)/n
* r=(b-a)\*c\*p
* s=sqrt(p\*(1.-p)/n)\*c\*(b-a)
* **write**(60,20)n,r,s
* **if** (n**.gt.**1d7) **go to** 30
* n=n\*10
* na=0
* **enddo**
* 30 **call** cpu\_time(finish)
* **write**(60,\*)"Temps = ",finish-start ,"s"
* **close**(60)
* 20 **Format**(I20,4X,F20.14,4X,E20.14)
* **end**
* *! definimos la funcion a integrar*
* **Function** g(x1,x2,x3)
* **Implicit Double Precision** (A-H,O-Z)
* g=exp(-x1\*\*2-x2\*\*2-x3\*\*2)\*cos(x2\*x1+x3\*x2+x3\*x1)
* **end**

Este programa tiene la siguiente salida :

N r s

|  |
| --- |
| 10 0.50000000000000 0.15811388300842E+00 |
| 100 0.33000000000000 0.47021271782035E-01 |
| 1000 0.32700000000000 0.14834790190630E-01 |
| 10000 0.33070000000000 0.47046520593982E-02 |
| 100000 0.33302000000000 0.14903612971357E-02 |
| 1000000 0.33410800000000 0.47167769115785E-03 |
| 10000000 0.33444860000000 0.14919542015693E-03 |
| 100000000 0.33436721000000 0.47176877692024E-04 |
| Temps = 29.9365919 s |

* + Monte -Carlo de muestreo uniforme:
* **Program** MONTECARLO\_MUESTREO\_UNIFORME\_3var
* **Implicit Double precision** (A-H,o-z)
* **DOUBLE PRECISION** g,g0,a,b,r1,s1,r,s,pi,u,z
* **real** finish ,start
* **integer** n
* a=0d0
* b=1d0
* n=10
* **open**(60,File="ejerc10mostrunif3var.xls")
* **call** cpu\_time(start)
* **do** j=n,10000000000
* r1=0.
* s1=0.
* **do** 1 i=1,n
* u=rand()
* v=rand()
* z=rand()
* g0=g(a+(b-a)\*u,a+(b-a)\*v,a+(b-a)\*z)
* r1=r1+g0
* s1=s1+g0\*g0
* 1 **continue**
* r1=r1/n
* s1=sqrt((s1/n-r1\*r1)/n)
* r=(b-a)\*r1
* s=(b-a)\*s1
* **write**(60,20)n,r,s
* **if** (n**.gt.**1d7) **go to** 30
* n=n\*10
* **enddo**
* 30 **call** cpu\_time(finish)
* **write**(60,\*)"Temps = ",finish-start ,"s"
* **close**(60)
* 20 **Format**(I20,4X,F20.14,4X,E20.14)
* **end**
* **Function** g(x1,x2,x3)
* **Implicit Double Precision** (A-H,O-Z)
* g=exp(-x1\*\*2-x2\*\*2-x3\*\*2)\*cos(x2\*x1+x3\*x2+x3\*x1)
* **end**

Con la siguiente salida :

N r s

|  |
| --- |
| 10 0.43062498105254 0.92767262048790E-01 |
| 100 0.32695438656329 0.25336356689296E-01 |
| 1000 0.33762788759208 0.82897591078333E-02 |
| 10000 0.32905399781140 0.24933933535245E-02 |
| 100000 0.32958392117928 0.79652401144633E-03 |
| 1000000 0.33128748784715 0.25240799896505E-03 |
| 10000000 0.33129563369469 0.79710425562943E-04 |
| 100000000 0.33123333688810 0.25210324886728E-04 |
| Temps = 29.328188 s |

* + Simpson simple :
* **Program** SIMPSON\_3VAR
* **IMPLICIT DOUBLE PRECISION** (A-H,O-Z)
* **call** cpu\_time(start)
* a=0d0
* b=1d0
* c=(a+b)/2d0
* r=(b-a)/6d0\*(g3(a)+4d0\*g3(c)+g3(b))
* **open**(20,File="ejerc10simp3var.xls")
* **write**(20,\*)r
* **call** cpu\_time(finish)
* **write**(20,\*)"Temps = ",finish-start ,"s"
* **close**(20)
* **END**
* **FUNCTION** g1(x1,x2,x3)
* **Implicit Double Precision** (A-H,O-Z)
* g1=exp(-(x1\*x1)-(x2\*x2)-(x3\*x3))\*cos(x1\*x2+x2\*x3+x1\*x3)
* **END**
* **Function** g2(x2,x3)
* **IMPLICIT DOUBLE PRECISION** (A-H,O-Z)
* a=0d0
* b=1d0
* c=(a+b)/2d0
* g2=(b-a)/6d0\*(g1(a,x2,x3)+4d0\*g1(c,x2,x3)+g1(b,x2,x3))
* **END**
* **Function** g3(x3)
* **IMPLICIT DOUBLE PRECISION** (A-H,O-Z)
* a=0d0
* b=1d0
* c=(a+b)/2d0
* g3=(b-a)/6d0\*(g2(a,x3)+4d0\*g2(c,x3)+g2(b,x3))
* **END**
  + Simpson compuesto :
* **Program** SIMPSON\_3VAR
* **IMPLICIT DOUBLE PRECISION** (A-H,O-Z)
* **DOUBLE PRECISION** X3(1000000)
* a=0d0
* b=1d0
* N=100
* h=(b-a)/N
* **do** i=1,N
* x3(i)=a+i\*h
* **enddo**
* sum1=0d0
* sum2=0d0
* nimp= n/2-1
* np = n/2
* **do** i=1,nimp
* sum1=sum1+2\*g3(x3(2\*i))
* **enddo**
* **do** i=1,np
* sum2=sum2+4\*g3(x3(2\*i-1))
* **enddo**
* r=h/3d0\*(g3(a)+sum1+sum2+g3(b))
* **open**(20,File="ejerc10simp\_comp\_3var.xls")
* **write**(20,\*)r
* **call** cpu\_time(finish)
* **write**(20,30)finish-start
* 30 **Format**(e18.8)
* **close**(20)
* **end**
* **FUNCTION** g1(x1,x2,x3)
* **Implicit Double Precision** (A-H,O-Z)
* g1=exp(-(x1\*x1)-(x2\*x2)-(x3\*x3))\*cos(x1\*x2+x2\*x3+x1\*x3)
* **END**
* **Function** g2(x2,x3)
* **IMPLICIT DOUBLE PRECISION** (A-H,O-Z)
* **DOUBLE PRECISION** x1(1000000)
* N=100
* sum1=0d0
* sum2=0d0
* nimp= n/2-1
* np = n/2
* a=0d0
* b=1d0
* h=(b-a)/N
* **do** i=1,N
* x1(i)=a+i\*h
* **enddo**
* **do** i=1,nimp
* sum1=sum1+2\*g1(x1(2\*i),x2,x3)
* **enddo**
* **do** i=1,np
* sum2=sum2+4\*g1(x1(2\*i-1),x2,x3)
* **enddo**
* g2=h/3d0\*(g1(a,x2,x3)+sum1+sum2+g1(b,x2,x3))
* **END**
* **Function** g3(x3)
* **IMPLICIT DOUBLE PRECISION** (A-H,O-Z)
* **DOUBLE PRECISION** x2(1000000)
* N=100
* sum1=0d0
* sum2=0d0
* nimp= n/2-1
* np = n/2
* a=0d0
* b=1d0
* h=(b-a)/N
* **do** i=1,N
* x2(i)=a+i\*h
* **enddo**
* **do** i=1,nimp
* sum1=sum1+2\*g2(x2(2\*i),x3)
* **enddo**
* **do** i=1,np
* sum2=sum2+4\*g2(x2(2\*i-1),x3)
* **enddo**
* g3=h/3d0\*(g2(a,x3)+sum1+sum2+g2(b,x3))
* **END**
  + Mathematica :

NIntegrate[NIntegrate[NIntegrate[Cos[x1\*x2+x1\*x3+x2\*x3]\*E^(-x1^2-x2^2-x3^2),{x1,0,1}],{x2,0,1}],{x3,0,1}]

Plot[NIntegrate[NIntegrate[Cos[x1\*x2+x1\*x3+x2\*x3]\*E^(-x1^2-x2^2-x3^2),{x1,0,1}],{x2,0,1}],{x3,0,1},AxesLabel{“x3”,”g(x3)”}]



* n = 5

La integral que debemos calcular es la siguiente :



* + Monte -Carlo de aciertos y fallos :
* **Program** MONTECARLO\_HIT\_AND\_MISS\_5var
* **Implicit Double precision** (A-H,o-z)
* **DOUBLE PRECISION** g ,a,b,c,p,r,s ,u,v,pi
* **integer** n , na
* a=0d0
* b=1d0
* c=1d0
* na=0
* n=10
* **open**(60,File="ejerc10aciertfallos5var.xls")
* **call** cpu\_time(start)
* **do** j=n,10000000000
* **do** 1 i=1,n
* u=rand()
* v=rand()
* z=rand()
* w=rand()
* y=rand()
* x=rand()
* **if** (g(a+(b-a)\*u,a+(b-a)\*z,a+(b-a)\*w,a+(b-a)\*y,a+(b-a)\*x)**.gt.**c\*v)
* & na=na+1
* 1 **continue**
* p= **real** (na)/n
* r=(b-a)\*c\*p
* s=sqrt(p\*(1.-p)/n)\*c\*(b-a)
* **write**(60,20)n,r,s
* **if** (n**.gt.**1d6) **go to** 30
* n=n\*10
* na=0
* **enddo**
* 30 **call** cpu\_time(finish)
* **write**(60,\*)"Temps = ",finish-start ,"s"
* **close**(60)
* 20 **Format**(I20,4X,F20.14,4X,E20.14)
* **end**
* **Function** g(x1,x2,x3,x4,x5)
* **Implicit Double Precision** (A-H,O-Z)
* g=exp(-x1\*\*2-x2\*\*2-x3\*\*2-x4\*\*2-x5\*\*2)\*cos(x2\*x1+x3\*x2+
* & x3\*x4+x4\*x5+x5\*x1)
* **end**

Salida :

N r s

|  |
| --- |
| 10 0.20000000000000 0.12649110640674E+00 |
| 100 0.15000000000000 0.35707142142714E-01 |
| 1000 0.13500000000000 0.10806248192597E-01 |
| 10000 0.13450000000000 0.34118873076349E-02 |
| 100000 0.13894000000000 0.10937809488193E-02 |
| 1000000 0.13927600000000 0.34623430769350E03 |
| 10000000 0.13920520000000 0.10946557097689E-03 |
| Temps = 3.3228212999999998 s |

* + Monte -Carlo de muestreo uniforme:
* **Program** MONTECARLO\_MUESTREO\_UNIFORME\_5var
* **Implicit Double precision** (A-H,o-z)
* **DOUBLE PRECISION** g,g0,a,b,r1,s1,r,s,pi,u,z
* **real** finish ,start
* **integer** n
* a=0d0
* b=1d0
* n=10
* **open**(60,File="ejerc10mostrunif5var.xls")
* **call** cpu\_time(start)
* **do** j=n,10000000000
* r1=0.
* s1=0.
* **do** 1 i=1,n
* u=rand()
* v=rand()
* z=rand()
* w=rand()
* y=rand()
* g0=g(a+(b-a)\*u,a+(b-a)\*v,a+(b-a)\*z,a+(b-a)\*w,a+(b-a)\*y)
* r1=r1+g0
* s1=s1+g0\*g0
* 1 **continue**
* r1=r1/n
* s1=sqrt((s1/n-r1\*r1)/n)
* r=(b-a)\*r1
* s=(b-a)\*s1
* **write**(60,20)n,r,s
* **if** (n**.gt.**1d6) **go to** 30
* n=n\*10
* **enddo**
* 30 **call** cpu\_time(finish)
* **write**(60,\*)"Temps = ",finish-start ,"s"
* **close**(60)
* 20 **Format**(I20,4X,F20.14,4X,E20.14)
* **end**
* **Function** g(x1,x2,x3,x4,x5)
* **Implicit Double Precision** (A-H,O-Z)
* g=exp(-x1\*\*2-x2\*\*2-x3\*\*2-x4\*\*2-x5\*\*2)\*cos(x2\*x1+x3\*x2+
* & x3\*x4+x4\*x5+x5\*x1)
* **end**

Salida :

N r s

|  |
| --- |
| 10 0.17522607898124 0.49793744103472E-01 |
| 100 0.15031867346043 0.20282692152686E-01 |
| 1000 0.13494325103640 0.58034631889643E-02 |
| 10000 0.12706645641786 0.17593107769349E-02 |
| 100000 0.12933726273468 0.56093431074361E-03 |
| 1000000 0.12968236861951 0.17789336948928E-03 |
| 10000000 0.12967208445152 0.56173709303541E-04 |
| Temps = 3.1824205 s |

* + Simpson simple :
* **Program** SIMPSON\_5VAR
* **IMPLICIT DOUBLE PRECISION** (A-H,O-Z)
* **call** cpu\_time(start)
* a=0d0
* b=1d0
* c=(a+b)/2d0
* r=(b-a)/6d0\*(g5(a)+4d0\*g5(c)+g5(b))
* **open**(20,File="ejerc10simp5var.xls")
* **write**(20,\*)r
* **call** cpu\_time(finish)
* **write**(20,\*)"Temps = ",finish-start ,"s"
* **close**(20)
* **END**
* **FUNCTION** g1(x1,x2,x3,x4,x5)
* **Implicit Double Precision** (A-H,O-Z)
* g1=exp(-x1\*\*2-x2\*\*2-x3\*\*2-x4\*\*2-x5\*\*2)\*cos(x2\*x1+x3\*x2+
* & x3\*x4+x4\*x5+x5\*x1)
* **END**
* **Function** g2(x2,x3,x4,x5)
* **IMPLICIT DOUBLE PRECISION** (A-H,O-Z)
* a=0d0
* b=1d0
* c=(a+b)/2d0
* g2=(b-a)/6d0\*(g1(a,x2,x3,x4,x5)+4d0\*g1(c,x2,x3,x4,x5)+
* & g1(b,x2,x3,x4,x5))
* **END**
* **Function** g3(x3,x4,x5)
* **IMPLICIT DOUBLE PRECISION** (A-H,O-Z)
* a=0d0
* b=1d0
* c=(a+b)/2d0
* g3=(b-a)/6d0\*(g2(a,x3,x4,x5)+4d0\*g2(c,x3,x4,x5)+g2(b,x3,x4,x5))
* **END**
* **Function** g4(x4,x5)
* **IMPLICIT DOUBLE PRECISION** (A-H,O-Z)
* a=0d0
* b=1d0
* c=(a+b)/2d0
* g4=(b-a)/6d0\*(g3(a,x4,x5)+4d0\*g3(c,x4,x5)+g3(b,x4,x5))
* **END**
* **Function** g5(x5)
* **IMPLICIT DOUBLE PRECISION** (A-H,O-Z)
* a=0d0
* b=1d0
* c=(a+b)/2d0
* g5=(b-a)/6d0\*(g4(a,x5)+4d0\*g4(c,x5)+g4(b,x5))
* **END**
  + Simpson compuesto :
* **Program** SIMPSON\_COMP\_5VAR
* **IMPLICIT DOUBLE PRECISION** (A-H,O-Z)
* **DOUBLE PRECISION** X5(1000000)
* a=0d0
* b=1d0
* N=10
* h=(b-a)/N
* **do** i=1,N
* x5(i)=a+i\*h
* **enddo**
* sum1=0d0
* sum2=0d0
* nimp= n/2-1
* np = n/2
* **do** i=1,nimp
* sum1=sum1+2\*g5(x5(2\*i))
* **enddo**
* **do** i=1,np
* sum2=sum2+4\*g5(x5(2\*i-1))
* **enddo**
* r=h/3d0\*(g5(a)+sum1+sum2+g5(b))
* **open**(20,File="ejerc10simp\_comp\_5var.xls")
* **write**(20,\*)r
* **call** cpu\_time(finish)
* **write**(20,30)finish-start
* 30 **Format**(e18.8)
* **close**(20)
* **end**
* **FUNCTION** g1(x1,x2,x3,x4,x5)
* **Implicit Double Precision** (A-H,O-Z)
* g1=exp(-(x1\*x1)-(x2\*x2)-(x3\*x3)-(x4\*x4)-(x5\*x5))\*
* & cos(x1\*x2+x2\*x3+x3\*x4+x4\*x5+x5\*x1)
* **END**
* **Function** g2(x2,x3,x4,x5)
* **IMPLICIT DOUBLE PRECISION** (A-H,O-Z)
* **DOUBLE PRECISION** x1(1000000)
* N=10
* sum1=0d0
* sum2=0d0
* nimp= n/2-1
* np = n/2
* a=0d0
* b=1d0
* h=(b-a)/N
* **do** i=1,N
* x1(i)=a+i\*h
* **enddo**
* **do** i=1,nimp
* sum1=sum1+2\*g1(x1(2\*i),x2,x3,x4,x5)
* **enddo**
* **do** i=1,np
* sum2=sum2+4\*g1(x1(2\*i-1),x2,x3,x4,x5)
* **enddo**
* g2=h/3d0\*(g1(a,x2,x3,x4,x5)+sum1+sum2+g1(b,x2,x3,x4,x5))
* **END**
* **Function** g3(x3,x4,x5)
* **IMPLICIT DOUBLE PRECISION** (A-H,O-Z)
* **DOUBLE PRECISION** x2(1000000)
* N=10
* sum1=0d0
* sum2=0d0
* nimp= n/2-1
* np = n/2
* a=0d0
* b=1d0
* h=(b-a)/N
* **do** i=1,N
* x2(i)=a+i\*h
* **enddo**
* **do** i=1,nimp
* sum1=sum1+2\*g2(x2(2\*i),x3,x4,x5)
* **enddo**
* **do** i=1,np
* sum2=sum2+4\*g2(x2(2\*i-1),x3,x4,x5)
* **enddo**
* g3=h/3d0\*(g2(a,x3,x4,x5)+sum1+sum2+g2(b,x3,x4,x5))
* **END**
* **Function** g4(x4,x5)
* **IMPLICIT DOUBLE PRECISION** (A-H,O-Z)
* **DOUBLE PRECISION** x3(1000000)
* N=10
* sum1=0d0
* sum2=0d0
* nimp= n/2-1
* np = n/2
* a=0d0
* b=1d0
* h=(b-a)/N
* **do** i=1,N
* x3(i)=a+i\*h
* **enddo**
* **do** i=1,nimp
* sum1=sum1+2\*g3(x3(2\*i),x4,x5)
* **enddo**
* **do** i=1,np
* sum2=sum2+4\*g3(x3(2\*i-1),x4,x5)
* **enddo**
* g4=h/3d0\*(g3(a,x4,x5)+sum1+sum2+g3(b,x4,x5))
* **END**
* **Function** g5(x5)
* **IMPLICIT DOUBLE PRECISION** (A-H,O-Z)
* **DOUBLE PRECISION** x4(1000000)
* N=10
* sum1=0d0
* sum2=0d0
* nimp= n/2-1
* np = n/2
* a=0d0
* b=1d0
* h=(b-a)/N
* **do** i=1,N
* x4(i)=a+i\*h
* **enddo**
* **do** i=1,nimp
* sum1=sum1+2\*g4(x4(2\*i),x5)
* **enddo**
* **do** i=1,np
* sum2=sum2+4\*g4(x4(2\*i-1),x5)
* **enddo**
* g5=h/3d0\*(g4(a,x5)+sum1+sum2+g4(b,x5))
* **END**
* + Mathematica :

Para hacer el cálculo de la integral :

NIntegrate[NIntegrate[NIntegrate[NIntegrate[NIntegrate[Cos[x1\*x2+x2\*x3+x3\*x4+x4\*x5+x5\*x1]\*Exp[-x1^2-x2^2-x3^2-x4^2x5^2],{x1,0,1}],{x2,0,1}],{x3,0,1}],{x4,0,1}],{x5,0,1}]

Si queremos ver la función g(x5) , el área bajo la curva g5(x5) es el resultado de la integral :

Plot[NIntegrate[NIntegrate[NIntegrate[NIntegrate[Cos[x1\*x2+x2\*x3+x3\*x4+x4\*x5+x5\*x1]\*Exp[-x1^2-x2^2-x3^2-x4^2-x5^2],{x1,0,1}],{x2,0,1}],{x3,0,1}],{x4,0,1}],{x5,0,1},AxesLabel{"x5","g(x5)"}]



Por último hacemos el caso n=10 :

* n = 10

La integral a calcular es la siguiente :



* + Monte -Carlo de aciertos y fallos :
* **Program** MONTECARLO\_HIT\_AND\_MISS\_10var
* **Implicit Double precision** (A-H,o-z)
* **DOUBLE PRECISION** g ,a,b,c,p,r,s ,u,v,pi
* **integer** n , na
* a=0d0
* b=1d0
* c=1d0
* na=0
* n=10
* pi=acos(-1d0)
* **open**(60,File="ejerc10aciertfallos10var.xls")
* **call** cpu\_time(start)
* **do** j=n,10000000000
* **do** 1 i=1,n
* u1=rand()
* v=rand()
* u2=rand()
* u3=rand()
* u4=rand()
* u5=rand()
* u6=rand()
* u7=rand()
* u8=rand()
* u9=rand()
* u10=rand()
* **if** (g(a+(b-a)\*u1,a+(b-a)\*u2,a+(b-a)\*u3,a+(b-a)\*u4,a+(b-a)\*u5,
* & a+(b-a)\*u6,a+(b-a)\*u7,a+(b-a)\*u8,a+(b-a)\*u9,a+(b-a)\*u10)**.gt.**c\*v)
* & na=na+1
* 1 **continue**
* p= **real** (na)/n
* r=(b-a)\*c\*p
* s=sqrt(p\*(1.-p)/n)\*c\*(b-a)
* **write**(60,20)n,r,s
* **if** (n**.gt.**1d7) **go to** 30
* n=n\*10
* na=0
* **enddo**
* 30 **call** cpu\_time(finish)
* **write**(60,\*)"Temps = ",finish-start ,"s"
* **close**(60)
* 20 **Format**(I20,4X,F20.14,4X,E20.14)
* **end**
* **Function** g(x1,x2,x3,x4,x5,x6,x7,x8,x9,x10)
* **Implicit Double Precision** (A-H,O-Z)
* g=exp(-x1\*\*2-x2\*\*2-x3\*\*2-x4\*\*2-x5\*\*2-x6\*\*2-x7\*\*2-x8\*\*2-
* & x9\*\*2-x10\*\*2)\*cos(x1\*x2+x2\*x3+x3\*x4+x4\*x5+x5\*x6+x6\*x7+x7
* & \*x8+x8\*x9+x9\*x10+x10\*x1)
* **end**

Salida :

N r s

|  |
| --- |
| 10 0.00000000000000 0.00000000000000E+00 |
| 100 0.00000000000000 0.00000000000000E+00 |
| 1000 0.00500000000000 0.22304708023195E-02 |
| 10000 0.00960000000000 0.97508153505233E-03 |
| 100000 0.00966000000000 0.30930057225941E-03 |
| 1000000 0.00988000000000 0.98905943198576E-04 |
| 10000000 0.00978940000000 0.31134494772904E-04 |
| 100000000 0.00981603000000 0.98588414912905E-05 |
| Temps = 42.8222745 s |

* + Monte -Carlo de muestreo uniforme:
* **Program** MONTECARLO\_MUESTREO\_UNIFORME\_10var
* **Implicit Double precision** (A-H,o-z)
* **DOUBLE PRECISION** g,g0,a,b,r1,s1,r,s,pi,u,z
* **real** finish ,start
* **integer** n
* a=0d0
* b=1d0
* n=10
* pi=acos(-1d0)
* **open**(60,File="ejerc10mostrunif10var.xls")
* **call** cpu\_time(start)
* **do** j=n,10000000000
* r1=0.
* s1=0.
* **do** 1 i=1,n
* u1=rand()
* u2=rand()
* u3=rand()
* u4=rand()
* u5=rand()
* u6=rand()
* u7=rand()
* u8=rand()
* u9=rand()
* u10=rand()
* g0=g(a+(b-a)\*u1,a+(b-a)\*u2,a+(b-a)\*u3,a+(b-a)\*u4,a+(b- a)\*u5,
* & a+(b-a)\*u6,a+(b-a)\*u7,a+(b-a)\*u8,a+(b-a)\*u9,a+(b-a)\*u10)
* r1=r1+g0
* s1=s1+g0\*g0
* 1 **continue**
* r1=r1/n
* s1=sqrt((s1/n-r1\*r1)/n)
* r=(b-a)\*r1
* s=(b-a)\*s1
* **write**(60,20)n,r,s
* **if** (n**.gt.**1d6) **go to** 30
* n=n\*10
* **enddo**
* 30 **call** cpu\_time(finish)
* **write**(60,\*)"Temps = ",finish-start ,"s"
* **close**(60)
* 20 **Format**(I20,4X,F20.14,4X,E20.14)
* **end**
* **Function** g(x1,x2,x3,x4,x5,x6,x7,x8,x9,x10)
* **Implicit Double Precision** (A-H,O-Z)
* g=exp(-x1\*\*2-x2\*\*2-x3\*\*2-x4\*\*2-x5\*\*2-x6\*\*2-x7\*\*2-x8\*\*2-
* & x9\*\*2-x10\*\*2)\*cos(x1\*x2+x2\*x3+x3\*x4+x4\*x5+x5\*x6+x6\*x7+x7
* & \*x8+x8\*x9+x9\*x10+x10\*x1)
* **end**
  + Simpson simple :
* **Program** SIMPSON\_10VAR
* **IMPLICIT DOUBLE PRECISION** (A-H,O-Z)
* **call** cpu\_time(start)
* a=0d0
* b=1d0
* c=(a+b)/2d0
* r=(b-a)/6d0\*(g10(a)+4d0\*g10(c)+g10(b))
* **open**(20,File="ejerc10simp10var.xls")
* **write**(20,\*)r
* **call** cpu\_time(finish)
* **write**(20,\*)"Temps = ",finish-start ,"s"
* **close**(20)
* **END**
* **FUNCTION** g1(x1,x2,x3,x4,x5,x6,x7,x8,x9,x10)
* **Implicit Double Precision** (A-H,O-Z)
* g1=exp(-x1\*\*2-x2\*\*2-x3\*\*2-x4\*\*2-x5\*\*2-x6\*\*2-x7\*\*2-x8\*\*2-
* &x9\*\*2-x10\*\*2)\*cos(x1\*x2+x2\*x3+x3\*x4+x4\*x5+x5\*x6+x6\*x7+x7
* &\*x8+x8\*x9+x9\*x10+x10\*x1)
* **END**
* **Function** g2(x2,x3,x4,x5,x6,x7,x8,x9,x10)
* **IMPLICIT DOUBLE PRECISION** (A-H,O-Z)
* a=0d0
* b=1d0
* c=(a+b)/2d0
* g2=(b-a)/6d0\*(g1(a,x2,x3,x4,x5,x6,x7,x8,x9,x10)+4d0\*
* & g1(c,x2,x3,x4,x5,x6,x7,x8,x9,x10)+
* & g1(b,x2,x3,x4,x5,x6,x7,x8,x9,x10))
* **END**
* **Function** g3(x3,x4,x5,x6,x7,x8,x9,x10)
* **IMPLICIT DOUBLE PRECISION** (A-H,O-Z)
* a=0d0
* b=1d0
* c=(a+b)/2d0
* g3=(b-a)/6d0\*(g2(a,x3,x4,x5,x6,x7,x8,x9,x10)+
* & 4d0\*g2(c,x3,x4,x5,x6,x7,x8,x9,x10)+
* & g2(b,x3,x4,x5,x6,x7,x8,x9,x10))
* **END**
* **Function** g4(x4,x5,x6,x7,x8,x9,x10)
* **IMPLICIT DOUBLE PRECISION** (A-H,O-Z)
* a=0d0
* b=1d0
* c=(a+b)/2d0
* g4=(b-a)/6d0\*(g3(a,x4,x5,x6,x7,x8,x9,x10)+
* & 4d0\*g3(c,x4,x5,x6,x7,x8,x9,x10)+g3(b,x4,x5,x6,x7,x8,x9,x10))
* **END**
* **Function** g5(x5,x6,x7,x8,x9,x10)
* **IMPLICIT DOUBLE PRECISION** (A-H,O-Z)
* a=0d0
* b=1d0
* c=(a+b)/2d0
* g5=(b-a)/6d0\*(g4(a,x5,x6,x7,x8,x9,x10)+
* & 4d0\*g4(c,x5,x6,x7,x8,x9,x10)+g4(b,x5,x6,x7,x8,x9,x10))
* **END**
* **Function** g6(x6,x7,x8,x9,x10)
* **IMPLICIT DOUBLE PRECISION** (A-H,O-Z)
* a=0d0
* b=1d0
* c=(a+b)/2d0
* g6=(b-a)/6d0\*(g5(a,x6,x7,x8,x9,x10)+
* & 4d0\*g5(c,x6,x7,x8,x9,x10)+g5(b,x6,x7,x8,x9,x10))
* **END**
* **Function** g7(x7,x8,x9,x10)
* **IMPLICIT DOUBLE PRECISION** (A-H,O-Z)
* a=0d0
* b=1d0
* c=(a+b)/2d0
* g7=(b-a)/6d0\*(g6(a,x7,x8,x9,x10)+
* & 4d0\*g6(c,x7,x8,x9,x10)+g6(b,x7,x8,x9,x10))
* **END**
* **Function** g8(x8,x9,x10)
* **IMPLICIT DOUBLE PRECISION** (A-H,O-Z)
* a=0d0
* b=1d0
* c=(a+b)/2d0
* g8=(b-a)/6d0\*(g7(a,x8,x9,x10)+
* & 4d0\*g7(c,x8,x9,x10)+g7(b,x8,x9,x10))
* **END**
* **Function** g9(x9,x10)
* **IMPLICIT DOUBLE PRECISION** (A-H,O-Z)
* a=0d0
* b=1d0
* c=(a+b)/2d0
* g9=(b-a)/6d0\*(g8(a,x9,x10)+
* & 4d0\*g8(c,x9,x10)+g8(b,x9,x10))
* **END**
* **Function** g10(x10)
* **IMPLICIT DOUBLE PRECISION** (A-H,O-Z)
* a=0d0
* b=1d0
* c=(a+b)/2d0
* g10=(b-a)/6d0\*(g9(a,x10)+
* & 4d0\*g9(c,x10)+g9(b,x10))
* **END**
  + Simpson compuesto :
* **Program** SIMPSON\_COMP\_10VAR
* **IMPLICIT DOUBLE PRECISION** (A-H,O-Z)
* **DOUBLE PRECISION** X10(1000000)
* a=0d0
* b=1d0
* N=4
* h=(b-a)/N
* **do** i=1,N
* x10(i)=a+i\*h
* **enddo**
* sum1=0d0
* sum2=0d0
* nimp= n/2-1
* np = n/2
* **do** i=1,nimp
* sum1=sum1+2\*g10(x10(2\*i))
* **enddo**
* **do** i=1,np
* sum2=sum2+4\*g10(x10(2\*i-1))
* **enddo**
* r=h/3d0\*(g10(a)+sum1+sum2+g10(b))
* **open**(20,File="ejerc10simp\_comp\_10var.xls")
* **write**(20,\*)r
* **call** cpu\_time(finish)
* **write**(20,30)finish-start
* 30 **Format**(e18.8)
* **close**(20)
* **end**
* **FUNCTION** g1(x1,x2,x3,x4,x5,x6,x7,x8,x9,x10)
* **Implicit Double Precision** (A-H,O-Z)
* g1=exp(-(x1\*x1)-(x2\*x2)-(x3\*x3)-(x4\*x4)-(x5\*x5)-(x6\*x6)-(x7\*x7)-
* &(x8\*x8)-(x9\*x9)-(x10\*x10))\*
* &cos(x1\*x2+x2\*x3+x3\*x4+x4\*x5+x5\*x6+x6\*x7+x7\*x8+x8\*x9+x9\*x10+x10\*x1)
* **END**
* **Function** g2(x2,x3,x4,x5,x6,x7,x8,x9,x10)
* **IMPLICIT DOUBLE PRECISION** (A-H,O-Z)
* **DOUBLE PRECISION** x1(1000000)
* N=4
* sum1=0d0
* sum2=0d0
* nimp= n/2-1
* np = n/2
* a=0d0
* b=1d0
* h=(b-a)/N
* **do** i=1,N
* x1(i)=a+i\*h
* **enddo**
* **do** i=1,nimp
* sum1=sum1+2\*g1(x1(2\*i),x2,x3,x4,x5,x6,x7,x8,x9,x10)
* **enddo**
* **do** i=1,np
* sum2=sum2+4\*g1(x1(2\*i-1),x2,x3,x4,x5,x6,x7,x8,x9,x10)
* **enddo**
* g2=h/3d0\*(g1(a,x2,x3,x4,x5,x6,x7,x8,x9,x10)+sum1+sum2+
* & g1(b,x2,x3,x4,x5,x6,x7,x8,x9,x10))
* **END**
* **Function** g3(x3,x4,x5,x6,x7,x8,x9,x10)
* **IMPLICIT DOUBLE PRECISION** (A-H,O-Z)
* **DOUBLE PRECISION** x2(1000000)
* N=4
* sum1=0d0
* sum2=0d0
* nimp= n/2-1
* np = n/2
* a=0d0
* b=1d0
* h=(b-a)/N
* **do** i=1,N
* x2(i)=a+i\*h
* **enddo**
* **do** i=1,nimp
* sum1=sum1+2\*g2(x2(2\*i),x3,x4,x5,x6,x7,x8,x9,x10)
* **enddo**
* **do** i=1,np
* sum2=sum2+4\*g2(x2(2\*i-1),x3,x4,x5,x6,x7,x8,x9,x10)
* **enddo**
* g3=h/3d0\*(g2(a,x3,x4,x5,x6,x7,x8,x9,x10)+sum1+sum2+
* & g2(b,x3,x4,x5,x6,x7,x8,x9,x10))
* **END**
* **Function** g4(x4,x5,x6,x7,x8,x9,x10)
* **IMPLICIT DOUBLE PRECISION** (A-H,O-Z)
* **DOUBLE PRECISION** x3(1000000)
* N=4
* sum1=0d0
* sum2=0d0
* nimp= n/2-1
* np = n/2
* a=0d0
* b=1d0
* h=(b-a)/N
* **do** i=1,N
* x3(i)=a+i\*h
* **enddo**
* **do** i=1,nimp
* sum1=sum1+2\*g3(x3(2\*i),x4,x5,x6,x7,x8,x9,x10)
* **enddo**
* **do** i=1,np
* sum2=sum2+4\*g3(x3(2\*i-1),x4,x5,x6,x7,x8,x9,x10)
* **enddo**
* g4=h/3d0\*(g3(a,x4,x5,x6,x7,x8,x9,x10)+sum1+sum2+
* & g3(b,x4,x5,x6,x7,x8,x9,x10))
* **END**
* **Function** g5(x5,x6,x7,x8,x9,x10)
* **IMPLICIT DOUBLE PRECISION** (A-H,O-Z)
* **DOUBLE PRECISION** x4(1000000)
* N=4
* sum1=0d0
* sum2=0d0
* nimp= n/2-1
* np = n/2
* a=0d0
* b=1d0
* h=(b-a)/N
* **do** i=1,N
* x4(i)=a+i\*h
* **enddo**
* **do** i=1,nimp
* sum1=sum1+2\*g4(x4(2\*i),x5,x6,x7,x8,x9,x10)
* **enddo**
* **do** i=1,np
* sum2=sum2+4\*g4(x4(2\*i-1),x5,x6,x7,x8,x9,x10)
* **enddo**
* g5=h/3d0\*(g4(a,x5,x6,x7,x8,x9,x10)
* & +sum1+sum2+g4(b,x5,x6,x7,x8,x9,x10))
* **END**
* **Function** g6(x6,x7,x8,x9,x10)
* **IMPLICIT DOUBLE PRECISION** (A-H,O-Z)
* **DOUBLE PRECISION** x5(1000000)
* N=4
* sum1=0d0
* sum2=0d0
* nimp= n/2-1
* np = n/2
* a=0d0
* b=1d0
* h=(b-a)/N
* **do** i=1,N
* x5(i)=a+i\*h
* **enddo**
* **do** i=1,nimp
* sum1=sum1+2\*g5(x5(2\*i),x6,x7,x8,x9,x10)
* **enddo**
* **do** i=1,np
* sum2=sum2+4\*g5(x5(2\*i-1),x6,x7,x8,x9,x10)
* **enddo**
* g6=h/3d0\*(g5(a,x6,x7,x8,x9,x10)
* & +sum1+sum2+g5(b,x6,x7,x8,x9,x10))
* **END**
* **Function** g7(x7,x8,x9,x10)
* **IMPLICIT DOUBLE PRECISION** (A-H,O-Z)
* **DOUBLE PRECISION** x6(1000000)
* N=4
* sum1=0d0
* sum2=0d0
* nimp= n/2-1
* np = n/2
* a=0d0
* b=1d0
* h=(b-a)/N
* **do** i=1,N
* x6(i)=a+i\*h
* **enddo**
* **do** i=1,nimp
* sum1=sum1+2\*g6(x6(2\*i),x7,x8,x9,x10)
* **enddo**
* **do** i=1,np
* sum2=sum2+4\*g6(x6(2\*i-1),x7,x8,x9,x10)
* **enddo**
* g7=h/3d0\*(g6(a,x7,x8,x9,x10)
* & +sum1+sum2+g6(b,x7,x8,x9,x10))
* **END**
* **Function** g8(x8,x9,x10)
* **IMPLICIT DOUBLE PRECISION** (A-H,O-Z)
* **DOUBLE PRECISION** x7(1000000)
* N=4
* sum1=0d0
* sum2=0d0
* nimp= n/2-1
* np = n/2
* a=0d0
* b=1d0
* h=(b-a)/N
* **do** i=1,N
* x7(i)=a+i\*h
* **enddo**
* **do** i=1,nimp
* sum1=sum1+2\*g7(x7(2\*i),x8,x9,x10)
* **enddo**
* **do** i=1,np
* sum2=sum2+4\*g7(x7(2\*i-1),x8,x9,x10)
* **enddo**
* g8=h/3d0\*(g7(a,x8,x9,x10)
* & +sum1+sum2+g7(b,x8,x9,x10))
* **END**
* **Function** g9(x9,x10)
* **IMPLICIT DOUBLE PRECISION** (A-H,O-Z)
* **DOUBLE PRECISION** x8(1000000)
* N=4
* sum1=0d0
* sum2=0d0
* nimp= n/2-1
* np = n/2
* a=0d0
* b=1d0
* h=(b-a)/N
* **do** i=1,N
* x8(i)=a+i\*h
* **enddo**
* **do** i=1,nimp
* sum1=sum1+2\*g8(x8(2\*i),x9,x10)
* **enddo**
* **do** i=1,np
* sum2=sum2+4\*g8(x8(2\*i-1),x9,x10)
* **enddo**
* g9=h/3d0\*(g8(a,x9,x10)
* & +sum1+sum2+g8(b,x9,x10))
* **END**
* **Function** g10(x10)
* **IMPLICIT DOUBLE PRECISION** (A-H,O-Z)
* **DOUBLE PRECISION** x9(1000000)
* N=4
* sum1=0d0
* sum2=0d0
* nimp= n/2-1
* np = n/2
* a=0d0
* b=1d0
* h=(b-a)/N
* **do** i=1,N
* x9(i)=a+i\*h
* **enddo**
* **do** i=1,nimp
* sum1=sum1+2\*g9(x9(2\*i),x10)
* **enddo**
* **do** i=1,np
* sum2=sum2+4\*g9(x9(2\*i-1),x10)
* **enddo**
* g10=h/3d0\*(g9(a,x10)
* & +sum1+sum2+g9(b,x10))
* **END**
  + Mathematica :

El programa al introducir la siguiente instrucción se queda pensando cierto tiempo , seguramente el tiempo necesario es demasiado alto :

NIntegrate[NIntegrate[NIntegrate[NIntegrate[NIntegrate[NIntegrate[NIntegrate[NIntegrate[NIntegrate[NIntegrate[Cos[x1\*x2+x2\*x3+x3\*x4+x4\*x5+x5\*x6+x6\*x7+x7\*x8+x8\*x9+x9\*x10+x10\*x1]\*Exp[-x1^2-x2^2-x3^2-x4^2-x5^2-x6^2-x7^2-x8^2-x9^2-x10^2],{x1,0,1}],{x2,0,1}],{x3,0,1}],{x4,0,1}],{x5,0,1}],{x6,0,1}],{x7,0,1}],{x8,0,1}],{x9,0,1}],{x10,0,1}]

…

…

Finalmente tenemos el resumen de los resultados obtenidos :

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | **1VAR (N=10\*\*8)** | **2VAR (N=10\*\*8)** |
| **SIMPSON COMPUESTA** | | 0.69871273 (N=1000 , t= 0.00..s ) | 0.4922686 (N=10000, t = 20.2s) |
| **SIMPSON SIMPLE** | | 0.702854 | 0.490687 |
| **ACIERTOS Y FALLOS** | | 0.69875±0.00004 (t=24.2s) | 0.49289±0.00005 (t=27.8s) |
| **MUESTREO UNIFORME** | | 0.69875±0.00002 (t=24.7s) | 0.49229±0.00003 (t=27.2s) |
| **Mathematica 5.2** | | 0.698713 | 0.492269 |

|  |  |  |
| --- | --- | --- |
|  | **3VAR (N=10\*\*8)** | **5VAR ( N=10\*\*7)** |
| **SIMPSON COMPUESTA** | 0.331235 (t=0.2s , N=100 ) | 0.129629 (t=0.14s , N=10) |
| **SIMPSON SIMPLE** | 0.327930 (t=0.01s) | 0.12466 |
| **ACIERTOS Y FALLOS** | 0.33437±0.00005 (t=29.8s) | 0.1392 ± 0.0001 (t=3.32s) |
| **MUESTREO UNIFORME** | 0.33123±0.00002 (t=28.7s) | 0.1297±0.00006 (t=3.186s) |
| **Mathematica 5.2** | 0.331236 | 0.129633 |

|  |  |
| --- | --- |
|  | **10VAR (N=10\*\*7)** |
| **SIMPSON COMPUESTA** | -0.0069836 (t=16.5s , N=4) |
| **SIMPSON SIMPLE** | -0.0097485 (t=0.01s) |
| **ACIERTOS Y FALLOS** | 0.0098±0.0001 (t=4.2s) |
| **MUESTREO UNIFORME** | -0.00688±0.00001 (t=4.4s) |
| **Mathematica 5.2** | … |

Nota : Los N=10\*\*7 y N=10\*\*8 presentes encima de las tablas al lado del Nº de variables , indican el número de iteraciones hechas para los métodos Monte-Carlo.

**Conclusiones :**

Como podemos ver es mejor el método de Simpson compuesto es mejor que los otros pero eligiendo un numero de subdivisiones ideal . A más variables podemos concluir que los métodos Monte-Carlo son mejores, ya que llega un momento en que las subdivisiones se hacen muy pequeñas con el fin de garantizar un tiempo de cálculo no muy grande y con ello se va perdiendo precisión en el cálculo. Además , la extrapolación a “N” variables resulta mucho más simple que el algoritmo de Simpson .