Práctica 5

En la quinta práctica usaremos dos métodos de integración numérica, el de Simpson y el de Montecarlo. En el segundo ejercicio trabajaremos con ecuaciones diferenciales.

Enunciado: Ej.1

Ejercicio 1: Dos métodos de integración numérica: uno determinista y otro estocástico

Se considera una sustancia cuya densidad viene dada por d(x) siendo x la coordenada espacial. La masa total en cierto intervalo [A,B] está dada por

$$M_{AB} = \int_A^B d(x)dx$$

Se desea realizar un programa en R para resolver dicha integral mediante una fórmula de Simpson compuesta, cuya expresión viene dada por:

Solo poner esta fórmula en la function Simpson (A,B,n)

$$\int_A^B d(x)dx \approx \frac{h}{6}\left(d(A) + 2\sum_{i=2}^{n-1} d(T_i) + 4\sum_{i=1}^{n-1} d\left(\frac{T_i + T_{i+1}}{2}\right) + d(B)\right)$$

Donde se ha realizado una subdivisión del intervalo [A,B] en (n-1) subintervalos iguales, es decir, considerando n puntos T_i, y siendo h la longitud de cada subintervalo.

El programa se ejecutará con los siguientes datos: A=0, B=3.05, densidad dada por d(x)=exp(x)*sin(x).
Como número de puntos para el desarrollo de la fórmula se tomará n=35.

Como dice el enunciado, debemos resolver la integral mediante la fórmula de Simpson compuesta.

Lo primero que tendremos que hacer es definir la función.

Programar la función "d"

La función "d" va a ser nuestra función densidad que vamos a integrar. Según el enunciado d=(e^x)*sinx

```
d=function(x){
exp(x)*sin(x)
}
```

Indicamos que "d" es una función que depende de "x" y escribimos su expresión.

Fórmula de Simpson compuesta

Para ello vamos a crear una función llamada Simpson. La función va a incorporar en su expresión la fórmula de Simpson compuesta.

$$\int_A^B d(x)dx \approx \frac{h}{6}\left(d(A) + 2\sum_{i=2}^{n-1} d(T_i) + 4\sum_{i=1}^{n-1} d\left(\frac{T_i + T_{i+1}}{2}\right) + d(B)\right)$$

```
Simpson=function(a,b,n){
      T=seq(a,b,length=n)
      h=(b-a)/(n-1)
      suma2=0
      for (i in 1:(n-1)){
            suma2=suma2+d((T[i]+T[i+1])/2)
      }
      suma1=0
      for (i in 2:(n-1)){
            suma1=suma1+d(T[i])
      }
      Masa=h*(d(a)+2*suma1+4*suma2+d(b))/6
return(Masa)
}
```

Como vemos, la función va a depender de tres factores, "a" y "b" que son los valores entre los que se integra, y "n" que es el número de puntos para el desarrollo de la fórmula.

T va a ser un vector donde va a haber "n" puntos equidistantes en el intervalo [a,b], y "h" la longitud de cada subintervalo.

Hacemos los 2 sumatorios en bucles *for* y luego igualamos la "Masa" a la fórmula de Simpson compuesta.

Ejecutar las funciones

Una vez creadas las funciones, les asignamos los valores que nos da el enunciado y llamamos a las funciones de vuelta.

Además, calcularemos el valor exacto de la integral usando el comando "integrate" (pondremos: integrate(d,A,B)).

```
options(digits=18)   #Para que opere con 18 digitos
A=0; B=3.05; n=35; h=(B-A)/(n-1)
T=seq(A,B,length=n)
Masa=Simpson(A,B,n)
Masa
Vex=integrate(d,A,B)
Vex
```

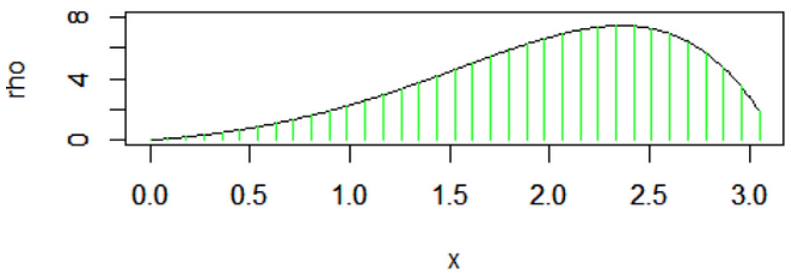
También se puede calcular el error que se comete, comparando el valor exacto con la aproximación.

Representar gráficamente

Lo primero que vamos a hacer antes de representar es crear un vector x que contenga puntos a intervalos iguales en el intervalo [A,B].

Después, representamos con le comando "plot" la función "d" y el vector T que va a indicar los valores que le corresponden a cada punto.

```
par(mfrow=c(2,1))
x=seq(A,B,0.01)
plot(x,d(x),xlim=c(A,B),ylim=c(0,8),xlab='x',ylab='rho',type='l')
par(new='TRUE')
plot(T,d(T),xlim=c(A,B),ylim=c(0,8),type='h',col='green',xlab='',ylab='')
```



Método de Montecarlo

El método de Montecarlo consiste en calcular una integral definiendo un rectángulo que contenga el área que se quiere obtener.

Se generan puntos aleatorios, que pueden estar dentro de ese área o fuera. Entonces, se calcula la fracción entre la cantidad de puntos que hay dentro del área y el total de puntos (es decir, la cantidad total de puntos de ese rectángulo).

El programa se ejecutará con los siguientes datos: A=0, B=3.05, densidad dada por d(x)=exp(x)*sin(x).
Como número de puntos para el desarrollo de la fórmula se tomará n=35.

Se usarán los mismos datos que con el ejercicio de Simpson.

Proceso

De nuevo, definimos la función "d" y empezamos con el proceso.

```
d=function(x){
exp(x)*sin(x)
}
num_puntos= 1000
AA=0
BB=7.4
A=0
B=3.05
ss=runif(num_puntos,A,B)
ff=runif(num_puntos,AA,BB)
puntos_dentro=0
puntos_fuera=0
jacinto=0
for (i in 1:num_puntos){
      if (ff[i]<=d(ss[i])){
            puntos_dentro=puntos_dentro+1
            jacinto[i]='blue'
      }else{
            puntos_fuera=puntos_fuera+1
            jacinto[i]='orange'
      }
}
Area=puntos_dentro/num_puntos*B*BB
Area
```

Después de definir la función, creamos una variable que indique cuantos puntos aleatorios queremos (num_puntos).

Después, definimos el intervalo de integración (A,B) y el intervalo en el que se va a situar el rectángulo que va a contener nuestro área (AA,BB).

El vector **ss** va a crear tantos puntos aleatorios como hayamos decidido en num_puntos en el intervalo [A, B].

El vector **ff** va a hacer lo mismo en el intervalo [AA,BB].

Después creamos otras tres variables: puntos_dentro, puntos_fuera y jacinto. En puntos_dentro se acumulan cuantos puntos están debajo de la función (en el área a analizar). En puntos_fuera se acumulan los puntos que están fuera del área. **Jacinto** va a ser un vector que va a indicar que si el punto se encuentra dentro su color va a ser azul, si está fuera, naranja.

Para hacerlo, usamos la estructura *if.* Este indica que si los puntos en el intervalo [AA,BB] están por debajo de la función, el punto está dentro y se considerará como tal, si el punto está fuera, se acumulará en la variable puntos_fuera.

Por último, calculamos el área.

Representar gráficamente

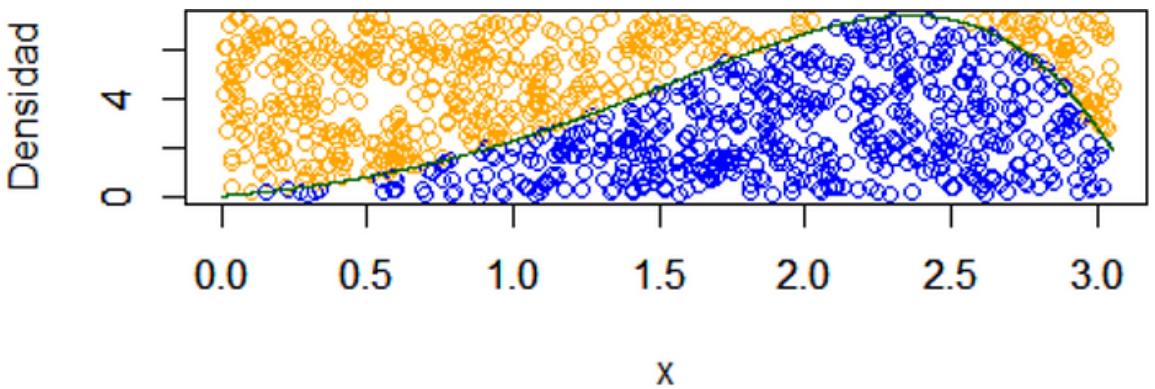
Vamos a representar conjuntamente ambas zonas, la que está fuera del área y la que está dentro.

```
plot(ss,ff,xlim=c(A,B),ylim=c(AA,BB),col=jacinto,ylab='Densidad',xlab='x')
par(new='TRUE')
ss=seq(A,B,0.001)

plot(ss,d(ss),type='l',col='dark green',xlim=c(A,B),
ylim=c(AA,BB),xlab='',ylab='')
```

Como vemos, dibujamos **ss** y **ff.** El color que se use será el que se haya acumulado en el vector **jacinto**.

El comando de abajo nos permite dibujar la función, ya que d(ss) otorga un valor de la función a cada punto de **ss**.

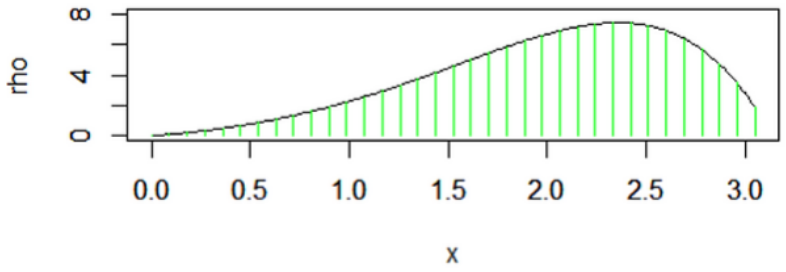


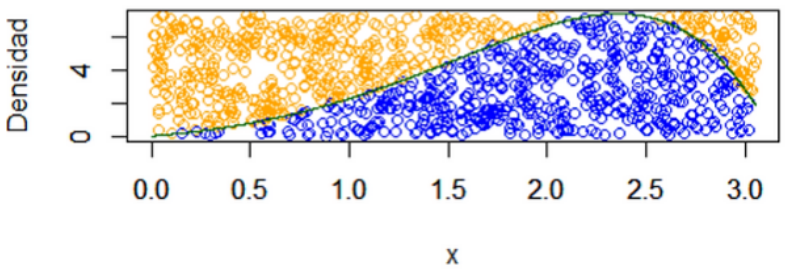
Comparación métodos

Ahora podemos comparar como son las gráficas de cada método o la exactitud de los mismos. Para ello, calculamos el valor exacto de la integral usando integrate y se lo asignaremos a la variable Vexact. Luego calculamos el error con cada una.

```
> #INTRODUCIR EL VALOR EXACTO DE LA INTEGRAL
> Vexact=11.97907159
> Vexact
[1] 11.9790715900000002
>
> #OBTENER LOS ERRORES COMETIDOS CON LA FÓRMULA DE SIMPSON COMPUESTA
> #Y EL MÉTODO DE MONTECARLO
> ErrorM=(Area-Vexact)/Vexact
> ErrorS=(Masa-Vexact)/Vexact
> ErrorM
[1] -0.0146056051744489802
> ErrorS
[1] -9.0197080204539698e-08
>
> #ESCRIBIR RESULTADOS EN FORMA DE TABLA (ESTRUCTURA data.frame)
> Metodo=c("Valor exacto","Simpson","Montecarlo")
> Valores=c(Vexact,Masa,Area)
> Error_Relativo=c(0,round(ErrorS,7),round(ErrorM,7))
> data.frame(Metodo,Valores,Error_Relativo)
        Metodo             Valores           Error_Relativo
1 Valor exacto 11.9790715900000002  0.0000000000000000e+00
2      Simpson 11.9790705095227192 -9.9999999999999955e-08
3   Montecarlo 11.8041100000000014 -1.4605599999999998e-02
```

Las gráficas quedarían de la siguiente manera.





Ecuación diferencial: Ej.2

Ejercicio 2:

Consideramos el problema:

(1) $$\begin{cases} y'(t) = ry(t).\left(1-\frac{y(t)}{K}\right), & t \in (0,3] \\ y(0) = 2 \end{cases}$$

donde la ecuación diferencial se denomina *ecuación logística*.
La variable t representa el tiempo, mientras que la variable y(t) representa la densidad de la población.
En la ecuación aparecen también las constantes positivas r, K.
El problema dado tiene una solución analítica (exacta) dada por la expresión:

$$y(t) = \frac{2\,K}{2 + e^{(-r\,t)}\,K - 2\,e^{(-r\,t)}}$$

Realiza un programa en R en el que:
a) Se genere un vector t de 20 componentes que son puntos equidistantes en [0,3].
b) Se asigne y_1=2 y w_1=2 (tanto y como w serán vectores de 20 componentes).
c) Se resuelva el problema (1) generando la sucesión (método de Euler)

$$y_{i+1} = y_i + hf(r,K,t_i,y_i), \quad (i = 1,\ldots,(N-1)) \qquad \text{siendo} \quad f(r,K,t,y) = r.y.\left(1-\frac{y}{K}\right)$$

d) Se resuelva nuevamente el problema (1) generando la sucesión (método de Heun):

$$z = w_i + hf(r,K,t_i,w_i)$$

$$w_{i+1} = w_i + \frac{h}{2}(f(r,K,t_i,w_i) + f(r,K,t_{i+1},z)), \quad (i = 1,\ldots,(N-1))$$

empleando, en ambos casos, los valores de las constantes: r=2, K=1, N=20.
e) Se represente, en un mismo gráfico, la soluciones: exacta (*) en línea continua verde oscuro, la solución del apartado c) con símbolos azules y la solución del apartado d) con símbolos rojos usando pch=18

Definir función

Definimos la ecuación diferencial como una función

```
f=function(r,K,y,t){
        r*y*(1-y)/K
}
```

Resolver el problema

El problema lo vamos a resolver de dos maneras, el método de Euler y el de Heun. Para el de Euler se sigue que:

$$y_{i+1} = y_i + hf(r,K,t_i,y_i), \quad (i = 1,\ldots,(N-1)) \qquad \text{siendo} \quad f(r,K,t,y) = r.y.\left(1 - \frac{y}{K}\right)$$

El de Heun seguiría lo siguiente:

$$z = w_i + hf(r,K,t_i,w_i)$$

$$w_{i+1} = w_i + \frac{h}{2}\left(f(r,K,t_i,w_i) + f(r,K,t_{i+1},z)\right), \quad (i = 1,\ldots,(N-1))$$

```
r=2;K=1;N=20
a=0; b=3; h=(b-a)/N
y=c(0)
w=c(0)
y[1]=2; w[1]=2
t=seq(a,b,length=N)
for (i in 1:(N-1)){
      y[i+1]=y[i]+h*f(r,K,y[i],t[i])
      z=w[i]+ h*f(r,K,y[i],t[i])
      w[i+1]=w[i]+h/2*(f(r,K,w[i],t[i])+f(r,K,z,t[i+1]))

}
```

Para ello, introducimos los datos de problema y creamos un vector **t** que tenga N puntos equidistantes en el intervalo [a,b].

Después elaboramos un bucle *for* para realizar las fórmulas de los respectivos métodos.

Representación

Queremos representar la solución exacta, la solución siguiendo el método de Euler y la solución del método de Heun en una misma gráfica.

Primero representamos los métodos de Euler y Heun.

```
plot(t,y,col='blue',xlab='',ylab='')
par(new='TRUE')
plot(t,w,col='red',xlab='',ylab='')
```

Para la representación de la solución exacta debemos crear un vector tt que tenga 10000 puntos equidistantes en [a,b]. Después, seguimos la fórmula que nos da el enunciado.

$$y(t) = \frac{2K}{2 + e^{(-rt)}K - 2e^{(-rt)}}$$

```
#EXACTA
Vex=c(0)
tt=seq(a,b,length=10000)
for (i in 1:10000){
      Vex[i]=2*K/(2+exp(-r*tt[i])*K-2*exp(-r*tt[i]))
}
par(new='TRUE')
plot(tt,Vex,type='l',col='dark green')
```