Área entre las curvas

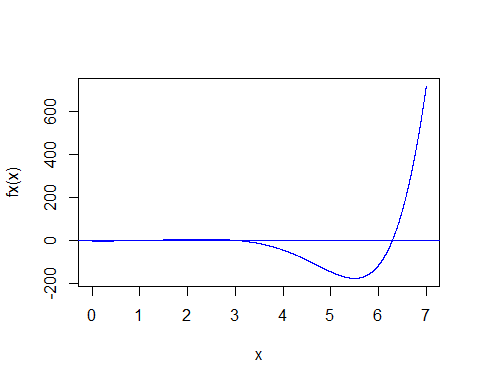
Andres David Gomez B

11 de abril de 2018

## taller

El siguiente ejercicio se ejecutará por pasos para hacer mas comprensible pero se puede integrar en una sola función. Primero se hallan los puntos de corte entre ambas funciones:

rm(list=ls())  
y1=function(x) (exp(x))\*(sin(x))  
y2=function(x) 4+cos(x+1)  
  
  
# programa busca intervalos donde es posible encontrar raices  
  
fx<- function(x) y1(x)-y2(x)  
  
x<-seq(0,7,0.001)  
plot(x,fx(x),type="l",col="blue")  
abline(h=0,col="blue")



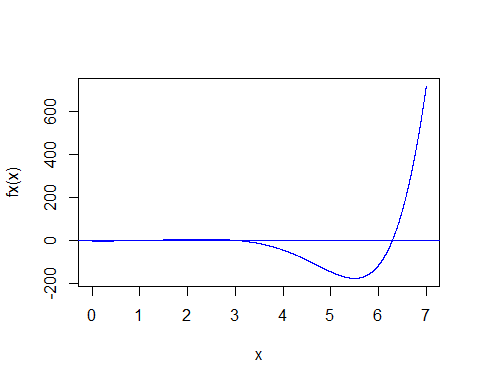
buscador <- function(a,b) {  
 i<-a  
 j<-a  
 x<-fx(a)  
 y<-fx(i)  
 while (b > i) {  
 y<-fx(i)  
 if (x\*y < 0)   
 {  
 cat("intervalo [",j," ",i,"]","\n")   
 x<-fx(i)  
 j<-i  
 }  
 i<-i+1  
 }  
}  
buscador(0,100)

## intervalo [ 0 2 ]   
## intervalo [ 2 3 ]   
## intervalo [ 3 7 ]   
## intervalo [ 7 10 ]   
## intervalo [ 10 13 ]   
## intervalo [ 13 16 ]   
## intervalo [ 16 19 ]   
## intervalo [ 19 22 ]   
## intervalo [ 22 26 ]   
## intervalo [ 26 29 ]   
## intervalo [ 29 32 ]   
## intervalo [ 32 35 ]   
## intervalo [ 35 38 ]   
## intervalo [ 38 41 ]   
## intervalo [ 41 44 ]   
## intervalo [ 44 48 ]   
## intervalo [ 48 51 ]   
## intervalo [ 51 54 ]   
## intervalo [ 54 57 ]   
## intervalo [ 57 60 ]   
## intervalo [ 60 63 ]   
## intervalo [ 63 66 ]   
## intervalo [ 66 70 ]   
## intervalo [ 70 73 ]   
## intervalo [ 73 76 ]   
## intervalo [ 76 79 ]   
## intervalo [ 79 82 ]   
## intervalo [ 82 85 ]   
## intervalo [ 85 88 ]   
## intervalo [ 88 92 ]   
## intervalo [ 92 95 ]   
## intervalo [ 95 98 ]

Como lo indica la función buscador existen multiples raíces por lo que la solución de penderá de los limites que el usuario desee. En este caso, por comodidad, se evalua el intervalo de 0 a 10.

Se modifica “buscador”" para que entregue todos los intervalos de existencia presentes entre los limtes

rm(list=ls())  
y1=function(x) (exp(x))\*(sin(x))  
y2=function(x) 4+cos(x+1)  
  
  
# programa busca intervalos donde es posible encontrar raices  
  
fx<- function(x) y1(x)-y2(x)  
  
x<-seq(0,7,0.001)  
plot(x,fx(x),type="l",col="blue")  
abline(h=0,col="blue")

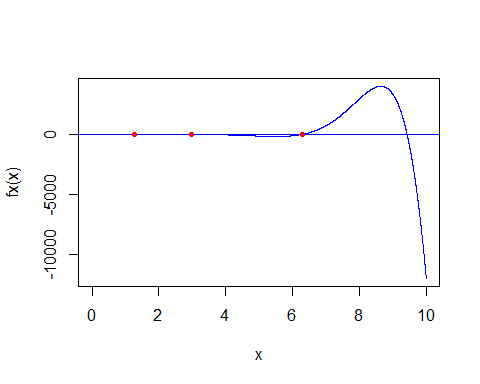


buscador <- function(a,b) {  
 i<-a  
 j<-a  
 x<-fx(a)  
 y<-fx(i)  
 resx=0  
 resy=0  
 k=1  
 while (b > i) {  
 y<-fx(i)  
 if (x\*y < 0)   
 {  
 cat("intervalo [",j," ",i,"]","\n")   
 resx[c(k)]=j  
 resy[c(k)]=i  
 k=k+1  
 x<-fx(i)  
 j<-i  
 }  
 i<-i+1  
 }  
 M=matrix(0,nrow=NROW(resx),ncol=2)  
 M[,1]=resx  
 M[,2]=resy  
 return(M)  
}  
  
M=buscador(0,10)

## intervalo [ 0 2 ]   
## intervalo [ 2 3 ]   
## intervalo [ 3 7 ]

Con los intervalos hallados se usa el metodo de newton, se elige por tener convergencia cuadratica, para hallar las raices de la función. El punto inicial es el 80 por ciento de la distancia entre los dos puntos para garantizar que las raices no se repitan y se maneja 1.e-5 como cota de error

rm(list=ls())  
y1=function(x) (exp(x))\*(sin(x))  
y2=function(x) 4+cos(x+1)  
  
  
# programa busca intervalos donde es posible encontrar raices  
  
fx<- function(x) y1(x)-y2(x)  
F1x = function(x) ((exp(x))\*(sin(x)))+((exp(x))\*(cos(x)))+(sin(x+1))  
  
x<-seq(0,10,0.001)  
plot(x,fx(x),type="l",col="blue")  
abline(h=0,col="blue")  
  
 # Metodo de Newton  
 # Halla la raiz de   
 Newton <- function(x0) {  
 x<-x0-(fx(x0)/F1x(x0))  
 error <-1  
 while (error > 1.e-5) {  
 x<-x-(fx(x)/F1x(x))  
 if (fx(x) == 0) break  
 error<-abs(fx(x)/F1x(x))  
 points(rbind(c(x,0)),pch=19,cex=0.7,col="red")  
 #cat("X=",x,"\t","E=",error,"\n")  
 }  
 cat("X=",x,"\t","E=",error,"\n")  
 return(x)  
 }  
  
  
M=matrix(c(0,2,3,2,3,7),nrow=3,ncol=2)  
  
i=1  
tam=NROW(M)  
#calcula la raiz en los intervalo de existencia encontrados y se guardan en res  
res=0  
while(i<=tam)  
{  
res[c(i)]=Newton(M[i,1]+0.8\*(M[i,2]-M[i,1]))  
 i=i+1  
}

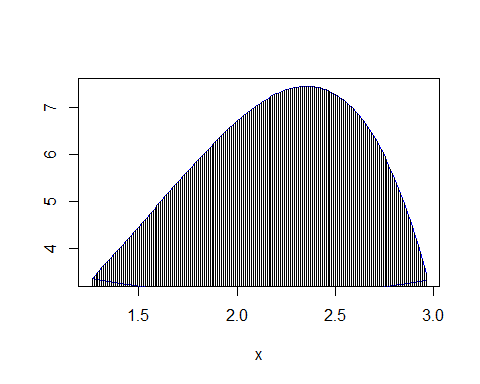


## X= 1.261451 E= 1.927353e-07   
## X= 2.970267 E= 6.696164e-06   
## X= 6.29158 E= 7.536476e-09

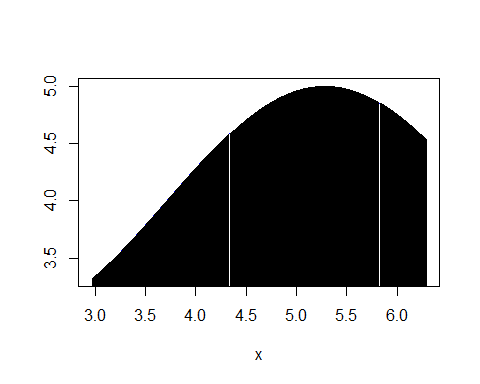
Se calcula el área entre las curvas usando la regla del trapecio.Para que se pueda ver la gráfica mas claramente se grafica una sección de ella y se divide en 100 particiones de manera que también se pueda hallar una buena aproximación. Es posible que si se visualiza en word la resolución sea muy mala.

rm(list=ls())  
y1=function(x) (exp(x))\*(sin(x))  
y2=function(x) 4+cos(x+1)  
  
fx<- function(x) y1(x)-y2(x)  
gx=function(x) y2(x)-y1(x)  
F1x <- function(x) (2\*exp(x)\*cos(x))+cos(x+1)  
  
  
integracion=function(fx,a,b,n){  
   
 particion=abs(a-b)/n  
 i=1  
 sum=0  
 b=a+particion  
 while(i<n)  
 {  
 diferencia=b-a  
 base=(fx(a)+fx(b))/2  
 res=base\*diferencia  
 sum=sum+res  
 i=i+1  
 a=b  
 b=b+particion  
 }  
   
 return(sum)  
}  
  
  
  
M=matrix(c(0,2,3,2,3,7),nrow=3,ncol=2)  
tam=NROW(M)  
res=(c(1.261451,2.970267,6.29158))  
i=1  
sum=0  
while(i<=tam-1)  
{  
 a=res[i]+((res[i+1]-res[i])/2)  
 if(y2(a)<y1(a))  
 {  
 aux=integracion(fx,res[i],res[i+1],100)  
 cat("integral ",i," ",aux,"\n")  
 sum=sum+aux  
 x<-seq(res[i],res[i+1],0.01)  
 plot(x,y1(x),type="l",col="blue",ylab =" ")  
 abline(h=0,b=0,col="black")  
 par(new=TRUE)  
 points(x,y2(x),type="l",col="blue")  
 lines(x, y1(x), type="h")   
 lines(x, y1(x), type="c")   
 }  
 else  
 {  
 aux=integracion(gx,res[i],res[i+1],100)  
 cat("integral ",i," ",aux,"\n")  
 sum=sum+aux  
   
 x<-seq(res[i],res[i+1],0.01)  
 plot(x,y2(x),type="l",col="blue",ylab =" ")  
 abline(h=0,b=0,col="black")  
 par(new=TRUE)  
 points(x,y1(x),type="l",col="blue")  
 lines(x, y2(x), type="h")   
 lines(x, y2(x), type="c")  
 }  
   
 i=i+1  
}

## integral 1 4.793485



## integral 2 293.5203



cat("El resultado de la integración es ",sum,"\n")

## El resultado de la integración es 298.3138