taller\_interpolacion

Andres David Gomez B

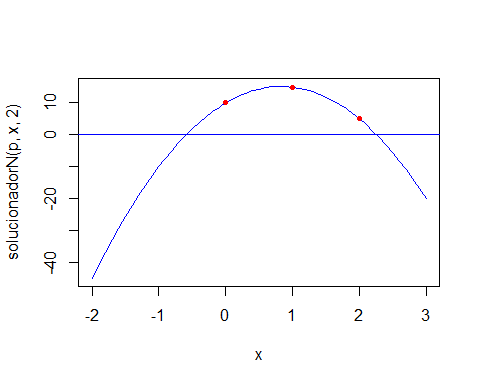
5 de abril de 2018

## punto 1:

rm(list=ls())  
  
interNewton<-function(M){  
 res<-0  
 k<-1  
 res<-M  
 tam<-NROW(res)  
 temp<-0  
 temp[c(1)]<-res[1,2]  
 while(k<tam)  
 {  
 k<-k+1  
 j<-k  
 i<-1  
 while((j-1)<=(tam-1))  
 {  
 #up<-(res[j,2]-res[j-1,2])  
 #down<-(res[j,1]-res[i,1])  
 temp[c(j)]<-(res[j,2]-res[j-1,2])/(res[j,1]-res[i,1])  
 #temp[c(j)]<-up/down  
 i<-i+1  
 j<-j+1  
   
 }  
 #cat("" solución: ",temp,"\n")  
 res[,2]<-temp  
 }  
 return(res)  
}  
  
solucionadorN=function(p,x,n){  
 i=2;  
 j=1;  
 if((0<=n)&&(n<=(NROW(p))))  
 {  
 acum=p[1,2]  
 while(i<=n+1){  
 multi=1  
 while(j<i){  
 multi=multi\*(x-p[j,1])  
 j=j+1;  
 }  
 multi=multi\*p[i,2]  
 i=i+1;  
 j=1  
 acum=acum+multi  
 }  
 return(acum)  
 }  
 else cat("Grado no posible","\n")  
}  
  
M<-matrix(c(0,1,2,10,15,5),ncol=2,nrow=3)  
p<-interNewton(M)  
x<-seq(-2,3,0.1)  
#plot(x,p5(x,p),type="l",col="blue")  
#abline(h=0,col="blue")  
#points(rbind(M),pch=19,cex=0.7,col="red")  
cat(solucionadorN(p,0,2))

## 10

plot(x,solucionadorN(p,x,2),type="l",col="blue")  
abline(h=0,col="blue")  
points(rbind(M),pch=19,cex=0.7,col="red")

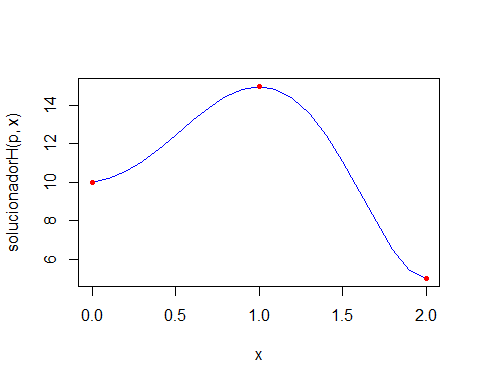


método de hermite: Debido a que la función necesita la primera derivada de los puntos iniciales y no se interesan las rectas tangentes de los otros dos puntos estas se consideran nulas.

rm(list=ls())  
hermite=function(M){  
 tam=NROW(M)  
 puntos=matrix(0,nrow=2\*tam,ncol=2)  
 res=matrix(0,nrow=2\*tam)  
 i=1  
 while(i<=tam){  
 puntos[(2\*i)-1,1]=M[i,1]  
 puntos[(2\*i),1]=M[i,1]  
 res[c(2\*i)]=M[i,3]  
 i=i+1  
 }  
 i=1  
 res[c(1)]=M[1,2]  
 while(i<tam){  
 res[c((2\*i)+1)]=(M[i+1,2]-M[i,2])/(M[i+1,1]-M[i,1])  
 i=i+1  
 }  
 #union  
 tam=tam\*2  
 k<-2  
 temp<-0  
 temp=res  
 while(k<tam)  
 {  
 k<-k+1  
 j=k  
 i<-1  
 while((j)<=(tam))  
 {  
 #a=res[j]  
 #b=res[j-1]   
 #up<-(a-b)  
 #c=puntos[j]  
 #d=puntos[i]   
 #down<-(c-d)  
 temp[c(j)]<-(res[j]-res[j-1])/(puntos[j,1]-puntos[i,1])  
 #temp[c(j)]<-up/down  
 i<-i+1  
 j<-j+1  
   
 }  
 #cat("" solución: ",temp,"\n")  
 res<-temp  
 }  
   
 puntos[,2]=res  
 return(puntos)  
   
}  
  
solucionadorH=function(p,x){  
 i=2;  
 j=1;  
 tam=NROW(p)  
 acum=p[1,2]  
 while(i<=(tam)){  
 multi=1  
 while(j<i){  
 multi=multi\*(x-p[j,1])  
 j=j+1;  
 }  
 multi=multi\*p[i,2]  
 i=i+1;  
 j=1  
 acum=acum+multi  
 }  
 return(acum)  
   
}  
  
M=matrix(c(0,1,2,10,15,5,1,0,0),ncol=3,nrow=3)   
p=hermite(M)  
  
x<-seq(0,2,0.1)  
cat(solucionadorH(p,1.5))

## 11.07812

plot(x,solucionadorH(p,x),type="l",col="blue")  
abline(h=0,col="blue")  
points(rbind(M),pch=19,cex=0.7,col="red")

 La interpolación de hermite permite interpolar los puntos a su vez que sus rspectivas derivadas, entonces es muchisimo mas eficiente ya que puede ejercer ambas tareas en paralelo.

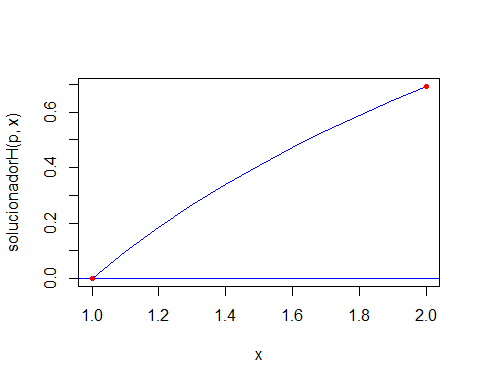
## punto 2

1. Dado que se conoce la función en ambos puntos sellena la matriz M con la evaluación de la función en esos puntos y sus respectivas derivadas.

rm(list=ls())  
hermite=function(M){  
 tam=NROW(M)  
 puntos=matrix(0,nrow=2\*tam,ncol=2)  
 res=matrix(0,nrow=2\*tam)  
 i=1  
 while(i<=tam){  
 puntos[(2\*i)-1,1]=M[i,1]  
 puntos[(2\*i),1]=M[i,1]  
 res[c(2\*i)]=M[i,3]  
 i=i+1  
 }  
 i=1  
 res[c(1)]=M[1,2]  
 while(i<tam){  
 res[c((2\*i)+1)]=(M[i+1,2]-M[i,2])/(M[i+1,1]-M[i,1])  
 i=i+1  
 }  
 #union  
 tam=tam\*2  
 k<-2  
 temp<-0  
 temp=res  
 while(k<tam)  
 {  
 k<-k+1  
 j=k  
 i<-1  
 while((j)<=(tam))  
 {  
 #a=res[j]  
 #b=res[j-1]   
 #up<-(a-b)  
 #c=puntos[j]  
 #d=puntos[i]   
 #down<-(c-d)  
 temp[c(j)]<-(res[j]-res[j-1])/(puntos[j,1]-puntos[i,1])  
 #temp[c(j)]<-up/down  
 i<-i+1  
 j<-j+1  
   
 }  
 #cat("" solución: ",temp,"\n")  
 res<-temp  
 }  
   
 puntos[,2]=res  
 return(puntos)  
   
}  
  
solucionadorH=function(p,x){  
 i=2;  
 j=1;  
 tam=NROW(p)  
 acum=p[1,2]  
 while(i<=(tam)){  
 multi=1  
 while(j<i){  
 multi=multi\*(x-p[j,1])  
 j=j+1;  
 }  
 multi=multi\*p[i,2]  
 i=i+1;  
 j=1  
 acum=acum+multi  
 }  
 return(acum)  
   
}  
  
  
M=matrix(c(1,2,0,0.69315,1,1/2),ncol=3,nrow=2)  
p=hermite(M)  
  
x<-seq(1,2,0.1)  
cat(solucionadorH(p,1.5))

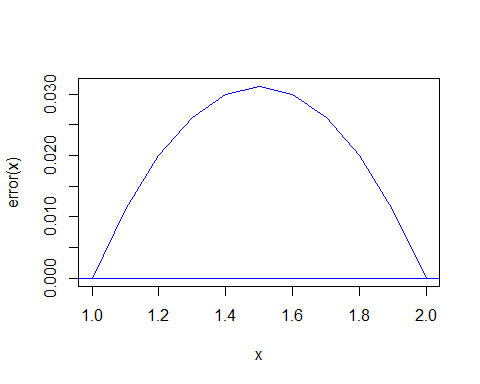
## 0.409075

plot(x,solucionadorH(p,x),type="l",col="blue")  
abline(h=0,col="blue")  
points(rbind(M),pch=19,cex=0.7,col="red")



1. error. Teniendo en cuenta que conocemos la función se puede hacer uso del metodo interpolante de lagrange.

rm(list=ls())  
segunda=function(x) -1/(x\*x)  
error=function(x) (x-1)\*(x-2)\*segunda(2)/2  
x<-seq(1,2,0.1)  
plot(x,error(x),type="l",col="blue")  
abline(h=0,col="blue")



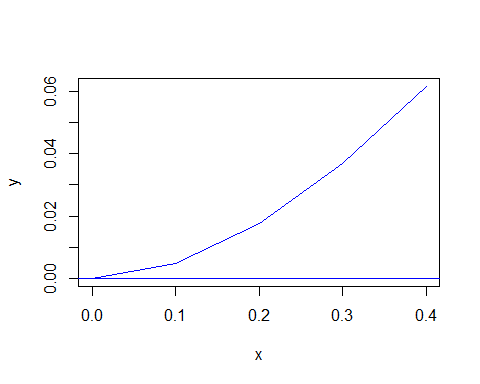
cat("El error en el intervalo [1,2] describe el siguiente comportamiento: ")

## El error en el intervalo [1,2] describe el siguiente comportamiento:

## 3.

a)gráfica

rm(list=ls())  
 x<-seq(0,0.4,0.1)  
 y=c(0,0.00467884,0.01752309,0.0369363,0.06155193)  
 plot(x,y,type="l",col="blue")  
 abline(h=0,col="blue")

 b)Método de neville

rm(list=ls())  
neville=function(M,x)  
{  
 tam<-NROW(M)  
 temp=matrix(0,nrow=tam,ncol=tam)  
 temp[,1]=M[,2]  
 i=2  
 j=1  
 while(i<=tam)  
 {  
 j=1  
 while(j<i)  
 {  
 a=((x-M[i-j,1])\*temp[i,j])  
 b=((x-M[i,1])\*temp[i-1,j])  
 abajo=M[i,1]-M[i-j,1]  
 temp[i,j+1]=(a-b)/abajo  
 #temp[i,j+1]=(((x-M[i-j,1])\*temp[i,j])-((x-M[i,1])\*temp[i-1,j]))/M[i,1]-M[i-j,1]  
   
 j=j+1  
 }  
   
 i=i+1  
 }  
 return(temp[tam,tam])  
}  
  
M<-matrix(c(0,0.1,0.2,0.3,0.4,0,0.00467884,0.01752309,0.0369363,0.06155193),ncol=2,nrow=5)  
cat(neville(M,0.25))

## 0.02649937

## 4. Demostración

Para ver la demostración ver el archivo adjunto:<https://docs.google.com/document/d/1LxlYIbw-R4bNqshdZ8uqnlKm8QlkTP3Hh6rspbti6WI/edit>