Taller\_1

Andres David Gomez B

11 de febrero de 2018

## 1. Horner

# Horner  
rm(list=ls())  
horner <- function(p,n,x) {  
 y<-p[c(1)]  
 i<-1  
 iter<-0  
 while (i < n+1) {  
 y<-x\*y+p[c(i+1)]  
 i<-i+1  
 iter<-iter+2  
 }  
 cat("RES: ",y,"\n")  
 cat("Operaciones: ",iter,"\n")  
 iter<-0  
 i<-n  
 j<-1  
 der<-0  
 while (i > 0) {  
 der[c(j)]<-(p[c(j)]\*i)  
 i<-i-1  
 j<-j+1  
 }  
 i<-1  
 y<-der[c(1)]  
 while (i < n) {  
 y<-x\*y+der[c(i+1)]  
 i<-i+1  
 iter<-iter+2  
 }  
 cat("RES derivada : ",y,"\n")  
 cat("Operaciones derivada: ",iter,"\n")  
}  
p<-c(2,0,-3,3,-4)  
horner(p,4,-2)

## RES: 10   
## Operaciones: 8   
## RES derivada : -49   
## Operaciones derivada: 6

cat("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" ,"\n")

## \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

p<-c(7,6,-6,0,3,-4)  
horner(p,5,3)

## RES: 2030   
## Operaciones: 10   
## RES derivada : 3324   
## Operaciones derivada: 8

cat("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" ,"\n")

## \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

p<-c(-5,0,3,0,2,-4,0)  
horner(p,6,-1)

## RES: 4   
## Operaciones: 12   
## RES derivada : 10   
## Operaciones derivada: 10

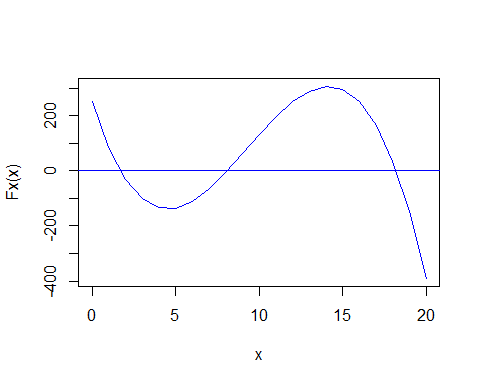
## 2.Caja rectangular

En este punto, dado que no sabemos que intervalo o punto es el adecuado, debemos buscar, estos datos. Entonces se usa el siguiente programa:

# programa busca intervalos donde es posible encontrar raices  
rm(list=ls())  
Fx <- function(x) -1\*x^3+28\*x^2-192\*x+250  
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)

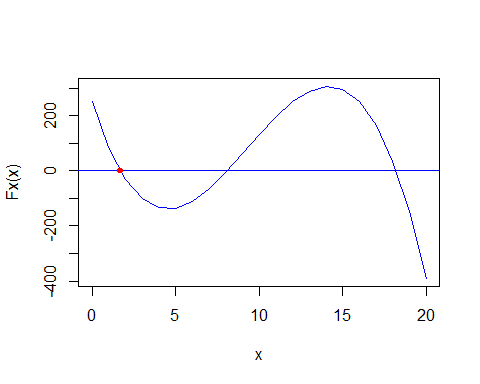
## intervalo [ 0 2 ]   
## intervalo [ 2 9 ]   
## intervalo [ 9 19 ]

x<-seq(0,20,1)  
plot(x,Fx(x),type="l",col="blue")  
abline(h=0,col="blue")



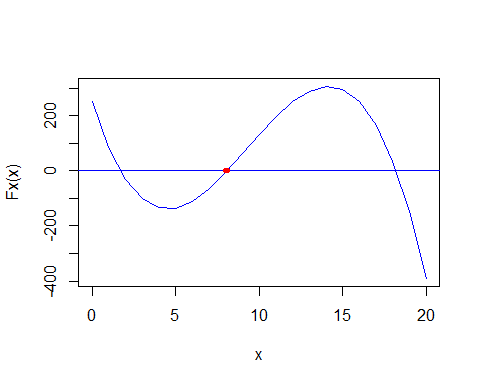
Se puede usar el buscador o la grafica para acercarse a las soluciones, pero ambos son limitados, por lo que son necesarios los dos para encontrar los puntos iniciales. Entonces en esos intervalos existe una inflexion, que garantiza la existencia de una raiz. Se usa un punto entre el intervalo, localizados con ayuda de los intervalos y de la grafica, como punto inicial y se procede a usar el metodo de newton.

# Metodo de Newton  
rm(list=ls())  
Fx <- function(x) -1\*x^3+28\*x^2-192\*x+250  
F1x <- function(x) -3\*x^2+56\*x-192  
# Metodo de Newton  
# Halla la raiz de   
Newton <- function(x0) {  
 x<-seq(0,20,1)  
 plot(x,Fx(x),type="l",col="blue")  
 abline(h=0,col="blue")  
 x<-x0-(Fx(x0)/F1x(x0))  
 error <-1  
 while (error > 1.e-4) {  
 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")  
 }  
}  
Newton(0)



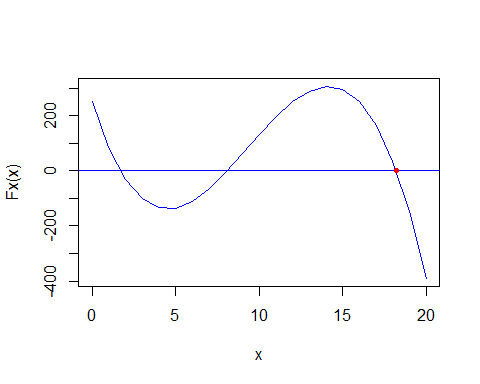
## X= 1.666619 E= 0.02946931   
## X= 1.696088 E= 0.0001888197   
## X= 1.696277 E= 7.732474e-09

Newton(5.5)



## X= 8.005242 E= 0.08844819   
## X= 8.09369 E= 0.0004708878   
## X= 8.09322 E= 1.274299e-08

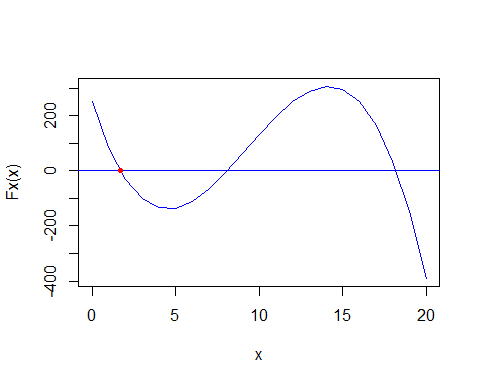
Newton(19)



## X= 18.21159 E= 0.00108474   
## X= 18.2105 E= 1.875686e-07

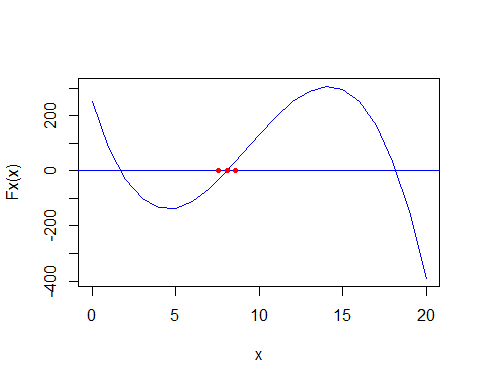
Como segundo metodo, se usa el metodo de la secante con el intervalo encontrado

# Metodo de secante  
# Remueve todos los objetos creados  
rm(list=ls())  
Fx <- function(x) -1\*x^3+28\*x^2-192\*x+250  
F1x <- function(x) -3\*x^2+56\*x-192  
# Halla la raiz de Fx  
secante <- function(x0,x1) {  
 x<-seq(0,20,1)  
 plot(x,Fx(x),type="l",col="blue")  
 abline(h=0,col="blue")  
 x<-(Fx(x1)\*x0-Fx(x0)\*x1)/(Fx(x1)-Fx(x0))  
 error <-1  
 while (error > 1.e-4) {  
 x0<-x1  
 x1<-x  
 x<-(Fx(x1)\*x0-Fx(x0)\*x1)/(Fx(x1)-Fx(x0))  
 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")  
 }  
}  
secante(0,2)



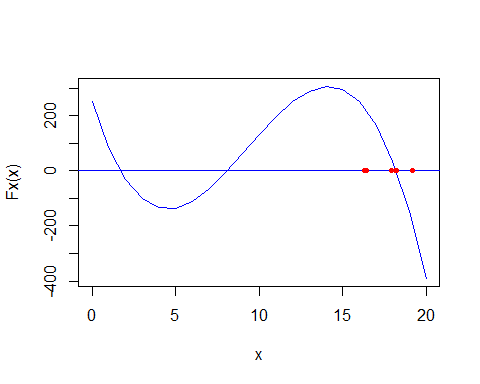
## X= 1.689955 E= 0.006313003   
## X= 1.696401 E= 0.0001244138   
## X= 1.696277 E= 1.703876e-07

secante(2,9)



## X= 7.557577 E= 0.5586044   
## X= 8.602794 E= 0.4992192   
## X= 8.077328 E= 0.01590585   
## X= 8.092825 E= 0.0003940794   
## X= 8.09322 E= 3.618945e-07

secante(9,19)



## X= 16.3192 E= 2.951724   
## X= 72.56216 E= 20.82413   
## X= 16.37069 E= 2.820269   
## X= 16.42121 E= 2.696209   
## X= 19.1534 E= 0.82764   
## X= 17.91445 E= 0.3111112   
## X= 18.16934 E= 0.04143279   
## X= 18.21253 E= 0.0020247   
## X= 18.21049 E= 1.335137e-05

La solucion puede ser cualquiera de las dos primeras puesto que la tercera respuesta arroja un resultado negativo.

¿Cual etapa del proceso de resolución de un problema numérico requiere más atención? la etapa que requiere mayor atención en la resolución de un problema es el análisis de este ya que realizar un análisis incorrecto de el problema puede generar un error en las posteriores etapas de diseño y la solución, y el hacer un buen análisis del problema facilita su posterior solución ya que da una visión más amplia del ejercicio y del como poder desarrollarlo. ¿Qué conocimientos son necesarios para formular un modelo matemático?

son necesarios los conocimientos respecto a los principios físicos y ecuaciones que rodean el problema, el poseer conocimientos sobre las ecuaciones permitirá tener una comprensión clara de como conectar la información del modelo, también es necesario conocer si existen modelos matemáticos que den solución al problema a tratar y en caso de no existir es necesario revisar otros modelos que trabajen problemas similares, lo cual permitirá plantear de manera más fácil el modelo del problema a trabajar.

En el ejemplo de la caja ¿Cual seria la desventaja de intentar obtener experimentalmente la solución mediante prueba y error en lugar de analizar el modelo matemático? se necesitaría realizar una cantidad indefinida de pruebas para encontrar una solución, al contrario de usar métodos numéricos los cuales nos darán de manera mas rapida y eficaz. ¿Que es más crítico: el error de truncamiento o el error de redondeo? El error de truncamiento es mas critico ya que este puede generar perdida de datos al usar aproximaciones al contrario de el error de redondeo que para de tomar cierta cantidad de cifras significativas pero las cuales corresponden al valor real de la respuesta lo que hace mas exacto el valor verdadero . ¿Cuál es la ventaja de instrumentar computacionalmente un método numérico? Permite realizar los cálculos del modelo con mayor rapidez y permite replicar los modelos y aplicarlos a diferentes modelos y la instrumentación computacional constituye la base para entender estos objetos matemáticos computacionales y extender el conocimiento al desarrollo de métodos para resolver problemas nuevos y más complejos. Cual es la desventaja de usar funciones definidas en paquetes de software para resolver los problemas.

Las desventajas de usar paquetes de software para resolver problemas son que no se conoce totalmente cómo se realiza los cálculos para resolver el problemas,las funciones aún no están completamente definidas o entendidas y el desconocer como están definidos y los posibles errores que estos paquetes pueden tener o los métodos que usen para resolver un problema, pueden afectar si no se ajusta hacia los métodos que se necesitan para nuestro problema.

¿Por que es importante validar los resultados obtenidos? ya que los métodos numéricos tienen distintas soluciones las cuales son verdaderas, pero que no se ajustan a el problema a tratar, por ejemplo en el problema de la caja los resultados posibles fueron 8.0932, 15.8136, 7.8136 de las cuales las respuestas apropiadas fueron las primeras dos puesto que la tercera no tiene interpretación física.

## 3. raiz cuadrada

# Remueve todos los objetos creados  
 # Remueve todos los objetos creados  
rm(list=ls())  
raices <- function(n,E,x) {  
 y<-0.5\*(x+n/x)  
 E2<-0  
 while (E < abs(x-y)) {  
 x<-y  
 y<-0.5\*(x+n/x)  
 E2<-abs(x-y)  
 cat("RES: ",y,"\t","E= ",E2,"Er ",E2/abs(y),"precision ",1-(E2/abs(y)),"\n")  
 }  
   
}  
raices(7,1.e-6,3)

## RES: 2.645833 E= 0.02083333 Er 0.007874016 precision 0.992126   
## RES: 2.645751 E= 8.2021e-05 Er 3.100102e-05 precision 0.999969   
## RES: 2.645751 E= 1.271367e-09 Er 4.805316e-10 precision 1

Precisión: Los cálculos muestran que conforme el error relativo disminuye con cada iteración, es cada vez mayor la precisión de la respuesta. convergencia: como se puede observar los resultados se aproximan cada vez más al resultado esperado lo que nos indica que el algoritmo converge al resultado esperado de raiz de 7, se puede observar que la fórmula iterativa converge ya que el error de truncamiento cada vez es menor lo que afirma la convergencia del algoritmo usado. validez: al realizar la validación de la respuesta obtenida nos da que 2.65751 elevado al cuadrado no nos da como resultado 7 si no 6. 5408574001, ya que las respuestas dadas por el algoritmo aun tienen error lo cual no permite que la respuesta sea exacta pero da una aproximación del posible valor esperado.

## 4. Error de redondeo

El computador guarda 0,5367x10∧3, que en comparación con los 0,53678x10∧3 originales, arroja un valor de 0,53678x10∧3-0,5367x10∧3=0,00008x10∧3 Según el criterio dado, la cota está dada por: (1x10∧3-4) =1x10∧-1  
entonces |0,08| < 1x10∧-1 El error relativo: (1x10∧-1)/(0.1x10∧4) =1x10∧-4

## 5. Error absoluto y relativo

d=20 Eabsoluto=4(0,1)+5(0,1)=0,9 Erelativo=(0,1/4) +(0,1/5)=0,045 d=20±0,9

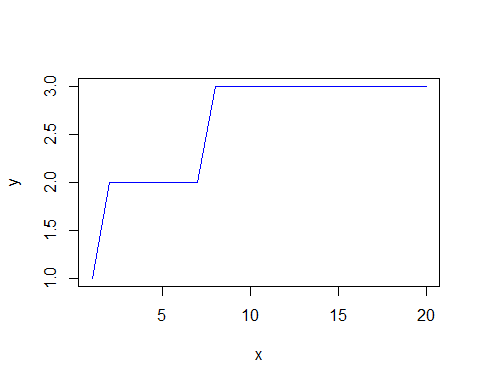
## 6. Eficiencia de algoritmos

# Remueve todos los objetos creados  
rm(list=ls())  
cociente <- function(n) {  
 i<- 1  
 y<- 0  
 while (n >0) {  
 d<- (n%/%2)  
 n<- (d/2)  
 y[c(i)]<-d  
 cat(i," ",d,"\n")  
 i=i+1  
   
 }  
   
}  
cociente(73)

## 1 36   
## 2 9   
## 3 2   
## 4 0

1. En este caso buscamos encontrar el comportamiento general, descrito por la grafica:

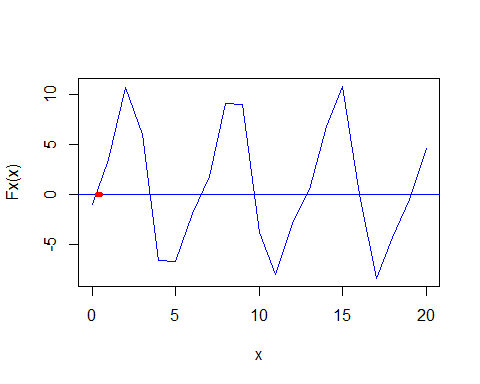
x<-seq(1,20,1)  
y<-c(1,2,2,2,2,2,2,3,3,3,3,3,3,3,3,3,3,3,3,3)  
plot(x,y,type="l",col="blue")  
abline(h=0,col="blue")

 Por lo tanto se puede aproximar a O([log(n)])

## 7. Particula en movimiento

Se usa la formula de distancia y se deriva para hallar la formula que descibe el tiempo y usamos el metodo de Newton para resolverlo, despues de hallar los intervalos.

# Metodo de Newton  
rm(list=ls())  
Fx <- function(x) 8\*sin(x)-3\*sin(2\*x)-2\*cos(x)/2\*sqrt((4\*(cos(x))^2)+((sin(x))^2)-8\*cos(x)-2\*sin(x)+5)  
F1x <- function(x) 18\*sin(x)-32\*sin(2\*x)+18\*sin(3\*x)+408\*cos(x)-240\*cos(2\*x)+72\*cos(3\*x)-9\*cos(4\*x)-231/(8\*(((sin(x))^2)-2\*sin(x)+(4\*(cos(x))^2)-8\*cos(x)+5)^1.5)  
# Halla la raiz de   
Newton <- function(x0) {  
 x<-seq(0,20,1)  
 plot(x,Fx(x),type="l",col="blue")  
 abline(h=0,col="blue")  
 x<-x0-(Fx(x0)/F1x(x0))  
 error <-1  
 while (error > 1.e-4) {  
 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")  
 }  
}  
Newton(0.5)

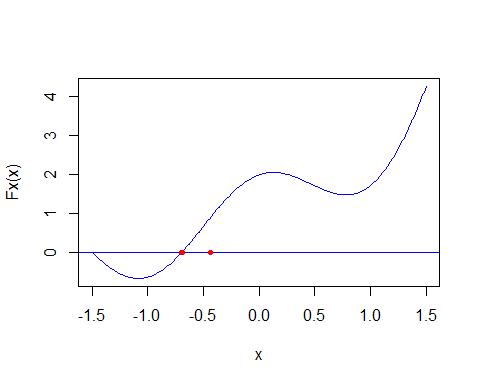


## X= 0.4818585 E= 0.007998083   
## X= 0.4738604 E= 0.007398797   
## X= 0.4664616 E= 0.006870485   
## X= 0.4595911 E= 0.006402411   
## X= 0.4531887 E= 0.005985531   
## X= 0.4472031 E= 0.005612307   
## X= 0.4415908 E= 0.005276479   
## X= 0.4363144 E= 0.004972851   
## X= 0.4313415 E= 0.004697094   
## X= 0.4266444 E= 0.004445596   
## X= 0.4221988 E= 0.00421532   
## X= 0.4179835 E= 0.004003708   
## X= 0.4139798 E= 0.003808589   
## X= 0.4101712 E= 0.003628114   
## X= 0.4065431 E= 0.003460698   
## X= 0.4030824 E= 0.003304976   
## X= 0.3997774 E= 0.003159769   
## X= 0.3966176 E= 0.003024049   
## X= 0.3935936 E= 0.002896923   
## X= 0.3906967 E= 0.002777605   
## X= 0.3879191 E= 0.002665403   
## X= 0.3852537 E= 0.002559707   
## X= 0.382694 E= 0.002459975   
## X= 0.380234 E= 0.002365725   
## X= 0.3778683 E= 0.002276527   
## X= 0.3755917 E= 0.002191994   
## X= 0.3733997 E= 0.00211178   
## X= 0.371288 E= 0.002035573   
## X= 0.3692524 E= 0.00196309   
## X= 0.3672893 E= 0.001894076   
## X= 0.3653952 E= 0.001828297   
## X= 0.3635669 E= 0.001765543   
## X= 0.3618014 E= 0.001705619   
## X= 0.3600958 E= 0.00164835   
## X= 0.3584474 E= 0.001593574   
## X= 0.3568538 E= 0.001541141   
## X= 0.3553127 E= 0.001490914   
## X= 0.3538218 E= 0.001442767   
## X= 0.352379 E= 0.001396584   
## X= 0.3509824 E= 0.001352257   
## X= 0.3496302 E= 0.001309685   
## X= 0.3483205 E= 0.001268776   
## X= 0.3470517 E= 0.001229444   
## X= 0.3458223 E= 0.001191607   
## X= 0.3446307 E= 0.001155192   
## X= 0.3434755 E= 0.001120127   
## X= 0.3423553 E= 0.001086348   
## X= 0.341269 E= 0.001053793   
## X= 0.3402152 E= 0.001022406   
## X= 0.3391928 E= 0.0009921312   
## X= 0.3382007 E= 0.000962919   
## X= 0.3372377 E= 0.0009347217   
## X= 0.336303 E= 0.0009074945   
## X= 0.3353955 E= 0.0008811951   
## X= 0.3345143 E= 0.0008557838   
## X= 0.3336585 E= 0.0008312229   
## X= 0.3328273 E= 0.0008074769   
## X= 0.3320198 E= 0.0007845122   
## X= 0.3312353 E= 0.0007622969   
## X= 0.330473 E= 0.0007408009   
## X= 0.3297322 E= 0.0007199955   
## X= 0.3290122 E= 0.0006998538   
## X= 0.3283124 E= 0.0006803497   
## X= 0.327632 E= 0.000661459   
## X= 0.3269706 E= 0.0006431581   
## X= 0.3263274 E= 0.0006254251   
## X= 0.325702 E= 0.0006082386   
## X= 0.3250938 E= 0.0005915787   
## X= 0.3245022 E= 0.0005754261   
## X= 0.3239268 E= 0.0005597624   
## X= 0.323367 E= 0.0005445702   
## X= 0.3228224 E= 0.0005298327   
## X= 0.3222926 E= 0.000515534   
## X= 0.3217771 E= 0.0005016588   
## X= 0.3212754 E= 0.0004881924   
## X= 0.3207872 E= 0.0004751209   
## X= 0.3203121 E= 0.0004624309   
## X= 0.3198497 E= 0.0004501094   
## X= 0.3193995 E= 0.0004381441   
## X= 0.3189614 E= 0.0004265233   
## X= 0.3185349 E= 0.0004152355   
## X= 0.3181196 E= 0.00040427   
## X= 0.3177154 E= 0.0003936162   
## X= 0.3173218 E= 0.000383264   
## X= 0.3169385 E= 0.0003732038   
## X= 0.3165653 E= 0.0003634264   
## X= 0.3162019 E= 0.0003539227   
## X= 0.3158479 E= 0.0003446842   
## X= 0.3155033 E= 0.0003357025   
## X= 0.3151676 E= 0.0003269698   
## X= 0.3148406 E= 0.0003184782   
## X= 0.3145221 E= 0.0003102205   
## X= 0.3142119 E= 0.0003021895   
## X= 0.3139097 E= 0.0002943783   
## X= 0.3136153 E= 0.0002867802   
## X= 0.3133285 E= 0.0002793888   
## X= 0.3130491 E= 0.000272198   
## X= 0.3127769 E= 0.0002652019   
## X= 0.3125117 E= 0.0002583945   
## X= 0.3122534 E= 0.0002517705   
## X= 0.3120016 E= 0.0002453243   
## X= 0.3117563 E= 0.0002390508   
## X= 0.3115172 E= 0.000232945   
## X= 0.3112843 E= 0.000227002   
## X= 0.3110573 E= 0.0002212172   
## X= 0.310836 E= 0.0002155859   
## X= 0.3106205 E= 0.0002101038   
## X= 0.3104104 E= 0.0002047667   
## X= 0.3102056 E= 0.0001995704   
## X= 0.310006 E= 0.0001945109   
## X= 0.3098115 E= 0.0001895844   
## X= 0.3096219 E= 0.0001847872   
## X= 0.3094371 E= 0.0001801156   
## X= 0.309257 E= 0.0001755661   
## X= 0.3090815 E= 0.0001711353   
## X= 0.3089103 E= 0.00016682   
## X= 0.3087435 E= 0.0001626169   
## X= 0.3085809 E= 0.0001585229   
## X= 0.3084224 E= 0.0001545352   
## X= 0.3082678 E= 0.0001506506   
## X= 0.3081172 E= 0.0001468665   
## X= 0.3079703 E= 0.0001431801   
## X= 0.3078271 E= 0.0001395887   
## X= 0.3076875 E= 0.0001360897   
## X= 0.3075514 E= 0.0001326808   
## X= 0.3074188 E= 0.0001293593   
## X= 0.3072894 E= 0.000126123   
## X= 0.3071633 E= 0.0001229697   
## X= 0.3070403 E= 0.0001198969   
## X= 0.3069204 E= 0.0001169028   
## X= 0.3068035 E= 0.000113985   
## X= 0.3066895 E= 0.0001111416   
## X= 0.3065784 E= 0.0001083707   
## X= 0.30647 E= 0.0001056702   
## X= 0.3063643 E= 0.0001030384   
## X= 0.3062613 E= 0.0001004734   
## X= 0.3061608 E= 9.797343e-05

## 8. Punto de interseccion

El punto de interseccion sera la diferencia entre ambas funciones, se buscan los intervalos iniciales y se resuelve por el metodo de Newton

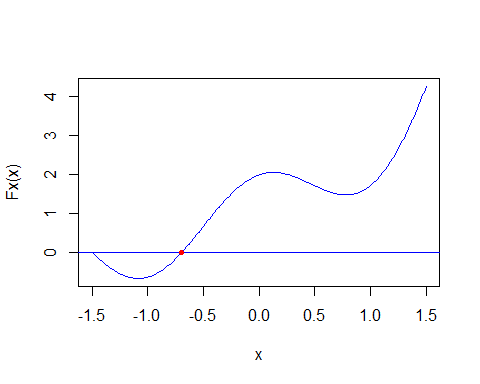
# Metodo de Newton  
rm(list=ls())  
Fx <- function(x) 2+cos(3\*x)-(2-exp(x))  
F1x <- function(x) exp(x)-3\*sin(3\*x)  
# Halla la raiz de   
Newton <- function(x0) {  
 x<-seq(-1.5,1.5,0.01)  
 plot(x,Fx(x),type="l",col="blue")  
 abline(h=0,col="blue")  
 x<-x0-(Fx(x0)/F1x(x0))  
 error <-1  
 while (error > 1.e-4) {  
 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")  
 }  
}  
Newton(0)



## X= -0.4414738 E= 0.2496774   
## X= -0.6911511 E= 0.006148202   
## X= -0.6972993 E= 2.978029e-05

Ahora por el metodo de la secante:

# Metodo de secante  
rm(list=ls())  
Fx <- function(x) 2+cos(3\*x)-(2-exp(x))  
F1x <- function(x) exp(x)-3\*sin(3\*x)  
# Halla la raiz de Fx  
secante <- function(x0,x1) {  
 x<-seq(-1.5,1.5,0.01)  
 plot(x,Fx(x),type="l",col="blue")  
 abline(h=0,col="blue")  
 x<-(Fx(x1)\*x0-Fx(x0)\*x1)/(Fx(x1)-Fx(x0))  
 error <-1  
 while (error > 1.e-4) {  
 x0<-x1  
 x1<-x  
 x<-(Fx(x1)\*x0-Fx(x0)\*x1)/(Fx(x1)-Fx(x0))  
 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")  
 }  
}  
secante(-1,0)



## X= -0.6962298 E= 0.00109839   
## X= -0.6973962 E= 6.709359e-05

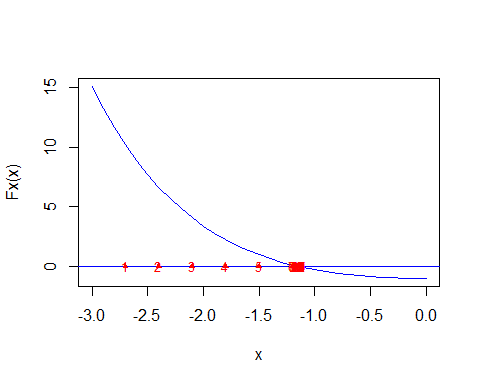
Al comparar ambas respuestas se puede observar que se asemejan en los tres primeros decimales, una incertidumbre de 1.e-4, que podemos aceptar como valido, entre otras caracteristicas, se encuentra que el metodo de secante es mas preciso y tuvo menos iteraciones que el de Newton.

## 9. Ejercicios

1. El metodo de Newton es un excelente ejemplo de metodo iterativo con convergencia cuadratica, porque su intervalo de convergencia existe mientras este sea continuo y se localiza en él, el corte con el eje x y las operaciones de las que se hace uso son elementales, suponiendo que se cuente con la derivada.

14.a Para que el algoritmo sea efectivo se necesita que la funcion sea continua, ademas que en el intervalo de entrada se encuentre una raiz unica o por lo menos limitar el intervalo a una sola raiz. Mientras estas condiciones sean satisfechas la raiz, puede ser calculada. 14.b El error de truncamiento es identico entre iteraciones hasta que encuentra el punto de inflexion, en donde cambia drasticamente, entonces las iteraciones vuelven a ser constantes, entonces se dice que la eficiencia es lineal a trozos y cuenta con un factor de convergencia de magnitud 1 y 0,1 entre ellos. 14.c Este es el algoritmo usando un ejemplo en clase

# Remueve todos los objetos creados  
rm(list=ls())  
Fx <- function(x) exp(-x) + x -2  
  
# Halla la raiz de Fx  
deci<- function(a,b) {  
 x<-seq(a,b,0.1)  
 plot(x,Fx(x),type="l",col="blue")  
 abline(h=0,col="blue")  
 x<-a  
 d<-(b-a)/10  
 i<-0  
 error<-abs(a-b)/2  
 while (error > 1.e-4) {  
 i<-i+1  
 x<-x+d  
 if (Fx(x) == 0) break  
 if (Fx(x)\*Fx(a) < 0)  
 {  
 x<-x-d  
 d<-d/10  
 }  
 points(rbind(c(x,0)),pch=17,cex=0.7,col="red")  
 text(x,0,i,cex=0.8,col="red")  
 error<-d  
 cat("X=",x,"\tE=",error,"\n")  
 }  
}  
deci(-3,0)



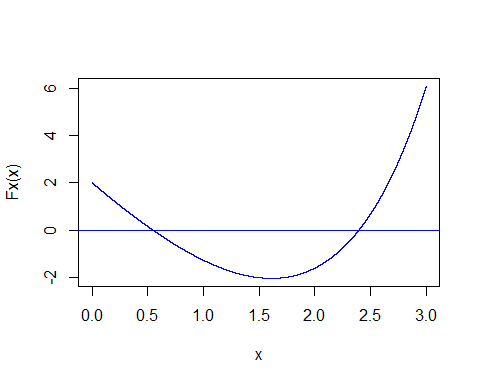
## X= -2.7 E= 0.3   
## X= -2.4 E= 0.3   
## X= -2.1 E= 0.3   
## X= -1.8 E= 0.3   
## X= -1.5 E= 0.3   
## X= -1.2 E= 0.3   
## X= -1.2 E= 0.03   
## X= -1.17 E= 0.03   
## X= -1.17 E= 0.003   
## X= -1.167 E= 0.003   
## X= -1.164 E= 0.003   
## X= -1.161 E= 0.003   
## X= -1.158 E= 0.003   
## X= -1.155 E= 0.003   
## X= -1.152 E= 0.003   
## X= -1.149 E= 0.003   
## X= -1.149 E= 3e-04   
## X= -1.1487 E= 3e-04   
## X= -1.1484 E= 3e-04   
## X= -1.1481 E= 3e-04   
## X= -1.1478 E= 3e-04   
## X= -1.1475 E= 3e-04   
## X= -1.1472 E= 3e-04   
## X= -1.1469 E= 3e-04   
## X= -1.1466 E= 3e-04   
## X= -1.1463 E= 3e-04   
## X= -1.1463 E= 3e-05

15.a La ecuacion es -5x+e^x+1 15.b Se usan los criterios de busqueda que se han venido usando en todo el taller, graficacion y el programa buscador para obtener:

# programa busca intervalos donde es posible encontrar raices  
rm(list=ls())  
Fx <- function(x) 1+exp(x)-5\*x  
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 1 ]   
## intervalo [ 1 3 ]

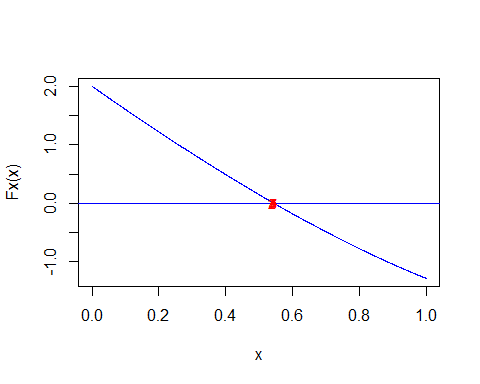
x<-seq(0,3,0.0001)  
plot(x,Fx(x),type="l",col="blue")  
abline(h=0,col="blue")



15.c

g(x)=(e^x+1)/5 Se usa el punto medio del primer intervalo hallado como punto inicial.

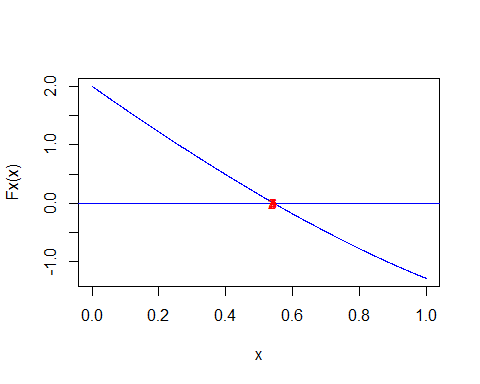
# Remueve todos los objetos creados  
rm(list=ls())  
Fx <- function(x) exp(x) +1-5\*x  
gx<- function(x) (exp(x) +1)/5  
# Halla la raiz de Fx  
fijo<- function(a) {  
 x<-seq(0,1,0.001)  
 plot(x,Fx(x),type="l",col="blue")  
 abline(h=0,col="blue")  
 x<-gx(a)  
 i<-1  
 error<-1  
 while (error > 1.e-4) {  
 error<-abs(gx(x)-x)  
 x<-gx(x)  
 i<-i+1  
 points(rbind(c(x,0)),pch=17,cex=0.7,col="red")  
 text(x,0,i,cex=0.8,col="red")  
 cat("X=",x,"\tE=",error,"\n")  
 }  
}  
fijo(0.5)



## X= 0.5396996 E= 0.00995532   
## X= 0.5430983 E= 0.003398707   
## X= 0.5442664 E= 0.001168075   
## X= 0.5446687 E= 0.0004023637   
## X= 0.5448074 E= 0.0001387101   
## X= 0.5448553 E= 4.783159e-05

15.d Manejando 5 iteraciones:

# Remueve todos los objetos creados  
rm(list=ls())  
Fx <- function(x) exp(x) +1-5\*x  
gx<- function(x) (exp(x) +1)/5  
# Halla la raiz de Fx  
fijo<- function(a) {  
 x<-seq(0,1,0.001)  
 plot(x,Fx(x),type="l",col="blue")  
 abline(h=0,col="blue")  
 x<-gx(a)  
 i<-1  
 error<-1  
 while (6 > i) {  
 error<-abs(gx(x)-x)  
 x<-gx(x)  
 i<-i+1  
 points(rbind(c(x,0)),pch=17,cex=0.7,col="red")  
 text(x,0,i,cex=0.8,col="red")  
 cat("X=",x,"\tE=",error,"\n")  
 }  
}  
fijo(0.5)



## X= 0.5396996 E= 0.00995532   
## X= 0.5430983 E= 0.003398707   
## X= 0.5442664 E= 0.001168075   
## X= 0.5446687 E= 0.0004023637   
## X= 0.5448074 E= 0.0001387101