Newton Raphson

Sebastian Angarita, Hector Rodriguez, Aldemar Ramirez

4/8/2019

Problema

Hallar la raiz de una función a partir de un x_0 a traves del metodo de Newton Raphson.

Solución

Lenguaje de programación: R

Función principal Newton Raphson

Parametros:

fun <- función.

x0 <- x_0 desde donde se comienza la busqueda de la raíz.

tol <- tolerancia mínima que debe tener la función.

maxiter <- cantidad máxima de iteraciones.

Valores de retorno:

x1 <- resultado de la raíz.

errorAbsoluto <- vector de errores absolutos de las x_n.

errorRelativo <- vector de errores relativos de las x_n.

x <- vector de las x_n.

Implementacion

```
newtonraphson = function(fun, x0, tol, maxiter){

  # f = string
  numiter = 0
  errorAbsoluto = c()
  errorRelativo = c()
  x = c()

  g = parse(text=fun) # parse devuelve tipo "expression"
  g. = D(g,"x")
  fx = function(x){eval(g)} # convertir f a función
  fp = function(x){eval(g.)} # convertir f' a función

  correccion = -fx(x0)/fp(x0)

  while (abs(correccion) >= tol && numiter <= maxiter) {
```

```
    numiter = numiter + 1
    if (fp(x0) == 0) stop("División por cero")
    x1 = x0 + correccion
    x0 = x1

    x <- c(x,x1)
    errorAbsoluto <- c(errorAbsoluto,abs(correccion))
    errorRelativo <- c(errorRelativo,abs(correccion)/(abs(x0)))

    correccion = -fx(x1)/fp(x1)

  }
  if (numiter > maxiter){ warning("Se alcanzó el máximo número de iteraciones.")
  } else {

    my_list <- list("resultado" = x1, "errorAbsoluto" = errorAbsoluto,
                    "errorRelativo" = errorRelativo, "x" = x)
    return(my_list)

  }
}
```

La función basicamente realiza la derivada de la función recibida y con el valor x_0 realiza iteraciones para encontrar un x^* cercano tal que $f(x^*) = 0$ con una tolerancia minima de tol o hasta completar la cantidad maxima de iteraciones.

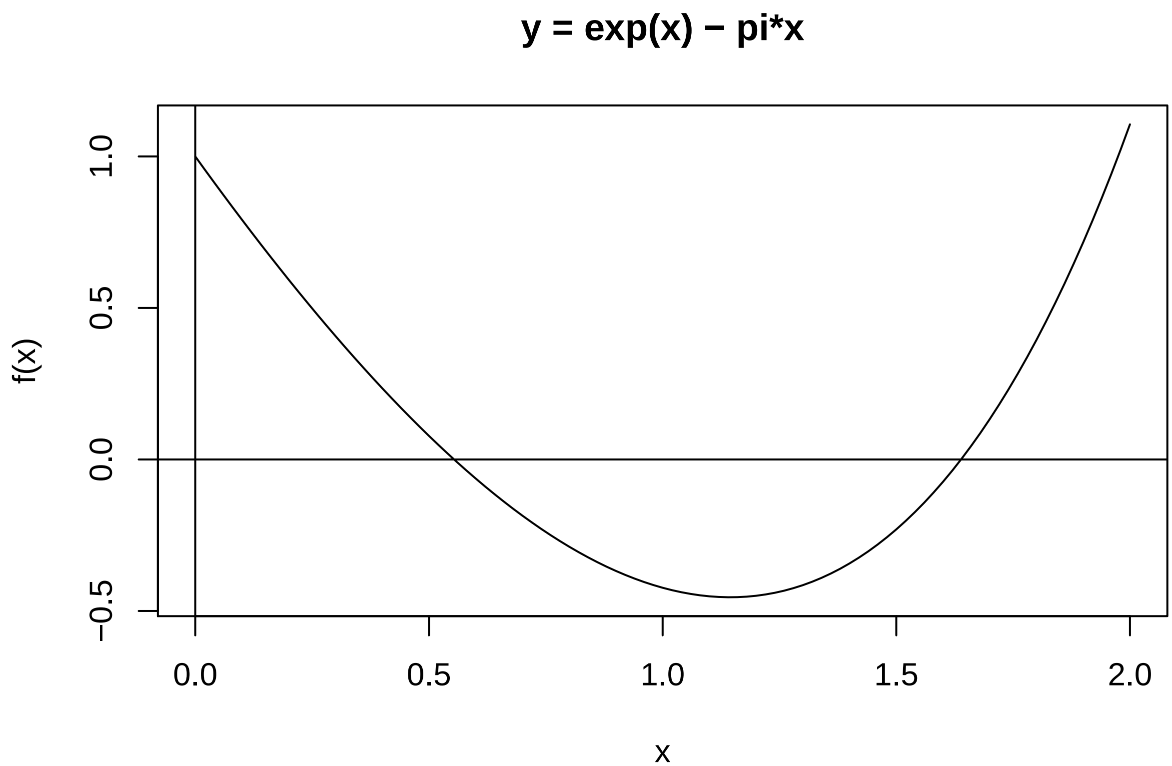
Suponemos que h es la correccion que necesita x_0 para alcanzar a x^*, es decir $x_0 + h = x^*$ y $f(x_0 + h) = 0$. Como $0 = f(x_0 + h) = f(x_0) + hf'(x_0)$ al despejar h, se tiene $h = -f(x_0)/f'(x_0)$ que se encuentra con el nombre de la variable correccion dentro de la implementacion de la función en el codigo.

De esta manera, una aproximacion corregida de x_0 seria

$$x_1 = x_0 - f(x_0)/f'(x_0)$$

Gráfica de la función

```
f = function(x) exp(x) - pi*x
curve(f, 0,2); abline(h=0, v=0) #gráfico para decidir un intervalo
title(main="y = exp(x) - pi*x")
```



Resultados

Se muestra una tabla con la cantidad de iteraciones y el resultado de cada una para la función

$$y = e^x - pi * x$$

Una buena aproximacion para seleccionar un x_0 inicial es que cumpla la siguiente formula.

$$f(x_0)f''(x_0) > 0$$

En este caso $x_0 = 1.3$.

```
## --- Pruebas
# recibe la función como una tira
resultados<-newtonraphson("exp(x)-pi*x",1.3, 1e-8, 100)

tablaErrores <- data.frame(
  "Iteraciones" = 1:length(resultados$errorAbsoluto),
  "x_n" = resultados$x,
  "Error Absoluto" = resultados$errorAbsoluto,
  "Error Relactivo" = resultados$errorRelativo
)
print(tablaErrores)
```

```
##   Iteraciones      x_n Error.Absoluto Error.Relactivo
## 1           1 2.085997   7.859970e-01    3.767968e-01
```

```
## 2           2 1.780712   3.052853e-01    1.714400e-01
## 3           3 1.659025   1.216864e-01    7.334812e-02
## 4           4 1.639047   1.997804e-02    1.218881e-02
## 5           5 1.638529   5.185442e-04    3.164694e-04
## 6           6 1.638528   3.451161e-07    2.106257e-07
```

```r
cat("Numero de iteraciones",length(resultados$errorAbsoluto),"\n")
```

```
## Numero de iteraciones 6
```

```r
cat("x = ",resultados$resultado,"\n")
```

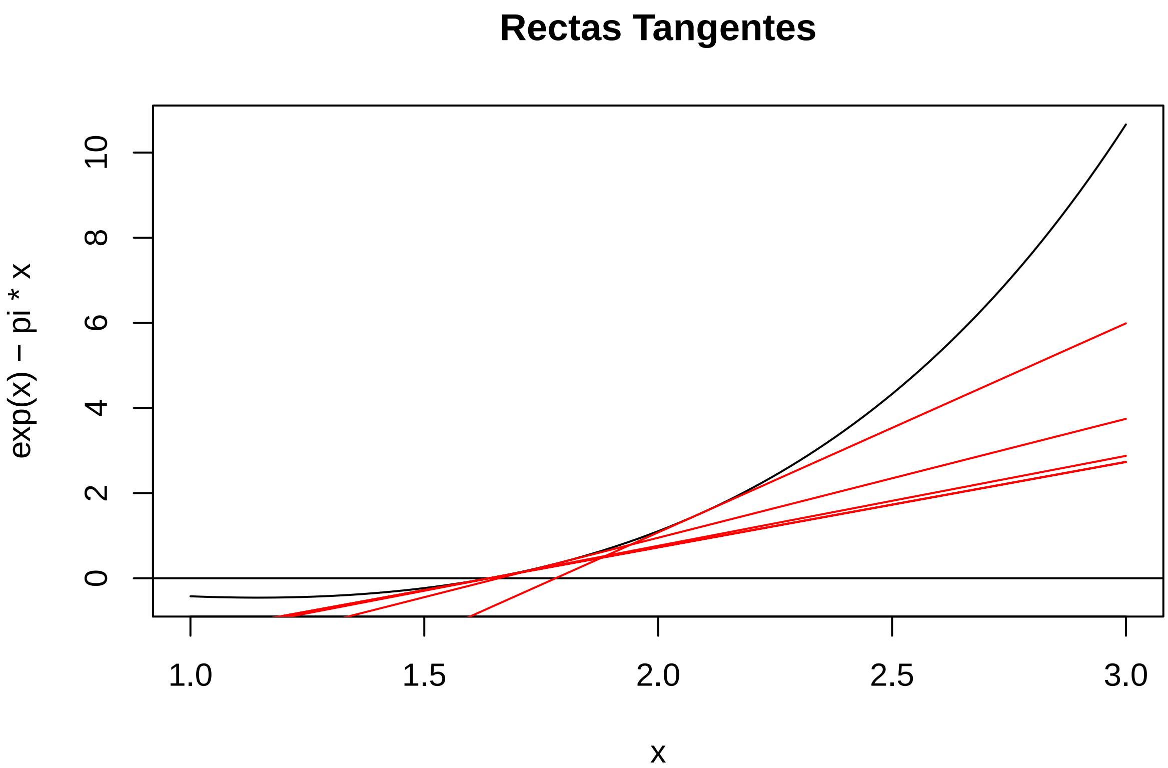
```
## x =  1.638528
```

```r
cat("Error absoluto estimado  = ", resultados$errorRelativo[length(resultados$errorAbsoluto)],"\n")
```

```
## Error absoluto estimado  =  2.106257e-07
```

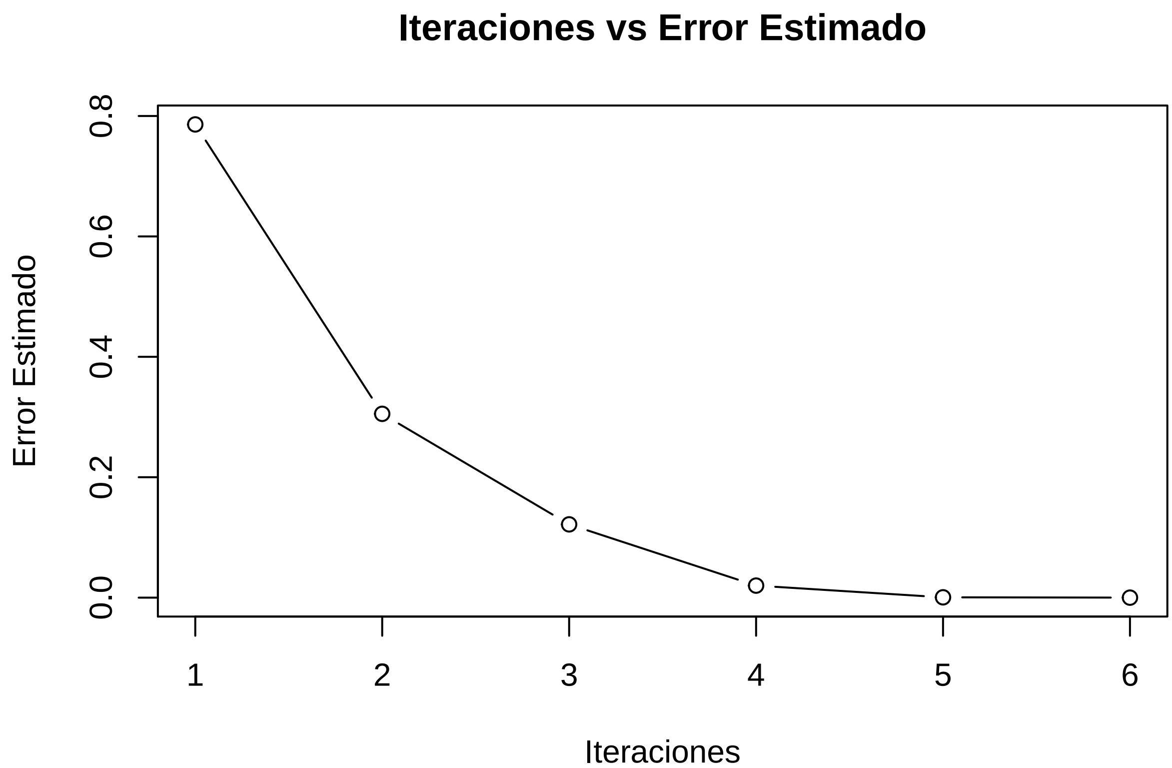
Grafica de las rectas tangentes

```r
curve(exp(x)-pi*x, 1,3)
title(main="Rectas Tangentes")
abline(h=0, v=0)

fx = function(x) exp(x)-pi*x
g = parse(text="exp(x)-pi*x") # parse devuelve tipo "expression"
g. = D(g,"x")
fp = function(x){eval(g.)} # convertir f' a función


for(x0 in resultados$x){
  curve(fp(x0)*(x-x0)+fx(x0), 0,3, add = T,col=rainbow(9), )
}
```



Grafica de iteraciones vs error estimado

```
plot(x = 1:length(resultados$x), y = resultados$errorAbsoluto,
     xlab = "Iteraciones", ylab = "Error Estimado", type="b",
     main = "Iteraciones vs Error Estimado")
```

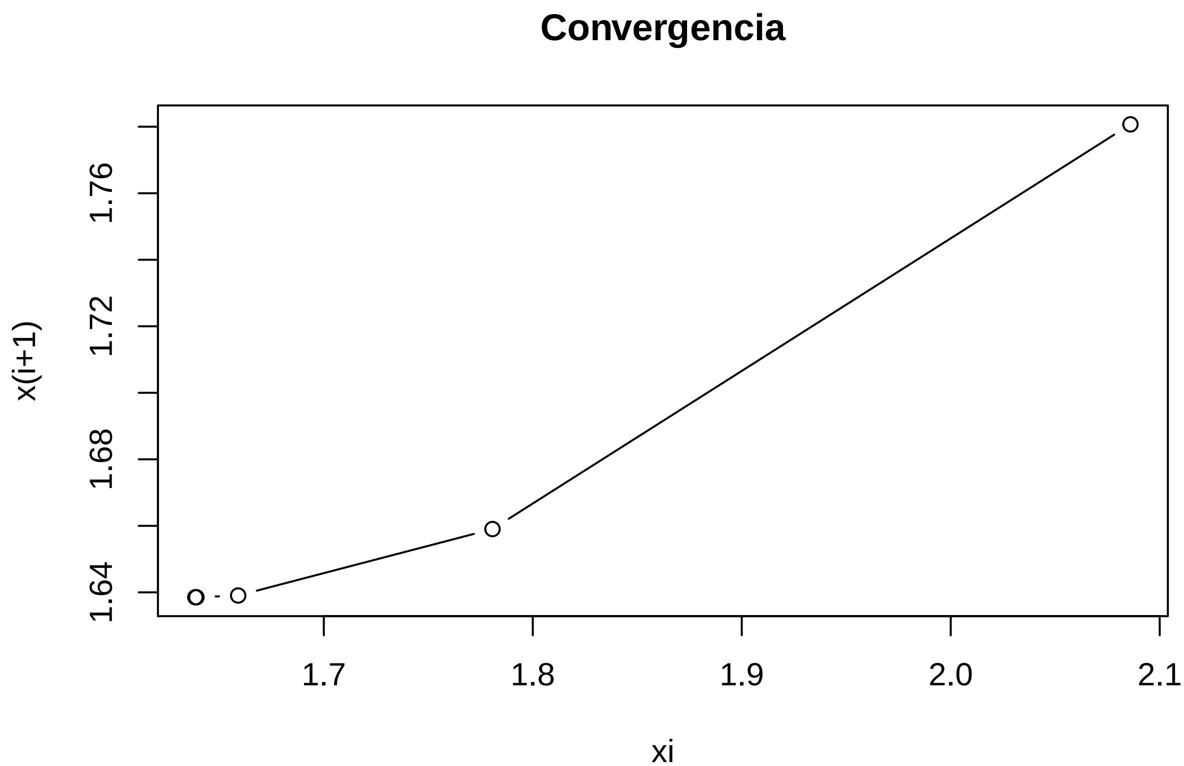


El error estimado siempre converge a cero.

$$Errorestimado : fx(x_0)/fp(x_0)$$

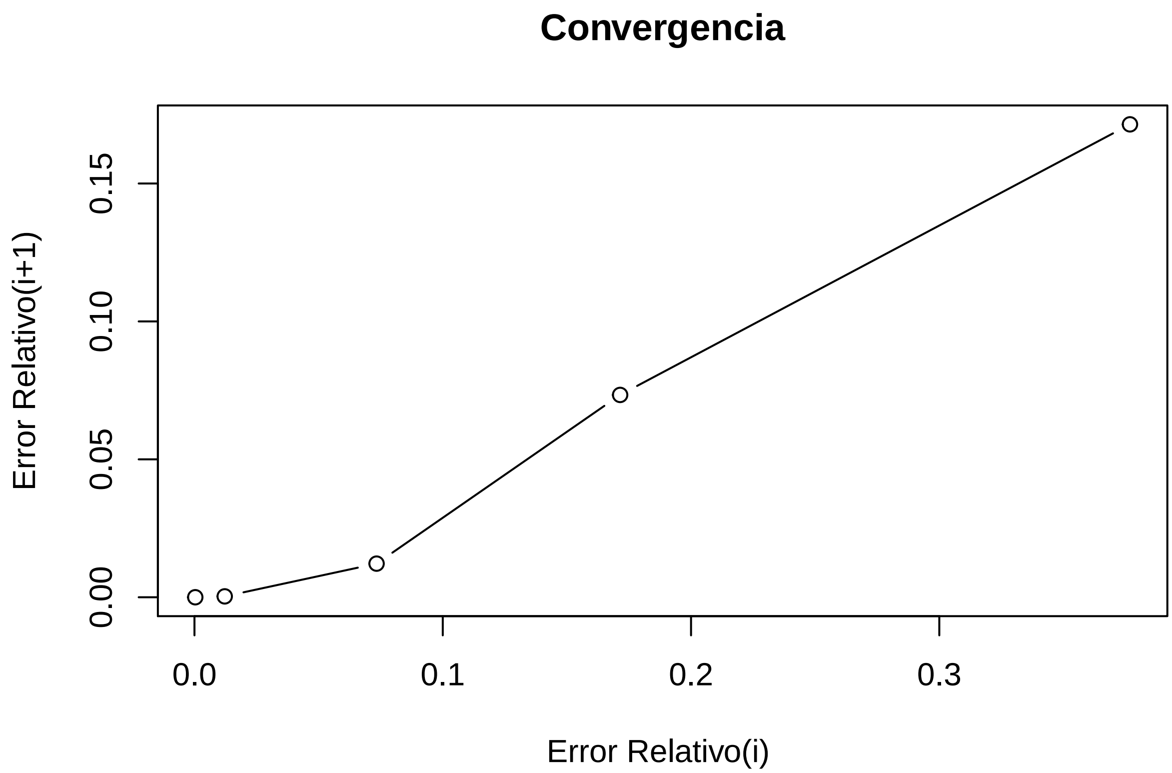
Grafica de x(i) vs x(i+1)

```
m_i =  resultados$x[-length(resultados$x)]
m_i2 = resultados$x
m_i2 = m_i2[-1]

plot(x =m_i, y =m_i2, xlab = "xi", ylab = "x(i+1)", type="b",main = "Convergencia")
```



Grafica de Error Relativo(i) vs Error Relativo(i+1)

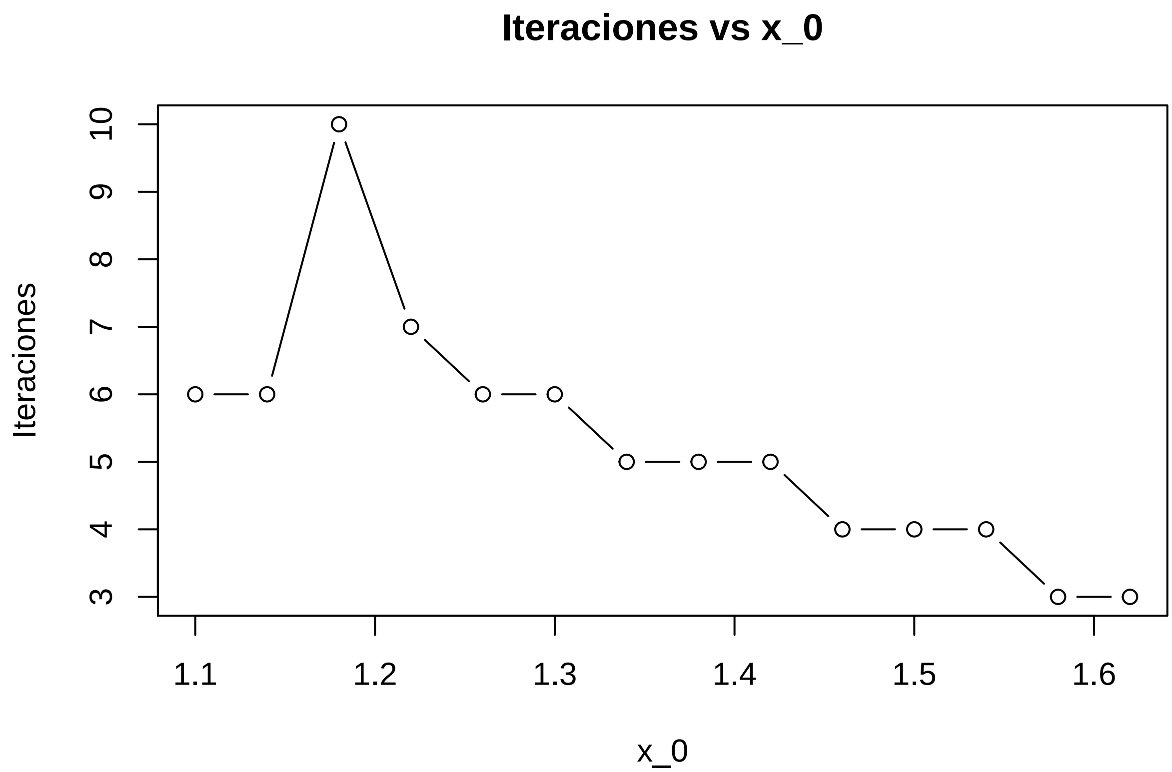
```
m_i =  resultados$errorRelativo[-length(resultados$errorRelativo)]
m_i2 = resultados$errorRelativo
m_i2 = m_i2[-1]

plot(x =m_i, y =m_i2, xlab = "Error Relativo(i) ",
     ylab = "Error Relativo(i+1)", type="b",main = "Convergencia")
```



De acuerdo con las gráficas el método tiene una convergencia cuadratica. Ademas en la tabla de resultados se puede apreciar como aproximadamente por cada iteración se duplican el número de cifras significativas correctas.

Grafica de iteraciones vs x0 (Aproximacion por la izquierda)

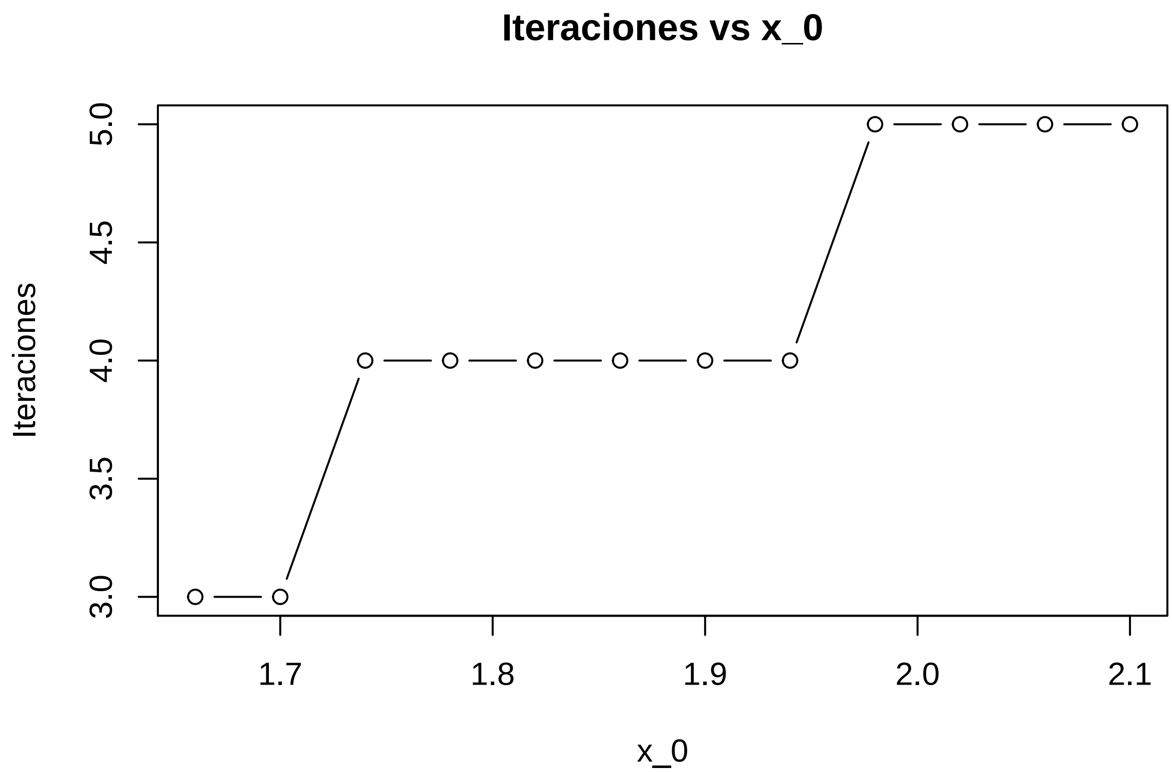
```
l=1.10
v = c()
iteraciones = c()
while(l<1.64) {
  v <- c(v,l)
  resultados<-newtonraphson("exp(x)-pi*x",l, 1e-8, 100)

  tablaErrores <- data.frame(
    "iteraciones" = 1:length(resultados$errorAbsoluto),
    "x_n" = resultados$x,
    "errorAbsoluto" = resultados$errorAbsoluto,
    "errorRelactivo" = resultados$errorRelativo
  )
  iteraciones <- c(iteraciones,length(resultados$errorAbsoluto))
  l = l+0.04
  #print(tablaErrores)
}
plot(x =v, y =iteraciones, xlab = "x_0", ylab = "Iteraciones",
     type="b",main = "Iteraciones vs x_0")
```



Mientras mas alejado se encuentre x_0 del x^* son necesarias mas cantidad de iteraciones.

Grafica de iteraciones vs x0 (Aproximacion por la derecha)

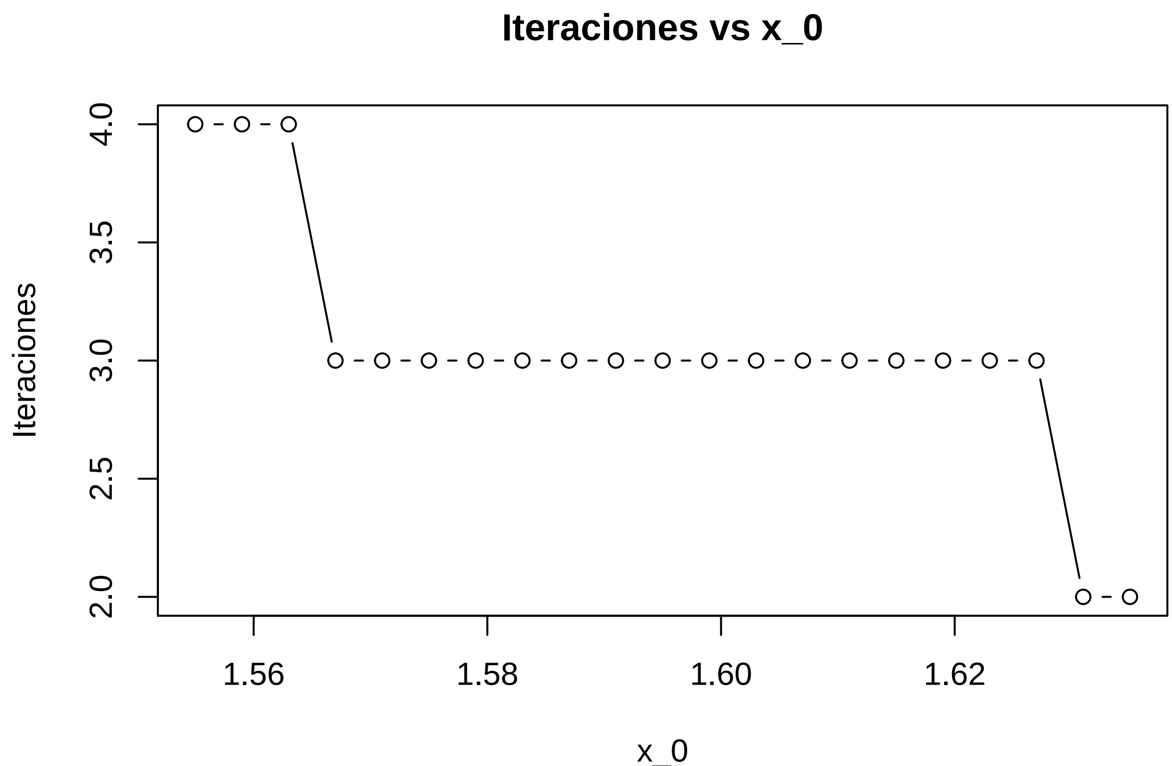
```
l=2.10
v = c()
iteraciones = c()
while(l>1.64) {
  v <- c(v,l)
  resultados<-newtonraphson("exp(x)-pi*x",l, 1e-8, 100)

  tablaErrores <- data.frame(
    "iteraciones" = 1:length(resultados$errorAbsoluto),
    "x_n" = resultados$x,
    "errorAbsoluto" = resultados$errorAbsoluto,
    "errorRelactivo" = resultados$errorRelativo
  )
  iteraciones <- c(iteraciones,length(resultados$errorAbsoluto))
  l = l-0.04
  #print(tablaErrores)
}
plot(x =v, y =iteraciones, xlab = "x_0", ylab = "Iteraciones",
     type="b",main = "Iteraciones vs x_0")
```



A comparación de la grafica de aproximación por la izquierda encontramos que la aproximación por derecha es mucho mas eficiente en la cantidad de iteraciones realizadas.

Grafica de iteraciones vs x0 (Aproximacion de un x_0 cercano a la raiz)

```
l=1.555
v = c()
iteraciones = c()
while(l<1.638) {
  v <- c(v,l)
  resultados<-newtonraphson("exp(x)-pi*x",l, 1e-8, 100)

  tablaErrores <- data.frame(
    "iteraciones" = 1:length(resultados$errorAbsoluto),
    "x_n" = resultados$x,
    "errorAbsoluto" = resultados$errorAbsoluto,
    "errorRelactivo" = resultados$errorRelativo
  )
  iteraciones <- c(iteraciones,length(resultados$errorAbsoluto))
  l = l+0.004
  #print(tablaErrores)
}
plot(x =v, y =iteraciones, xlab = "x_0", ylab = "Iteraciones",
     type="b",main = "Iteraciones vs x_0")
```



Al realizar una aproximación de x_0 con un valor cercano a la raiz encontramos que el número de iteraciones no cambia con variaciones bajas de x_0(es este caso variamos x_0 de a 0.004).