Newton Steffensen

Sebastian Angarita, Hector Rodriguez, Aldemar Ramirez

4/8/2019

Problema

Hallar la raíz de una función a partir de un x_0 a traves del metodo de Newton en combinacion con el metodo de Steffensen.

Solución

Lenguaje de programación: R

Función principal Newton Steffensen

Parametros:

fun <- función

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

tol <- tolerancia minima que debe tener la función

maxiter <- cantidad maxima 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

```
newtonsteffensen = function(fun, x0, tol, maxiter){
  # f = string
  numiter = 0
  errorAbsoluto = c()
  errorRelativo = c()
  x = c()

  g = parse(text=fun) # parse devuelve tipo "expression"

  fx = function(x){eval(g)} # convertir f a función


  res <- fx(x0)

  correccion = -(res)^2/(fx(x0+res)-res)
```

```
  while (abs(correccion) >= tol && numiter <= maxiter) {
    numiter = numiter + 1
    if ((fx(x0+fx(x0))-fx(x0)) == 0) stop("DivisiÃ³n por cero")
    x1 = x0 + correccion
    x0 = x1

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

    res <- fx(x1)

    correccion = -(res)^2/(fx(x1+res)-res)

  }
  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)
    #return(list(cero = x0, f.cero = fx(x0), numeroiter=numiter, error.est = correccion))

  }
}

##-------------------------------------------------------------------------------

## SE REALIZA LA COPIA DEL METODO NEWTON RAPHSON CON EL OBJETIVO DE REALIZAR LA COMPARACION

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 de newtonsteffensen basicamente realiza las operaciones sin recurrir al uso de derivadas en comparacion con la de newtonraphson, 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.

En esta ocasion la variable llamada correccion dentro de la implementacion de la función newtonsteffensen en el codigo tendra una variacion con respecto a la funcion newtonraphson.

De esta manera, una aproximacion corregida de x_0 en la funcion newtonraphson se realiza como

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

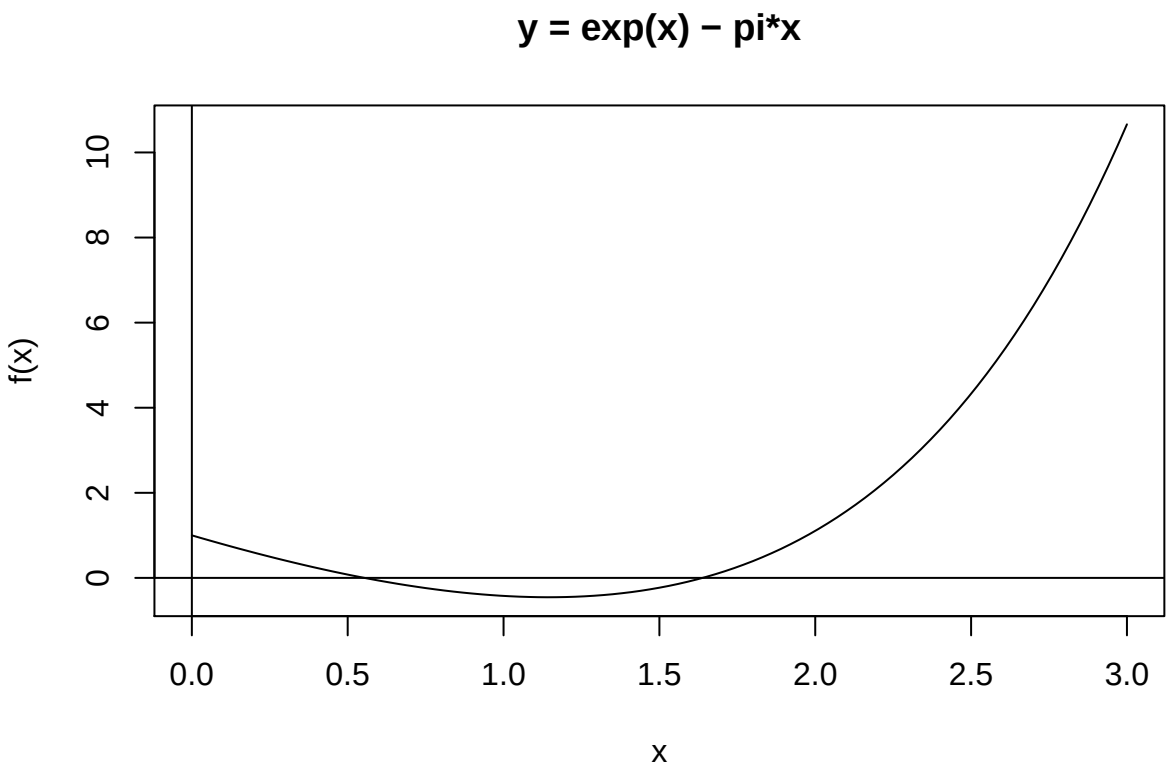
Mientras que en la funcion newtonsteffensen se realiza como

$$x_1 = x_0 - [f(x_0)]^2/[f(x_0 + f(x_0)) - f(x_0)]$$

Resultados

Gráfica de la función

```
f = function(x) exp(x) - pi*x
curve(f, 0,3); abline(h=0, v=0) #gráfico para decidir un intervalo
title(main="y = exp(x) - pi*x")
```



Se puede apreciar que una de las raices de la función se encuentra en el intervalo [1.4,2], por esta razón una buena aproximación para un x_0 seria $x_0 = 1.5$

Se muestra una tabla con la cantidad de iteraciones y el resultado de cada una para la función $y = e^x - pi * x$. Se realizan separaciones para diferenciar la implementación de la función newton_steffensen de la newton_raphson.

```
## --- Pruebas
# recibe la función como una tira
## Newton Steffensen
resultados<-newtonsteffensen("exp(x)-pi*x",1.5, 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 1.768038   2.680384e-01    1.516021e-01
## 2           2 1.685838   8.220031e-02    4.875932e-02
## 3           3 1.646118   3.972022e-02    2.412963e-02
## 4           4 1.638746   7.372151e-03    4.498654e-03
## 5           5 1.638529   2.171210e-04    1.325097e-04
## 6           6 1.638528   1.820057e-07    1.110788e-07
```

```r
cat("Resultado: iteraciones",length(resultados$errorAbsoluto),"\n")
```

```
## Resultado: iteraciones 6
```

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

```
## x =  1.638528
```

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

```
## Error estimado <=  1.110788e-07
```

```r
##-------------------------------------------------------------------------------------------------

## Newton Raphson
resultado2<-newtonraphson("exp(x)-pi*x",1.5, 1e-8, 100)

tablaErrores2 <- data.frame(
  "Iteraciones" = 1:length(resultado2$errorAbsoluto),
  "x_n" = resultado2$x,
  "Error Absoluto" = resultado2$errorAbsoluto,
  "Error Relactivo" = resultado2$errorRelativo
)
print(tablaErrores2)
```

```
##   Iteraciones      x_n Error.Absoluto Error.Relactivo
## 1           1 1.672152   1.721517e-01    1.029522e-01
## 2           2 1.639892   3.225952e-02    1.967173e-02
## 3           3 1.638531   1.361402e-03    8.308674e-04
## 4           4 1.638528   2.380175e-06    1.452630e-06
```

```r
cat("Numero de iteraciones",length(resultado2$errorAbsoluto),"\n")
```

```
## Numero de iteraciones 4
```

```r
cat("x = ",resultado2$resultado,"\n")
```

```
## x =  1.638528
```

```r
cat("Error estimado = ", resultado2$errorRelativo[length(resultado2$errorAbsoluto)],"\n")
```

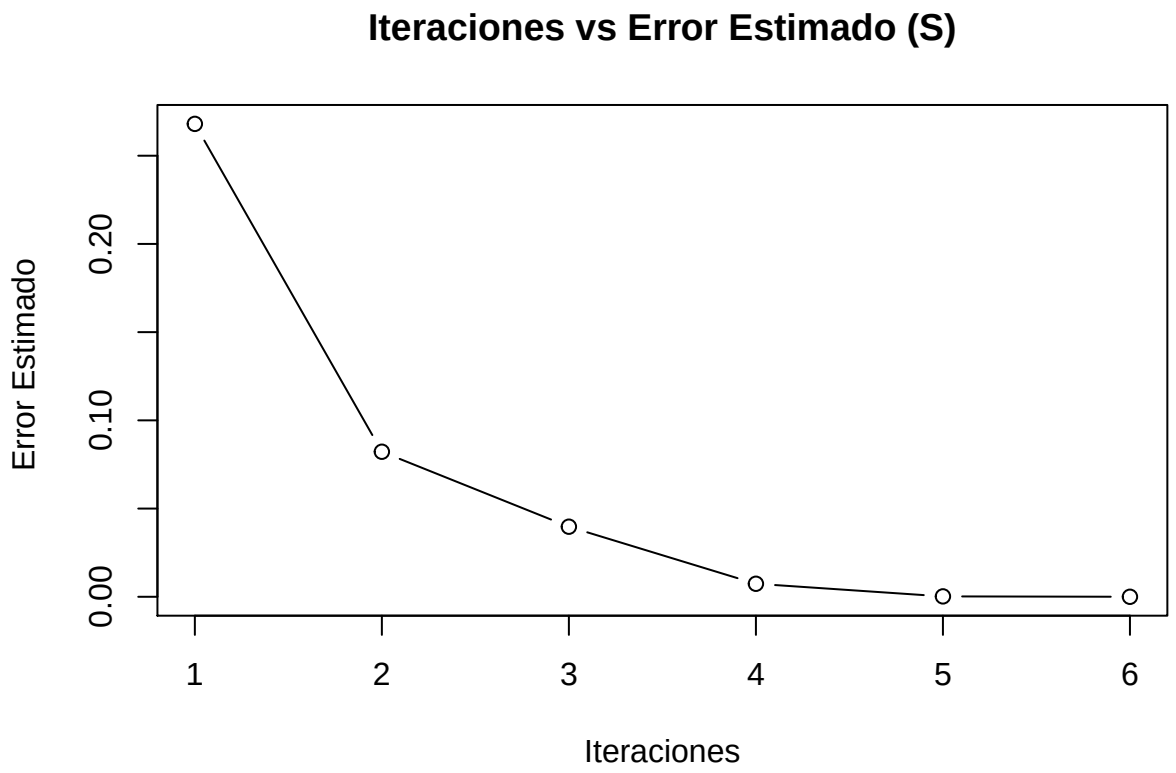
```
## Error estimado =  1.45263e-06
```

En un primer analisis observamos que la función newton_Steffensen realiza mas iteraciones, aunque se ahorra la complejidad de realizar la derivada de la función.

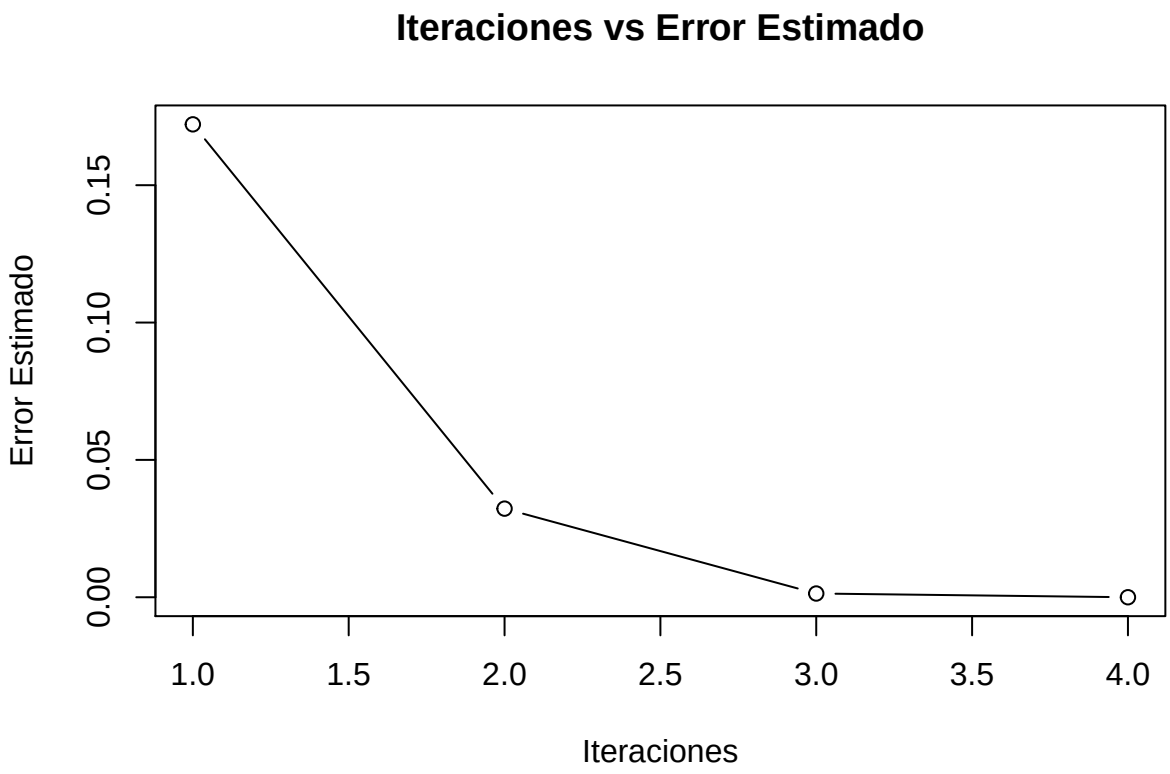
Grafica de iteraciones vs error estimado

Las graficas que representan la implementación de la función newton_steffensen se identifican con una (S) en el titulo.

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




```r
##-------------------------------------------------------------------------------------------
plot(x = 1:length(resultado2$x), y = resultado2$errorAbsoluto,
     xlab = "Iteraciones", ylab = "Error Estimado",
     type="b", main = "Iteraciones vs Error Estimado")
```



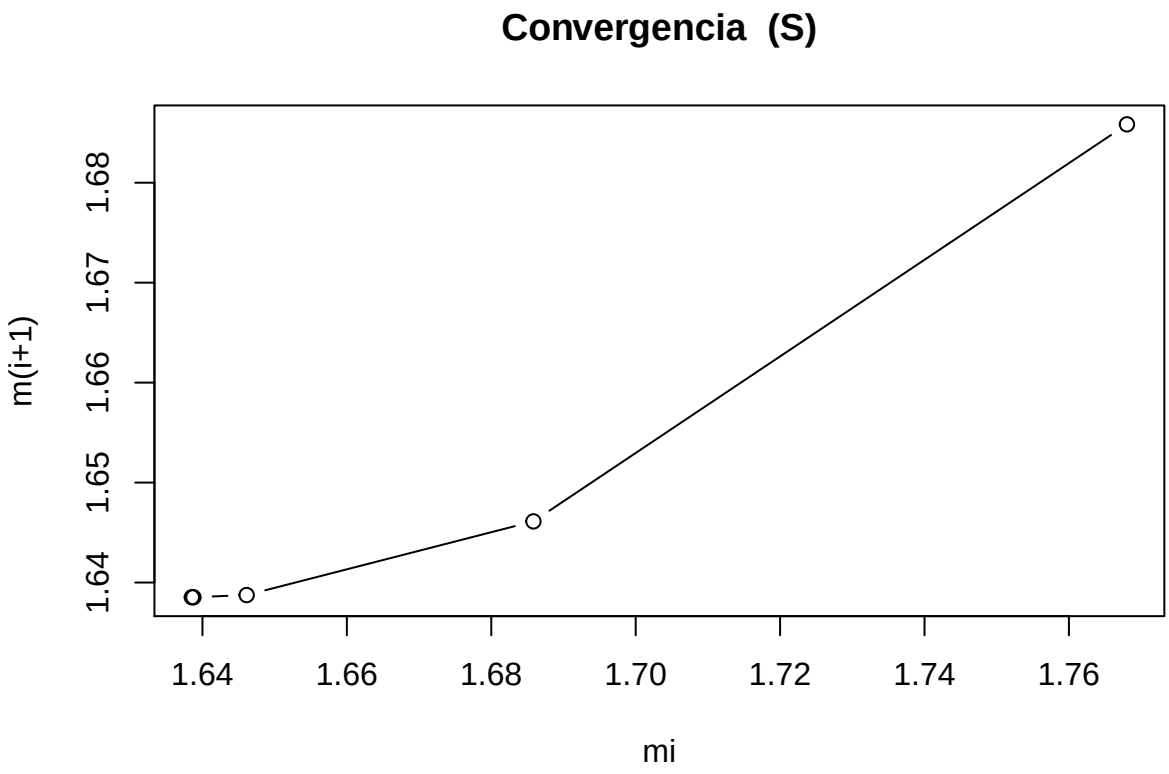
El error absoluto estimado en ambos casos converge a 0, aunque en el método de Newton_Raphson con menos iteraciones.

$Errorestimado : fx(x_0)/fp(x_0)$ para la función newtonraphson.

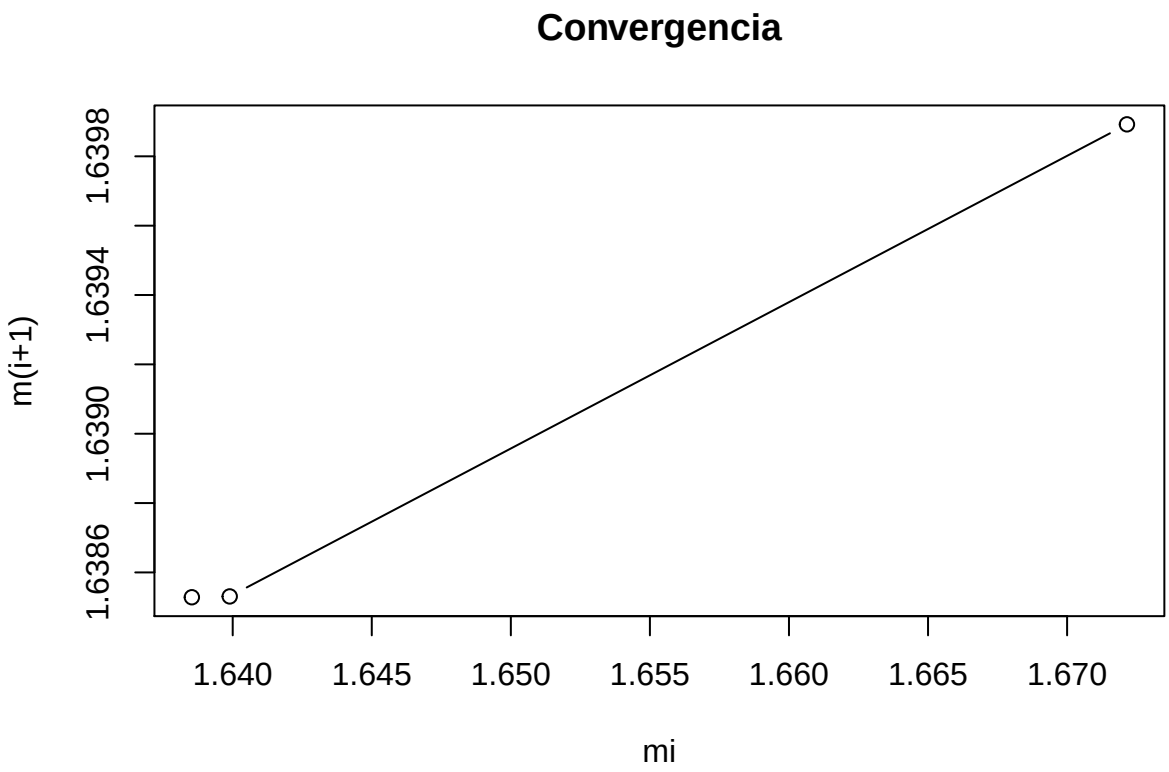
$Errorestimado : (res)^2/(fx(x0 + res) - res)$ para la función newtonsteffensen donde $res = fx(x0)$.

Grafica de m(i) vs m(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 = "mi", ylab = "m(i+1)", type="b",main = "Convergencia  (S)")
```




```
##-------------------------------------------------------------------------------------------
m_i =  resultado2$x[-length(resultado2$x)]
m_i2 = resultado2$x
m_i2 = m_i2[-1]

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

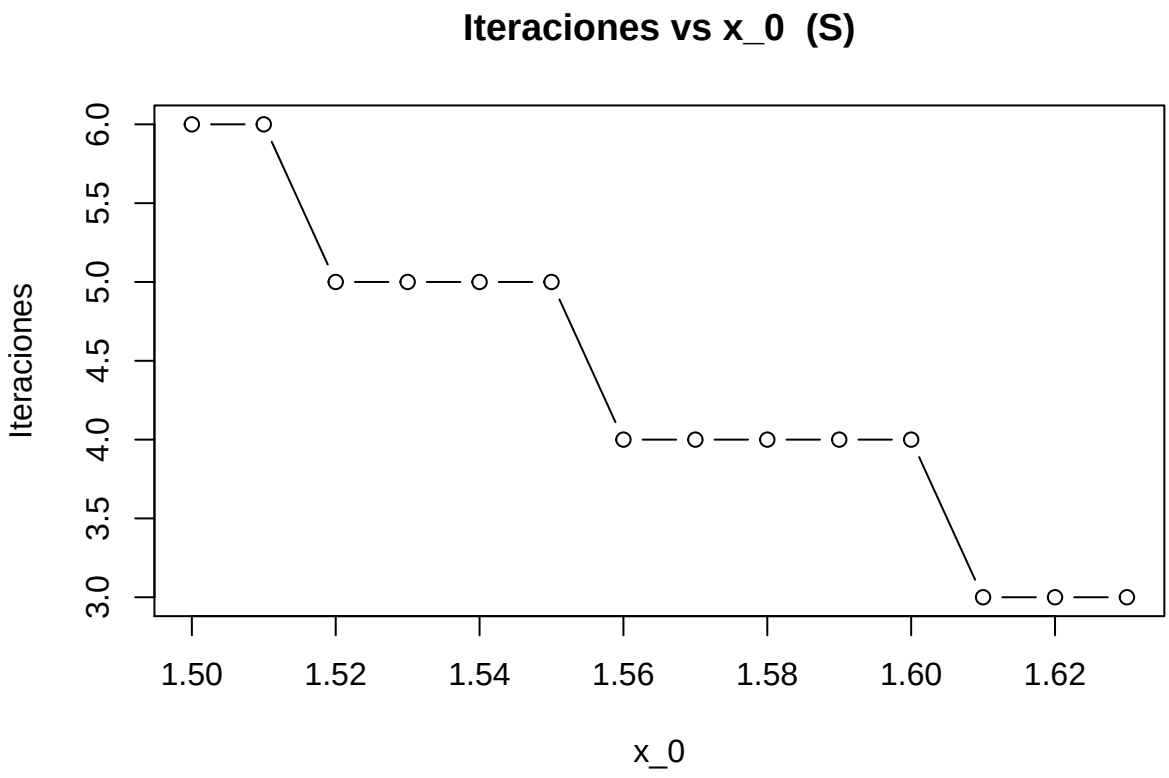


De acuerdo con la grafica el metodo tiene una convergencia cuadratica.

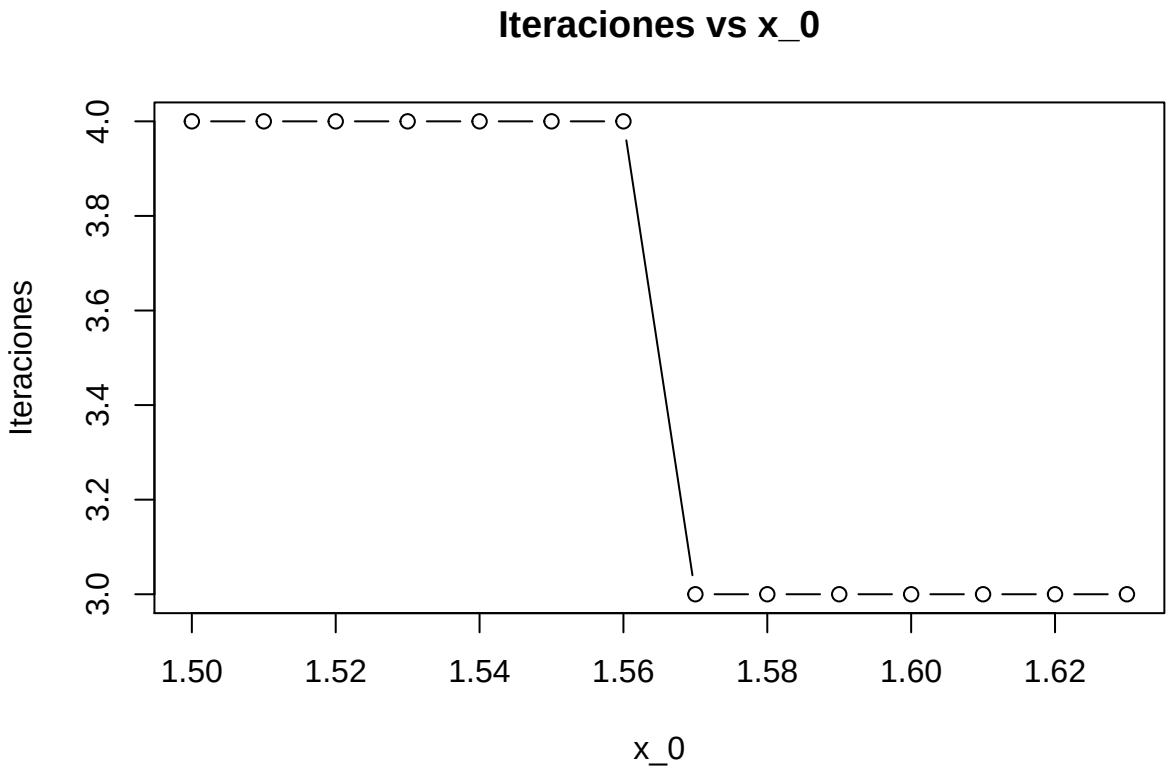
Con la función newtonsteffensen se observa mucho mejor la convergencia cuadratica del Método Newton.

Grafica de iteraciones vs x0

```
ls=1.5
vs = c()
iteracioness = c()
while(ls<1.64) {
  vs <- c(vs,ls)
  resultados<-newtonsteffensen("exp(x)-pi*x",ls, 1e-8, 100)

  tablaErrores <- data.frame(
    "iteraciones" = 1:length(resultados$errorAbsoluto),
    "x_n" = resultados$x,
    "errorAbsoluto" = resultados$errorAbsoluto,
    "errorRelactivo" = resultados$errorRelativo
  )
  iteracioness <- c(iteracioness,length(resultados$errorAbsoluto))
  ls = ls+0.01
  #print(tablaErrores)
}
plot(x =vs, y =iteracioness, xlab = "x_0", ylab = "Iteraciones", type="b",main = "Iteraciones vs x_0  (S
```




```
##-------------------------------------------------------------------------------------------

l=1.5
v = c()
iteraciones = c()
while(l<1.64) {
  v <- c(v,l)
  resultado2<-newtonraphson("exp(x)-pi*x",l, 1e-8, 100)

  tablaErrores <- data.frame(
    "iteraciones" = 1:length(resultado2$errorAbsoluto),
    "x_n" = resultado2$x,
    "errorAbsoluto" = resultado2$errorAbsoluto,
    "errorRelactivo" = resultado2$errorRelativo
  )
  iteraciones <- c(iteraciones,length(resultado2$errorAbsoluto))
  l = l+0.01
  #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.

La implementacion de la función newtonsteffensen a comparacion de la newtonraphson es más "sensible" en cuanto a la escogencia el x_0. Si en Newton_Raphson la diferencia ente escoger un x_0 alejado en 0.5 de la raíz lleva a una iteración de más, en newton_steffensen puede llegar a ganar dos iteraciones de más.