Método de Newton aplicado a funciones parametrizadas

Juan Angarita, Hector Hernandez, Aldemar Ramirez

Agosto 8,2019

Problema

Hallar la itercepción una función de 3 variables con el eje z.

Solución

Lenguaje de programación: R

Función principal - método de Método de Newton

Parametros:

fun <- función

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

tol <- tolerancia minima que debe tener la función

maxiter <- cantidad maxima de iteraciones

Valores de retorno:

x1 <- resultado de la raiz (parametro t).

errorAbsoluto <- vector de errores absolutos de las x_n

errorRelativo <- vector de errores relativos de las x_n

x <- vector de las x_n

```r
library(pracma)

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


    } else {
    my_list <- list("resultado" = x1, "errorAbsoluto" = errorAbsoluto,
                    "errorRelativo" = errorRelativo, "x" = x, "i" = numiter)
    return(my_list)
  }
}
```

El método en general nos da la siguiente serie:

$$x_{(n+1)} = x_n - f(x_n)/f'(x_n)$$

Ya que la función está parametrizada el método devuelve el parametro t donde la función de intercepta con el eje z.

Implementación

Caso 1: interecepción de gráfica en R3 con el eje z.

Funcion en R^3 parametrizada:

$$a(t) = (cos(t) - 1, sin(t), t)$$

Para hallar la intercepción con el eje z seria un punto en la función tal que:

$$a(t) = (0, 0, t)$$

Por ende:

$$cos(t) - 1 = 0, sin(t) = 0$$

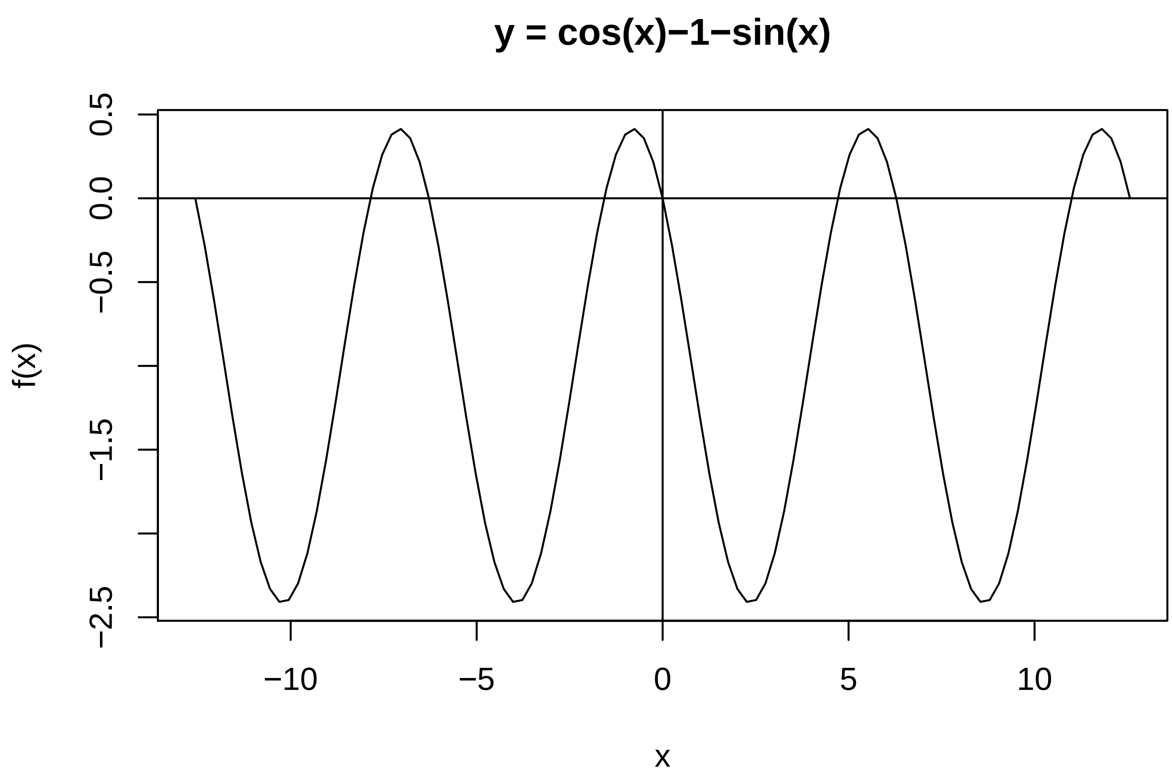
Igualando ambas ecuaciones tenemos:

$$((cos(t) - 1 - (sin(t)) = 0)$$

Esta ecuación es la que hay que resolver a través del método de Newton.

Gráficas

```r
f = function(x) cos(x)-1-sin(x)
curve(f, -4*pi,4*pi); abline(h=0, v=0) #gráfico para decidir un intervalo
title(main="y = cos(x)-1-sin(x)")
```



A través de la gráfica se puede apreciar que la función tiene sus ceros en las cercanias de los valores $2pi * n$.

```r
resultados <- newtonraphson("cos(x)-1-sin(x)",0.9,1e-7,100)

g = parse(text="cos(x)-1")

fx = function(x){eval(g)}

cat("Ambas ecuaciones en t = : ", resultados$resultado,
    " con error absoluto: ",resultados$errorAbsoluto[resultados$i] ,"\n")
```

```
## Ambas ecuaciones en t = :  3.928209e-12  con error absoluto:  2.802926e-06
```

```r
resultados <- newtonraphson("cos(x)-1-sin(x)",5/6*pi,1e-7,100)

g = parse(text="cos(x)-1")

fx = function(x){eval(g)}

cat("Ambas ecuaciones en t = : ", resultados$resultado,
    " con error absoluto: ",resultados$errorAbsoluto[resultados$i] ,"\n")
```

```
## Ambas ecuaciones en t = :  12.56637  con error absoluto:  0.000289788
```

Podemos generalizar: la ecuación se intercepta con el eje z en valores de:

$$2 * pi * n$$

con n un número perteneciente a los naturales.

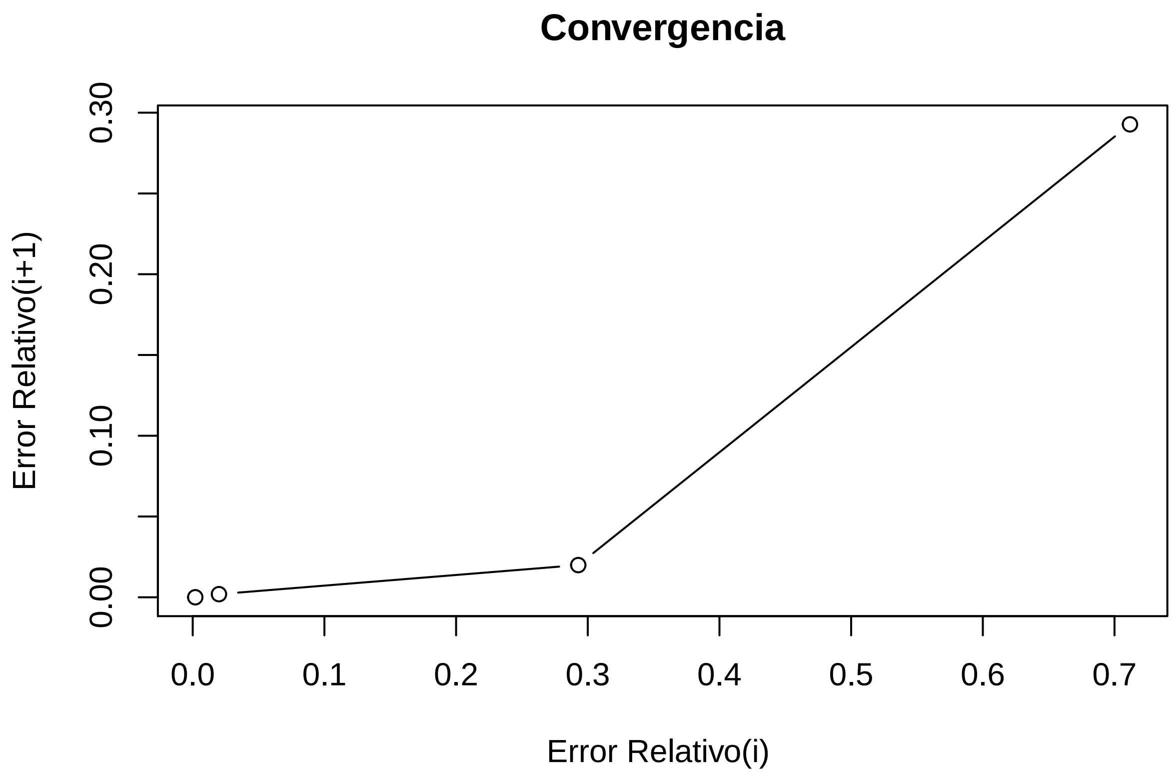
Tabla de resultados

```r
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  9.082095    6.464101615    0.7117412077
## 2           2 12.841940    3.759844728    0.2927785571
## 3           3 12.590941    0.250999189    0.0199349031
## 4           4 12.566660    0.024280588    0.0019321432
## 5           5 12.566371    0.000289788    0.0000230606
```

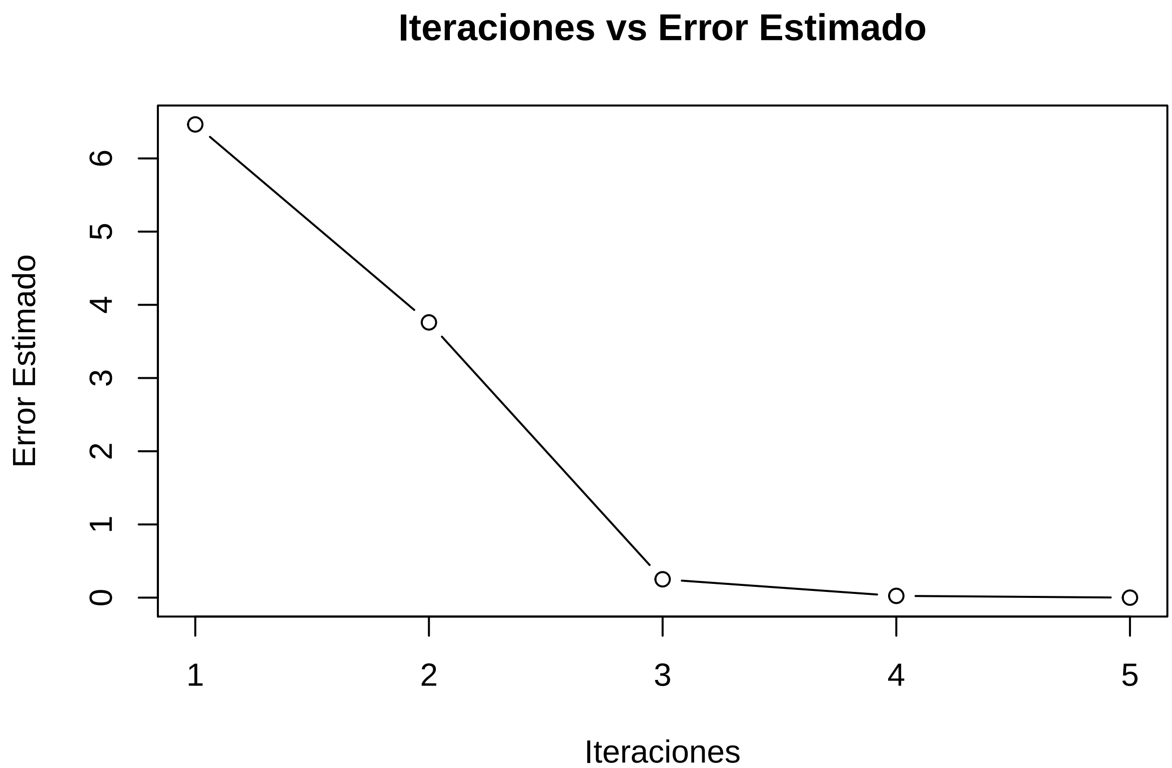
Convergencia

```r
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")
```



El método tiene una convergencia cuadratica.

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



A través de las iteraciones el error tiende a cero.

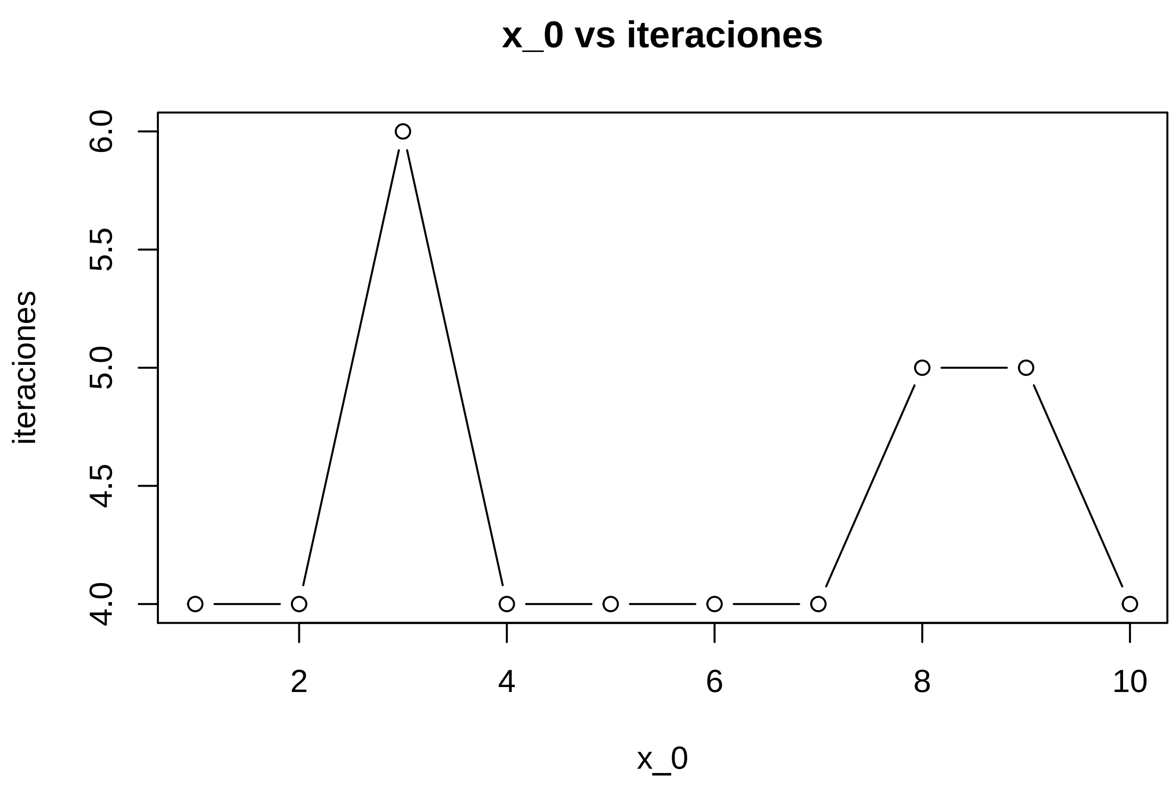
Modificando el x_0

```
a = c(1:10)
iter = c()
error =c()
resultados = c()
n = 1

for(k in a){
  iter = c(iter,newtonraphson("cos(x)-1-sin(x)", a[n], 1e-7, 100)$i)
    error = c(error,
              newtonraphson("cos(x)-1-sin(x)", a[n], 1e-7, 100)$errorAbsoluto[iter[n]])
  resultados = c(resultados,
                 newtonraphson("cos(x)-1-sin(x)", a[n], 1e-7, 100)$resultado)
  n = n+1
}

tabla <- data.frame(
  "x_0"= c(1:10),
  "resultado"= resultados,
  "iteraciones" = iter,
  "Error est." = error
```

```
)

print(tabla)
```

```
##    x_0     resultado iteraciones   Error.est.
## 1    1  7.110261e-13           4 1.192482e-06
## 2    2 -1.570796e+00           4 9.249861e-05
## 3    3  2.984513e+01           6 2.766304e-05
## 4    4  4.712389e+00           4 3.326567e-06
## 5    5  4.712389e+00           4 2.595293e-06
## 6    6  6.283185e+00           4 2.223430e-06
## 7    7  6.283185e+00           4 3.359291e-06
## 8    8 -6.283185e+00           5 1.571544e-06
## 9    9  1.256637e+01           5 1.598294e-07
## 10  10  1.099557e+01           4 1.263711e-06
```

```
plot(x = c(1:10), y = iter,
xlab = "x_0", ylab = "iteraciones", type="b",
main = "x_0 vs iteraciones")
```



Como estamos trabajando con funciones periodicas, la función es cercanda al eje z $2 * pi * n$.