Newton-Raphson

Uriel Paluch

5/9/2021

Método de Newton-Raphson

El método de la forma presentada en el libro esta basada en los polinomios de Taylor.

Supongamos que p_0 existe en un intervalo $[a; b]$ y es una aproximación para p, de tal forma que $f'(p_0) \neq 0$ y $|p - p_0|$ es "pequeño". Consideremos e primer polinomio de Taylor expandido de p_0 y evaluado en $x = p$:

$$f(p) = f(p_0) + (p - p_0) * f'(p_0) + \frac{(p - p_0)^2}{2} * f''(\xi(p))$$

Puesto que, $f(p) = 0$ y $(p - p_0)^2$ es pequeño, reexpresamos de modo que:

$$p = p_0 - \frac{f(p_0)}{f'(p_0)}$$

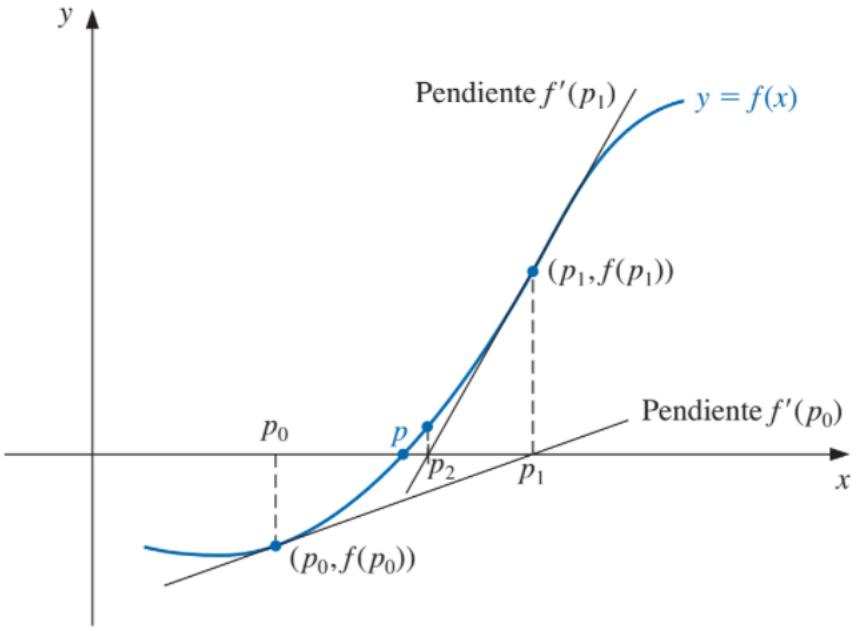


Figure 1: Método de Newton-Raphson

```
Newton <- function(p0, tol, n = 100){
  #Donde p0 es la aproximación inicial
  #El número máximo de iteraciones n viene por default en 100
```

```
  #Y tol es la toleranacia al error

  for (i in 1:n) {

    #Calculo p
    p <- p0 - (f(p0)/fprima(p0))

    if(abs(p-p0) <= tol){
      return(p)
    }

    p0 <- p

  }

  #En el caso de que falle el método
  return(paste('El método falla luego de: ', n, ' iteraciones'))
}
```

En el libro se presenta como el método mas eficaz ¿eso es realmente así? sí, pero hay que tener algo en cuenta. Al comienzo supusimos que el término $(p - p_0)^2$ es tan pequeño, que es despreciable. Esto implica que la primera aproximación de p, necesariamente, es buena.

Ejercicios:

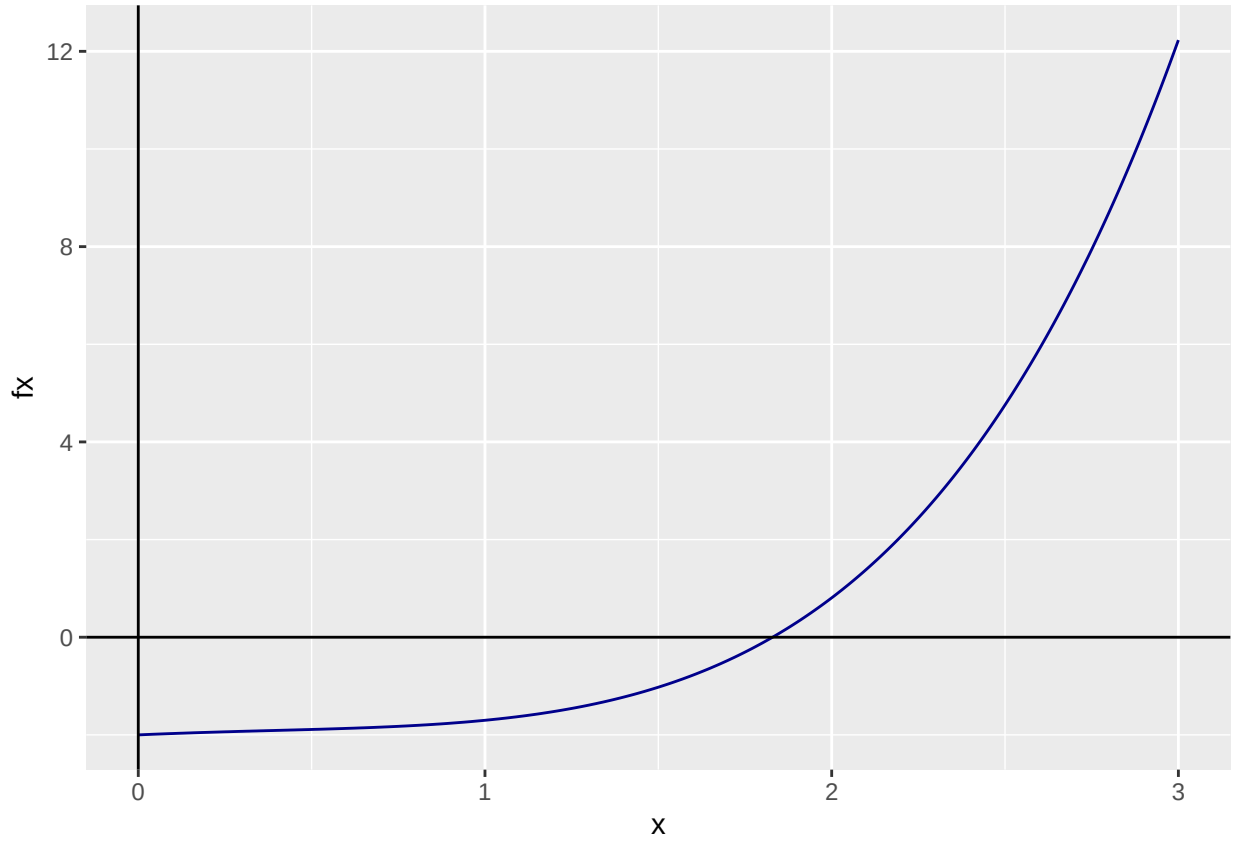
- Hallar las soluciones de (si es posible):
  1. $e^x + 2^{-x} + 2cos(x) - 6 = 0 \quad 1 \leq x \leq 2$
  2. $ln(x-1) + cos(x-1) = 0 \quad 1.3 \leq x \leq 2$
  3. $2x * cos(2x) - (x-2)^2 = 0 \quad 2 \leq x \leq 3 \text{ and } 3 \leq x \leq 4$
  4. $(x-2)^2 - ln(x) = 0 \quad 1 \leq x \leq 2 \text{ and } e \leq x \leq 4$
  5. $e^x - 3x^2 = 0 \quad 0 \leq x \leq 1 \text{ and } 3 \leq x \leq 5$
  6. $sin(x) - e^{-x} = 0 \quad 0 \leq x \leq 1 \text{ and } 3 \leq x \leq 4 \text{ and } 6 \leq x \leq 7$
  7. $cos(x) = \sqrt{(x)}$
  8. $2 + cos(e^x - 2) = e^x$
  9. $x^3 - 7x^2 + 14x - 6 = 0$
  10. $-x^3 - cos(x) = 0$

Solución:

Ejercicio 1:

```
f <- function(x){
  return(exp(x) + 2^(-x) + 2*cos(x)-6)
}

#Instancio un vector que me va a indicar los puntos en la función
x <- seq(0, 3, by = 0.01)

#Genero los puntos
fx <- f(x)

#Creo un data frame con los x e y
df <- data.frame(x, fx)

#Instancio los datos
```

```r
gg_fx <- ggplot(data = df)

#Agrego la capa con los datos
gg_fx <- gg_fx + aes(x = x, y = fx)

#Est grafica una linea
gg_fx <- gg_fx + geom_line(linetype = 1, colour = "darkblue")

#Agrego el eje X
gg_fx <- gg_fx + geom_vline(xintercept = 0, linetype = 1)

#Agrego el eje Y
gg_fx <- gg_fx + geom_hline(yintercept = 0, linetype = 1)

#Grafico
gg_fx
```




```r
fprima <- D(expression(exp(x) + 2^(-x) + 2*cos(x)-6), "x")

fprima
```

```
## exp(x) - 2^(-x) * log(2) - 2 * sin(x)
```

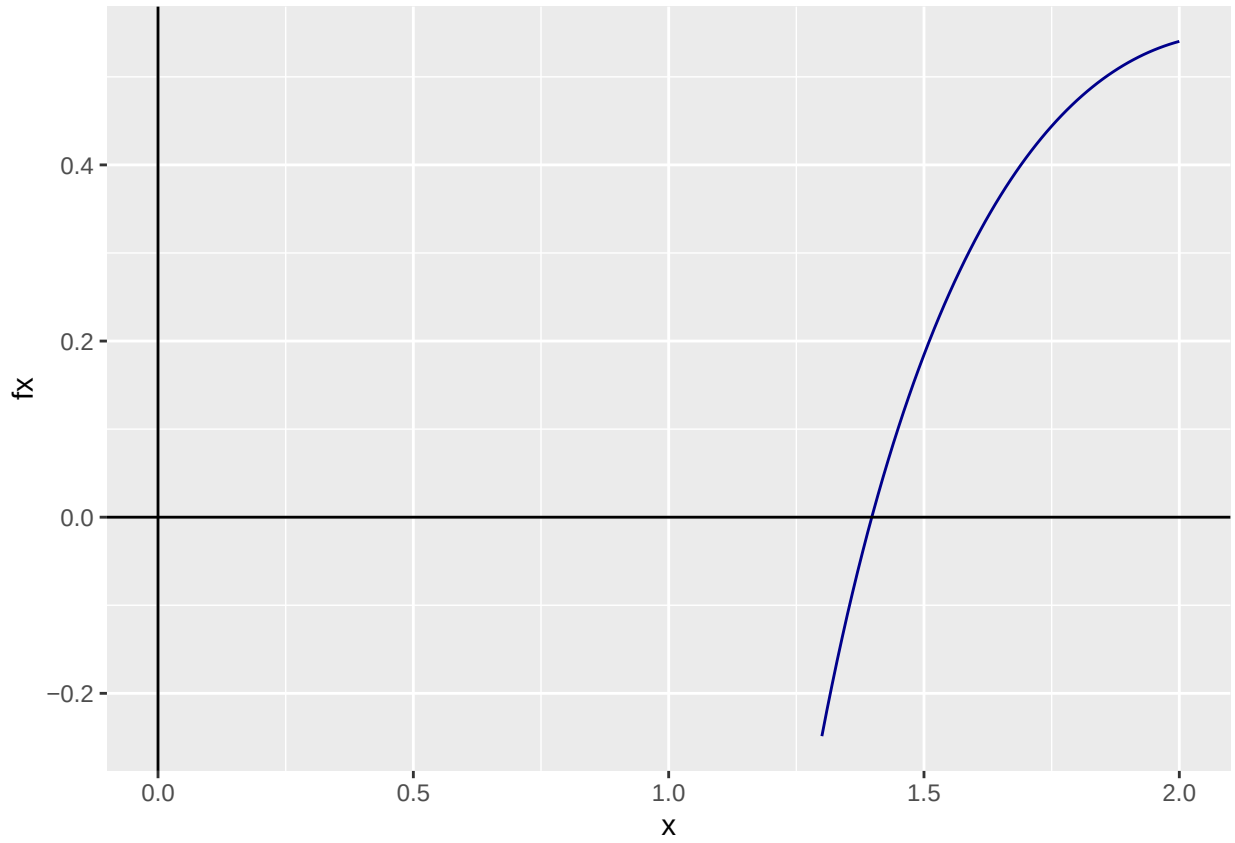
```r
fprima <- function(x){
  return(exp(x) - 2^(-x) * log(2) - 2 * sin(x))
}
```

```
Newton(p0 = 1.5, tol = 0.001)
```

```
## [1] 1.829384
```

Ejercicio 2:

```
f <- function(x){
  return( log(x-1) + cos(x-1))
}

#Instancio un vector que me va a indicar los puntos en la función
x <- seq(1.3, 2, by = 0.01)

#Genero los puntos
fx <- f(x)

#Creo un data frame con los x e y
df <- data.frame(x, fx)

#Instancio los datos
gg_fx <- ggplot(data = df)

#Agrego la capa con los datos
gg_fx <- gg_fx + aes(x = x, y = fx)

#Est grafica una linea
gg_fx <- gg_fx + geom_line(linetype = 1, colour = "darkblue")

#Agrego el eje X
gg_fx <- gg_fx + geom_vline(xintercept = 0, linetype = 1)

#Agrego el eje Y
gg_fx <- gg_fx + geom_hline(yintercept = 0, linetype = 1)

#Grafico
gg_fx
```




```
fprima <- D(expression(log(x-1) + cos(x-1)), "x")

fprima
```

```
## 1/(x - 1) - sin(x - 1)
```

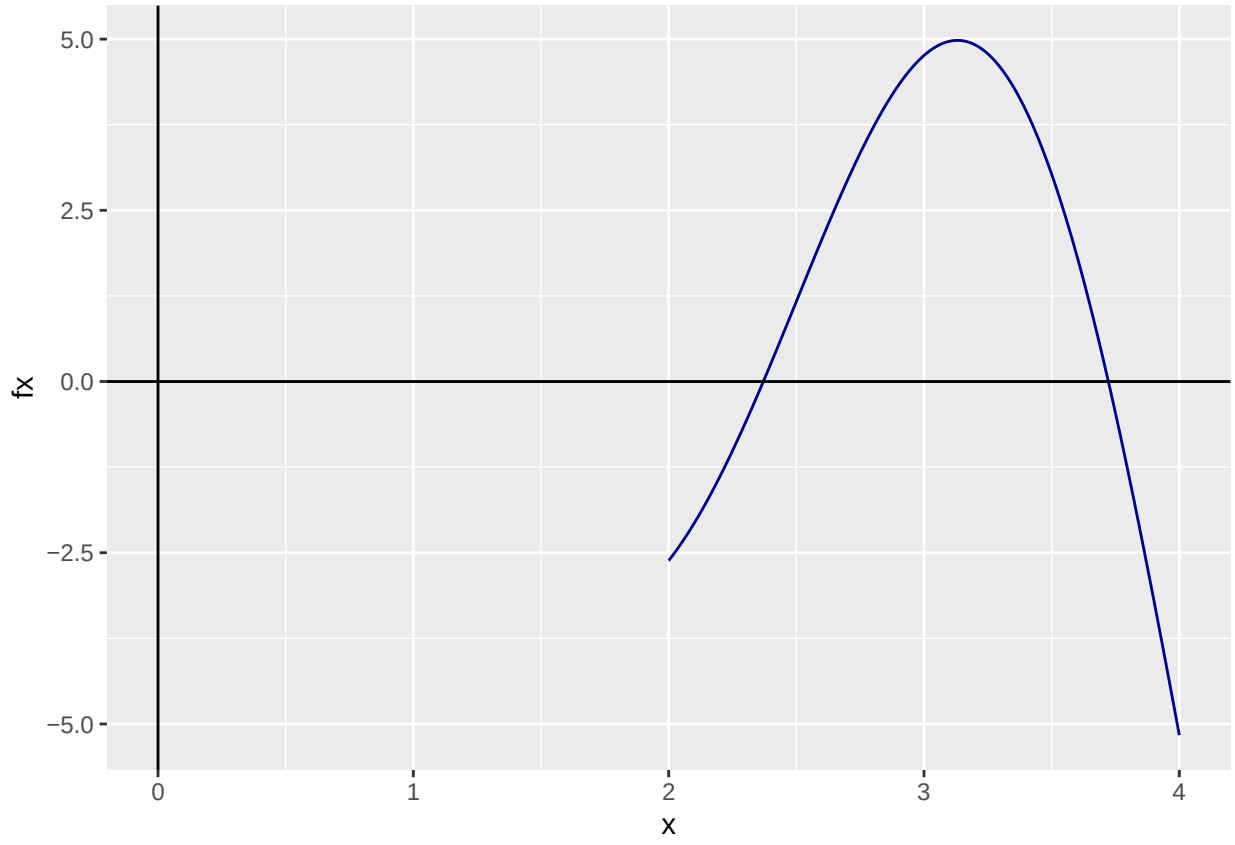
```
fprima <- function(x){
  return(1/(x - 1) - sin(x - 1))
}

Newton(p0 = 1.5, tol = 0.001)
```

```
## [1] 1.397748
```

Ejercicio 3:

```
f <- function(x){
  return( 2*x * cos(2*x) - (x-2)^2)
}

#Instancio un vector que me va a indicar los puntos en la función
x <- seq(2, 4, by = 0.01)

#Genero los puntos
fx <- f(x)

#Creo un data frame con los x e y
df <- data.frame(x, fx)
```

```r
#Instancio los datos
gg_fx <- ggplot(data = df)

#Agrego la capa con los datos
gg_fx <- gg_fx + aes(x = x, y = fx)

#Est grafica una linea
gg_fx <- gg_fx + geom_line(linetype = 1, colour = "darkblue")

#Agrego el eje X
gg_fx <- gg_fx + geom_vline(xintercept = 0, linetype = 1)

#Agrego el eje Y
gg_fx <- gg_fx + geom_hline(yintercept = 0, linetype = 1)

#Grafico
gg_fx
```




```r
fprima <- D(expression(2*x * cos(2*x) - (x-2)^2), "x")

fprima
```

```
## 2 * cos(2 * x) - 2 * x * (sin(2 * x) * 2) - 2 * (x - 2)
```

```r
fprima <- function(x){
  return(2 * cos(2 * x) - 2 * x * (sin(2 * x) * 2) - 2 * (x - 2))
```

```
}

Newton(p0 = 2.2, tol = 0.001)
```

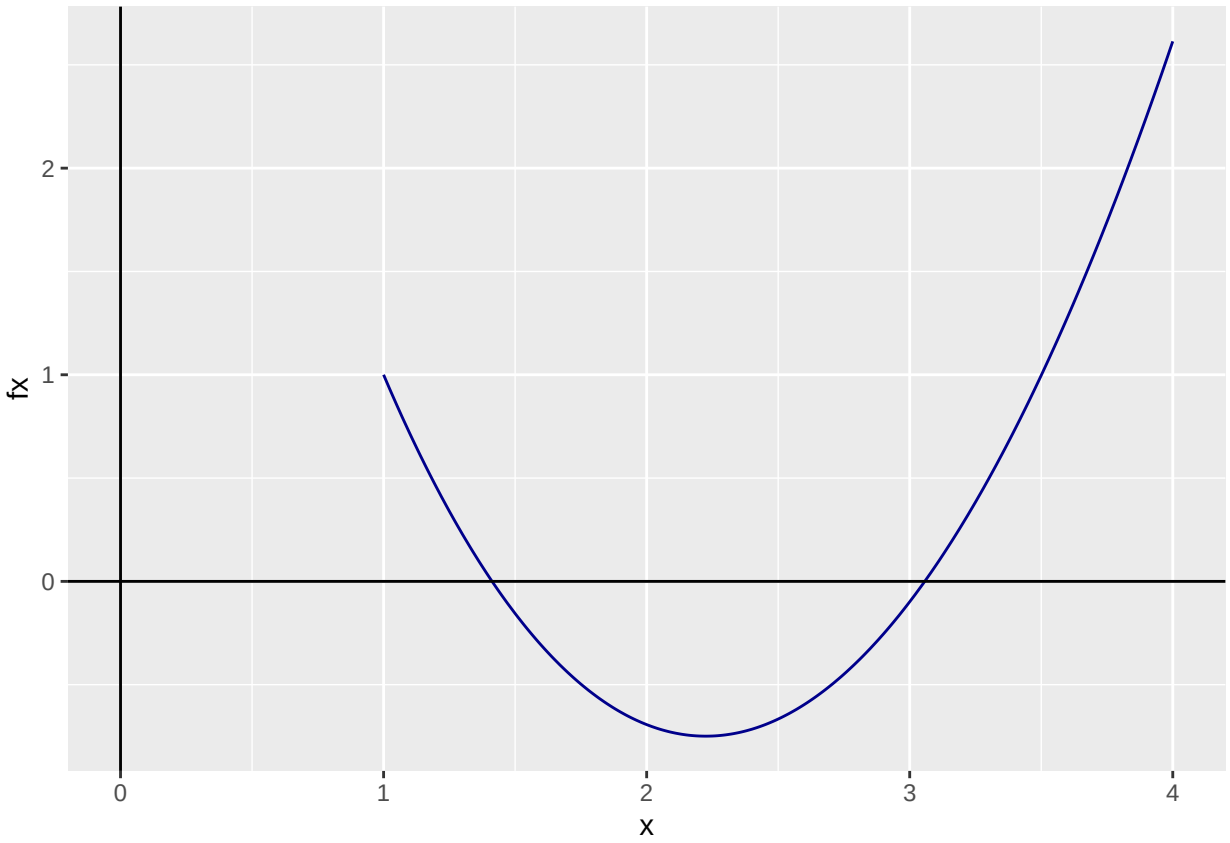
```
## [1] 2.370687
```

```
Newton(p0 = 3.7, tol = 0.001)
```

```
## [1] 3.722113
```

Ejercicio 4:

```
f <- function(x){
  return((x-2)^2 - log(x))
}

#Instancio un vector que me va a indicar los puntos en la función
x <- seq(1, 4, by = 0.01)

#Genero los puntos
fx <- f(x)

#Creo un data frame con los x e y
df <- data.frame(x, fx)

#Instancio los datos
gg_fx <- ggplot(data = df)

#Agrego la capa con los datos
gg_fx <- gg_fx + aes(x = x, y = fx)

#Est grafica una linea
gg_fx <- gg_fx + geom_line(linetype = 1, colour = "darkblue")

#Agrego el eje X
gg_fx <- gg_fx + geom_vline(xintercept = 0, linetype = 1)

#Agrego el eje Y
gg_fx <- gg_fx + geom_hline(yintercept = 0, linetype = 1)

#Grafico
gg_fx
```




```
fprima <- D(expression((x-2)^2 - log(x)), "x")

fprima
```

```
## 2 * (x - 2) - 1/x
```

```
fprima <- function(x){
  return(2 * (x - 2) - 1/x)
}

Newton(p0 = 1.5, tol = 0.001)
```

```
## [1] 1.412391
```

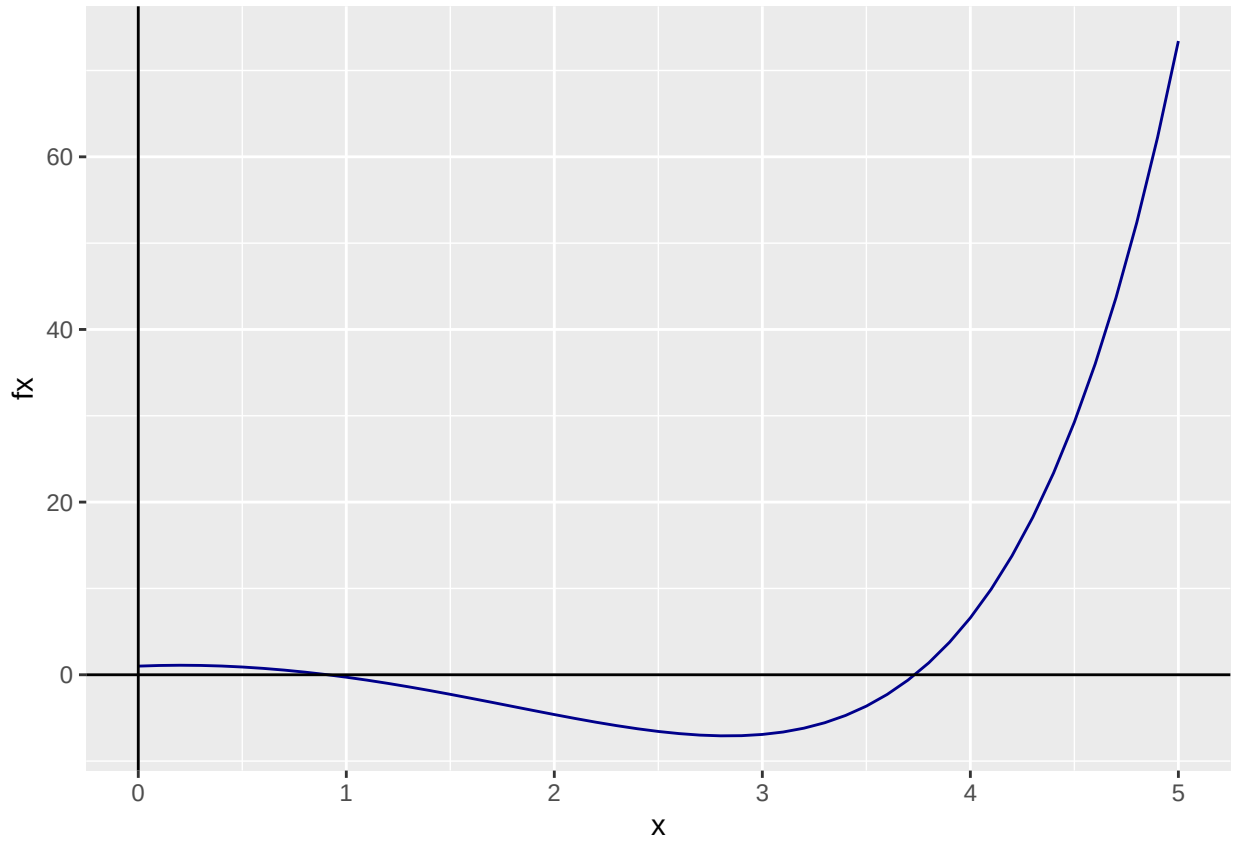
```
Newton(p0 = 3, tol = 0.001)
```

```
## [1] 3.057104
```

Ejercicio 5:

```
f <- function(x){
  return(exp(x) - 3*x^2)
}

#Instancio un vector que me va a indicar los puntos en la función
x <- seq(0, 5, by = 0.1)

#Genero los puntos
fx <- f(x)
```

```r
#Creo un data frame con los x e y
df <- data.frame(x, fx)

#Instancio los datos
gg_fx <- ggplot(data = df)

#Agrego la capa con los datos
gg_fx <- gg_fx + aes(x = x, y = fx)

#Est grafica una linea
gg_fx <- gg_fx + geom_line(linetype = 1, colour = "darkblue")

#Agrego el eje X
gg_fx <- gg_fx + geom_vline(xintercept = 0, linetype = 1)

#Agrego el eje Y
gg_fx <- gg_fx + geom_hline(yintercept = 0, linetype = 1)

#Grafico
gg_fx
```




```r
fprima <- D(expression(exp(x) - 3*x^2), "x")

fprima
```

```
## exp(x) - 3 * (2 * x)
```

```r
fprima <- function(x){
  return(exp(x) - 3 * (2 * x))
}

Newton(p0 = 1, tol = 0.001)
```

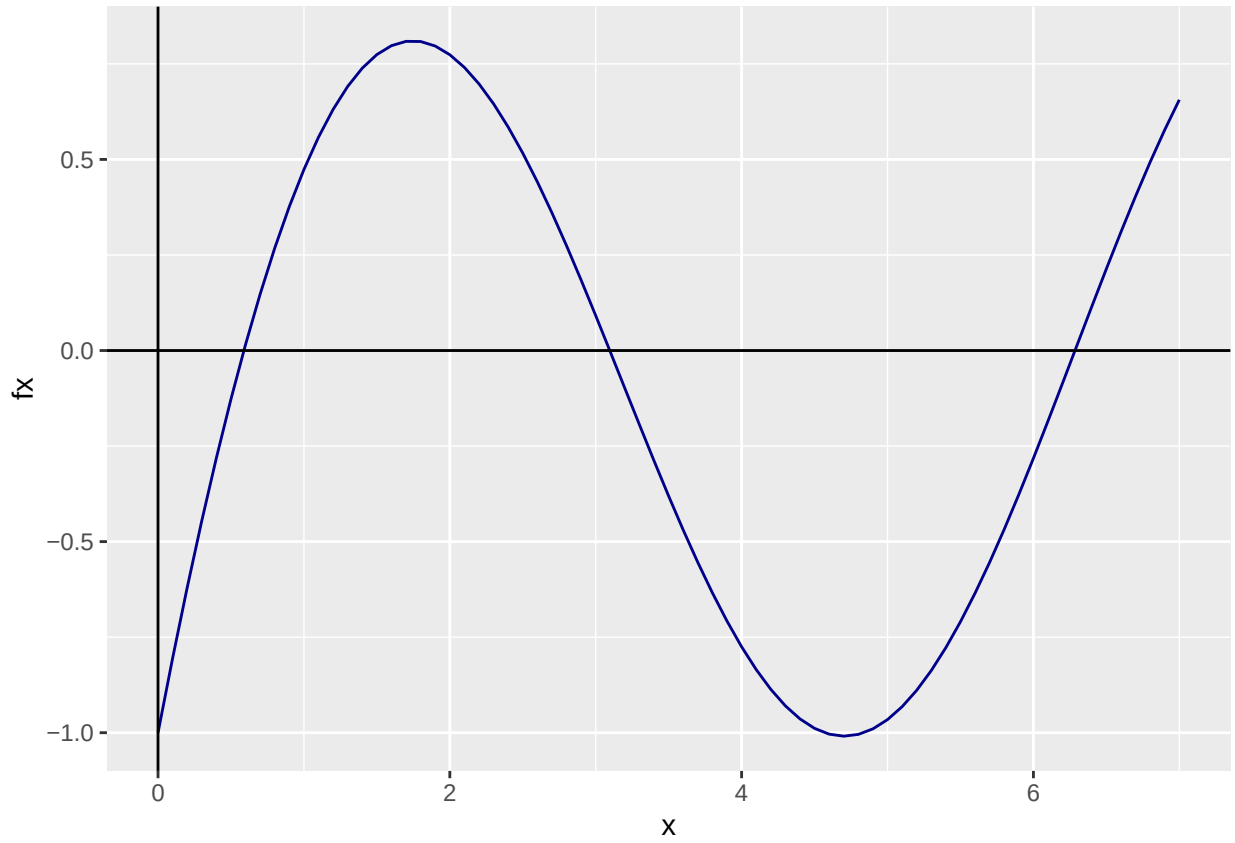
```
## [1] 0.9100076
```

```r
Newton(p0 = 3.5, tol = 0.001)
```

```
## [1] 3.733079
```

Ejercicio 6:

```r
f <- function(x){
  return(sin(x) - exp(-x))
}

#Instancio un vector que me va a indicar los puntos en la función
x <- seq(0, 7, by = 0.1)

#Genero los puntos
fx <- f(x)

#Creo un data frame con los x e y
df <- data.frame(x, fx)

#Instancio los datos
gg_fx <- ggplot(data = df)

#Agrego la capa con los datos
gg_fx <- gg_fx + aes(x = x, y = fx)

#Est grafica una linea
gg_fx <- gg_fx + geom_line(linetype = 1, colour = "darkblue")

#Agrego el eje X
gg_fx <- gg_fx + geom_vline(xintercept = 0, linetype = 1)

#Agrego el eje Y
gg_fx <- gg_fx + geom_hline(yintercept = 0, linetype = 1)

#Grafico
gg_fx
```




```
fprima <- D(expression(sin(x) - exp(-x)), "x")

fprima
```

```
## cos(x) + exp(-x)
```

```
fprima <- function(x){
  return(cos(x) + exp(-x))
}

Newton(p0 = 0.5, tol = 0.001)
```

```
## [1] 0.5885327
```

```
Newton(p0 = 3.5, tol = 0.001)
```

```
## [1] 3.096364
```

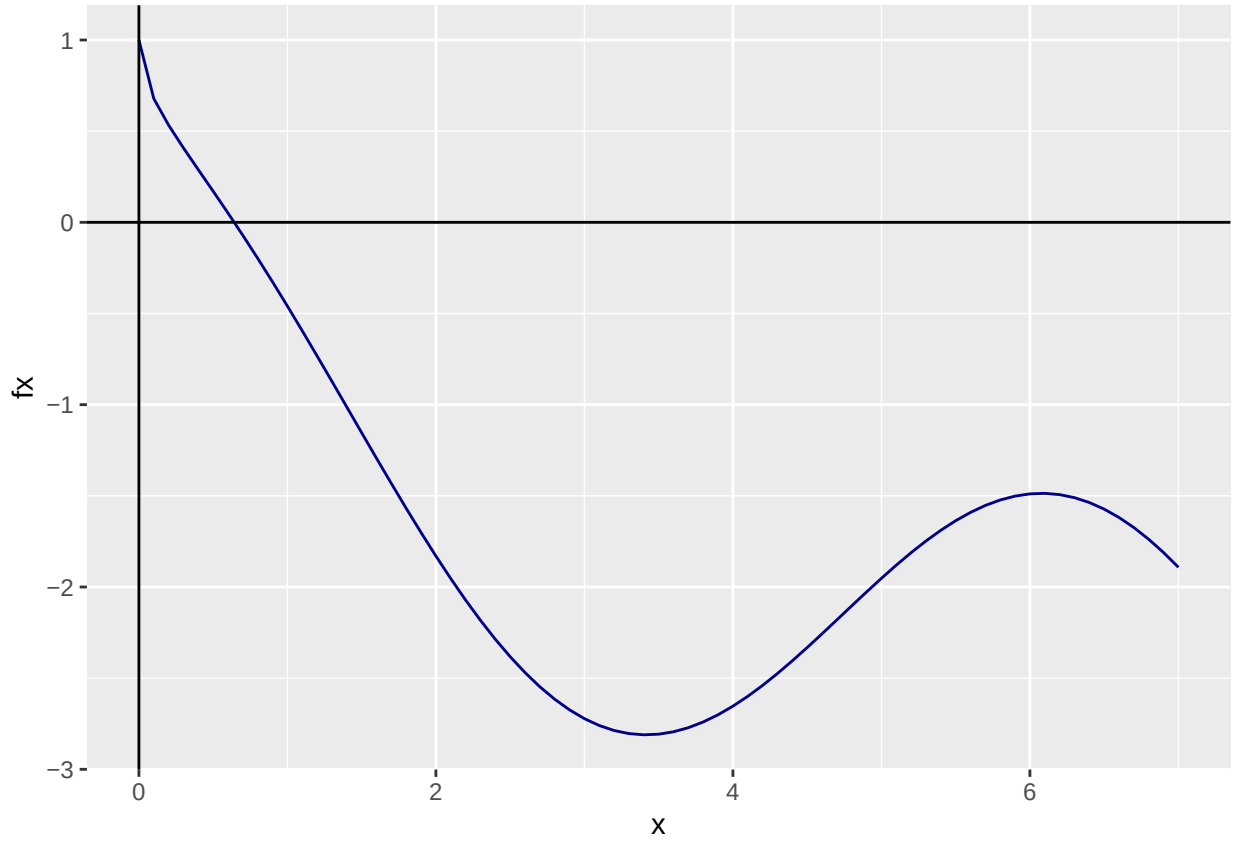
```
Newton(p0 = 6, tol = 0.001)
```

```
## [1] 6.285049
```

Ejercicio 7:

```
f <- function(x){
  return(cos(x) - sqrt(x))
}

#Instancio un vector que me va a indicar los puntos en la función
```

```r
x <- seq(0, 7, by = 0.1)

#Genero los puntos
fx <- f(x)

#Creo un data frame con los x e y
df <- data.frame(x, fx)

#Instancio los datos
gg_fx <- ggplot(data = df)

#Agrego la capa con los datos
gg_fx <- gg_fx + aes(x = x, y = fx)

#Est grafica una linea
gg_fx <- gg_fx + geom_line(linetype = 1, colour = "darkblue")

#Agrego el eje X
gg_fx <- gg_fx + geom_vline(xintercept = 0, linetype = 1)

#Agrego el eje Y
gg_fx <- gg_fx + geom_hline(yintercept = 0, linetype = 1)

#Grafico
gg_fx
```



```r
fprima <- D(expression(cos(x) - sqrt(x)), "x")

fprima
```

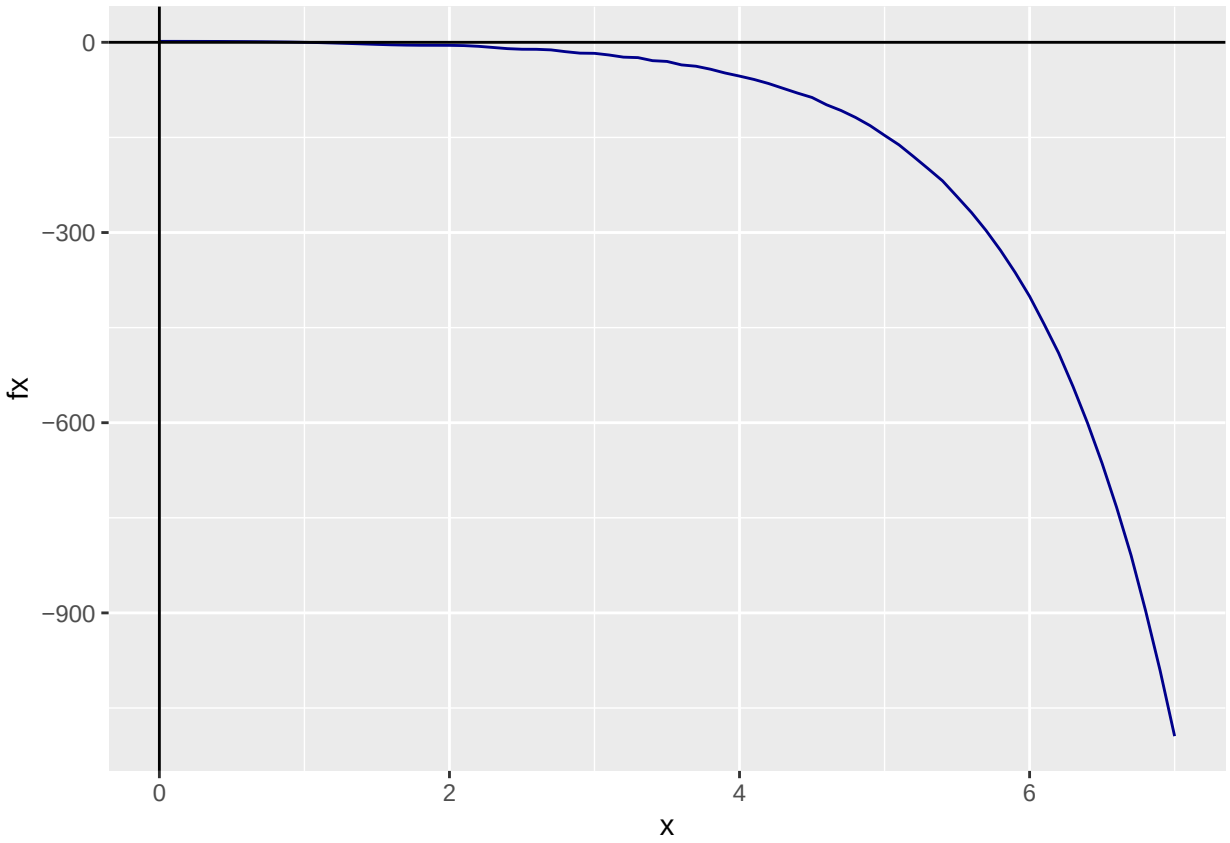
```
## -(sin(x) + 0.5 * x^-0.5)
```

```r
fprima <- function(x){
  return(-(sin(x) + 0.5 * x^-0.5))
}

Newton(p0 = 0.5, tol = 0.001)
```

```
## [1] 0.6417144
```

Ejercicio 8:

```r
f <- function(x){
  return(2 + cos(exp(x) - 2) - exp(x))
}

#Instancio un vector que me va a indicar los puntos en la función
x <- seq(0, 7, by = 0.1)

#Genero los puntos
fx <- f(x)

#Creo un data frame con los x e y
df <- data.frame(x, fx)

#Instancio los datos
gg_fx <- ggplot(data = df)

#Agrego la capa con los datos
gg_fx <- gg_fx + aes(x = x, y = fx)

#Est grafica una linea
gg_fx <- gg_fx + geom_line(linetype = 1, colour = "darkblue")

#Agrego el eje X
gg_fx <- gg_fx + geom_vline(xintercept = 0, linetype = 1)

#Agrego el eje Y
gg_fx <- gg_fx + geom_hline(yintercept = 0, linetype = 1)

#Grafico
gg_fx
```




```
fprima <- D(expression(2 + cos(exp(x) - 2) - exp(x)), "x")

fprima
```

```
## -(sin(exp(x) - 2) * exp(x) + exp(x))
```

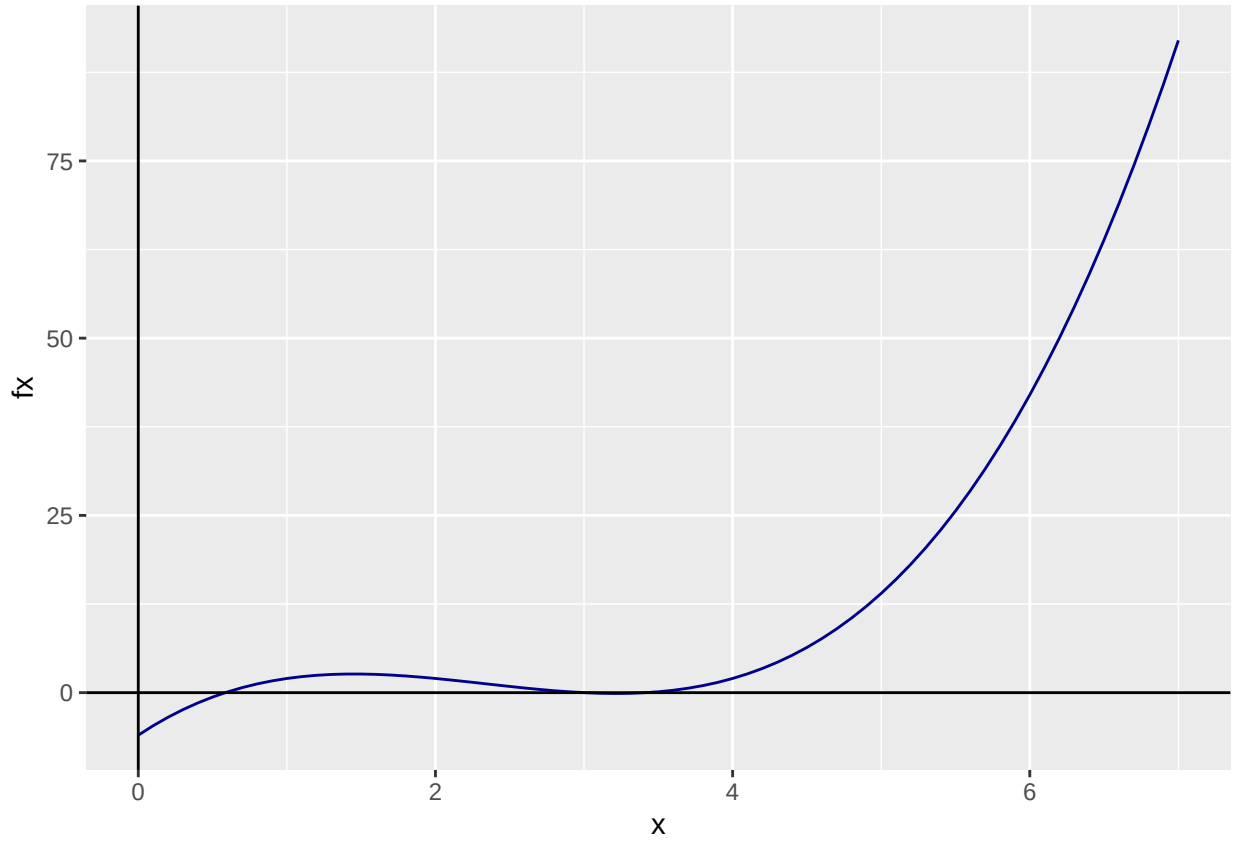
```
fprima <- function(x){
  return(-(sin(exp(x) - 2) * exp(x) + exp(x)))
}

Newton(p0 = 1, tol = 0.1)
```

```
## [1] 1.007689
```

Ejercicio 9:

```
f <- function(x){
  return(x^3 - 7*x^2 + 14*x - 6)
}

#Instancio un vector que me va a indicar los puntos en la función
x <- seq(0, 7, by = 0.1)

#Genero los puntos
fx <- f(x)

#Creo un data frame con los x e y
df <- data.frame(x, fx)
```

```r
#Instancio los datos
gg_fx <- ggplot(data = df)

#Agrego la capa con los datos
gg_fx <- gg_fx + aes(x = x, y = fx)

#Est grafica una linea
gg_fx <- gg_fx + geom_line(linetype = 1, colour = "darkblue")

#Agrego el eje X
gg_fx <- gg_fx + geom_vline(xintercept = 0, linetype = 1)

#Agrego el eje Y
gg_fx <- gg_fx + geom_hline(yintercept = 0, linetype = 1)

#Grafico
gg_fx
```




```r
fprima <- D(expression(x^3 - 7*x^2 + 14*x - 6), "x")

fprima
```

```
## 3 * x^2 - 7 * (2 * x) + 14
```

```r
fprima <- function(x){
  return(3 * x^2 - 7 * (2 * x) + 14)
```

```
}

Newton(p0 = 0.5, tol = 0.001)
```

```
## [1] 0.5857864
```

```
Newton(p0 = 3, tol = 0.001)
```

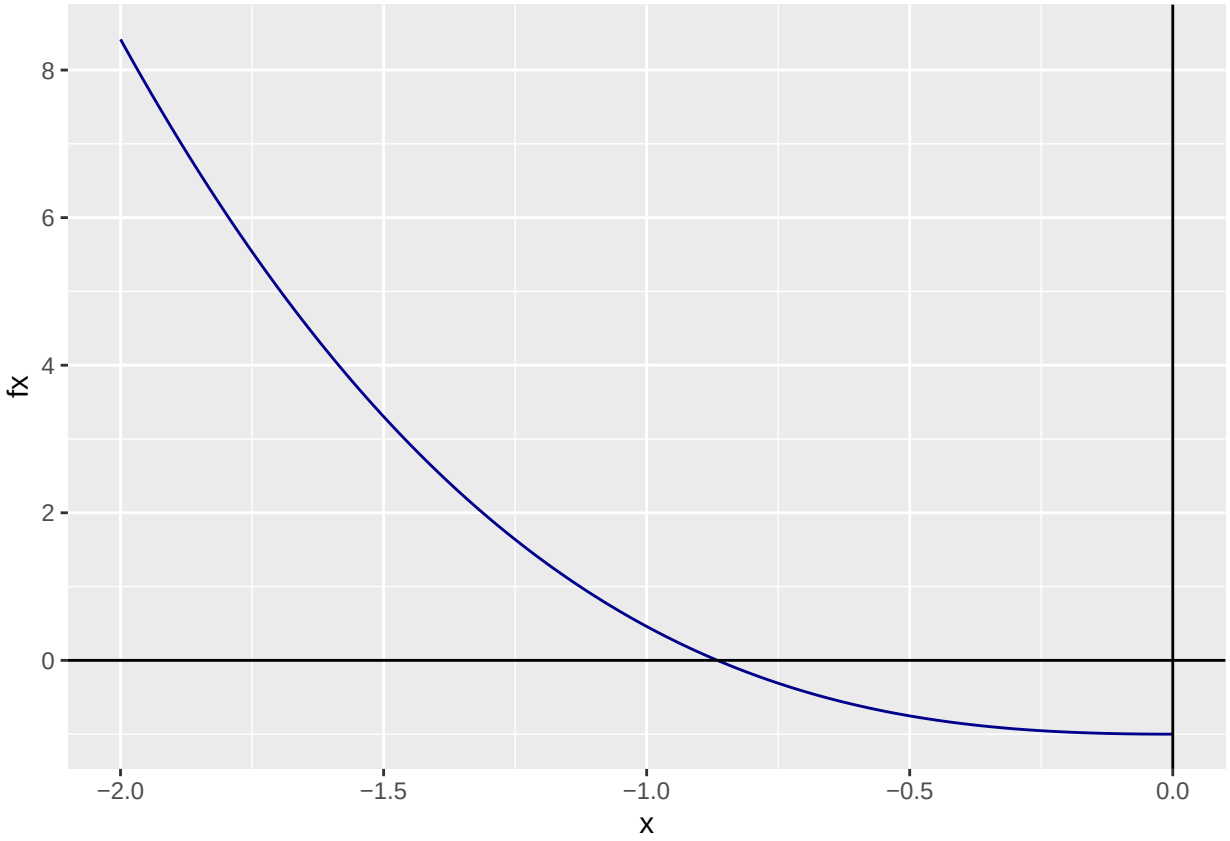
```
## [1] 3
```

```
Newton(p0 = 3.5, tol = 0.001)
```

```
## [1] 3.414214
```

Ejercicio 10:

```
f <- function(x){
  return(-x^3 - cos(x))
}

#Instancio un vector que me va a indicar los puntos en la función
x <- seq(-2, 0, by = 0.01)

#Genero los puntos
fx <- f(x)

#Creo un data frame con los x e y
df <- data.frame(x, fx)

#Instancio los datos
gg_fx <- ggplot(data = df)

#Agrego la capa con los datos
gg_fx <- gg_fx + aes(x = x, y = fx)

#Est grafica una linea
gg_fx <- gg_fx + geom_line(linetype = 1, colour = "darkblue")

#Agrego el eje X
gg_fx <- gg_fx + geom_vline(xintercept = 0, linetype = 1)

#Agrego el eje Y
gg_fx <- gg_fx + geom_hline(yintercept = 0, linetype = 1)

#Grafico
gg_fx
```




```
fprima <- D(expression(-x^3 - cos(x)), "x")

fprima
```

```
## -(3 * x^2 - sin(x))
```

```
fprima <- function(x){
  return(-(3 * x^2 - sin(x)))
}

Newton(p0 = -0.6, tol = 0.001)
```

```
## [1] -0.865474
```