Optimización y metaheurísticas I

Unidad 3: Métodos de búsqueda y optimización

Dr. Jonás Velasco Álvarez

jvelascoa@up.edu.mx

Optimización con derivadas

Optimización con derivadas

Un enfoque popular para encontrar puntos extremos (máximos y mínimos) o puntos silla de una función diferenciable $f(x)$, es establecer un conjunto de condiciones necesarias para los extremos de f en cero

$$\frac{\partial' f}{\partial x_i} = 0$$

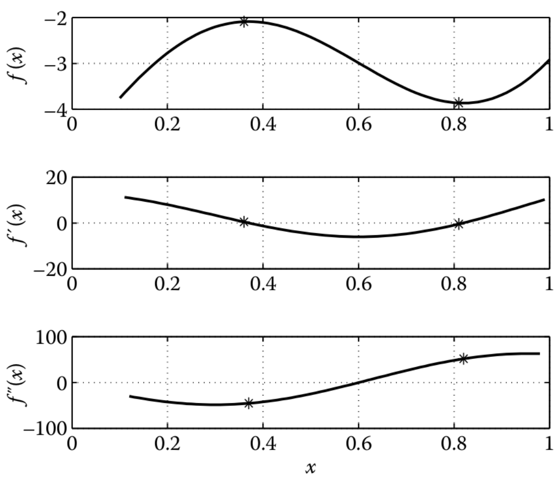
donde la segunda derivada

$$\frac{\partial'' f}{\partial x_i}$$

es negativa para un **máximo**, positiva para un **mínimo** y cero para los **puntos silla**.

Optimización con derivadas

$$f(x) = 2\sin 5x + 3x^3 - 2x^2 + 3x - 5$$



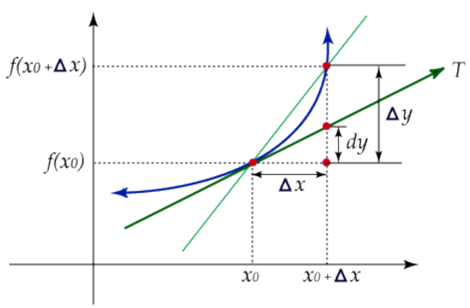
¿Qué es la derivada?

¿Qué es la derivada?

En términos generales, la derivada es una medida de cómo varian los valores de una función con respecto al valor que toman sus variables.

Por ejemplo, si tenemos una función que describe la posición de un objeto en cada instante de tiempo, la derivada de esa función describirá cómo varía la posición del objeto a medida que varía el tiempo (es decir: su velocidad).

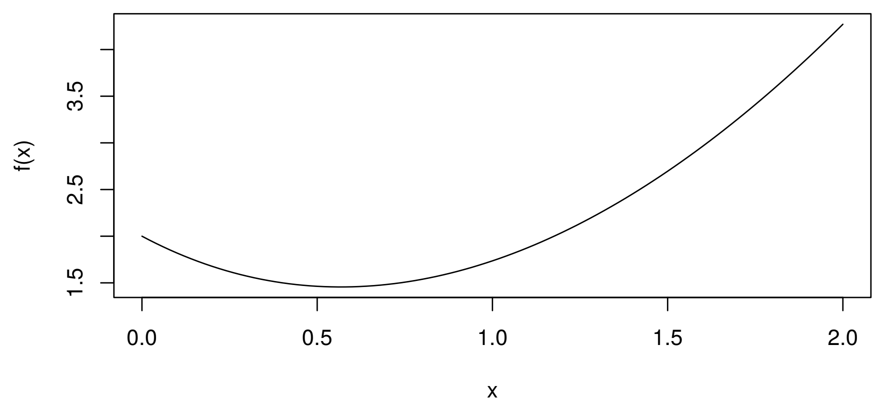
¿Qué es la derivada?



$$f'(x) = \lim_{\Delta x \to 0} \frac{f(x+\Delta x) - f(x)}{\Delta x}$$

min $f_1(x) = x^2 + 2e^{-x}$

```
curve(x**2+2*exp(-x), from=0, to=2, ylab='f(x)')
```



min $f_1(x) = x^2 + 2e^{-x}$

min $f_1(x) = x^2 + 2e^{-x}$

$$\frac{d'}{dx}(x^2 + 2e^{-x}) \quad = \quad 2x - 2e^{-x}$$

Dada la ecuación $2x - 2e^{-x} = 0$, hay que calcular sus raíces. Su raíz real es $x \approx 0.567143$ y evaluada en $f(x) = 1.455938$.

min $f_1(x) = x^2 + 2e^{-x}$

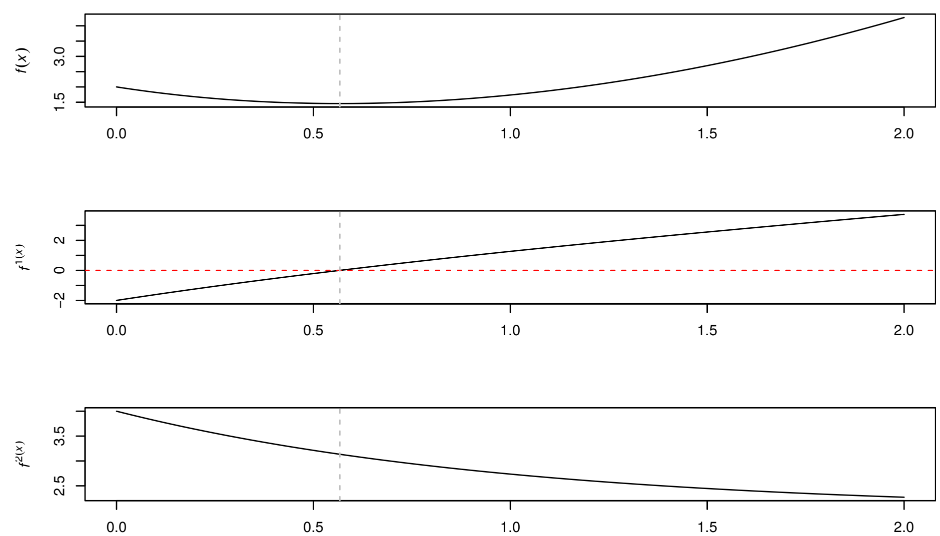
$$\frac{d'}{dx}(x^2 + 2e^{-x}) \quad = \quad 2x - 2e^{-x}$$

Dada la ecuación $2x - 2e^{-x} = 0$, hay que calcular sus raíces. Su raíz real es $x \approx 0.567143$ y evaluada en $f(x) = 1.455938$.

$$\frac{d''}{dx}(2x - 2e^{-x}) \quad = \quad 2e^{-x} + 2$$

Es positiva, entonces f tiene un **mínimo** relativo en $(x, f(x))$.

min $f_1(x) = x^2 + 2e^{-x}$

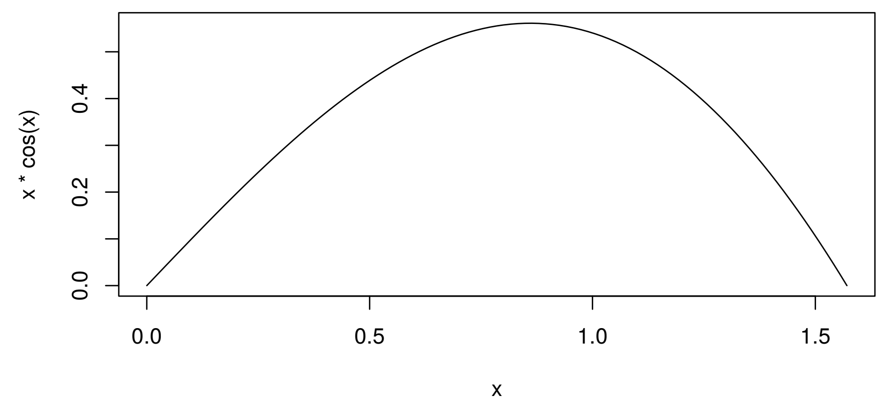


min $f_1(x) = x^2 + 2e^{-x}$

```
library('Deriv')
f <- function(x) x^2 + 2*exp(-x)
# Primer derivada
primer <- Deriv(f)
print(primer)

## function (x)
## 2 * x - 2 * exp(-x)

# Segunda derivada
segunda <- Deriv(primer)
print(segunda)

## function (x)
## 2 + 2 * exp(-x)

segunda(0.567143)

## [1] 3.134287
```


max $f_2(x) = xcosx$

```
curve(x*cos(x), from=0, to=pi/2)
```



max $f_2(x) = xcosx$

max $f_2(x) = xcosx$

$$\frac{d'}{dx}(xcosx) = \cos x - x \sin x$$

Dada la ecuación $(\cos x - x \sin x) = 0$, hay que calcular sus raíces. Su raíz real es $x \approx 0.860334$ y evaluada en $f(x) = 0.5610963$.

max $f_2(x) = xcosx$

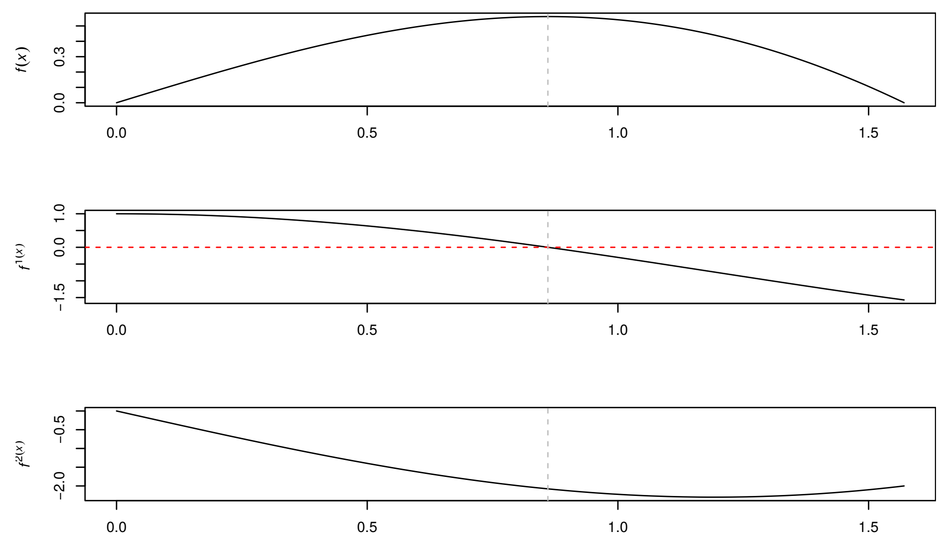
$$\frac{d'}{dx}(xcosx) = \cos x - x \sin x$$

Dada la ecuación $(\cos x - x \sin x) = 0$, hay que calcular sus raíces. Su raíz real es $x \approx 0.860334$ y evaluada en $f(x) = 0.5610963$.

$$\frac{d''}{dx}(\cos x - x \sin x) = -2 \sin x - xcosx$$

Es negativa, entonces f tiene un **máximo** relativo en $(x, f(x))$.

max $f_2(x) = xcosx$

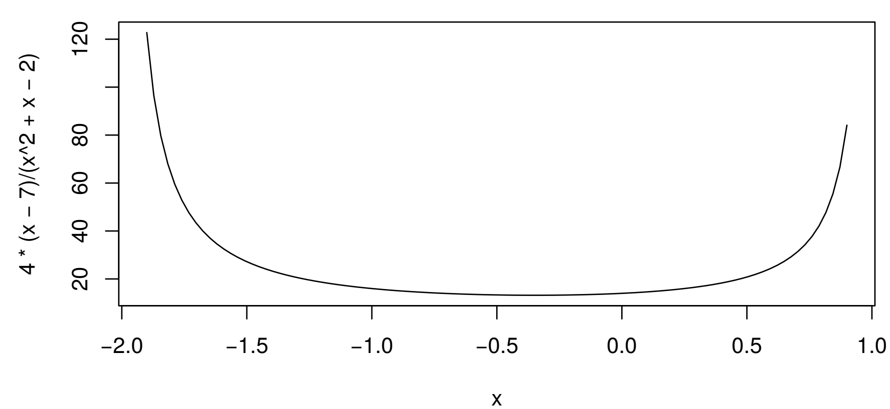


max $f_2(x) = xcosx$

```
f <- function(x) x*cos(x)
# Primer derivada
primer <- Deriv(f)
print(primer)

## function (x)
## cos(x) - x * sin(x)

# Segunda derivada
segunda <- Deriv(primer)
print(segunda)

## function (x)
## -(2 * sin(x) + x * cos(x))

segunda(0.860334)

## [1] -2.077217
```


min $f_3(x) = 4(x-7)/(x^2+x-2)$

```
curve(4*(x-7)/(x**2+x-2), from=-1.9, to=0.9)
```



min $f_3(x) = 4(x-7)/(x^2+x-2)$

min $f_3(x) = 4(x-7)/(x^2+x-2)$

$$\frac{d'}{dx}\left(\frac{4(x-7)}{(x^2+x-2)}\right) = -\frac{4(x^2-14x-5)}{(x^2+x-2)^2}$$

Dada la ecuación $-\frac{4(x^2-14x-5)}{(x^2+x-2)^2} = 0$, hay que calcular sus raíces. Su raíz real es $x \approx -0.34847$ y evaluada en $f(x) = 13.19864$.

min $f_3(x) = 4(x-7)/(x^2+x-2)$

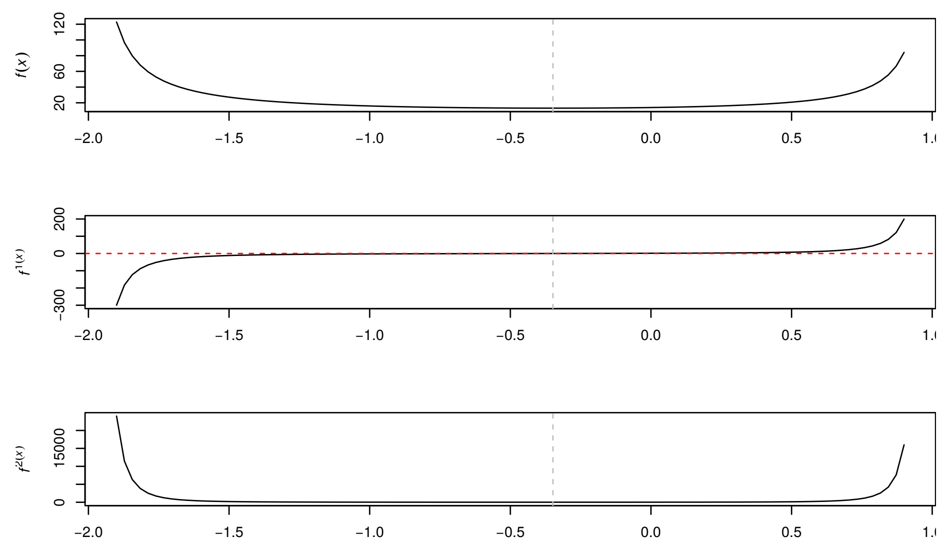
$$\frac{d'}{dx}\left(\frac{4(x-7)}{(x^2+x-2)}\right) = -\frac{4(x^2-14x-5)}{(x^2+x-2)^2}$$

Dada la ecuación $-\frac{4(x^2-14x-5)}{(x^2+x-2)^2} = 0$, hay que calcular sus raíces. Su raíz real es $x \approx -0.34847$ y evaluada en $f(x) = 13.19864$.

$$\frac{d''}{dx}\left(-\frac{4(x^2-14x-5)}{(x^2+x-2)^2}\right) = \frac{8(x^3-21x^2-15x-19)}{(x^2+x-2)^3}$$

Es positiva, entonces f tiene un **mínimo** relativo en $(x, f(x))$.

min $f_3(x) = 4(x-7)/(x^2+x-2)$



min $f_3(x) = 4(x-7)/(x^2+x-2)$

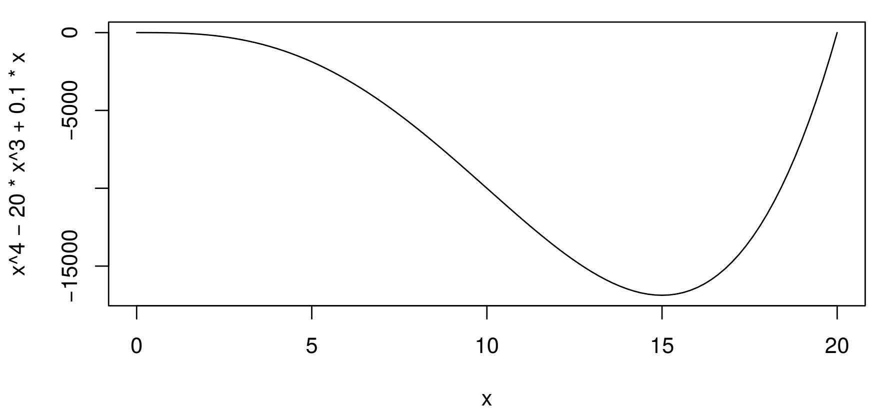
```r
f <- function(x) 4*(x-7) / (x^2 + x-2)
# Primer derivada
primer <- Deriv(f)
print(primer)

## function (x)
## {
##     .e3 <- x * (1 + x) - 2
##     4 * ((1 - (1 + 2 * x) * (x - 7)/.e3)/.e3)
## }
```


min $f_3(x) = 4(x-7)/(x^2+x-2)$

```r
# Segunda derivada
segunda <- Deriv(primer)
print(segunda)

## function (x)
## {
##     .e1 <- 2 * x
##     .e2 <- 1 + .e1
##     .e5 <- x * (1 + x) - 2
##     .e6 <- x - 7
##     -(4 * (((1 - .e2 * .e6/.e5) * .e2 + (2 - .e2^2/.e5) * .e6 +
##         1 + .e1)/.e5^2))
## }

segunda(-0.34847)

## [1] 11.85308
```


min $f_4(x) = x^4 - 20x^3 + 0.1x$

```r
curve(x**4-20*x**3+0.1*x, from=0, to=20)
```



min $f_4(x) = x^4 - 20x^3 + 0.1x$

min $f_4(x) = x^4 - 20x^3 + 0.1x$

$$\frac{d'}{dx}(x^4 - 20x^3 + 0.1x) \quad = \quad 4x^3 - 60x^2 + 0.1$$

Dada la ecuación $4x^3 - 60x^2 + 0.1 = 0$, hay que calcular sus raíces. Su raíz real es $x = 14.9999$ y evaluada en $f(x) = -16873.5$.

min $f_4(x) = x^4 - 20x^3 + 0.1x$

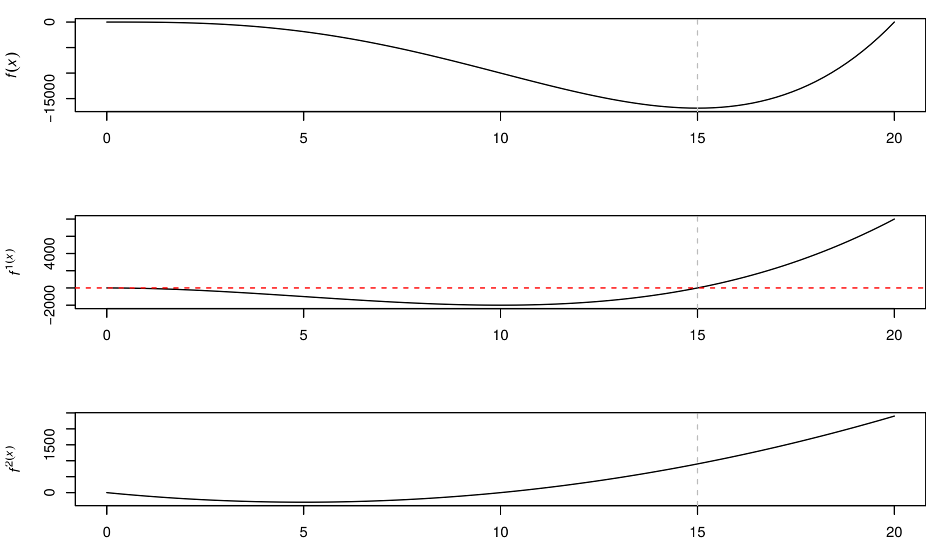
$$\frac{d'}{dx}(x^4 - 20x^3 + 0.1x) \quad = \quad 4x^3 - 60x^2 + 0.1$$

Dada la ecuación $4x^3 - 60x^2 + 0.1 = 0$, hay que calcular sus raíces. Su raíz real es $x = 14.9999$ y evaluada en $f(x) = -16873.5$.

$$\frac{d''}{dx}(4x^3 - 60x^2 + 0.1) \quad = \quad 12(x - 10)x$$

Es positiva, entonces f tiene un **mínimo** relativo en $(x, f(x))$.

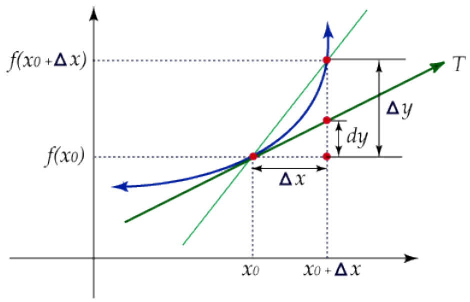
min $f_4(x) = x^4 - 20x^3 + 0.1x$



min $f_4(x) = x^4 - 20x^3 + 0.1x$

```
f <- function(x) x^4-20*x^3+0.1*x
# Primer derivada
primer <- Deriv(f)
print(primer)

## function (x)
## 0.1 + x^2 * (2 * (x - 20) + 2 * x - 20)

# Segunda derivada
segunda <- Deriv(primer)
print(segunda)

## function (x)
## x * (2 * (2 * (x - 20) + 2 * x - 20) + 4 * x)

segunda(14.9999)

## [1] 899.976
```

¿Qué es la derivada?



$$f'(x) = \lim_{\Delta x \to 0} \frac{f(x+\Delta x) - f(x)}{\Delta x}$$

min $f_1(x) = x^2 + 2e^{-x}$

```r
# Función a optimizar
f <- function(x) x^2 + 2*exp(-x)

aprox <- 0.01 # tamaño de paso
x <- seq(0, 2, aprox)

valor <- (f(x + aprox)-f(x))/aprox

# buscar la menor diferencia
indices <- which(valor <= 1e-12)
solucion <- x[indices[length(indices)]]

cat('x = ', solucion, ' con f(x) = ', f(solucion), '\n')

## x =  0.56  con f(x) =  1.456018
```


min $f_1(x) = x^2 + 2e^{-x}$

```r
# Función a optimizar
f <- function(x) x^2 + 2*exp(-x)

aprox <- 0.000001 # tamaño de paso
x <- seq(0, 2, aprox)

valor <- (f(x + aprox)-f(x))/aprox

# buscar la menor diferencia
indices <- which(valor <= 1e-12)
solucion <- x[indices[length(indices)]]

cat('x = ', solucion, ' con f(x) = ', f(solucion), '\n')

## x =  0.567142  con f(x) =  1.455938
```


min $f_1(x) = x^2 + 2e^{-x}$

Usando las fórmulas.

```r
library(Deriv)
f <- function(x) x^2 + 2*exp(-x)

paso <- 0.000001
x <- seq(0, 2, paso)

primera <- Deriv(f)
valoresX <- primera(x)

mejor <- which.min(abs(valoresX)) # ritmo de variación
solucion <- x[mejor]

cat('x = ', solucion, ' con f(x) = ', f(solucion), '\n')

## x =  0.567143  con f(x) =  1.455938
```

min $f_1(x) = x^2 + 2e^{-x}$

```r
segunda <- Deriv(primera)

if(segunda(solucion) < 0){
  cat('Es un maximo ')
}else if( segunda(solucion) == 0 ){
  cat('Es un punto silla ')
}else{
  cat('Es un minimo ')
}

## Es un minimo
```


Criterios de paro

- Diferencia absoluta entre dos valores de la función,

$$|f(x_{k+1}) - f(x_k)| < \varepsilon$$

- La diferencia relativa,

$$\frac{|f(x_{k+1}) - f(x_k)|}{|f(x_k)|} < \varepsilon$$

- La norma de la diferencia entre dos puntos para cada dos interaciones sucesivas,

$$\|x_{k+1} - x_k\| < \varepsilon$$

- La norma de la función gradiente,

$$\|\nabla f(x_k)\| < \varepsilon$$

Algoritmo de la bisección

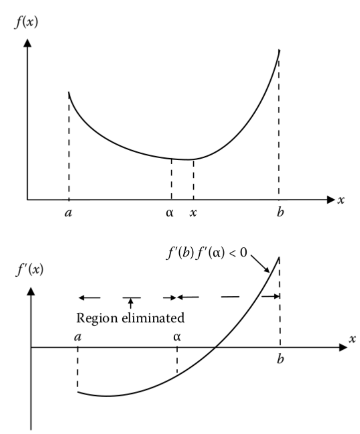
Algorithm for the Bisection Method

Step 1: Given a, b, ε, and Δx

Step 2: Compute $\alpha = \frac{a+b}{2}$, $f'(a)$ and $f'(\alpha)$

If $f'(a) f'(\alpha) < 0$
  then $b = \alpha$
  else $a = \alpha$
If $|a - b| > \varepsilon$
  then goto Step 2
  else goto Step 3

Step 3: Converged. Print $x^* = a$, $f(x^*) = f(a)$



Algoritmo de gradiente ascendente

Algorithm 1 *Gradient Ascent*

1: $\vec{x} \leftarrow$ random initial vector
2: **repeat**
3: $\quad \vec{x} \leftarrow \vec{x} + \alpha \nabla f(\vec{x})$ ▷ In one dimension: $x \leftarrow x + \alpha f'(x)$
4: **until** $\vec{x}$ is the ideal solution or we have run out of time
5: **return** $\vec{x}$

donde α es un valor positivo muy pequeño.

Algoritmo de gradiente ascendente

Algorithm 1 *Gradient Ascent*

1: $\vec{x} \leftarrow$ random initial vector
2: **repeat**
3: $\quad \vec{x} \leftarrow \vec{x} + \alpha \nabla f(\vec{x})$ $\quad \triangleright$ In one dimension: $x \leftarrow x + \alpha f'(x)$
4: **until** $\vec{x}$ is the ideal solution or we have run out of time
5: **return** $\vec{x}$

donde α es un valor positivo muy pequeño. Para establecer la minimización de la función tenemos que definir a $\vec{x} \leftarrow \vec{x} - \alpha \nabla f(\vec{x})$. Este algoritmo se llama gradiente descendente.

Algoritmo de Newton-Raphson

Este método es uno de los más utilizados para encontrar raíces de la ecuación $f'(x) = 0$. En general, el algoritmo es muy eficiente y siempre converge para una función polinomial.

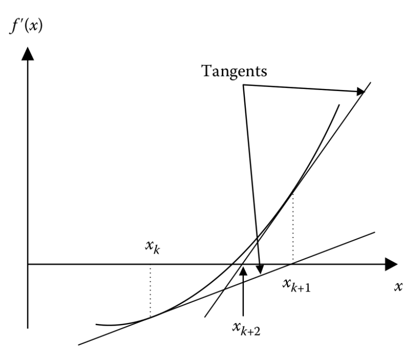
Desventajas:

- La convergencia es sensible a los valores iniciales. Para ciertos valores iniciales puede tener una tendencia divergente.
- La convergencia al óptimo se vuelve lenta cuando el valor del gradiente es cercano a cero.
- Tiene que existir la segunda derivada.

Algoritmo de Newton-Raphson

Algorithm for the Newton–Raphson Method

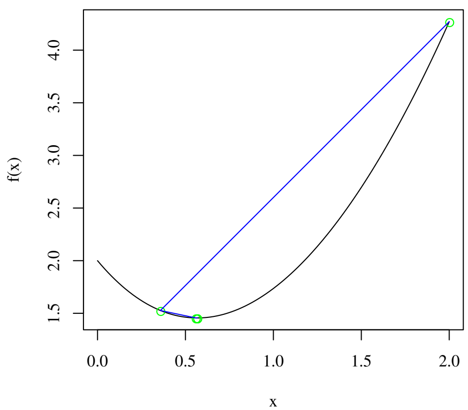
Step 1: Given x and Δx
Step 2: Compute, $f'(x)$ and $f''(x)$
Store, $xprev = x$
Update $x = xprev - \frac{f'(x)}{f''(x)}$
If $|x - xprev| > \varepsilon$
then goto Step 2
else goto Step 3
Step 3: Converged. Print $x^* = x$, $f(x^*) = f(x)$, $f'(x^*)$, $f''(x^*)$



Algoritmo de Newton-Raphson

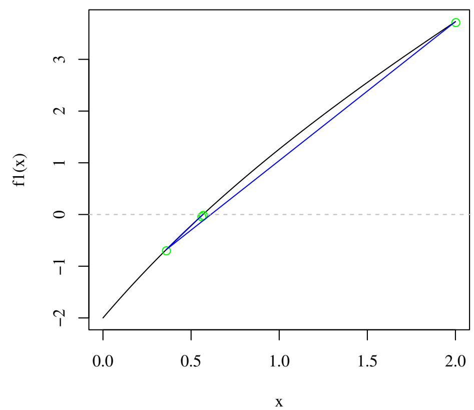
```
library('Deriv')

f <- function(x){
  x^2+ 2*exp(-x)
}
# Paso 1
x <- 2
epsilon <- 0.001 # nivel de precision
xprev <- 0
# Paso 2
f1 <- Deriv(f)
f2 <- Deriv(f1)

while(abs(x-xprev) > epsilon){
xprev <- x
x <- xprev - (f1(x)/f2(x))
}
cat('x= ',x,' f(x)= ',f(x),' f1(x)= ',f1(x),' f2(x)= ',f2(x),'\n')
```

min $f(x) = x^2 + 2e^{-x}$



x	f(x)	f1(x)	f2(x)
2.0000000	4.270671	3.729329e+00	2.270671
0.3576088	1.526577	-6.834757e-01	3.398693
0.5587083	1.456050	-2.647805e-02	3.143895
0.5671304	1.455938	-4.045515e-05	3.134301
0.5671433	1.455938	-9.448486e-11	3.134287

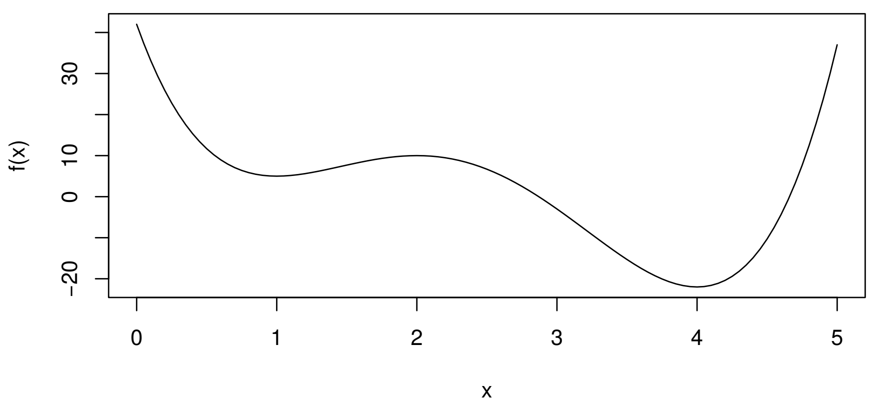
$f'(x) = 0$



x	f(x)	f1(x)	f2(x)
2.0000000	4.270671	3.729329e+00	2.270671
0.3576088	1.526577	-6.834757e-01	3.398693
0.5587083	1.456050	-2.647805e-02	3.143895
0.5671304	1.455938	-4.045515e-05	3.134301
0.5671433	1.455938	-9.448486e-11	3.134287

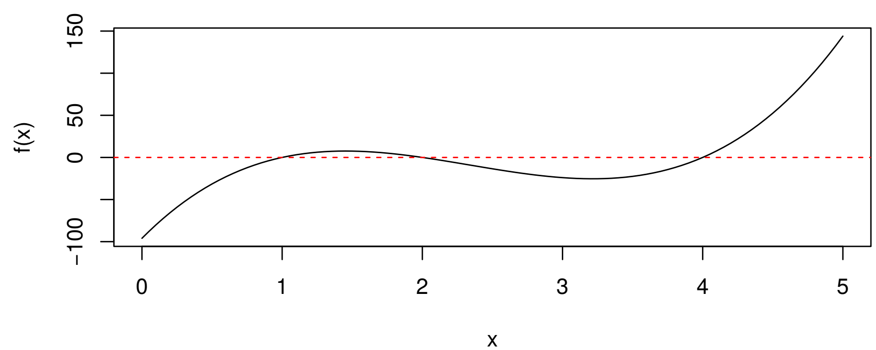
min $f(x) = 3x^4 - 28x^3 + 84x^2 - 96x + 42$

```r
curve(3*x^4-28*x^3+84*x^2-96*x+42, from=0, to=5, ylab='f(x)')
```



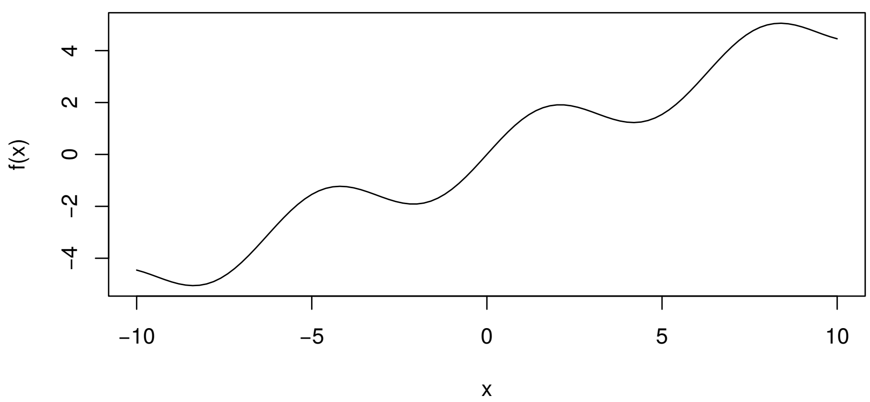
$f'(x) = 0$

```r
f <- function(x) 3*x^4-28*x^3+84*x^2-96*x+42
f1 <- Deriv(f)
curve(f1(x), from=0, to=5, ylab='f(x)')
abline(h = 0, lty=2, col='red')
```



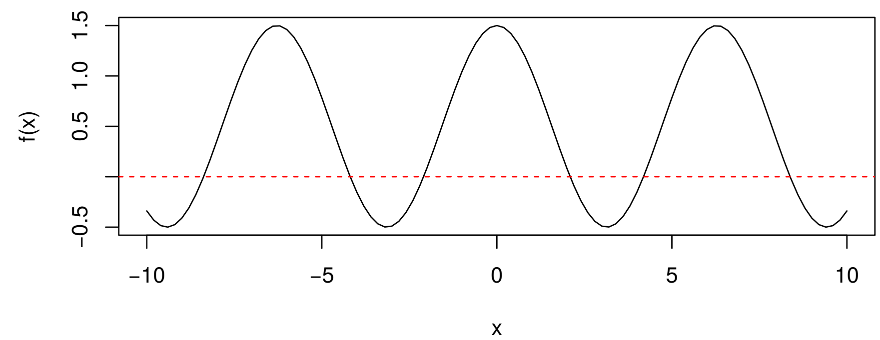
min $f(x) = sin(x) + 0.5x$

```
curve(sin(x)+0.5*x, from=-10, to=10, ylab='f(x)')
```



$f'(x) = 0$

```
f <- function(x) sin(x)+0.5*x
f1 <- Deriv(f)
curve(f1(x), from=-10, to=10, ylab='f(x)')
abline(h = 0, lty=2, col='red')
```



Algoritmo de la secante

Algorithm for the Secant Method

Step 1: Given a, b, ε, and Δx, flag = 0;

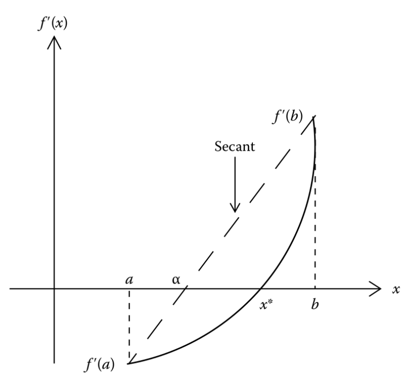
Step 2: Compute $\alpha = \frac{a+b}{2}$, $f'(a)$ and $f'(\alpha)$

If $f'(a)\,f'(\alpha) < 0$
then $b = \alpha$
set flag = 1(zero is bracketed)
else $a = \alpha$
If flag = 1
then goto Step 3
else goto Step 2

Step 3: Compute $\alpha = x_2 - \frac{f'(x_2)}{\left(f'(x_2) - f'(x_1)\right)/(x_2 - x_1)}$

If $f'(\alpha) > 0$
then $b = \alpha$
else $a = \alpha$
If $|f'(\alpha)| < \varepsilon$
then goto Step 4
else goto Step 3

Step 4: Converged. Print $x^* = \alpha$, $f(x^*) = f(\alpha)$



Algoritmo de la secante

```
library('Deriv')

# funcion a optimizar
f <- function(x){
  6*((300/x) + (x/3)) + 7*(600/x)
}

# Paso 1
a <- 10 # cota inferior
b <- 110 # cota superior
epsilon <- 0.00001 # precision
flag <- 0
contador <- 1 # iteraciones

# Paso 2
alpha <- (a+b)/2
f1 <- Deriv(f) # primer derivada
f2 <- Deriv(f1) # segunda derivada
historial <- c(contador,alpha,f(alpha),f1(alpha),f2(alpha))

while( abs(f1(alpha)) > epsilon){

if (f1(a)*f1(alpha)< 0){
  b <- alpha
  flag <- 1
}else{
  a <- alpha
}
```

Algoritmo de la secante

```
if(flag == 1){

  # Paso 3
  alpha <- b - (f1(b)/((f1(b)-f1(a))/(b-a)))

  contador <- contador + 1
  historial <- rbind(historial, c(contador,alpha,f(alpha),f1(alpha),f2(alpha)))

  if(f1(alpha) > 0){
    b <- alpha
  }else{
    a <- alpha
  }
}

}# while

cat('alpha = ', alpha,' f(alpha)= ', f(alpha),' f1(alpha)= ', f1(alpha),'\n')
cat('f2(alpha)= ', f2(alpha),'\n')
print(historial)
```