Universidad Corporativa Coppel | Programas Académicos

Ingeniería en Desarrollo de Software

Actividad 2 Método de Secante y Newton

Métodos Numéricos

Ingeniería en Desarrollo de Software

TUTOR: Miguel Angel Rodríguez Vega

ALUMNO: Karol Ochoa Beltran

FECHA: 13x de noviembre 2025

Índice

Desarrollo

Ecuación método Secante

El método de la Secante es un método que permite aproximar el cero de una función, este caso se utilizan dos valores iniciales para calcular la aproximación de la pendiente y obtener una nueva aproximación, la cual servirá para encontrar la próxima iteración y así sucesivamente.

Ecuación 1: Mètodo SECANTE

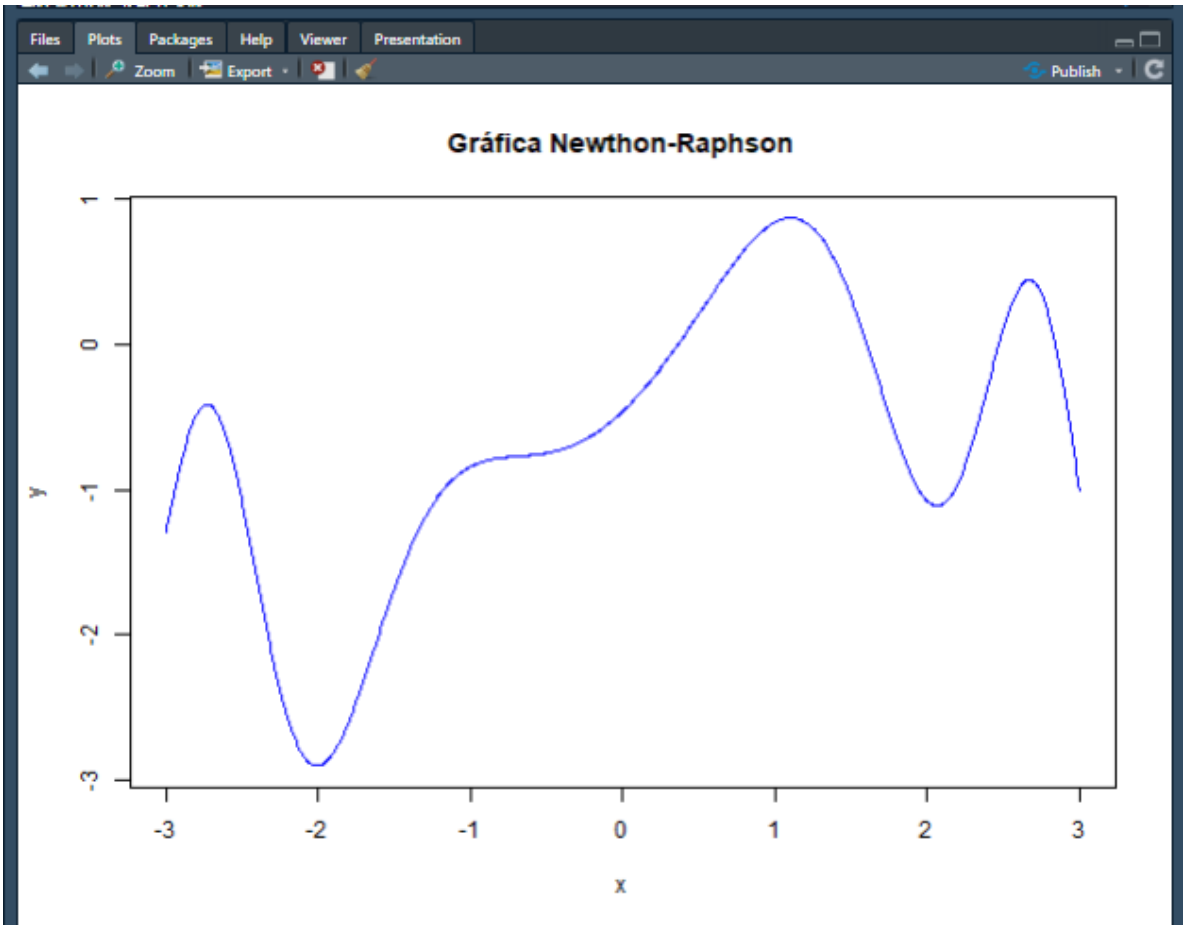
$$f(\theta) = \sin(\theta) + \cos(1 - \theta^2) - 1$$

Transformaciones.R | Transformación2.R | Metodo de NewtonRaphson (1).R | Untitled1* | Untitled2* | Untitled3*

Source on Save | Run

```r
#METODO DE SECANTE

#VALORES INICIALES
x0= 1; xant= 0; Error=0;

#ERROR PERMITIDO (DELTA)
delt= 0.000001

#NUEMRO DE ITERACIONES
n= 100

#FUNCION
f= function(x) sin(x) + cos(1-x^2) - 1

#GRÁFICA
plot (f, -3, 3,
      lwd = 1,
      main = "Gráfica Newthon-Raphson",
      col = "blue",
      xlab = "x",
      ylab = "y",
      axes = TRUE,
      n = 1000
)

#CALCULO
for(i in 1:n){
  numera = f(x0)*(xant-x0)
  denomi = f(xant)-f(x0)
  x1 = x0 - (numera/denomi)
  print(c(i, x0, xant, x1)); error = abs(x1-x0)

  if(error<delt){
    cat("la solución converge en ", i, "iteraciones, raiz= ", x1);
    break()
  }

  x0 = x1

  if(i==n) {
    print("maximo numero de iteraciones alcanzado");
    break()
  }
}
```

```
Console  Terminal  Background Jobs
R 4.5.2 · ~/
> #METODO DE SECANTE
>
> #VALORES INICIALES
> x0= 1; xant= 0; Error=0;
>
> #ERROR PERMITIDO (DELTA)
> delt= 0.000001
>
> #NUEMRO DE ITERACIONES
> n= 100
>
> #FUNCION
> f= function(x) sin(x) + cos(1-x^2) - 1
>
> #GRÁFICA
> plot (f, -3, 3,
+       lwd = 1,
+       main = "Gráfica Newthon-Raphson",
+       col = "blue",
+       xlab = "x",
+       ylab = "y",
+       axes = TRUE,
+       n = 1000
+ )
>
> #CALCULO
> for(i in 1:n){
+   numera = f(x0)*(xant-x0)
+   denomi = f(xant)-f(x0)
+   x1 = x0 - (numera/denomi)
+   print(c(i, x0, xant, x1)); error = abs(x1-x0)
+
+   if(error<delt){
+     cat("la solución converge en ", i, "iteraciones, raiz= ", x1);
+     break()
+   }
+
+   x0 = x1
+     cat("la solución converge en ", i, "iteraciones, raiz= ", x1);
+     break()
+   }
+
+   x0 = x1
+
+   if(i==n) {
+     print("maximo numero de iteraciones alcanzado");
+     break()
+   }
+ }
[1] 1.000000 1.000000 0.000000 0.353296
[1] 2.0000000 0.3532960 0.0000000 0.3637005
[1] 3.0000000 0.3637005 0.0000000 0.3618890
[1] 4.0000000 0.3618890 0.0000000 0.3622008
[1] 5.0000000 0.3622008 0.0000000 0.3621470
[1] 6.0000000 0.3621470 0.0000000 0.3621563
[1] 7.0000000 0.3621563 0.0000000 0.3621547
[1] 8.0000000 0.3621547 0.0000000 0.3621549
la solución converge en  8 iteraciones, raiz=  0.3621549
>
```



Ecuación método Newton-Raphson

El método de Newton-Raphson consiste en un algoritmo numérico que calcula aproximaciones a las raíces de una función. Esto se realiza utilizando la siguiente formula: $x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$ Con esta fórmula se calcula la siguiente aproximación (x_{i+1}) a partir de la resta de la aproximación actual (x_i) y el resultado de dividir la función $(f(x_i))$ entre su derivada $(f'(x_i))$. Para esta actividad usaremos la ecuación $f(x) = 2x^3 - 8x^3 + 10x - 15$ y su derivada $f'(x) = 6x^2 - 16x + 10$.

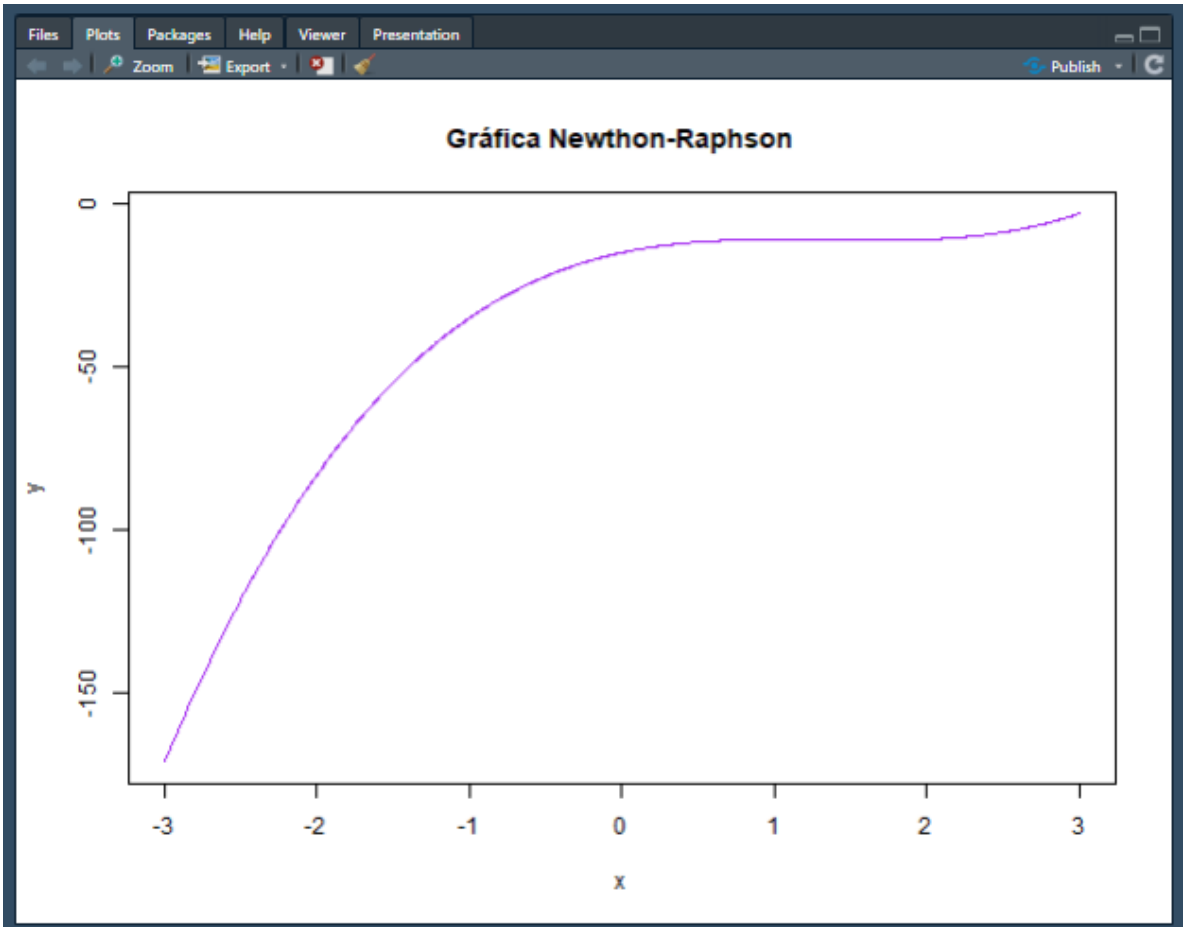
Transformaciones.R | Transformación2.R | Metodo de NewtonRaphson (1).R | Untitled1* | Untitled2*

Source on Save

```r
#NEWTHON-RAPHSON

#VALOR INICIAL DE X0 (valor supuesto)
x0 = 0

#VALOR DE PRESICIÓN (delta)
delt = 0.000001

#NUMERO DE ITERACIONES
n=100

#FUNCION
f = function(x) 2*x^3 - 8*x^2 + 10*x - 15

#GRÁFICA
plot (f, -3, 3,
      lwd = 1,
      main = "Gráfica Newthon-Raphson",
      col = "purple",
      xlab = "x",
      ylab = "y",
      axes = TRUE,
      n = 1000
      )

#DERIVADO DE LA FUNCION
df = function(x) 6*x^2 - 16*x + 10

#CICLO DE ITERACIONES Y RESULTADOS
for( i in 1:n) {
  x1 = x0 - f(x0)/df(x0)
  print (c(i, x0, x1)); error = abs(x1-x0)
  if(error<delt) {
    cat("La solución converge en", i, "iteraciones, raiz =", x1);
  break()
}

x0 = x1

if(i==n) {
  print ("Máximo numero de iteraciones alcanzado");
  break()
  }
}
```

```
R 4.5.2 · ~/
> #NEWTHON-RAPHSON
>
> #VALOR INICIAL DE X0 (valor supuesto)
> x0 = 0
>
> #VALOR DE PRESICIÓN (delta)
> delt = 0.000001
>
> #NUMERO DE ITERACIONES
> n=100
>
> #FUNCION
> f = function(x) 2*x^3 - 8*x^2 + 10*x - 15
>
> #GRÁFICA
> plot (f, -3, 3,
+       lwd = 1,
+       main = "Gráfica Newthon-Raphson",
+       col = "purple",
+       xlab = "x",
+       ylab = "y",
+       axes = TRUE,
+       n = 1000
+       )
>
> #DERIVADO DE LA FUNCION
> df = function(x) 6*x^2 - 16*x + 10
>
> #CICLO DE ITERACIONES Y RESULTADOS
> for( i in 1:n) {
+   x1 = x0 - f(x0)/df(x0)
+   print (c(i, x0, x1)); error = abs(x1-x0)
+   if(error<delt) {
+     cat("La solución converge en", i, "iteraciones, raiz =", x1);
+   break()
+ }
+
+ x0 = x1
+
+ if(i==n) {
+   print ("Máximo numero de iteraciones alcanzado");
+   break()
+   }
+ }
[1] 1.0 0.0 1.5
[1]   2.0   1.5 -21.0
[1]   3.00000 -21.00000 -13.55515
[1]   4.000000 -13.555147  -8.588912
```

```
+ x0 = x1
+
+ if(i==n) {
+   print ("Máximo numero de iteraciones alcanzado");
+   break()
+   }
+ }
[1] 1.0 0.0 1.5
[1]   2.0   1.5 -21.0
[1]   3.00000 -21.00000 -13.55515
[1]   4.000000 -13.555147  -8.588912
[1]  5.000000 -8.588912 -5.270077
[1]  6.000000 -5.270077 -3.037467
[1]  7.000000 -3.037467 -1.499752
[1]  8.000000 -1.499752 -0.347164
[1]  9.0000000 -0.3471640  0.8519877
[1] 10.0000000  0.8519877 16.1254805
[1] 11.00000 16.12548 11.20827
[1] 12.00000 11.20827  7.94321
[1] 13.000000  7.943210  5.793789
[1] 14.000000  5.793789  4.417583
[1] 15.000000  4.417583  3.611432
[1] 16.000000  3.611432  3.251145
[1] 17.000000  3.251145  3.172611
[1] 18.000000  3.172611  3.169045
[1] 19.000000  3.169045  3.169038
[1] 20.000000  3.169038  3.169038
La solución converge en 20 iteraciones, raiz = 3.169038
```



Interpretación de resultados

Metodo Newton-Raphson: Podemos observar que el bucle del código se ejecutó con éxito, la solución converge en 20 iteraciones y dio como resultado la raíz = 3.169038. Por lo tanto, si se reemplaza este valor en la función el resultado debería ser un número muy cercano a 0 ya que se encuentra dentro de la precisión que especificamos.

Metodo de la Secante: El bucle de código fue ejecutado con éxito, en este caso la solución converge en 8 iteraciones, dando una raíz = 0.3621549 que es el valor más cercano de x donde $f(x) = 0$.

Referencias

Métodos Numéricos 3: Raíces de ecuaciones: Métodos de Newton-Raphson y de la secante. (s. f.). https://estadistica-dma.ulpgc.es/FCC/05-3-Raices-de-Ecuaciones-2.html

Ijraset. (s. f.). The Newton-Raphson Method: A Detailed analysis. IJRASET. https://www.ijraset.com/research-paper/newton-raphson-method-a-detailed-analysis

Método de Newton-Raphson. (2017, 19 febrero). GeoGebra. https://www.geogebra.org/m/XCrwWHzy

Admin. (2022, 7 diciembre). Secant method. BYJUS. https://byjus.com/maths/secant-method/

Método de la secante. (s. f.). https://aplicacionmetodosnumericos.blogspot.com/p/metodo-de-la-secante.html