



Actividad | #2 | Método de Secante y Newton

Métodos Numéricos

Ingeniería en Desarrollo de Software



TUTOR: MIGUEL ANGEL RODRIGUEZ VEGA

ALUMNO: ANWAR DAVID CARAVANTES PERAZA

FECHA: 15 DE JULIO DE 2025

ÍNDICE

ÍNDICE.....	2
INTRODUCCIÓN.....	3
DESCRIPCIÓN.....	4
JUSTIFICACIÓN.....	5
DESARROLLO.....	8
Ecuación Método Secante.....	8
Ecuación Método Newton-Raphson.....	15
Interpretación de Resultados.....	20
CONCLUSIÓN.....	22
REFERENCIAS.....	24

INTRODUCCIÓN

En la actividad anterior estuvimos abordando el lenguaje R por medio del software RStudio, aprendimos un poco de su estructura, operaciones y funciones básicas, entre otras cositas. Eso nos generó las bases para poder iniciar en lo que son los métodos numéricos con el apoyo de herramientas computacionales.

Cuando hablamos de métodos numéricos, tenemos una variedad de métodos por los cuales pudiéramos llegar a una solución y cada uno se desarrolla de diferente manera aunque algunos son similares.

En esta actividad vamos a ver el Método de la Secante, y también estaremos trabajando con el Método de Newton-Raphson, ambos por medio del entorno RStudio, donde nos ayudará a graficar funciones, introducir valores, determinar condicionales, analizar información, generar ciclos, etc. y con eso podremos ver las diferencias de cada uno de los métodos para poderlos aplicar a nuestra conveniencia y realizar la mejor selección de acuerdo a la problemática propuesta.

DESCRIPCIÓN

En esta actividad se nos pide que estemos trabajando con dos métodos numéricos:

Método de Secante y Método de Newton-Raphson.

Para llevar a cabo esta actividad se nos compartieron dos archivos los cuales cada uno contiene uno de los métodos que estaremos desarrollando y nos piden resolver la ecuación que nos indican para cada una de ellas.

Ecuación a resolver por medio del Método de SECANTE

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

Ecuación a resolver por medio del Método Newton-Raphson.

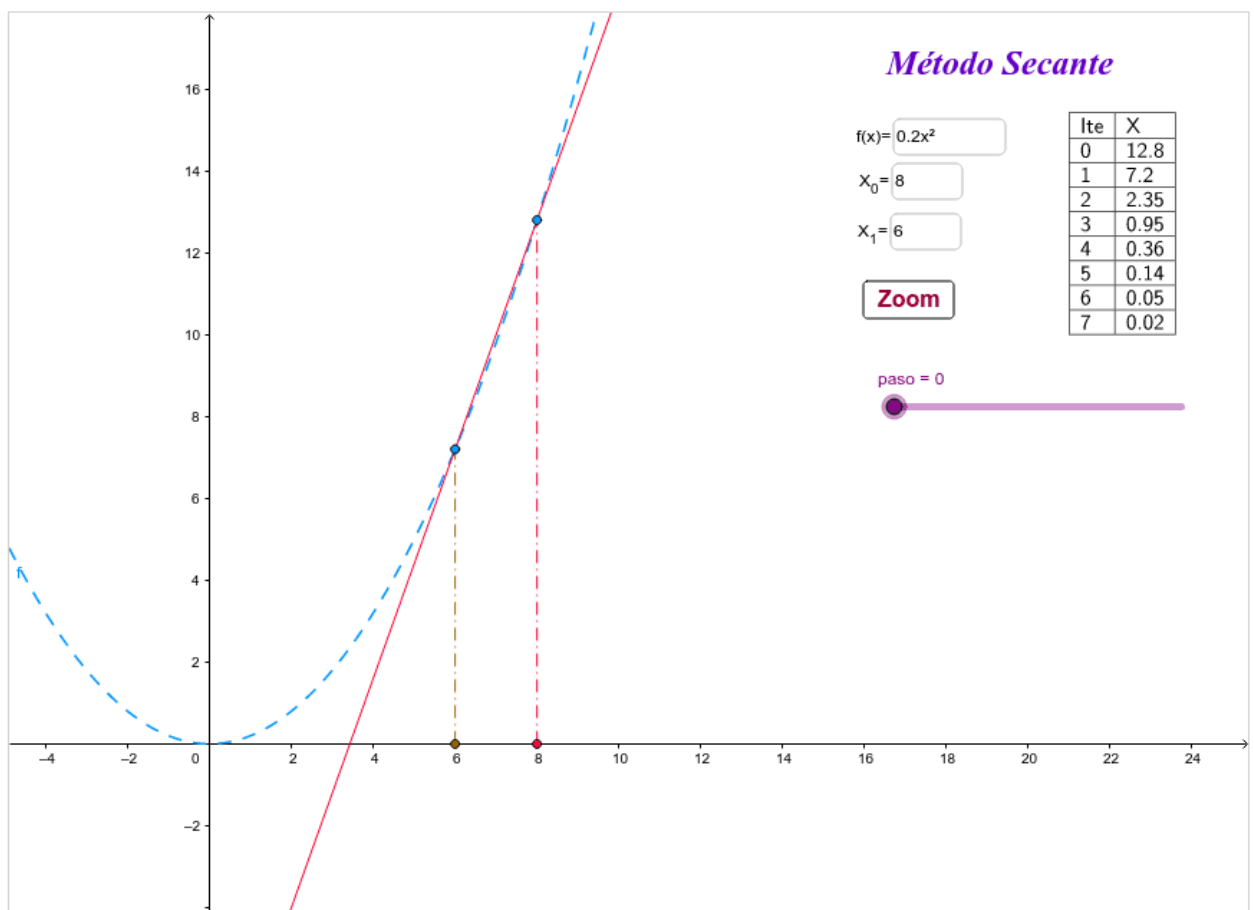
$$f(x) = 2x^3 - 8x^2 + 10x - 15$$

En los archivos que nos comparten por parte de la academia, ya vienen ciertos scripts que podemos trabajar en RStudio para resolver la ecuación correspondiente, por lo que debemos cargarlos al software, introducir la ecuación dada, y debemos tomar capturas y explicar paso a paso cómo se va desarrollando cada uno de los métodos, una vez realizadas las resoluciones nos pide interpretar los resultados obtenidos de cada ecuación.

JUSTIFICACIÓN

(*Método de la Secante*, s. f.) En análisis numérico el método de la secante es un método para encontrar los ceros de una función de forma iterativa.

Es una variación del método de Newton-Raphson donde en vez de calcular la derivada de la función en el punto de estudio, teniendo en mente la definición de derivada, se aproxima la pendiente a la recta que une la función evaluada en el punto de estudio y en el punto de la iteración anterior.



Sobre el método de Newton-Raphson encontramos en nuestra investigación que “En análisis numérico, el método de Newton (conocido también como el método de Newton-Raphson o el método de Newton-Fourier) es un algoritmo para encontrar aproximaciones de los ceros o raíces de una función real. También puede ser usado para encontrar el máximo o mínimo de una función, encontrando los ceros de su primera derivada. El método de Newton es un método abierto, en el sentido de que no está garantizada su convergencia global. La única manera de alcanzar la convergencia es seleccionar un valor inicial lo suficientemente cercano a la raíz buscada. Así, se ha de comenzar la iteración con un valor razonablemente cercano al cero (denominado punto de arranque o valor supuesto). La relativa cercanía del punto inicial a la raíz depende mucho de la naturaleza de la propia función; si presentara múltiples puntos de inflexión o pendientes grandes en el entorno de la raíz, entonces las probabilidades de que el algoritmo diverge aumentan. La divergencia se evita si se selecciona un valor inicial cercano a la raíz. Una vez iniciado el procedimiento a partir de un valor de x asumido como raíz presunta, el método prosigue trazando la recta tangente a la función en el valor x de la presunta raíz. La abscisa al origen de la intersección de esa recta con el eje x sería una mejor aproximación a la raíz que el valor anterior. Se realizan sucesivas iteraciones acorde al método, hasta alcanzar suficiente convergencia.” (colaboradores de Wikipedia, 2025).

También de acuerdo con colaboradores de Wikipedia (2023), “en análisis numérico el método de la secante es un método para encontrar los ceros de una función de forma iterativa. Es

una variación del método de Newton-Raphson donde en vez de calcular la derivada de la función en el punto de estudio, teniendo en mente la definición de derivada, se aproxima la pendiente a la recta que une la función evaluada en el punto de estudio y en el punto de la iteración anterior. Este método es de especial interés cuando el coste computacional de derivar la función de estudio y evaluarla es demasiado elevado, por lo que el método de Newton no resulta atractivo.”

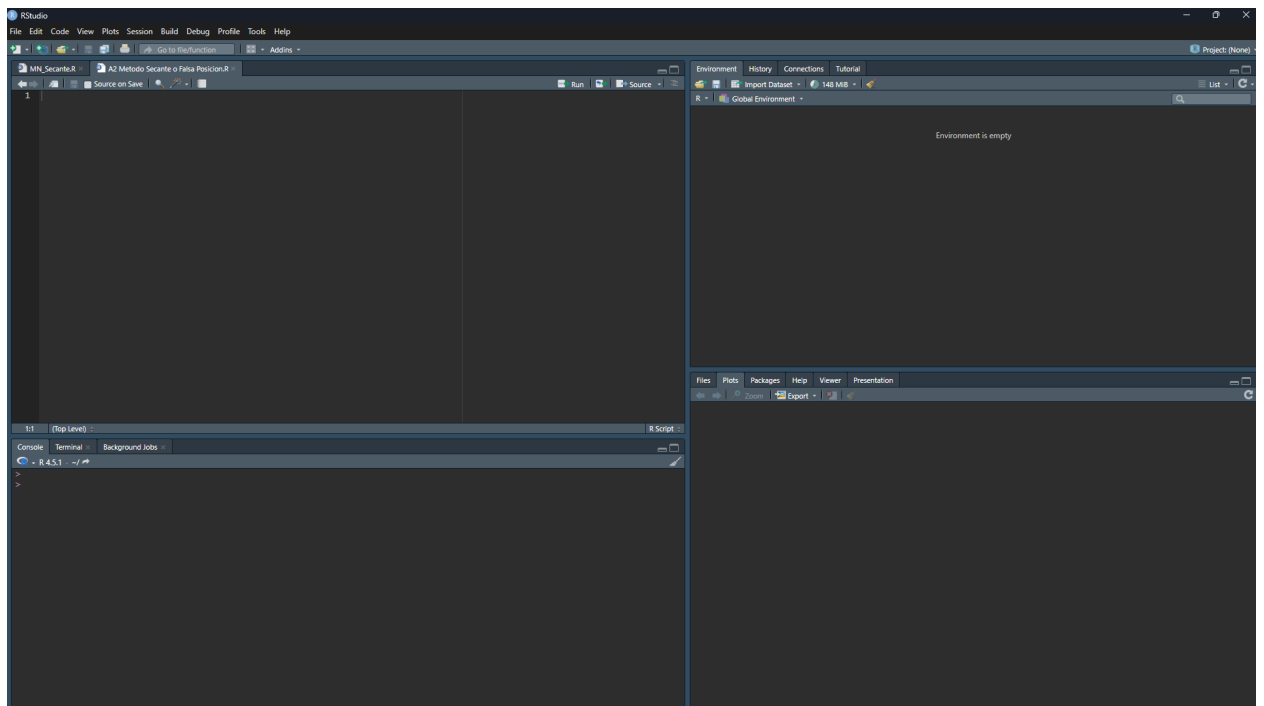
DESARROLLO

Ecuación Método Secante

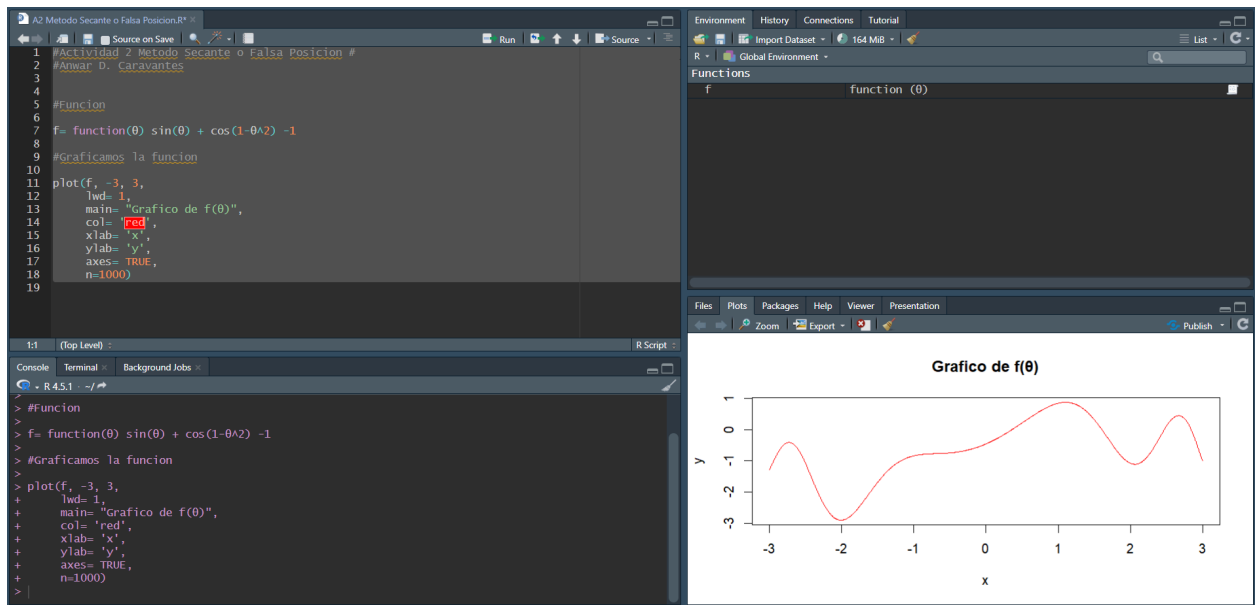
Ecuación 1: Método SECANTE

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

Abrimos un archivo nuevo y comenzamos

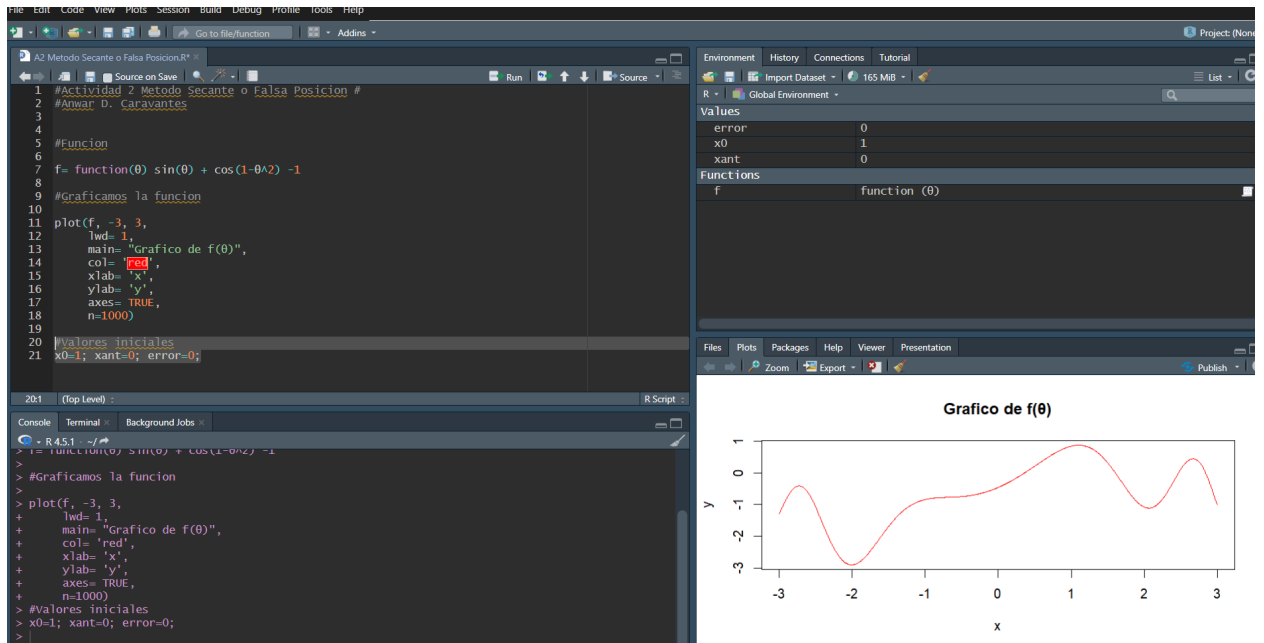


Primero vamos a colocar la función y graficarla

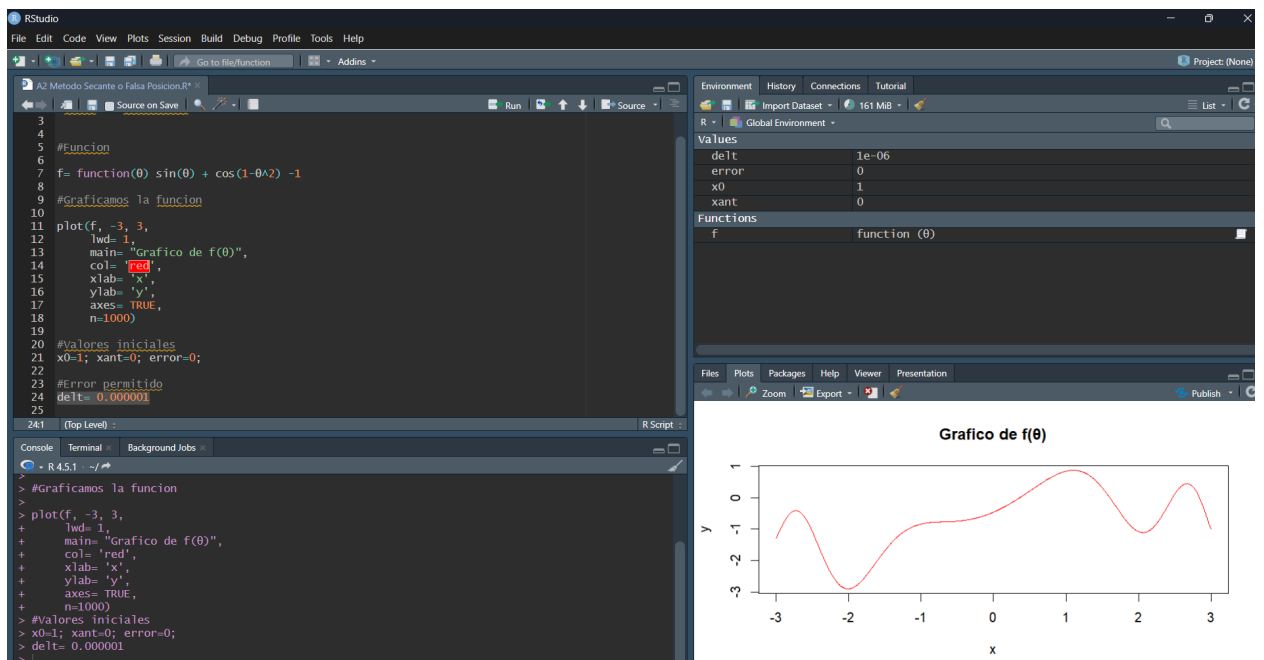


Observando la gráfica podemos observar que la gráfica cruza el eje x entre 0-1, 1-2 y 2-3 por lo que tiene al menos 3 posibles soluciones.

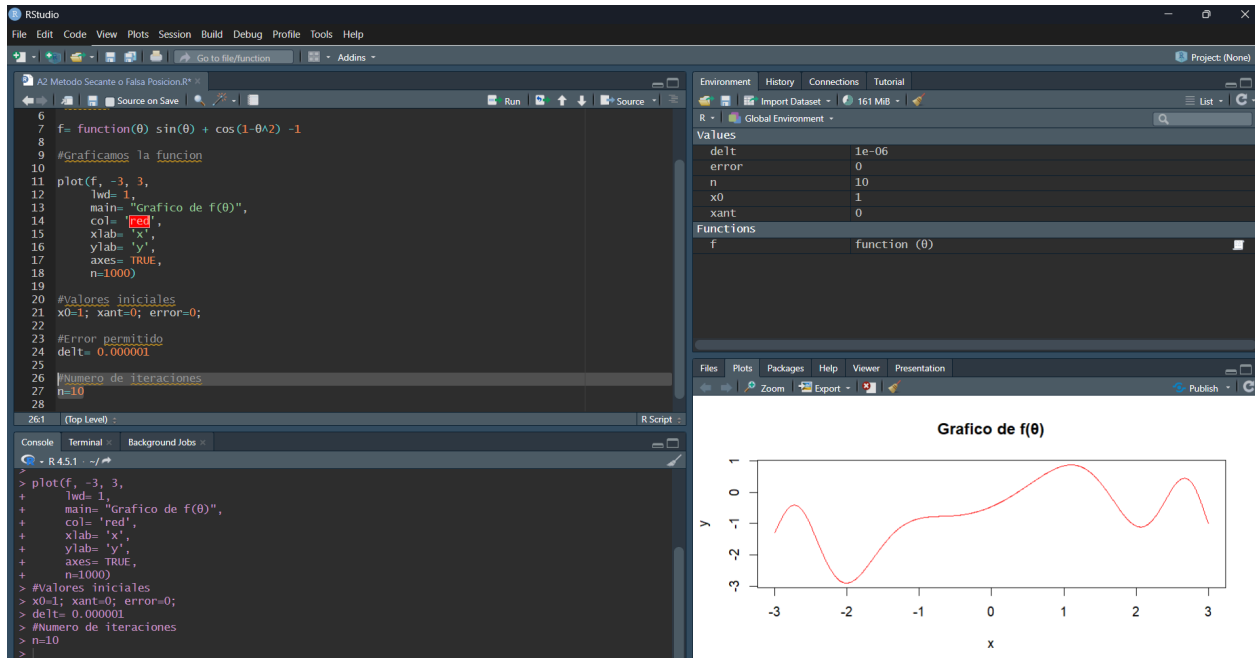
Ahora con esta información vamos a introducir los valores iniciales de a y b.



Agregamos el error permitido para la aproximación de 10^{-6}



Y agregamos el número de iteraciones para este caso n=10



Iniciamos el cálculo con el método de la secante

$$x_y = x_a - \frac{f(x_a) \cdot (x_b - x_a)}{f(x_b) - f(x_a)}$$

Entonces vamos a iniciar con un ciclo de iteraciones donde va a iterar desde 1 hasta las n iteraciones que establecimos.

```

28
29 #Calculo usando el metodo Secante
30
31 for(i in 1:n)
32

```

De acuerdo con la codificación dada, vamos a calcular primero numerador y denominador para facilitar el cálculo

```

31 for(i in 1:n) {
32     numera= f(x0) * (xant - x0)
33     denomi= f(xant) - f(x0)
34

```

y sustituyendo en la formula para encontrar el nuevo valor de x nos queda

```

35
36     x1= x0 - numera/denomi
37

```

y mandamos a imprimir el numero de iteracion en el que va el proceso, el punto anterior y el punto actual.

```

37
38     print (c(i,x0,x1))
39

```

Calculamos el error

```

40     error= abs(x1-x0)

```

y comprobamos si el error es menor al error permitido de 10^{-6} vamos mostrar en pantalla

la solución

```
40 error= abs(x1-x0)
41
42 if(error<delt) {
43     cat('La solucion converge en ', i, 'iteraciones. Raiz= ',x1);
44     break()
45 }
46
```

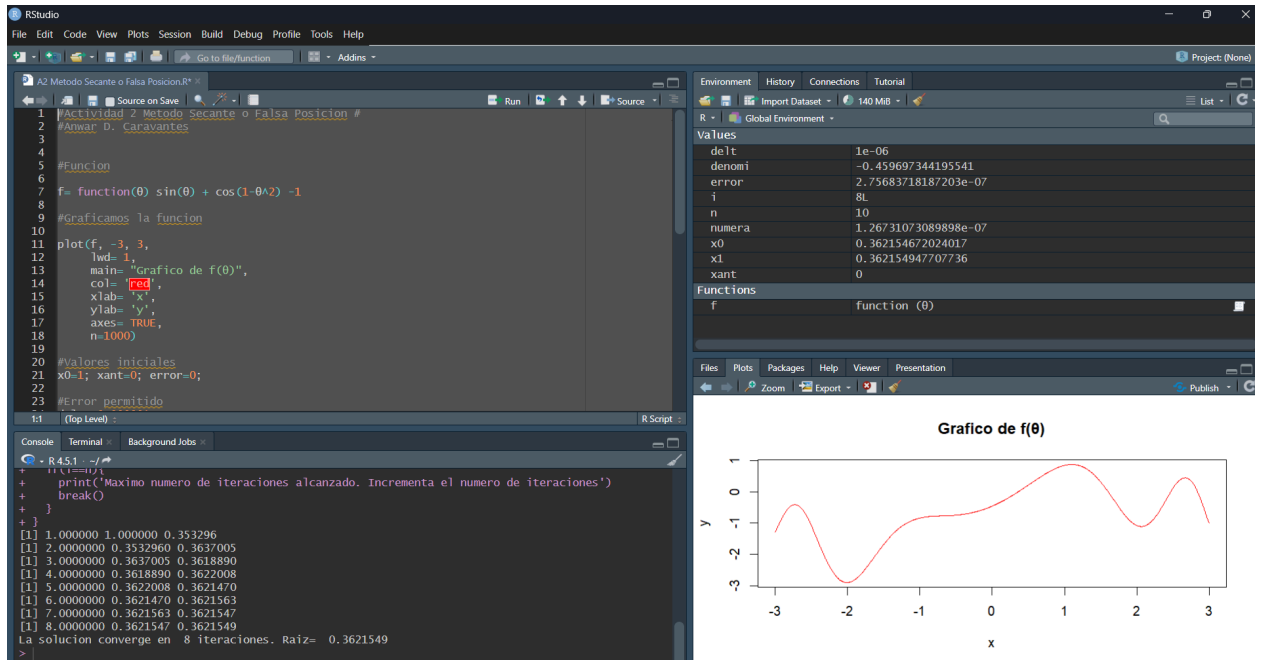
Convertimos el punto en el nuevo punto

```
46
47 x0= x1
48
```

y nos queda establecer que si el número de iteraciones ya es el máximo permitido de $n=10$ nos informe.

```
48
49 if(i==n){
50     print('Maximo numero de iteraciones alcanzado. Incrementa el numero de iteraciones')
51     break()
52 }
53 }
54
55
```

Por último corremos el script completo



Console Terminal Background Jobs

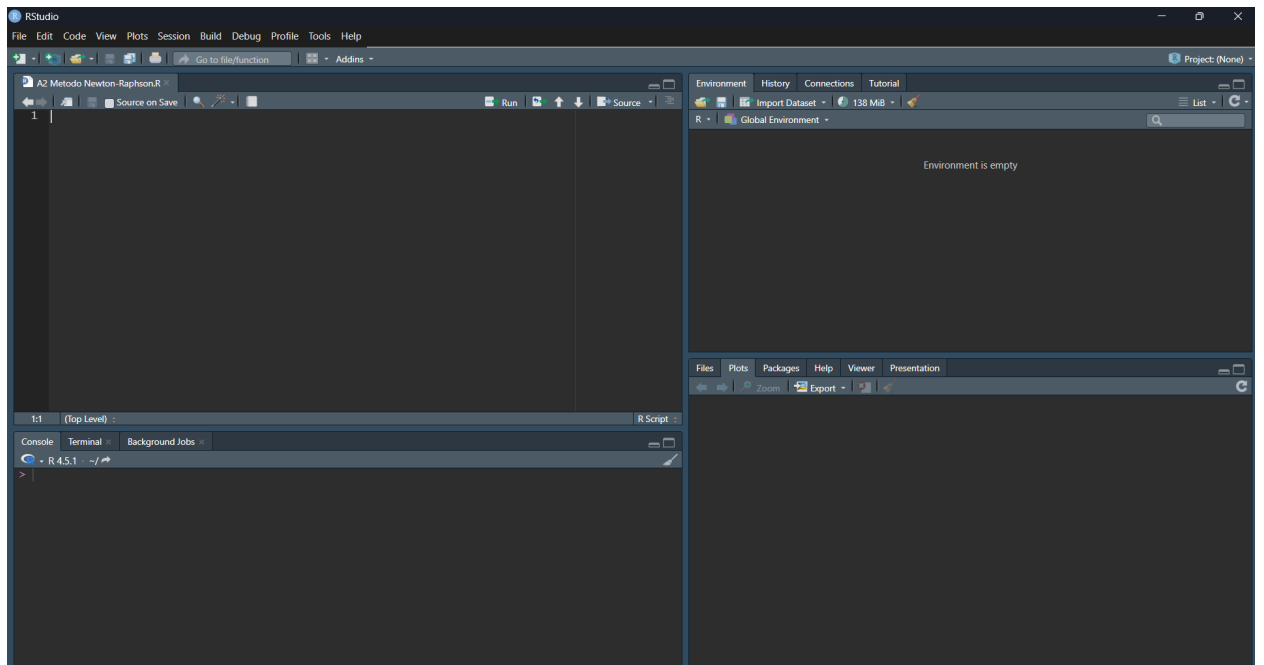
```
R 4.5.1 ~/>
+ print('Maximo numero de iteraciones alcanzado. Incrementa el numero de iteraciones')
+ break()
+ }
+ }
[1] 1.000000 1.000000 0.353296
[1] 2.000000 0.353296 0.3637005
[1] 3.000000 0.3637005 0.3618890
[1] 4.000000 0.3618890 0.3622008
[1] 5.000000 0.3622008 0.3621470
[1] 6.000000 0.3621470 0.3621563
[1] 7.000000 0.3621563 0.3621547
[1] 8.000000 0.3621547 0.3621549
La solucion converge en 8 iteraciones. Raiz= 0.3621549
>
```

Ecuación Método Newton-Raphson

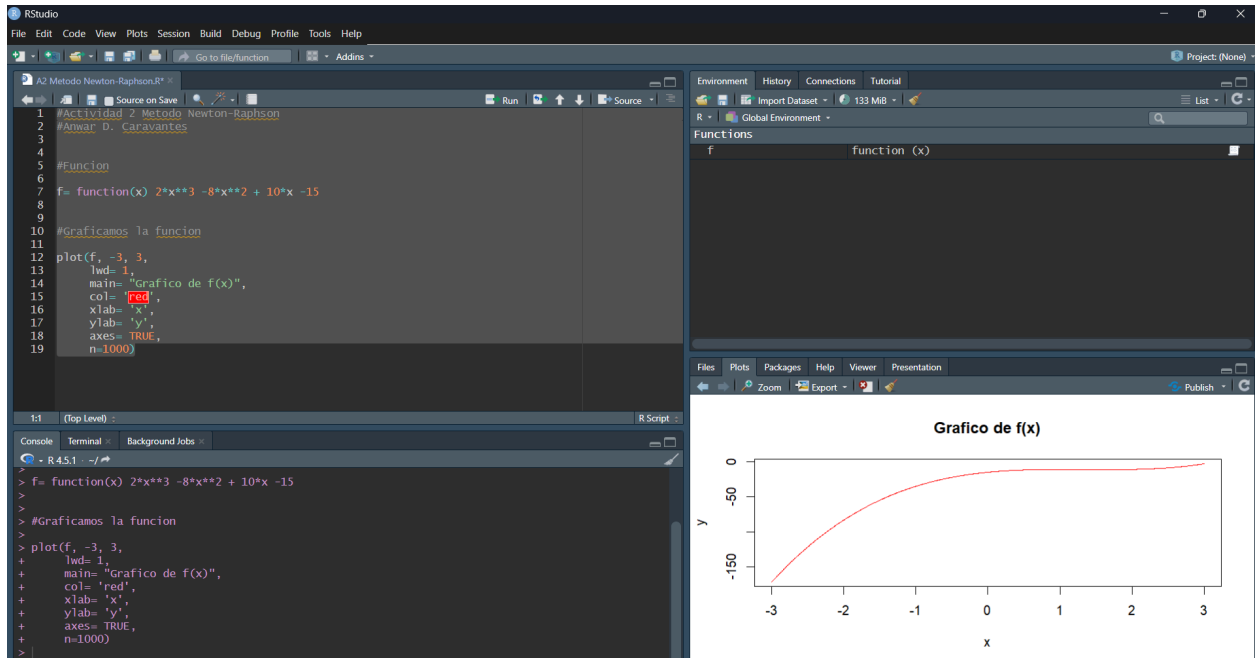
Ecuación 2: Método Newton-Raphson.

$$f(x)=2x^3-8x^2+10x-15$$

Abriremos RStudio



De igual manera que en el método anterior, iniciamos insertando la función y graficando



El método de Newton-Raphson nos pide la primer derivada de la función:

$$f'(x)=6x^2-16x+10$$

```
3
4
5 #Funcion
6
7 f= function(x) 2*x**3 -8*x**2 + 10*x -15
8
9
10 #Graficamos la funcion
11
12 plot(f, -3, 3,
13      lwd= 1,
14      main= "Grafico de f(x)",
15      col= 'red',
16      xlab= 'x',
17      ylab= 'y',
18      axes= TRUE,
19      n=1000)
20
21 #Derivada de la funcion
22
23 df= function(x) 6*x**2 -16*x +10
24
```

Ahora colocamos los valores iniciales como el punto supuesto que observando la gráfica la solución está próxima al 3 por lo que iniciaremos con 3, el número de iteraciones lo definiremos en 10 y el valor del error permitido de 10^{-6}

```
25
26 #valores iniciales
27
28 x0=3; delt= 0.000001; n=10;
29
```

Entonces vamos a iniciar con el cálculo de la función con un ciclo de iteraciones donde va a iterar desde 1 hasta las n iteraciones que establecimos y sustituimos valores. Colocamos que nos muestre el número de iteración actual, el valor actual y el valor anterior, así como calcular el error en cada iteración.

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}$$

```
29
30 #Ciclo de iteraciones y resultados
31
32
33 for (i in 1:n) {
34     x1= x0 - f(x0)/df(x0)
35     print (c(i,x0,x1)); error= abs(x1-x0)
36 }
```

Comprobamos si el error es menor al error permitido de 10^{-6} vamos mostrar en pantalla la solución

```
37 if(error<delt) {
38     cat('La solucion converge en ', i, 'iteraciones. Raiz= ',x1);
39     break()
40 }
```

Convertimos el punto en el nuevo punto

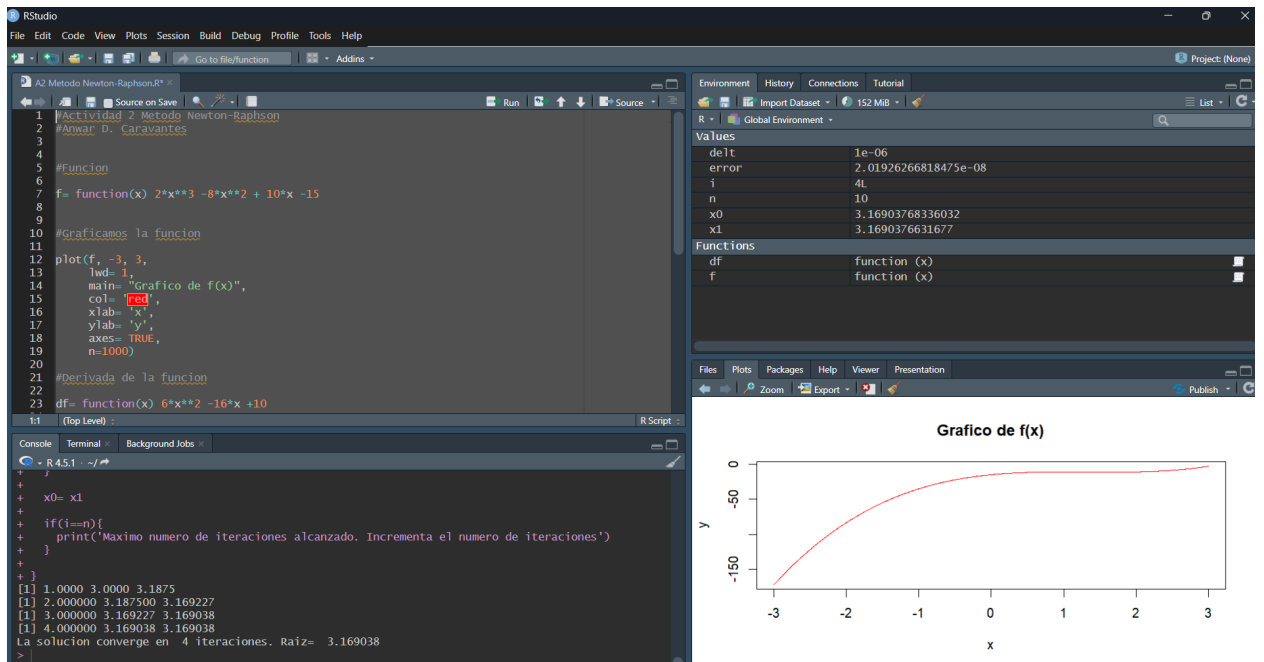
```
41
42 x0= x1
43
```

Y nos queda establecer que si el número de iteraciones ya es el máximo permitido de

n=10 nos informe.

```
44   if(i==n){  
45     print('Maximo numero de iteraciones alcanzado. Incrementa el numero de iteraciones')  
46   }  
47
```

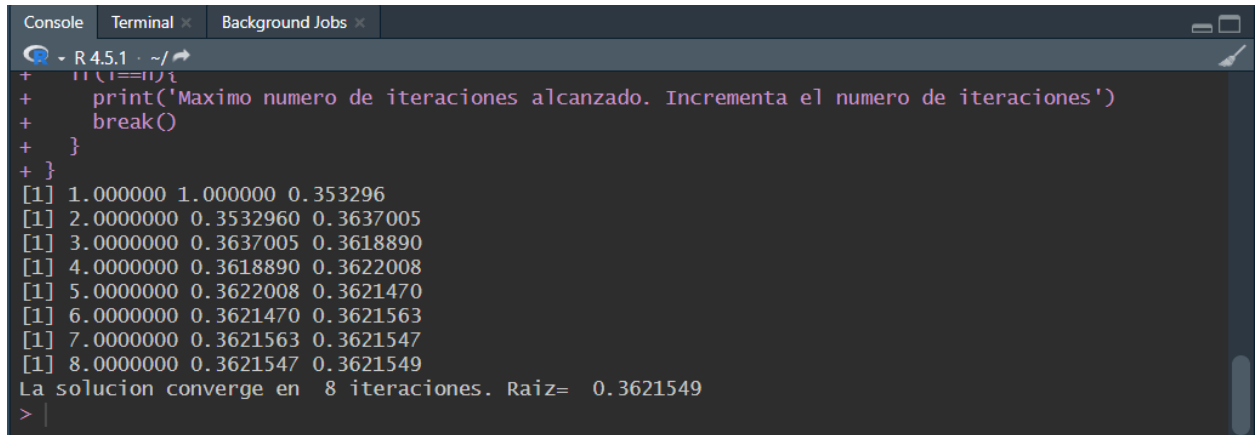
Por último corremos el script completo



```
Console Terminal Background Jobs  
R 4.5.1 ~/  
+ }  
+  
+ x0= x1  
+  
+ if(i==n){  
+   print('Maximo numero de iteraciones alcanzado. Incrementa el numero de iteraciones')  
+ }  
+ }  
+ }  
[1] 1.0000 3.0000 3.1875  
[1] 2.000000 3.187500 3.169227  
[1] 3.000000 3.169227 3.169038  
[1] 4.000000 3.169038 3.169038  
La solucion converge en 4 iteraciones. Raiz= 3.169038  
>
```

Interpretación de Resultados

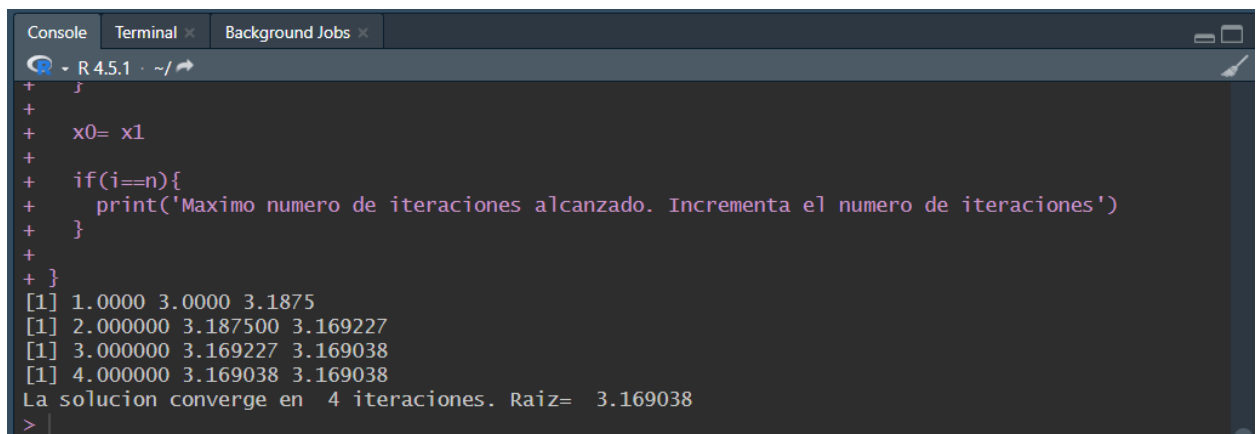
Método Secante



```
Console Terminal Background Jobs
R 4.5.1 ~ /
+ if(i==n){
+   print('Maximo numero de iteraciones alcanzado. Incrementa el numero de iteraciones')
+   break()
+ }
+ }
[1] 1.000000 1.000000 0.353296
[1] 2.000000 0.353296 0.3637005
[1] 3.000000 0.3637005 0.3618890
[1] 4.000000 0.3618890 0.3622008
[1] 5.000000 0.3622008 0.3621470
[1] 6.000000 0.3621470 0.3621563
[1] 7.000000 0.3621563 0.3621547
[1] 8.000000 0.3621547 0.3621549
La solución converge en 8 iteraciones. Raíz= 0.3621549
>
```

Tomando los valores iniciales conforme a la interpretación de la gráfica, se toman 0 y 1, la solución converge después de 8 iteraciones, dando como resultado 0.3621549

Método Newton-Raphson



```
Console Terminal Background Jobs
R 4.5.1 ~ /
+ }
+
+ x0= x1
+
+ if(i==n){
+   print('Maximo numero de iteraciones alcanzado. Incrementa el numero de iteraciones')
+ }
+ }
[1] 1.0000 3.0000 3.1875
[1] 2.000000 3.187500 3.169227
[1] 3.000000 3.169227 3.169038
[1] 4.000000 3.169038 3.169038
La solución converge en 4 iteraciones. Raíz= 3.169038
>
```

Tomando el valor supuesto de 3 conforme a la visualización de la gráfica, la solución

converge después de 4 iteraciones, dando como resultado 3.169038, esto ya que tomamos un valor supuesto muy cercano, al tomar un valor más lejano, también se resuelve la función pero costará realizar más iteraciones.

Para ambos métodos, es importante saber identificar en las gráficas cuales son los puntos raíz de la función, aunque no lo podemos identificar con mucha exactitud pero una buena aproximación bastará para reducir el número de iteraciones que se ejecuten mejorando la eficiencia.

CONCLUSIÓN

En el desarrollo de esta tarea hemos logrado implementar dos métodos numéricos muy conocidos, el método de Newton-Raphson y el método de Secante que se deriva del primer método.

Ambos métodos se basan en la aproximación iterativa para buscar el punto donde la raíz de la función converge, pero pues si existe una diferencia muy grande entre ambos métodos.

Al utilizar el método de Newton-Raphson necesitamos encontrar la primer derivada de la función, lo cual dificulta o puede llegar a ser un poco complejo para algunas personas y de igual manera esa necesidad de obtener la derivada puede resultar imposible de obtener en algunas funciones, pero una vez pasando este punto y al obtener la primera derivada de la función, se logra una un proceso muy eficiente llevando a encontrar la solución de una manera muy rápida y eficiente sin tantas iteraciones, obviamente que si tomamos el valor supuesto muy lejano al real vamos a incrementar el número de iteraciones pero de igual manera se mantienen muy por debajo de otros métodos.

Y al utilizar el método de Secante eliminamos esta necesidad de llegar a obtener la primera derivada de la función, por lo que se convierte en un método más atractivo para muchas personas cuando esta derivada es difícil de obtener, pero a diferencia de método de newton-raphson es un poco menos eficiente ya que realizar más iteraciones para poder encontrar el punto Raíz de la función.

Ambos métodos son muy buenos ya que logramos determinar el punto de convergencia de la raíz, y el apoyarnos por medio de algoritmos de programación para evitar el trabajo a mano reduce significativamente los tiempos y errores del desarrollo de la metodología.

REFERENCIAS

Método de la secante. (s. f.). GeoGebra. <https://www.geogebra.org/m/SvjeTquB>

colaboradores de Wikipedia. (2023, 15 enero). Método de la secante. Wikipedia, la Enciclopedia Libre. https://es.wikipedia.org/wiki/M%C3%A9todo_de_la_secante

colaboradores de Wikipedia. (2025, 15 junio). Método de Newton. Wikipedia, la Enciclopedia Libre. https://es.wikipedia.org/wiki/M%C3%A9todo_de_Newton