

Actividad | #2 | Método de Secante y

Newton

Ingeniería en Desarrollo de Software



TUTOR: Miguel Angel Rodríguez Vega

ALUMNO: Edgar Enrique Cuamea Ochoa

FECHA: 11 de marzo del 2024

Contenido

Introducción.	3
Descripción.	4
Justificación.	5
Desarrollo.	6
Conclusión.	16

Introducción.

Utilizaremos Rstudio para la realización de distintos algoritmos para poder solucionar algunos problemas matemáticos utilizando operaciones menos complejas, por lo que utilizaremos este programa para realizar diferentes funciones tales como poder realizar representaciones de cantidades y variables, guardar valores fraccionados y realizar operaciones básicas como divisiones o multiplicaciones y obtener un resultado, así como poder modificar el cómo es que nos puede entregar el resultado, esto con el fin de poder utilizar los métodos numéricos para poder realizar aproximaciones de operaciones matemáticas complejas para poder solucionar un problema complejo de una forma más fácil utilizando este programa o por lo que utilizaremos los métodos de secante y el método de newton-Raphson, con estos métodos resolveremos algunas ecuaciones en Rstudio para encontrar soluciones aproximadas para problemas complejos así mismo poder interpretar los resultados que obtendremos de estas ecuaciones complejas ya que utilizaremos diferentes ecuaciones para cada método, una para el método de secante y otra ecuación para el método de newton-Raphson

Descripcion.

Tendremos en cuenta los diferentes métodos numéricos para poder encontrar el valor aproximado de una incógnita de un problema matemático complejo por lo que existen diferentes métodos que nos ayudaran a encontrar el valor aproximado, a través de estos métodos podremos utilizar diferentes formas de encontrar la incógnita utilizando diferentes funciones como tangentes o derivar funciones para optimizar el algoritmo y así poder encontrar el resultado aproximado de la forma más rápida posible sin tener que hacer tantas interacciones así como poder realizar operaciones más sencillas que nos tomaran más interacciones para obtener el mismo resultado por lo que usaremos el programa de Rstudio para la realización de la actividad por lo que podremos utilizar el método de la secante que evita el calculo de la derivada mientras la de newton-raphson es la manera mas eficiente ya que esta puede resolverse en menos interacciones ya que calcula la derivada correctamente mientras que el método de la secante solo se aproxima

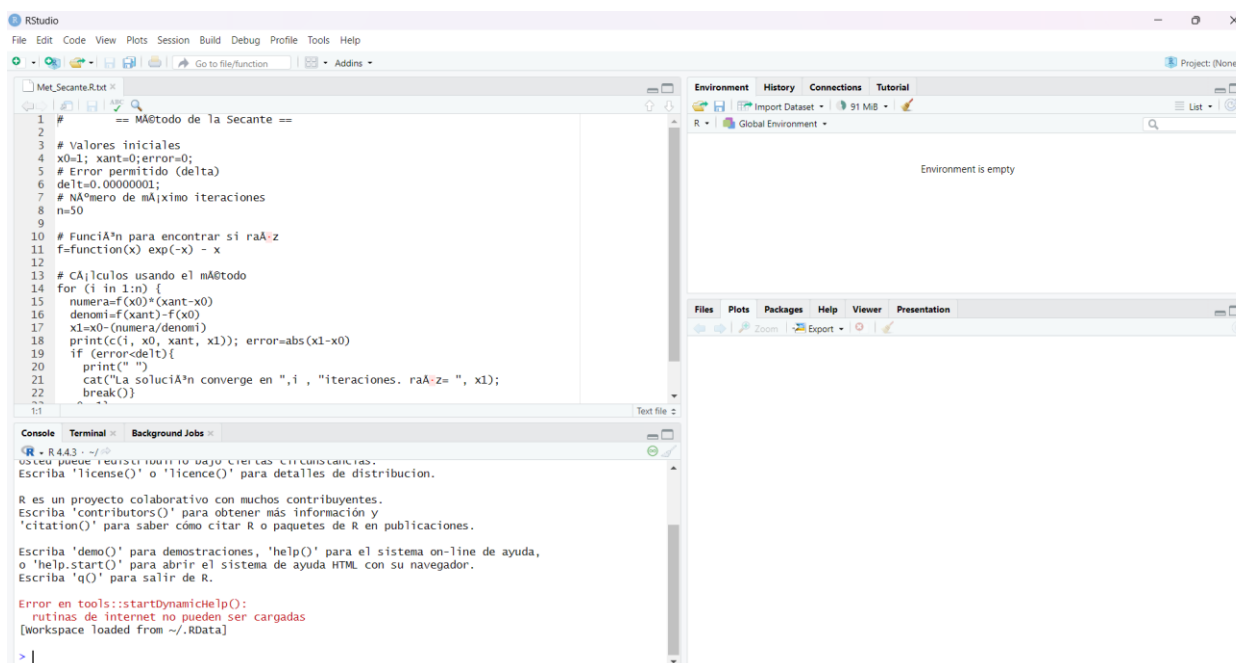
Justificación.

Aprenderemos a utilizar Rstudio para la realización de la actividad ya que es un programa para la manipulación de datos que nos funcionara para la realización de algoritmos complejos así como utilizar un lenguaje que está especializado en el análisis estadístico, este programa también nos proporciona la visualización de datos ya que podremos crear graficas que nos ayudaran a saber qué tipo de método podremos utilizar para la resolución del problema y obtener el resultado aproximado, este programa contiene funciones matemáticas y numéricas que incluyen cálculos, integraciones, diferenciación, interpolación, resolución de ecuaciones diferenciales, entre otras, podremos crear simulaciones para así poder realizar análisis de errores, además cuenta con paquetes especializados para el uso de métodos numéricos para el análisis matemático para ecuaciones diferenciales y funciones por lo que lo hace un programa practico para la resolución de problemas matemáticos complejos con el uso de funciones e interacciones repetitivas para llegar al resultado aproximado del problema.

Desarrollo.

Método de la secante.

Abrimos Rstudio para el desarrollo de la actividad donde tendremos que encontrar el valor aproximado de diferentes ecuaciones utilizando el método de la secante, así como el método de newton-raphson por lo que empezaremos utilizando el método de la secante:



Ingresamos el archivo de metodo de la secante en Rstudio donde empezaremos a trabajar con para resolver las ecuaciones utilizando este programa por lo que usaremos la siguiente ecuacion para la resolucion de este

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

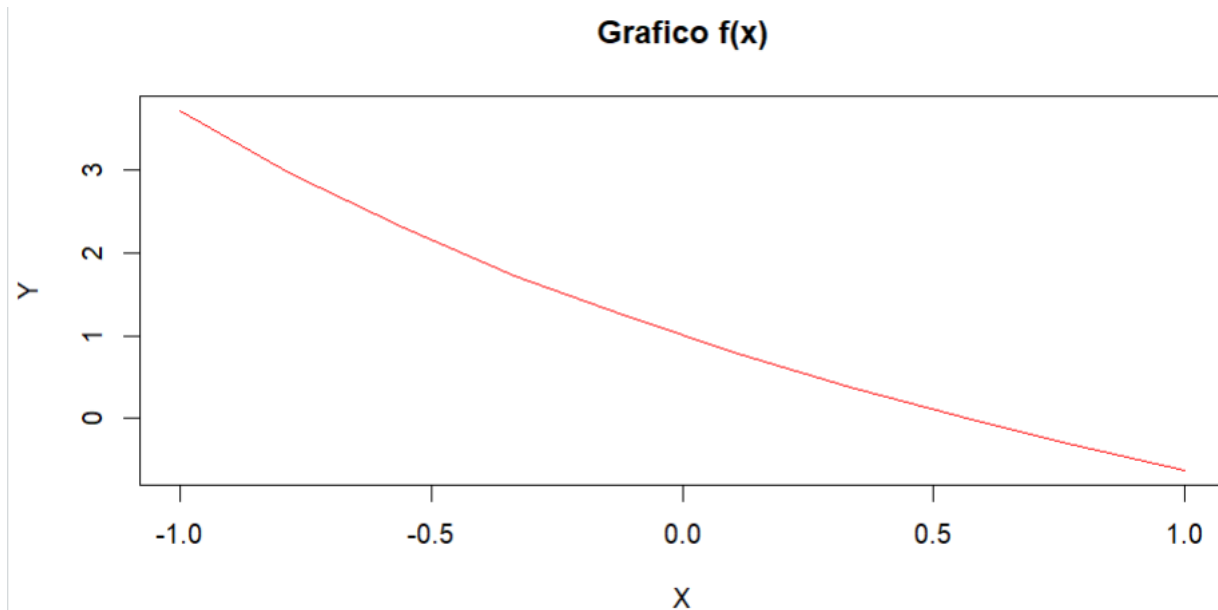
Tenemos nuestros valores iniciales $x_0=1$ que es nuestro valor inicial ; $x_{ant}=0$ ya que este es nuestro valor anterior a x_0 por lo que el metodo de la secante necesita 2 puntos para poder trazar una linea y nuestra variable $error=0$ por lo que esta variable guarda nuestra precisión del

algoritmo definimos nuestro error permitido en nuestra variable delta donde este tendra un valor de 0.00000001, ademas de nuestro numero de interacciones de $n=50$

Ademas de definir la funcion para calcular nuestra raiz utilizando el metodo de la secante

```
> x0=1; xant=0; error=0;  
> delt=0.00000001;  
> n=50  
>  
> f=function(x) exp(-x) - x  
> |
```

Es importante definir los valores iniciales donde establercemos nuestro x_0 y x_{ant} o tambien podremos nombrarla con otras variables como A y B ademas de el numero de interacciones, nuestro error permitido delta y la funcion para encontrar la raiz antes de ejecutar nuestro codigo, ya que si ingresamos un valor diferente en estos parametros inicales, hata que nuestro resultado final sea distinto



Graficamos nuestra funcion y vemos que existe una solucion donde hay un cambio de signo en el eje y mientras que x se encuentra aproximadamente en 0.5

```

> x0=1; xant=0; error=0;
> delt=0.00000001;
> n=50
>
> f=function(x) exp(-x) - x
> 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){
        print(" ")
        cat("La soluci n converge en ",i , "iteraciones. ra z= ", x1);
        break()}
    x0=x1}

print(" ximo n mero de iteraciones alcanzada !!!")

```

Tenemos nuestro codigo donde al ejecutarlo realiza la siguiente funcion, tenemos que for (i in 1:n), esta funcion realiza un blucle que se ejecutara un numero de veces ya que estamos utilizando la variable n y definimos que n=50 por lo que se ejecutara 50 veces, la variable i representa las interacciones que van desde la interaccion 1 hasta n=50.

Pasamos a la siguiente linea donde tenemos el siguiente codigo: numera=f(x0)*(xant-x0), esta linea calcula el numerador de la formula donde numera = $f(1) * (0 - 1)$

En la siguiente linea utilizamos el siguiente codigo denomi = f(xant) - f(x0) para calcular el denominador de la ecuacion, donde denomi = $f(0) - f(1)$

En la siguiente linea utilizamos el codigo $x1 = x0 - (\text{numera} / \text{denomi})$ para calucular la aproximacion de la raiz x1 restandole a la variable x0 elvalor de la division de los valores de el numerador y denominador

En la siguiente linea utilizamos print(c(i, x0, xant, x1)); error = abs(x1 - x0) este codigo imprime en la consola los valores de la interaccion (i) así como la aproximacion del valor (x0) y el valor (xant) y nuestra raiz expresada en (x1), ademas de mostrarnos nuestro error absoluto este es igual a el valor aproximado de x1 menos el valor de x0

En la siguiente linea tenemos una condicion donde if (error < delt) verifica que si el error es menor a nuestra variable delta que es nuestro error permitido de 0.00000001 ejecutara la siguiente linea:

```
print(" ") cat("La soluci3n converge en ",i , "iteraciones. raÅ-z= ", x1);

break() } x0=x1 }
```

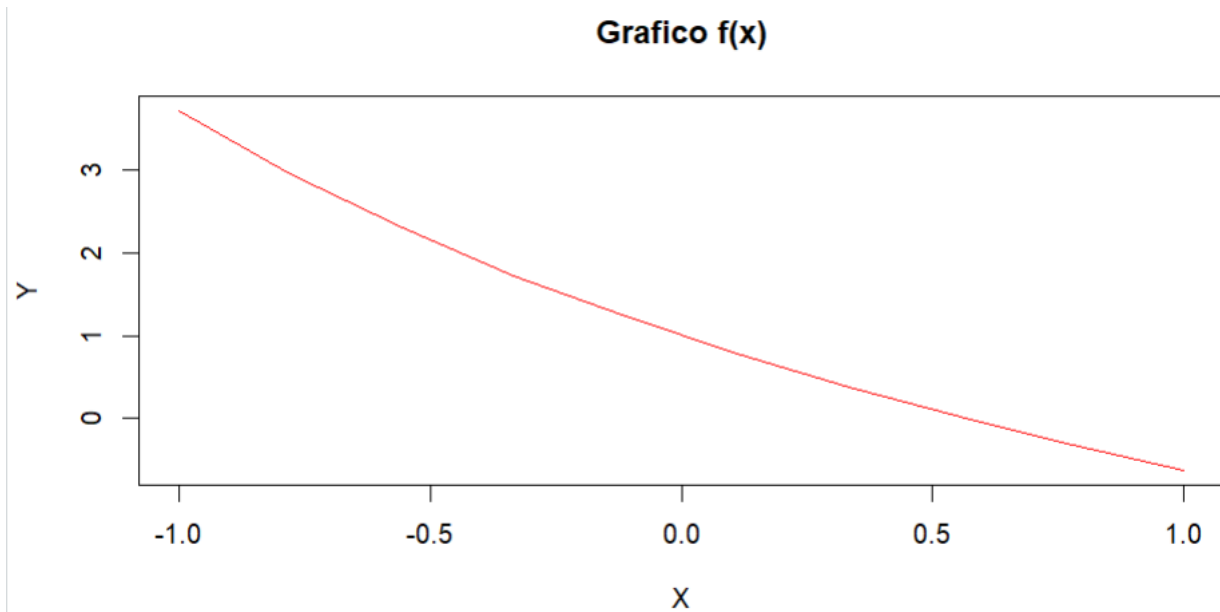
Esta linea imprime en la consola que la solucion converge e imprime el numero de interacciones ademas de la raiz de la ecuacion expresada en x1 ademas de mostrar todas las interacciones ralizadas por lo que muestra una lista ademas de mostrar que se alcanzo el numero maximo de interacciones al encotrar x1 antes de la interaccion 50 que definimos anteriormente

```
+      x0=x1,f
[1] 1.0000000 1.0000000 0.0000000 0.6126998
[1] 2.0000000 0.6126998 0.0000000 0.5721814
[1] 3.0000000 0.5721814 0.0000000 0.5677032
[1] 4.0000000 0.5677032 0.0000000 0.5672056
[1] 5.0000000 0.5672056 0.0000000 0.5671502
[1] 6.0000000 0.5671502 0.0000000 0.5671441
[1] 7.0000000 0.5671441 0.0000000 0.5671434
[1] 8.0000000 0.5671434 0.0000000 0.5671433
[1] 9.0000000 0.5671433 0.0000000 0.5671433
[1] " "
La soluci3n converge en 9 iteraciones. raÅz= 0.5671433>
> print("MÁximo número de iteraciones alcanzada !!!")
[1] "MÁximo número de iteraciones alcanzada !!!"
> |
```

Por lo que tenemos en nuestra ventana los valores que utilizamos

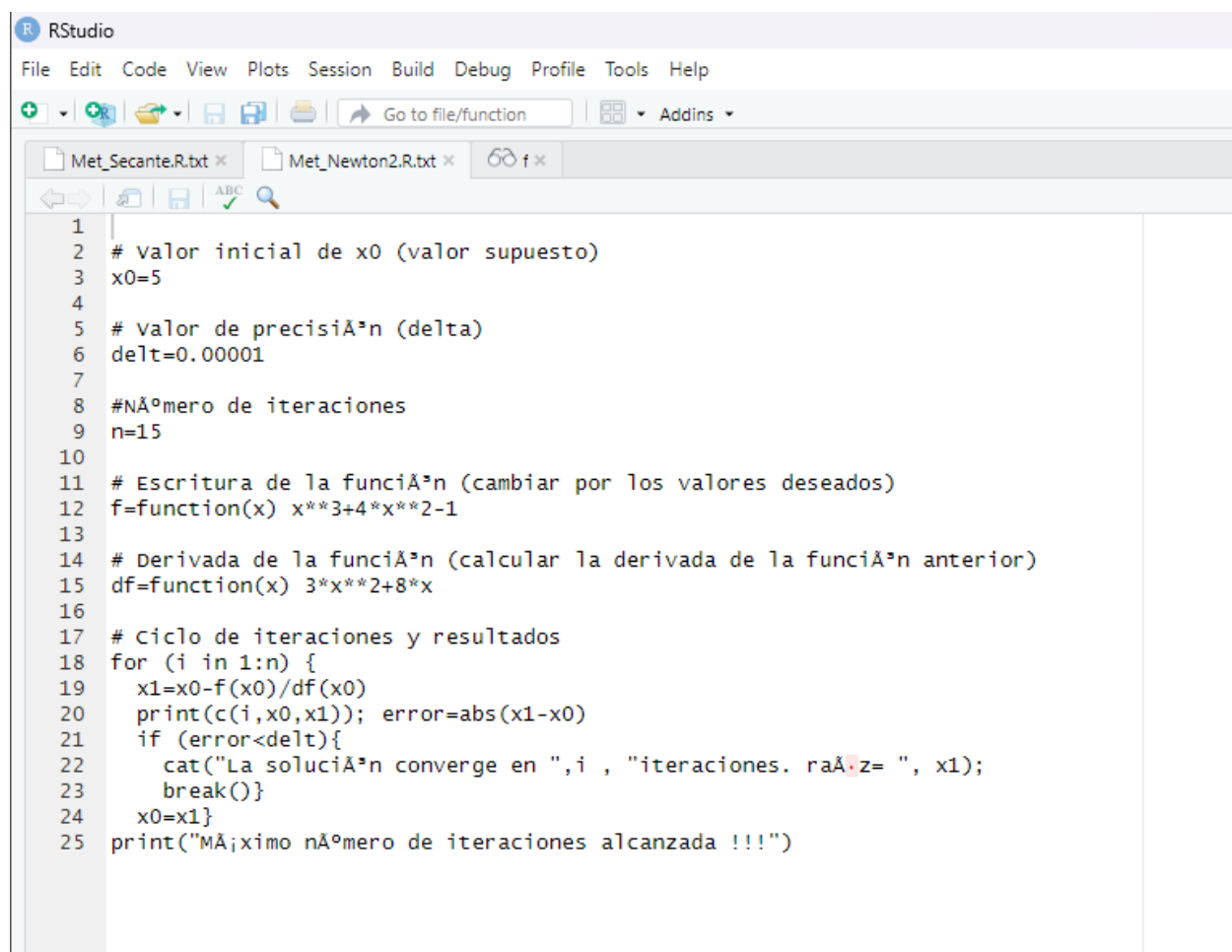
Values	
delt	1e-08
denomi	1.0000000149221
error	8.4629698804406e-09
i	9L
n	50
numera	8.46297004341093e-09
x0	0.567143299931633
x1	0.567143291468663
xant	0
Functions	
f	function (x)

Concluimos la raíz x_1 es de 0.5671433 y converge o se soluciona en la interacción número 9 donde podemos verificar que se aproxima al resultado



Metodo de Newton-Raphson.

Abrimos nuestro archivo en r studio para continuar con el siguiente método de newton-Raphson



```

1
2 # valor inicial de x0 (valor supuesto)
3 x0=5
4
5 # valor de precisión (delta)
6 delt=0.00001
7
8 #Número de iteraciones
9 n=15
10
11 # Escritura de la función (cambiar por los valores deseados)
12 f=function(x) x**3+4*x**2-1
13
14 # Derivada de la función (calcular la derivada de la función anterior)
15 df=function(x) 3*x**2+8*x
16
17 # ciclo de iteraciones y resultados
18 for (i in 1:n) {
19   x1=x0-f(x0)/df(x0)
20   print(c(i,x0,x1)); error=abs(x1-x0)
21   if (error<delt){
22     cat("La solución converge en ",i , "iteraciones. raíz= ", x1);
23     break()}
24   x0=x1}
25 print("Máximo número de iteraciones alcanzada !!!")

```

Ingresamos nuestro valor inicial para x0 que sera de 5 ademas de establecer nuestra variable delta de 0.00001 que sera nuestro error permitido en la ecuacion y el numero maximo de las interacciones de n=15

```

> x0=5
> delt=0.00001
> n=15
>

```

Por lo que nuestros valores iniciales son los siguientes

values	
delt	1e-05
n	15
x0	5

Escribimos la funcion para encontrar la raiz x1

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

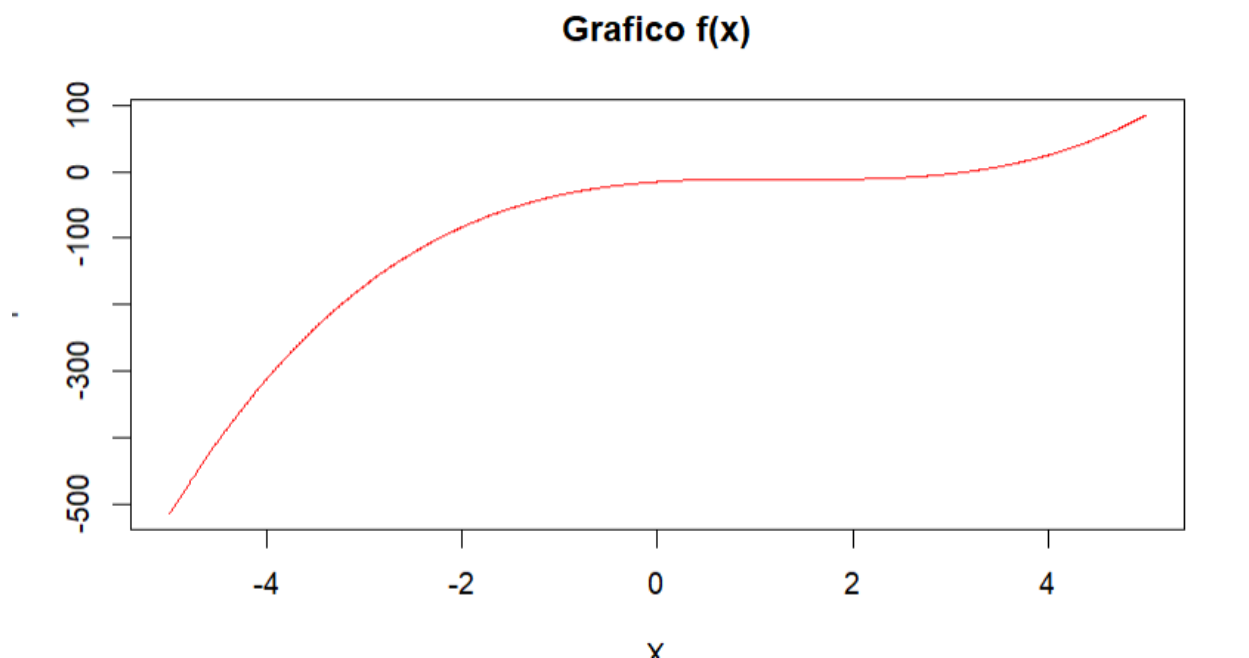
```
f=function(x) 2*x^3 - 8*x^2 +10*x -15
```

```
> x0=5
> delt=0.00001
> n=15
> f=function(x) 2*x^3 - 8*x^2 + 10*x - 15
> |
```

Una vez que agreguemos la funcion, realizamos la grafica para verificar que el problema tiene una solucion o mas por lo que ingresamos los siguientes datos para graficar esta funcion

```
> plot(f,-5,5,
+      lwd=1,
+      main= "Grafico f(x)",
+      col="red",
+      xlab="X",
+      ylab="Y",
+      axes=TRUE,
+      n=1000)
> |
```

Seleccionamos los valores y graficamos la funcion



Podemos ver que en el eje x la linea cruza desde $x=0$ hasta $x=2$ de manera horizontal para despues subir y vemos que desde el eje Y se mantiene en el valor cercano a 0 mientras pasa por los valores de $x=0$ a $x=2$ por lo tanto esta ecuacion tiene solucion, procedemos a realizar la derivada de la funcion para calcular x_1 por lo que derivamos la funcion

$$f = \text{function}(x) \ 2 \cdot x^3 - 8 \cdot x^2 + 10 \cdot x - 15$$

que nos da $2 \cdot 3 = 6$ y aplicamos la regla de la potencia $n-1$ por lo que al ser potencia de 3, se le resta 1 y nos quedara $6x^2$, hacemos la misma operación con la siguiente formula de $8x^2$ que nos da resultado como $16x$ y por ultimo la operación de $10x$ y como esta x se representa como x^1 nos dara 10 ya que su potencia se le resta 1

en este caso necesitamos una x para el numero final de 15 por lo que al derivarla nos da 0 y no la agregamos en la funcion derivada, por lo que la funcion derivada es la siguiente

$$df = \text{function}(x) \ 6x^2 - 16x + 10$$

```
> df=function(x) 6x**2 - 16x + 10
Error: unexpected symbol en "df=function(x) 6x"
> df=function(x) 6*x**2 - 16*x + 10
>
```

Ingresamos la funcion en Rstudio respetando las reglas de como es que se representan las potencias y las variables en Rstudio ya que al escribirlas mal, nos dara un error de sintaxis por lo que ejecutamos el siguiente codigo

```
> for (i in 1:n) {
  x1=x0-f(x0)/df(x0)
  print(c(i,x0,x1)); error=abs(x1-x0)
  if (error<delta){
    cat("La solución converge en ",i , "iteraciones. raíz= ", x1);
    break()}
  x0=x1}
print("Máximo número de iteraciones alcanzada !!!")
```

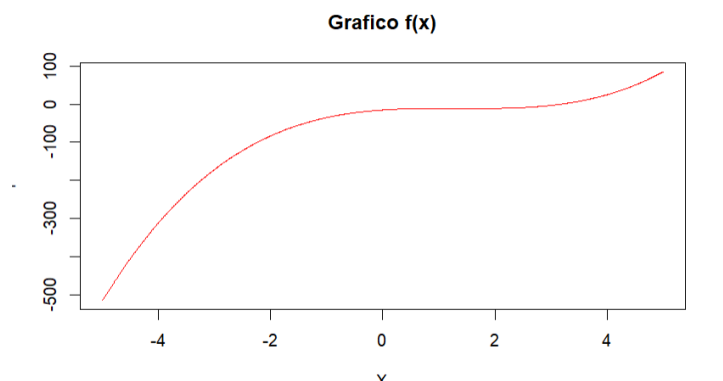
Esto definirá una variable i para las interacciones donde se ejecutará hasta encontrar x1 o hasta alcanzar el número máximo de interacciones que definimos anteriormente que es 15 interacciones máximas, para ello definirá (x1) como resultado de la división entre la función de (x0) entre la derivada de la función (x0) restandola a la variable de (x0) por lo que en la siguiente línea se pide que imprima en consola el número de interacciones, el valor de x0 y el valor calculado de x1, además calcula el error absoluto restando el valor calculado de x1 menos el valor de x0 calculado anteriormente y en caso de que el error calculado sea menor a nuestro error permitido guardado en la variable delta, esta mostrará un texto donde nos diga en qué interacción converge la solución mostrando la variable i además de la variable x1 mostrando la raíz de la función calculada, y en el caso contrario, volverá a repetir la operación hasta encontrar el error menor a delta o al llegar al número máximo de interacciones

```

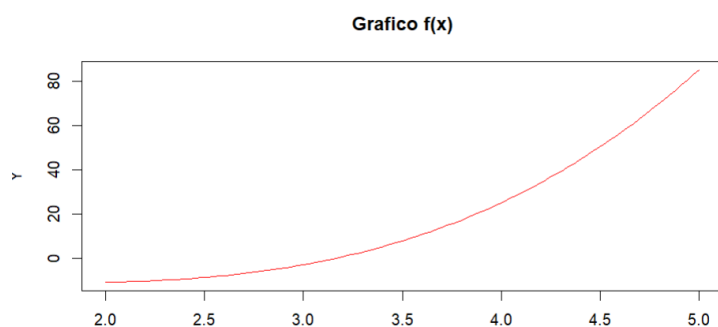
[1] 1.0000 5.0000 3.9375
[1] 2.000000 3.937500 3.376903
[1] 3.000000 3.376903 3.190023
[1] 4.000000 3.190023 3.169282
[1] 5.000000 3.169282 3.169038
[1] 6.000000 3.169038 3.169038
La solución converge en 6 iteraciones. raíz= 3.169038>
anzada !!!")
[1] "Máximo número de iteraciones alcanzada !!!"
>

```

Por lo que vemos en la tabla, podemos deducir que la solución converge en la 6ta iteración además de mostrarnos la raíz de la ecuación de 3.169038 por lo que verificamos que el valor es aproximado a la solución, ya que verificamos que en nuestro gráfico, la línea cruza el eje Y de ser negativo a positivo pasando por el valor de 0 mientras el eje x se encuentra cercano al valor de 3.169038



Al cambiar los datos para visualizar la tabla vemos que la ecuación tiene solución acercándose al valor que encontramos de 3.169038



Conclusión.

Para poder realizar las ecuaciones primero tendremos que hacer la gráfica de la función ya que esta grafica nos dará la respuesta para saber si la gráfica tiene solución o no, por lo que puede tener muchas soluciones o no tener ninguna solución, es de mucha utilidad el tener la opción de poder graficar una función ya que nos da la idea de donde estará el valor aproximado para la función a calcular o si no tiene solución y no tener que calcularla, es un proceso fácil el nombrar las variables así como poder realizar las ecuaciones ya que las fórmulas matemáticas son fáciles de representar conociendo la fórmula de la función así como las divisiones de la función entre la derivada de la misma función para obtener un valor para calcular el error absoluto de la ecuación utilizando el método de Newton-Raphson ya que es la mas eficiente para calcular estas ecuaciones, por lo que utilizando r studio, nos permite realizar ciclos para automatizar el proceso de la solución para la función seleccionada.