Universidad Corporativa Coppel | Programas Académicos

Ingeniería en Desarrollo de Software

Actividad | 2 | Método de Secante y Newton Rahpson

Métodos Numéricos

Ingeniería en Desarrollo de Software

academiaglobal

TUTOR: Miguel Ángel Rodríguez Vega

ALUMNO: Adriana Esteban López

FECHA: 12 de noviembre de 2024

INDICE

e

INTRODUCCIÓN

En el transcurso de la presente actividad se estará trabajando con los siguientes métodos:

1. **Método de Newton Rahpson** el cual consiste en encontrar aproximaciones de las raíces de una función real que pueda ser derivable, aunque no es un método infalible si es uno de los más utilizados.

   Es una técnica numérica con la cual se puede encontrar la solución a sistemas de ecuaciones no lineales a partir de aproximaciones sucesivas.

   En esencia la idea de este método consiste en comenzar con una aproximación inicial, aproximar la función por su línea tangente y finalmente calcular la intersección con el eje x de esta línea tangente.

2. **Método de la Secante** es un método para encontrar los ceros de una función de forma interactiva, es una variación el método de Newton Rahpson, en la cual, en lugar de estar calculando derivadas, se aproxima la pendiente en la recta que une la función evaluada en el punto de estudio y en el punto de la iteración anterior.

   Se podría llegar a considerar este método como una simplificación al método de Newton Rahpson ya que en lugar de tener que realizar una derivada se aproxima por una recta secante

DESCRIPCIÓN

1. Ejecutar los siguientes archivos en lenguaje R de acuerdo a las funciones que se tienen que resolver con cada uno de los métodos

```
#        == MÃ©todo de la Secante ==

# Valores iniciales
x0=1; xant=0;error=0;
# Error permitido (delta)
delt=0.00000001;
# NÃºmero de mÃ¡ximo iteraciones
n=50

# FunciÃ³n para encontrar si raÃz
f=function(x) exp(-x) - x

# CÃ¡lculos usando el mÃ©todo
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 soluciÃ³n converge en ",i , "iteraciones. raÃz= ", x1);
    break()}
  x0=x1}

print("MÃ¡ximo nÃºmero de iteraciones alcanzada !!!")
```

```
# Valor inicial de x0 (valor supuesto)
x0=5

# Valor de precisiÃ³n (delta)
delt=0.00001

#NÃºmero de iteraciones
n=15

# Escritura de la funciÃ³n (cambiar por los valores deseados)
f=function(x) x**3+4*x**2-1

# Derivada de la funciÃ³n (calcular la derivada de la funciÃ³n anterior)
df=function(x) 3*x**2+8*x

# 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 soluciÃ³n converge en ",i , "iteraciones. raÃz= ", x1);
    break()}
  x0=x1}
print("MÃ¡ximo nÃºmero de iteraciones alcanzada !!!")
```

2. Resolver la ecuación $f(\theta)=\sin(\theta)+\cos(1-\theta 2)-1$ con el método de la secante
3. Resolver la ecuación $f(x)=2x3-8x2+10x-15$ con el método de Newton-Raphson
4. Análisis e interpretación de resultados de cada una de las funciones

JUSTIFICACIÓN

Ambos métodos son usados dentro de los Métodos numéricos en la solución de ecuaciones no lineales y cada uno tiene su ventajas y desventajas de acuerdo a su aplicación:

Newton Rahpson, se aplica a una ecuación o función de cualquier grado, proporciona una respuesta en forma numérica mientras que el método de la Secante de entrada no requiere del cálculo de la derivada, aunque este se base en el método de Newton Rahpson.

Aunque cada uno tiene sus ventajas de pendiendo de la problemática que se presenten, ninguno de los dos está exentos de presentar fallas, en el caso de Newton Rahpson puede fallar cuando la función tiene varios puntos de inflexión mientras que el Método de la Secante puede fallar en situaciones poco predecibles.

Es cuestión de analizar la función o la problemática que se presente para determinar cuál de los dos métodos nos sería de mayor utilidad.

DESARROLLO

En el desarrollo de esta actividad estaremos haciendo la programación de los Métodos de Newton Rahpson y de la Secante con las siguientes funciones:

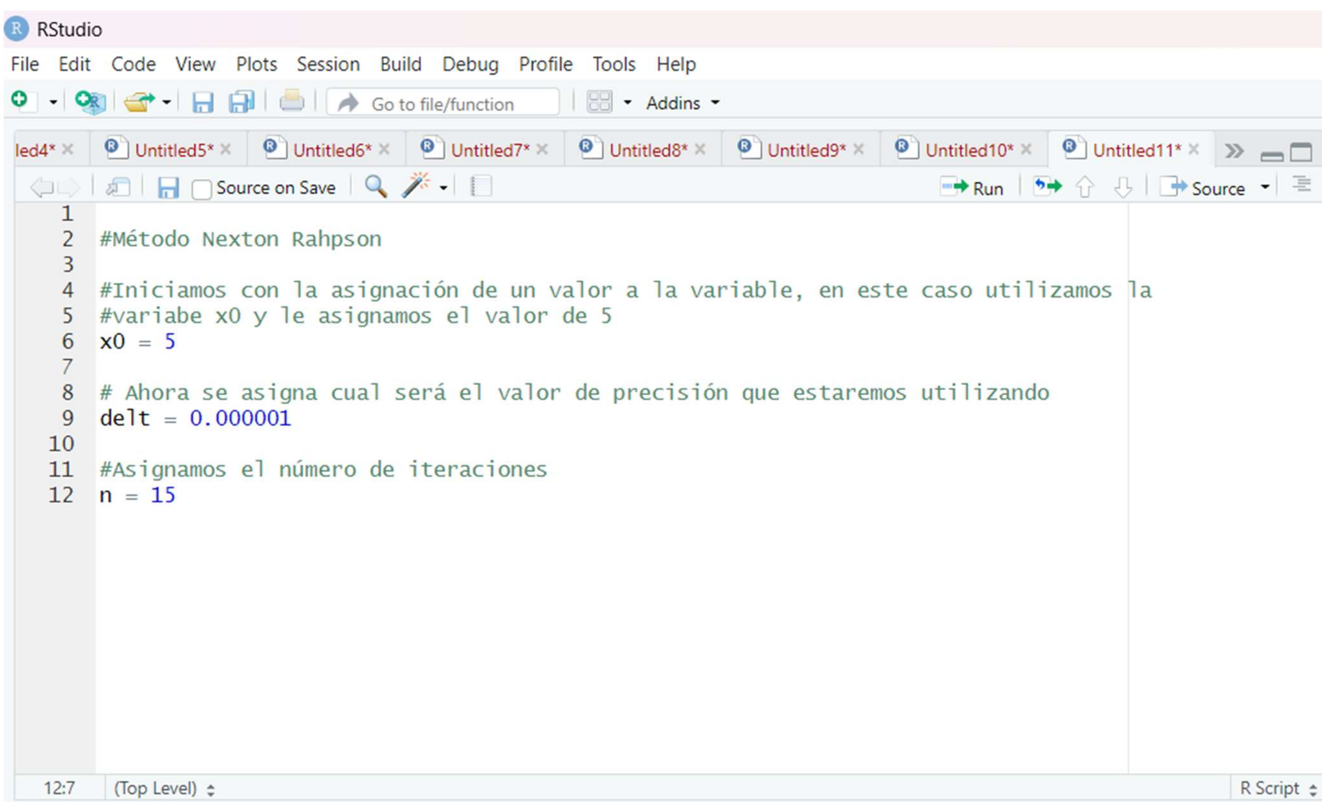
1. $f(\theta)=\sin(\theta)+\cos(1-\theta2)-1$ con el Método de la Secante
2. $\mathbf{f}(x)=2x3-8x2+10x-15$ con el Método de Newton Rahpson

Por lo que es necesario dar solución a las funciones que se nos indican a través de la programación en Rstudio

Método de Newton Rahpson

Para este método es necesario que tengamos un **valor inicial** y un **valor de precisión** para poder determinar en qué momento vamos a detener el cálculo; las variables a utilizar son: **x0 = 5** y **delt = 0.000001**

Otro dato importante es el número de iteraciones (número de veces que estaremos realizando un determinado cálculo), para lo cual estaremos utilizando la vacante **n = 15**

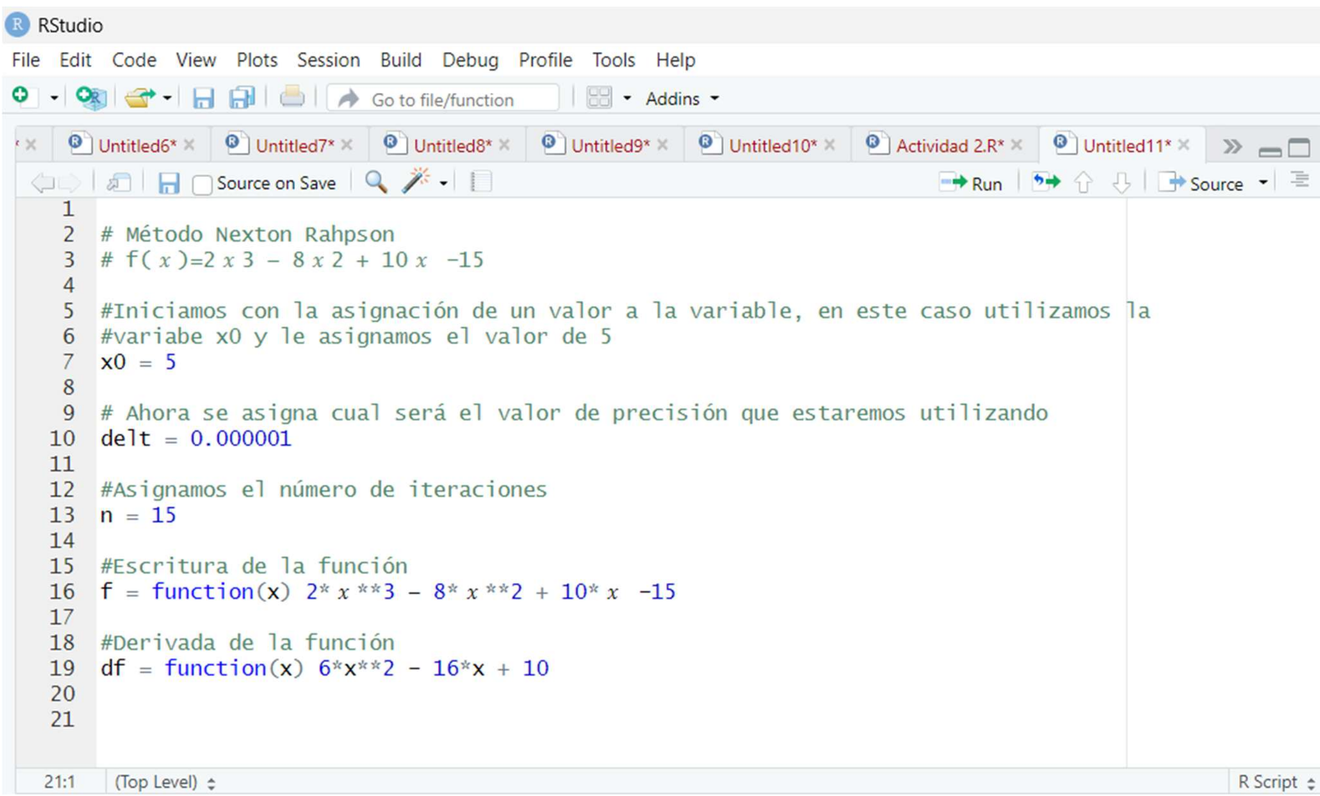


Con estos datos iniciales ya dentro de nuestro código, pasamos a la parte de la programación de la función, para lo cual estaremos haciendo uso de la variable **f:**

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

function es un comando que nos ayuda a estar mandando llamar la función cada vez que así lo indiquemos dentro de nuestra programación.

Una vez que se tiene la función lo primero que se realiza es obtener la **derivada de la función**, para lo cual utilizando la siguiente fórmula: $\mathbf{f'(x) = nx^{n-1}}$, entonces tenemos que: $\mathbf{f'\,(x) = 6x^2 - 16x + 10}$ y esta derivada la estaremos guardando en la variable **df**.

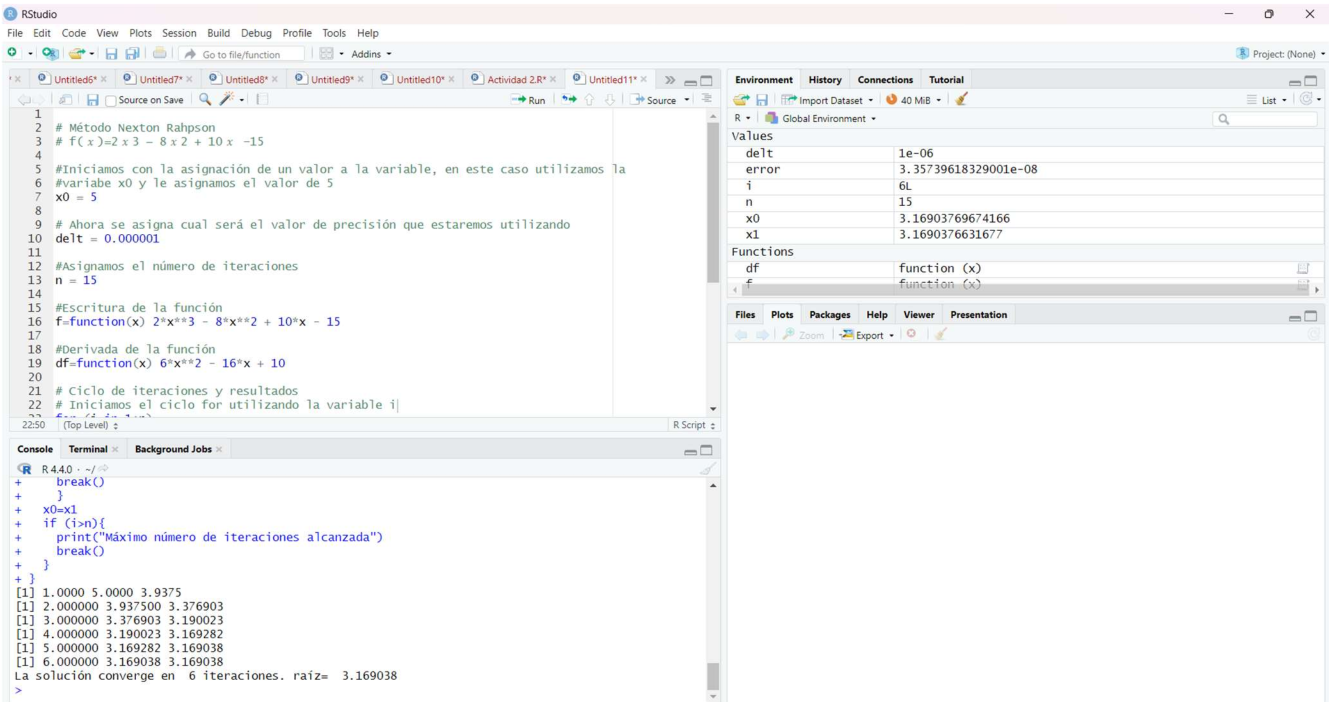


Continuamos ahora con la parte de la programación del ciclo de iteraciones que se requieren para encontrar el resultado; dentro de la programación es común utilizar los ciclos **while** y **for**; sin embargo, en esta ocasión al ser una serie de repeticiones lo que vamos a realizar se ajusta mejor el uso de **for.**

Iniciamos el ciclo for utilizando la variable i, en la cual se asigna un valor inicial de 1 hasta donde i valga n, que en este caso n vale 15; por tanto, estará realizando esta ejecución del ciclo 15 veces.

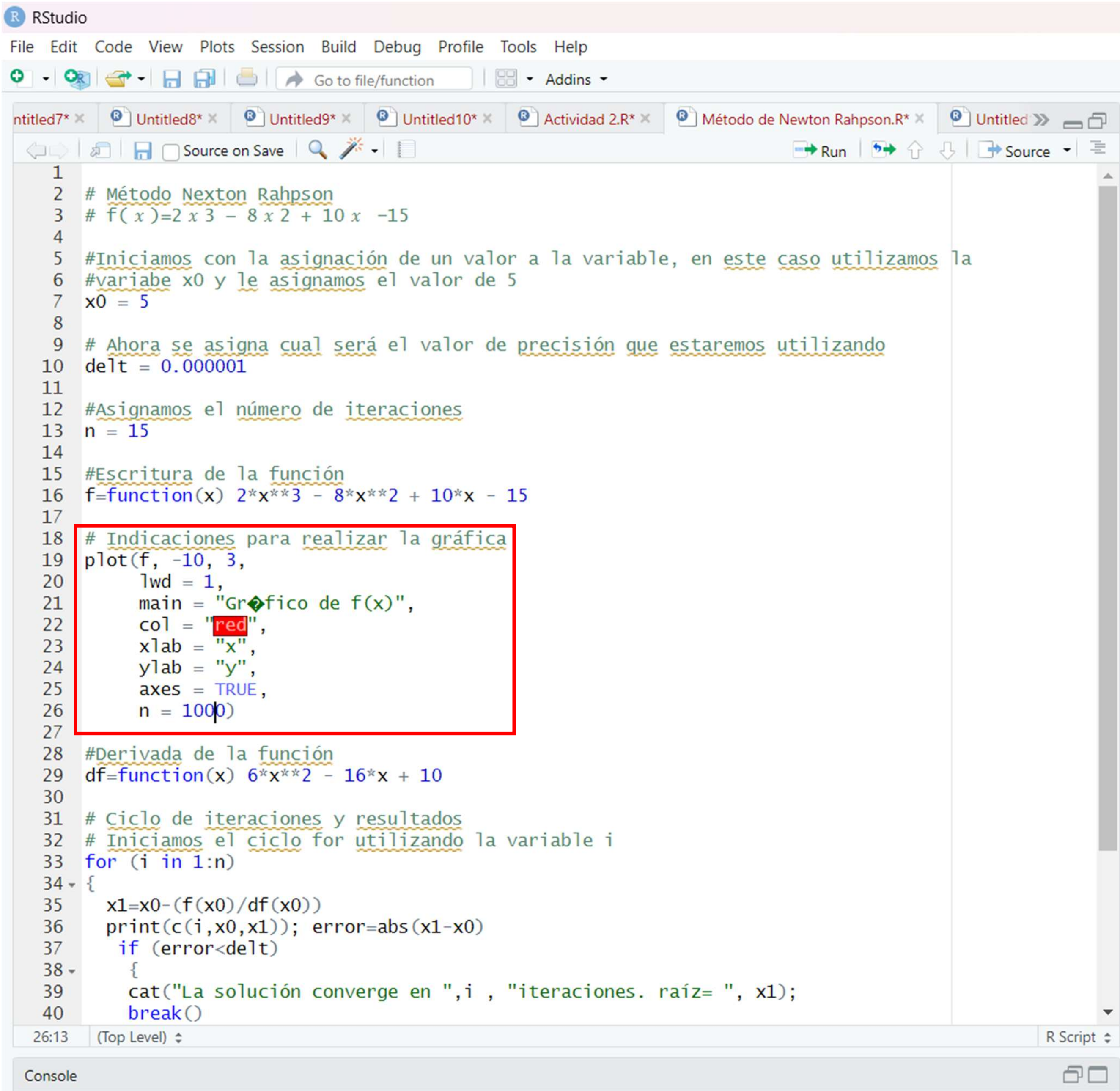
```r

# Método Nexton Rahpson
# f( x )=2 x 3 − 8 x 2 + 10 x  -15

#Iniciamos con la asignación de un valor a la variable, en este caso utilizamos la
#variabe x0 y le asignamos el valor de 5
x0 = 5

# Ahora se asigna cual será el valor de precisión que estaremos utilizando
delt = 0.000001

#Asignamos el número de iteraciones
n = 15

#Escritura de la función
f = function(x) 2*x**3 - 8*x**2 + 10*x -15

#Derivada de la función
df = function(x) 6*x**2 - 16*x + 10

# Ciclo de iteraciones y resultados
# Iniciamos el ciclo for utilizando la variable i
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. raíz= ", x1);
    break()
    }
  x0=x1
}
print("Máximo número de iteraciones alcanzada !!!")
```

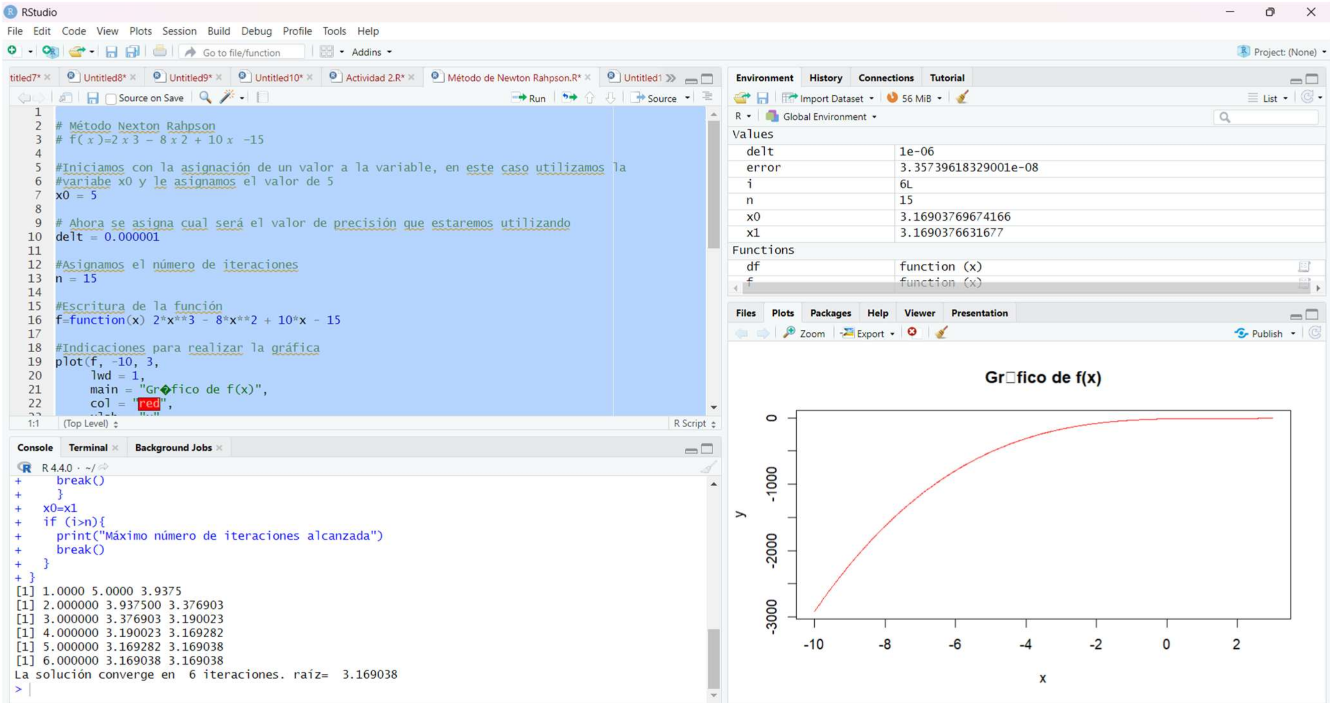
Ahora veamos la ejecución del programa que se acaba de capturar:



Esto nos indica que en la interacción 6 obtenemos el resultado 3.169038

Así mismo podemos implementar el uso de la gráfica para poder verificar cuántas soluciones tiene el cálculo que acabamos de realizar, y para ellos vamos a insertar el siguiente código:

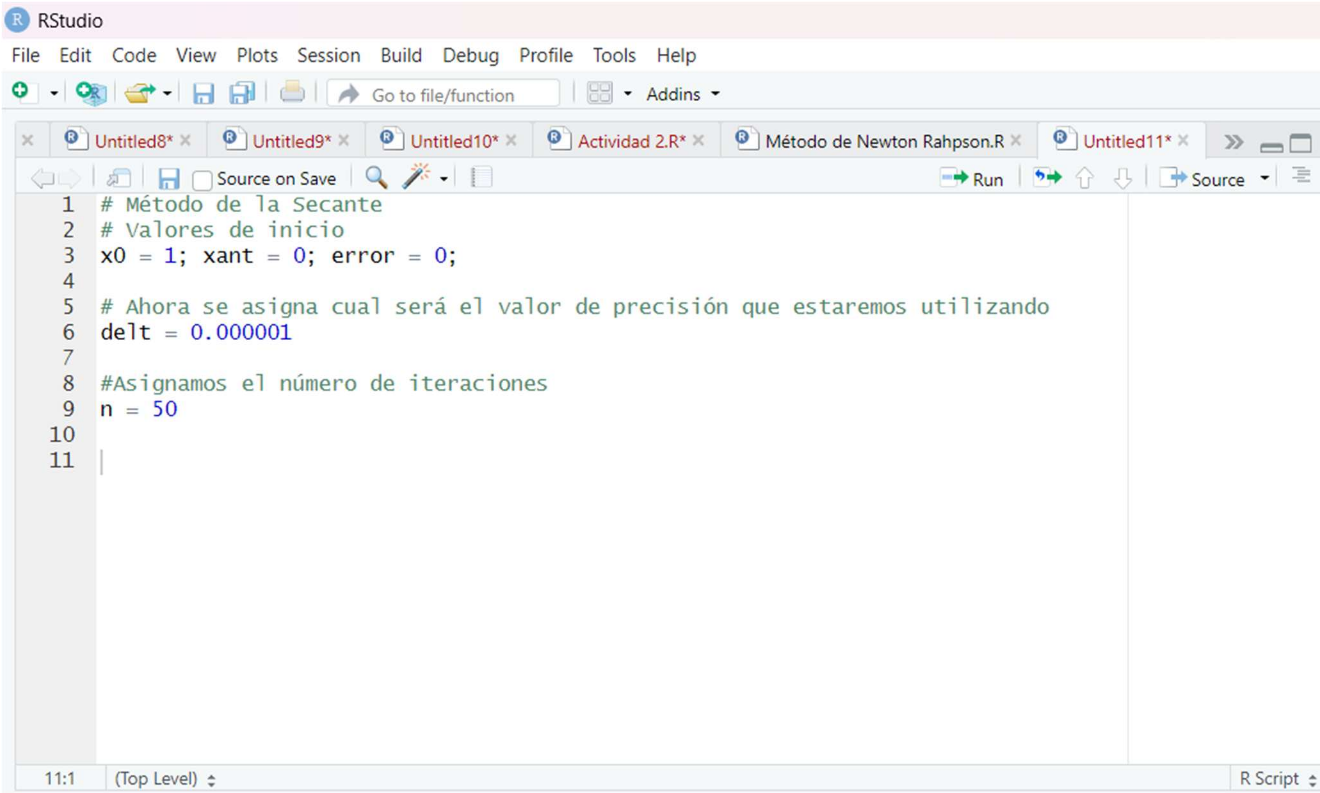

RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
ntitled7*
Untitled8*
Untitled9*
Untitled10*
Actividad 2.R*
Método de Newton Rahpson.R*
Untitled
Source on Save
Run
Source
Método Nexton Rahpson
f(x)=2 x 3 − 8 x 2 + 10 x -15
#Iniciamos con la asignación de un valor a la variable, en este caso utilizamos la
#variabe x0 y le asignamos el valor de 5
x0 = 5
# Ahora se asigna cual será el valor de precisión que estaremos utilizando
delt = 0.000001
#Asignamos el número de iteraciones
n = 15
#Escritura de la función
f=function(x) 2*x**3 - 8*x**2 + 10*x - 15
# Indicaciones para realizar la gráfica
plot(f, -10, 3,
lwd = 1,
main = "Gr�fico de f(x)",
col = "red",
xlab = "x",
ylab = "y",
axes = TRUE,
n = 1000)
#Derivada de la función
df=function(x) 6*x**2 - 16*x + 10
# Ciclo de iteraciones y resultados
# Iniciamos el ciclo for utilizando la variable i
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. raíz= ", x1);
break()
26:13
(Top Level)
R Script
Console



Método de la secante

Retomando el mismo proceso de inicio con el Método de Newton tenemos que ingresar los datos iniciales; el método de la secante requiere que se tengan 2 valores iniciales que vamos a identificar con la variable **x0 y xant**, también se estará definiendo un error que será 0.

Otro dato a considerar es un error permitido, el cual también vamos a definir a través de la variable **delt** con un valor de 10^{-6} (0.000001), así como también el número de iteraciones que será de 50 y vamos almacenar en la variable **n**



Ahora pasamos a la parte de establecer la función con la que vamos a trabajar y a programar: **$f(\theta)$=sin(θ)+cos(1−θ2)–1**

Para el Método de la Secante se tiene que seguir la siguiente fórmula (que estaremos aplicando a nuestra función):

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

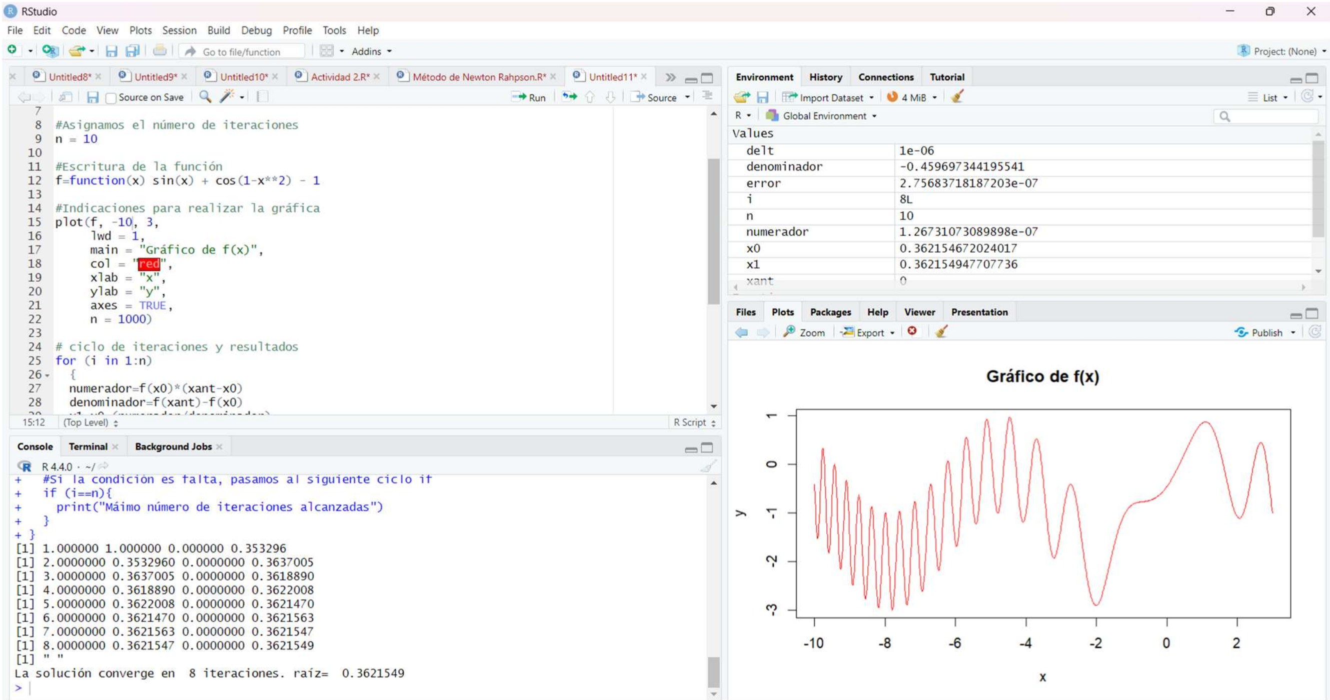
Al ser una operación un tanto "larga" vamos primero a obtener por separado cada una de las partes:

1. Numerador $f(x_a)\cdot(x_b - x_a)$ variable numerador
2. Denominador $f(x_b) - f(x_a)$ variable denominador
3. Resultado (Variable x0) = x0 – (numerador / denominador)

Así mismo, como en el método anterior, vamos a considerar realizar la gráfica de la función:

```r

# Ahora se asigna cual será el valor de precisión que estaremos utilizando
delt = 0.000001

#Asignamos el número de iteraciones
n = 50

#Escritura de la función
f=function(x) sin(x) + cos(1-x**2) - 1

#Indicaciones para realizar la gráfica
plot(f, -3, 3,
     lwd = 1,
     main = "Gráfico de f(x)",
     col = "red",
     xlab = "x",
     ylab = "y",
     axes = TRUE,
     n = 1000)

# ciclo de iteraciones y resultados
for (i in 1:n)
  {
  numerador=f(x0)*(xant-x0)
  denominador=f(xant)-f(x0)
  x1=x0-(numerador/denominador)
  print(c(i, x0, xant, x1)); error=abs(x1-x0)
  #si esta consicional es verdadera se imprime un resultado
  if (error<delt)
    {
    print(" ")
    cat("La solución converge en ",i , "iteraciones. raíz= ", x1);
    break()
    }
  x0=x1
  #Si la condición es falta, pasamos al siguiente ciclo if
  if (i==n){
    print("Máimo número de iteraciones alcanzadas")
  }
}
```

Ahora pasamos a la ejecución del programa:



CONCLUSIÓN

El Método de Newton Rahpson puede ser eficiente y rápido para obtener raíces de una ecuación o función, ya que generalmente se requiere de 5 a 6 iteraciones para obtener un resulta con una buena precisión, pero su uso requiere que se tenga el conocimiento para obtener la 1ra derivada de la función en cuestión para poderla trabajar, sin embargo, uno de los posibles errores que puede llegar a presentar es que se presente la división en 0, en cuyo momento el resultado es error.

En cuanto al método de la secante se puede ver que no requiere del uso de la derivada, solo requiere de una evaluación en cada iteración, sin embargo, y es muy efectivo cuando la ecuación o función a resolver es demasiado compleja para hacer la derivada aunque suele ser más lento que el de Newton Rahpson.

Se agrega dicha actividad a la plataforma de GitHub a través del siguiente link:

https://github.com/22HADRIA/M-todos-Num-ricos

REFERENCIAS ELÉCTRONICAS

El Método de Newton

http://www.objetos.unam.mx/matematicas/matema/Daplica/da_aplicacion08_d.html#:~:text=El%20M%C3%A9todo%20de%20Newton&text=Se%20trata%20de%20un%20procedimiento,variable%20real%20que%20sea%20derivable.

Método de Newton Rahpson

http://personal.cimat.mx:8181/~julio/courses/progra01/clase18/MetododeNewtonRaphson.pdf

Aplicación de la técnica numérica en la solución de una ecuación no lineal

https://www.urp.edu.pe/pdf/id/2555/n/application-of-numerical-methods-to-solve-nonlinear-equations-for-sea-wave-modeling

El Método

file:///C:/Users/adri2/Downloads/Dialnet-ElMetodoDeNewtonraphsonParaLaObtencionDeRaicesDeEc-7894459.pdf

El método de Newton Rahpson Univariado

https://rpubs.com/Jaimemosg/431885

Métodos Numéricos

https://www3.fi.mdp.edu.ar/metodos/apuntes/secante_rodrigo.pdf

Matemáticas avanzadas

https://es.slideshare.net/slideshow/ventajas-y-desventajas-de-mtodos-de-biseccin-secante-y-newton-raphson/46204722#3