

Actividad 2 - Método de Secante y Newton.

Métodos Numéricos.

Ingeniería en Desarrollo de Software

Tutor: Miguel Ángel Rodríguez Vega

Alumno: Ramón Ernesto Valdez Felix

Fecha: 23/10/2023

Índice

Introducción	3
Descripción	3
Justificación	4
Desarrollo	4
Ecuación método Secante	5
Ecuación método Newton-Raphson	6
Interpretación de Resultados	8
Conclusión	11
Referencias	11

Introducción

En la actividad 2 de la materia de métodos numéricos nos pide resolver, ejecutar e interpretar los archivos script del lenguaje r que se nos proporciona en la actividad dos de la materia, se debe ejecutar en la herramienta de rstudio para obtener el resultado de las funciones a remplazar en el script y así obtener el resultado de las iteraciones recorridas para llegar a la solución de dicha expresión, el método numérico son proceso matemático iterativo cuyo objetivo es encontrar la aproximación a una solución específica con un cierto error previamente determinado. Las cuales son aplicaciones en algoritmos mediante las posible formular y solucionar problemas matemáticos utilizando operaciones aritméticas menos complejas. También son conocidas como métodos indirectos, un análisis numérico idealiza y concibe métodos para aprobar, de forma eficiente, las soluciones de problemas expresados matemáticamente. El objetivo principal del análisis numérico es encontrar soluciones aproximadas para problemas complejos.

Descripción

En esta actividad 2 se nos solicita la creación del documento con el nombre siguiente de la actividad :“ Método de Secante y Newton” esto nos dará el derecho a ser calificada para así obtenerte la puntuación de la calificación final de la materia métodos numéricos impartida por el docente o maestro asignado, En esta actividad se requiera la ejecución, solución e interpretación de los script que comprenden la actividad, así como el realizar el cambio de la funciona a ejecutar en los métodos de la secante y newton-raphson utilizando la aplicación de rstudio que es asignada por el mismo material de la académico a ser usado para llegar a la solución correcta de la función y la ejecución de los script asignados.

Justificación

En esta actividad se trabajará con dos scripts los cuales son proporcionado por la academia en el material de la actividad 2 esto para obtener el resultado del funcionamiento correcto del script utilizando dos métodos: uno método de Secante y como segundo método newton-raphson, adicional se anexan algunas actividades que se deben realiza para llegar la solución final:

- Descargar script y ejecutar el archivo con el Lenguaje R.
- Subirlo al GitHub el documento realizado compartiendo el link para que pueda consultar el docente o maestro.
- Resolver la ecuación por el método de Secante.
- Resolver la ecuación por el método de Newton-Raphson.
- Analizar e interpretar los resultados.

Desarrollo:

En esta actividad realizaremos las actividades de la solución de dos ecuaciones usando dos métodos de la materia de métodos números que son los siguientes: el método de secante y newton-raphson. Donde la realizar el cambio de las funciones proporcionadas para cada script se continuará con la ejecución para llegar a la solución e interpretar el desarrollo de cada una de las ecuaciones para el llenado de las evidencias que se requieren en los puntos siguientes a documentar.

Link: [GitHub](#)

Ecuacion método Secante.

En esta actividad se requiere utilizar la ecuación de método de secante para la solución, se realiza la descarga del script a utilizar con el método de secante “http://umi.edu.mx/coppel/IDS/plataforma/files/programasenr/MN_Secante.R” y la función que será utilizada para la solución del método de secante es la siguiente: $f(\theta) = \sin(\theta) + \cos(1 - \theta^2) - 1$ utilizando herramientas de rstudio que se nos pide usar en el documentó de la actividad 2.

Se agrega la evidencia del script utilizado para sacar la ecuación con el método de secante y se anexan imágenes de la ejecución del código con su resultado final, donde la solución nos dio el siguiente dato: converge en 10 iteraciones. Raíz= 0.3621549[1] ;Máximo número iteraciones alcanzadas!

##Método de Secante##

Valores iniciales

x0 = 1

xant = 0

error = 0

Error permitido (delta)

delt = 0.00000001

Numero de máximo iteraciones

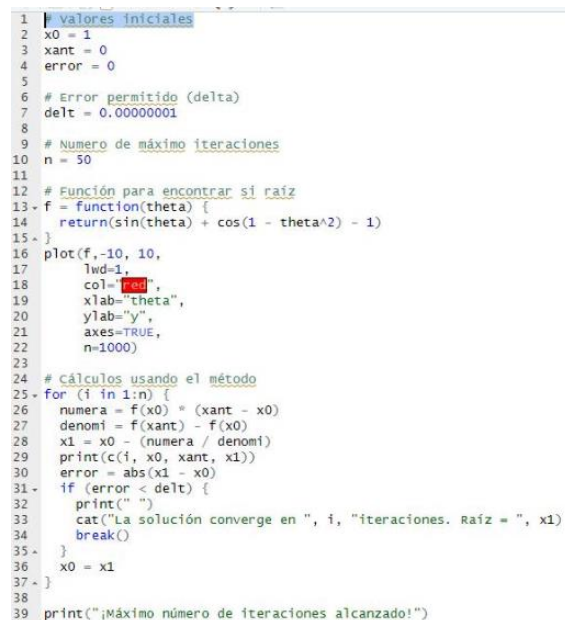
n = 50

Función para encontrar si raíz

```
f = function(theta) {
  return(sin(theta) + cos(1 - theta^2) - 1)
}
```

Grafica function(theta)

plot(f,-10, 10,



```
1 # Valores iniciales
2 x0 = 1
3 xant = 0
4 error = 0
5
6 # Error permitido (delta)
7 delt = 0.00000001
8
9 # Numero de máximo iteraciones
10 n = 50
11
12 # Función para encontrar si raíz
13 f = function(theta) {
14   return(sin(theta) + cos(1 - theta^2) - 1)
15 }
16 plot(f,-10, 10,
17       lwd=1,
18       col="red",
19       xlab="theta",
20       ylab="y",
21       axes=TRUE,
22       n=1000)
23
24 # cálculos usando el método
25 for (i in 1:n) {
26   numera = f(x0) * (xant - x0)
27   denomi = f(xant) - f(x0)
28   x1 = x0 - (numera / denomi)
29   print(c(i, x0, xant, x1))
30   error = abs(x1 - x0)
31   if (error < delt) {
32     print(" ")
33     cat("La solución converge en ", i, " iteraciones. Raíz = ", x1)
34     break()
35   }
36   x0 = x1
37 }
38
39 print("Máximo número de iteraciones alcanzado!")
40
```

```

lwd=1,

col="red",

xlab="theta",

ylab="y",

axes=TRUE,

n=1000)

# Cálculos usando el método

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 solución converge en ", i, "iteraciones. Raíz = ",

x1)

    break()

  }

  x0 = x1

}

print("¡Máximo número de iteraciones alcanzado!")

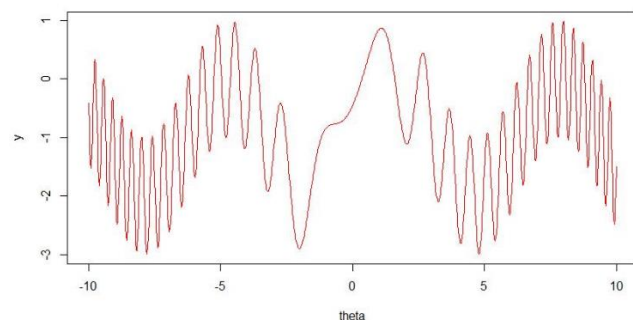
```

```

> source("C:/users/ranon.valdez/Desktop/umi/Periodo_3/04_Métodos_Numéricos/A2/secante_A2.R")
[1] 1.000000 1.000000 0.000000 0.353296
[1] 2.000000 0.353296 0.000000 0.3637005
[1] 3.000000 0.3637005 0.000000 0.3618890
[1] 4.000000 0.3618890 0.000000 0.3622008
[1] 5.000000 0.3622008 0.000000 0.3621470
[1] 6.000000 0.3621470 0.000000 0.3621563
[1] 7.000000 0.3621563 0.000000 0.3621547
[1] 8.000000 0.3621547 0.000000 0.3621549
[1] 9.000000 0.3621549 0.000000 0.3621549
[1] 10.000000 0.3621549 0.000000 0.3621549
[1] " "
[1] " "
La solución converge en 10 iteraciones. Raíz = 0.3621549[1] "¡Máximo número de iteraciones alcanzado!"
>

```

values	
delt	1e-08
denomi	-0.459697683728096
error	8.19619994096854e-09
i	10L
n	50
numera	3.76777413908576e-09
x0	0.362154900172911
x1	0.362154908369111
xant	0
Functions	
f	function (theta)



Ecuación método Newton-Raphson.

En esta actividad se requiere utilizar la ecuación de método de secante para la solución, se realiza la descarga del script a utilizar con el método de secante “<http://umi.edu.mx/coppel/IDS/plataforma/files/programasenr/ewton2.R>” y la función que será utilizada para la solución del método de secante es la siguiente: $f(x) = 2x^3 - 8x^2 + 10x - 15$ utilizando herramientas de rstudio que se nos pide usar en el documentó de la actividad 2.

Se agrega la evidencia del script utilizado para sacar la ecuación con el método de newton-raphson y se anexan imágenes de la ejecución del código con su resultado final, donde la solución nos dio el siguiente dato: converge en 6 iteraciones. Raíz= 3.169030[1] ;máximo número de iteraciones alcanzado!

##Método de Newton##

Valor inicial de x0 (valor supuesto)

x0 = 5

Valor de precisión (delta)

delt = 0.00001

Numero de iteraciones

n = 15

Escritura de la función (cambiar por los valores deseados)

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

Derivada de la función (calcular la derivada de la función anterior)

df = function(x) 6*x^2 - 16*x + 10

Grafica function(x)

plot(f,-10, 10,

lwd=1,

col="red",

xlab="x",

ylab="y",

axes=TRUE,

n=1000)

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) {

```
1 # Valor inicial de x0 (valor supuesto)
2 x0 = 5
3
4 # Valor de precisión (delta)
5 delt = 0.00001
6
7 # Numero de iteraciones
8 n = 15
9
10 # Escritura de la función (cambiar por los valores deseados)
11 f = function(x) 2*x^3 - 8*x^2 + 10*x - 15
12
13
14 # Derivada de la función (calcular la derivada de la función anterior)
15 df = function(x) 6*x^2 - 16*x + 10
16
17 plot(f,-10, 10,
18      lwd=1,
19      col="red",
20      xlab="x",
21      ylab="y",
22      axes=TRUE,
23      n=1000)
24
25 # Ciclo de iteraciones y resultados
26 for (i in 1:n) {
27   x1 = x0 - f(x0) / df(x0)
28   print(c(i, x0, x1))
29   error = abs(x1 - x0)
30   if (error < delt) {
31     cat("La solución converge en ", i, " iteraciones. raíz = ", x1)
32     break()
33   }
34   x0 = x1
35 }
36 print("Máximo número de iteraciones alcanzadas !!!")
```

```
> source("C:/users/ramon.valdez/Desktop/unl/Periodo_3/04_Método")
[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[1]
> |
```

Values	
delt	1e-05
error	3.35739618329001e-08
i	6L
n	15
x0	3.16903769674166
x1	3.1690376631677
Functions	
df	function (x)
f	function (x)

```

cat("La solución converge en ", i, " iteraciones. raíz = ", x1)

break()

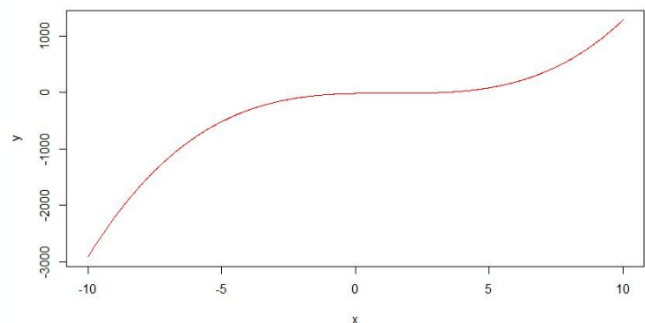
}

x0 = x1

}

print("Máximo número de iteraciones alcanzadas !!!")

```



Interpretación de resultados.

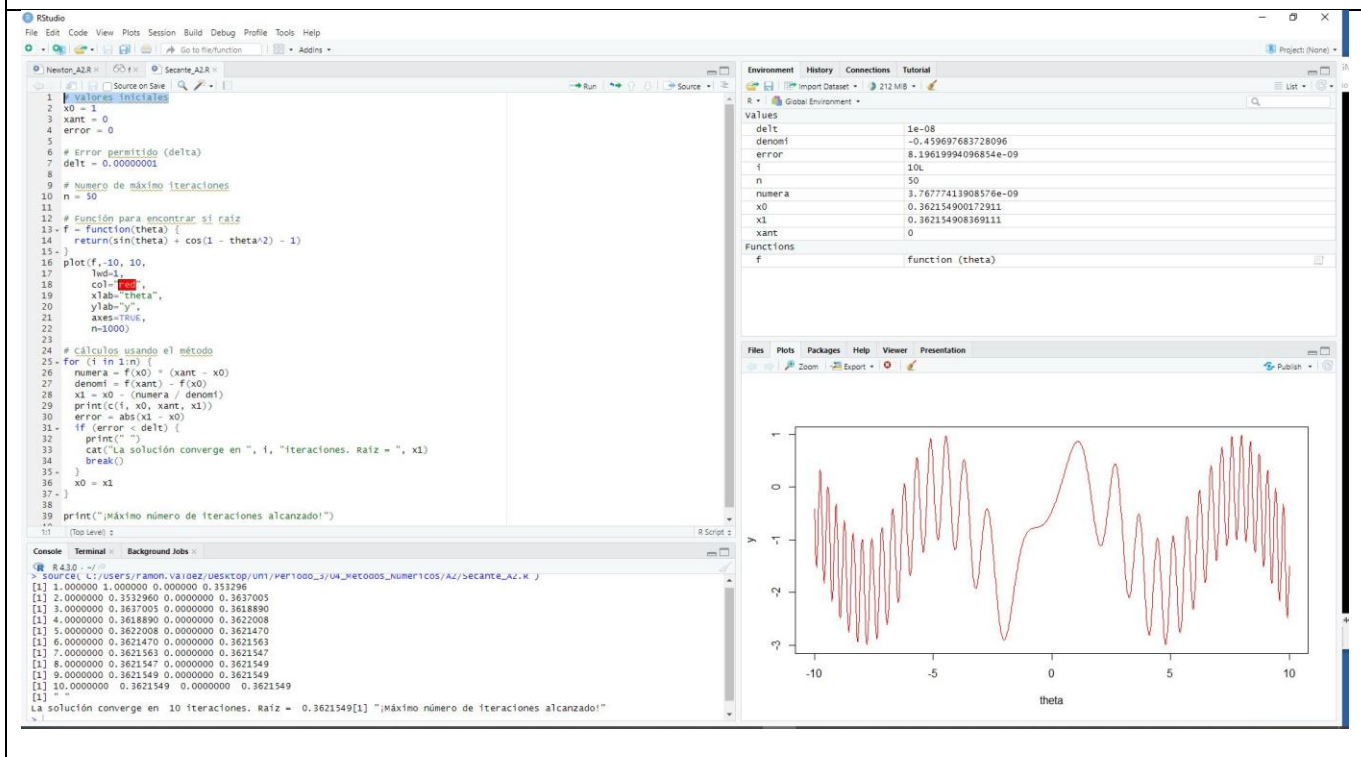
En este punto de la actividad daremos una breve interpretación personal de las ecuaciones de los métodos de secantes y newton-raphson anexaremos las pantallas de evidencia de la ejecución del código y el resultado obtenido en la herramienta rstudio del código proporcionado.

Método de secante breve descripción de lo realizado y evidencia de solución: converge en 10 iteraciones. Raíz= 0.3621549[1] ;Máximo número iteraciones alcanzadas!

En la siguiente ecuación se utilizó el método de secante para dar la solución: “ $f(\theta) = \sin(\theta) + \cos(1 - \theta^2) - 1$ ”.

- En primer paso para el uso de metodo de secante se ingresan las variable ($x_0=1$, $x_{ant}=0$ y $error=0$).
- Se anexa el numero de error permitido a encontrar en las iteraciones de la ejecucion $delt=0.0000001$
- Se asigna el valor maximo de las iteraciones a realizar con el siguiente dato $n=50$.
- Se utiliza la funcion $f(\theta) = \sin(\theta) + \cos(1 - \theta^2) - 1$ en el script y se trasforma de la siguiente manera $f=funcion(theta) \{return(\sin(theta) + \cos(1 - theta^2) - 1)\}$
- Se genera la grafica de la ecuacion a resolver.

- Se calcula utilizando el ciclo for para la solución de la ecuación dejando como resultado cada uno de los ciclos de las iteraciones hasta que se encuentre el error o solución que en este caso llega a una cantidad de 10 iteraciones, Raíz= 0.3621549[1], en caso de que no llegara a la solución el ciclo seguiría hasta que se cumplirán el máximo número de iteraciones que se asignó con una cantidad de 50 si esto se cumpliera nos diría que es necesario el incremento de las iteraciones para su solución.



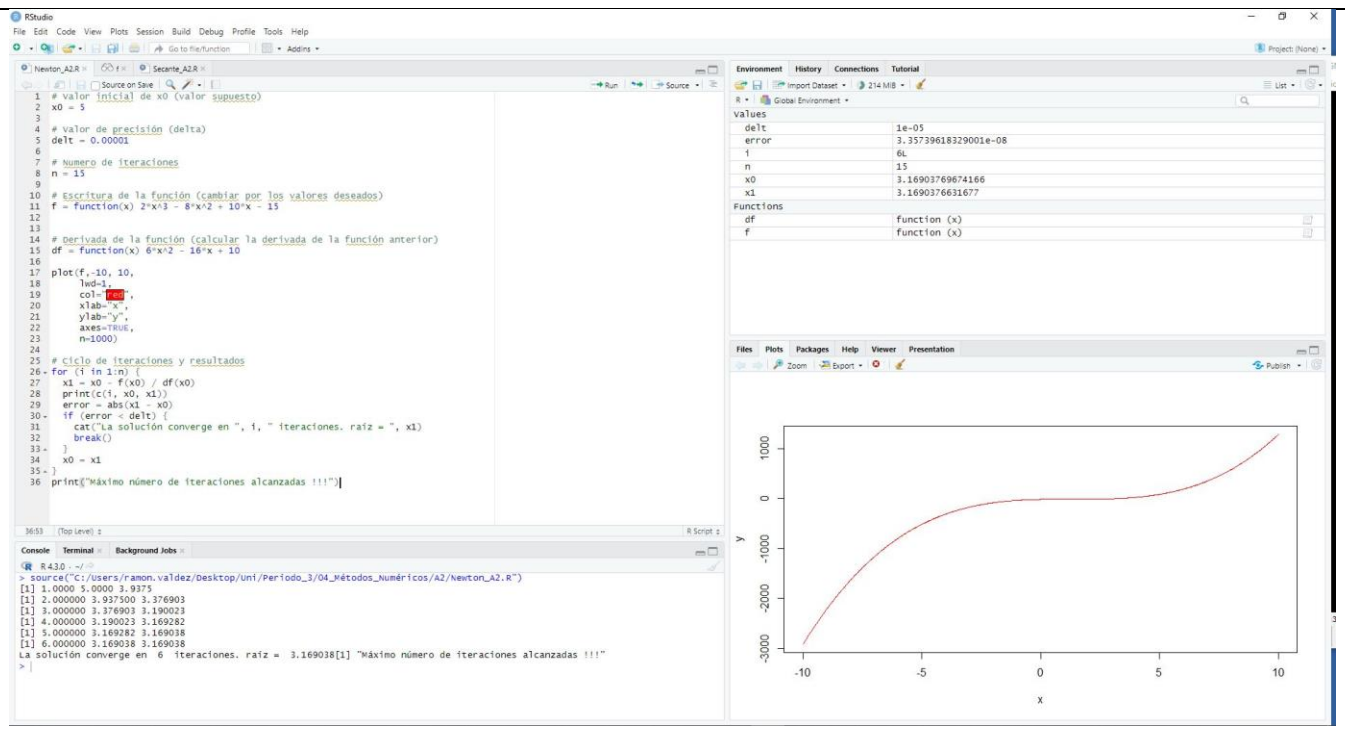
Método de newton-raphson breve descripción de lo realizado y evidencia de solución: converge en 6 iteraciones. Raíz= 3.169030[1] ;máximo número de iteraciones alcanzado!

En la siguiente ecuación se utilizó el método de secante para dar la solución: “ $f(x) = 2x^3 - 8x^2 + 10x - 15$ ”.

- En primer paso para el uso de metodo de secante se ingresan las variable ($x_0=5$).
- Se anexa el numero de error permitido a encontrar en las iteraciones de la ejecucion

delt=0.00001.

- Se asigna el valor maximo de las iteraciones a realizar con el siguiente dato n=15.
- Se utiliza la funcion $f(x) = 2x^3 - 8x^2 + 10x - 15$ en el script y trasformando de la siguiente manera $f=funcion(x) \ 2*x^3-8*x^2+10x-15$.
- Sacamos la derivada de la funcion quedando de la siguiente manera $df=funcion(x) \ 6*x^2-16*x+10$
- Se genera la grafica de la ecuacion a resolver.
- Se calcula utilizando el ciclo for para la solucion de la ecuacion de la derivada de la funcion dejando como resultado cada uno de los ciclos de las iteraciones has que se encuentre el error o solucion que en este caso llega a una cantidad de 6 iteraciones. Raíz= 3.169030[1], en caso de que no llegara a la solución el ciclo for seguiría hasta que se cumplirán el máximo número de iteraciones que se asignó con una cantidad de 15 si esto se cumpliera nos diria que es necesario el incremento de las iteraciones para su solucion.



Conclusión

En conclusión, en la vida cotidiana como podemos ver que los métodos numéricos han venido a ser muy importantes porque pueden aplicarse en distintos campos para encontrar resultados aproximados a sistemas complejos utilizando solo las operaciones matemáticas simples. También dentro de lo que es el análisis numérico podemos observar que aquí se identifican todos los errores que puedan analizar en pocas palabras el análisis de errores son algoritmos que los cuales son conjuntos de instrucciones cuyo fin es calcular la exactitud del error. El método de la secante se basa en el método de Newton-Raphson, donde no se requiere calcular la derivada. Este método casi nunca falla ya que solo requiere de dos puntos al principio, y después el mismo método se va retroalimentando, es decir, se va acomodando hasta que encuentra la raíz.

Referencias

GitHub: Let's build from here. (n.d.)

(N.d.). Edu.Mx. Retrieved October 26, 2023, from

https://umi.edu.mx/coppel/IDS/plataforma/files/programasenr/MN_Secante.R

(N.d.-b). Edu.Mx. Retrieved October 26, 2023, from

<https://umi.edu.mx/coppel/IDS/plataforma/files/programasenr/ewton2.R>

De jorgeyfloreth, L. T. las E. (2017, March 6). *Método Secante*. Métodos Numéricos.

<https://jorgeyfloreth.wordpress.com/2017/03/06/metodo-secante/>