Universidad Corporativa Coppel | Programas Académicos

Ingeniería en Desarrollo de Software

Actividad | 2 | Método de Secante y Newton

Nombre del curso

Ingeniería en Desarrollo de Software

TUTOR: Miguel Angel Rodriguez Vega.

ALUMNO: Uziel de Jesús López Ornelas.

FECHA: 10 de Noviembre del 2024

Índice

Introducción

Continuaremos con la materia de métodos numéricos, pero ahora nos toca resolver dos ecuaciones con métodos correspondientes, al principio se nos pide ejecutar el archivo en el lenguaje R, ese archivo contiene en código para correrlo de acuerdo con la sintaxis correcta de la ecuación, en lo personal no copiare el archivo y lo pegare tal cual, con la reunión del maestro lo hare paso por paso para acostumbrarme a lo que significa cada comando y función, después de ello se nos pide resolver una ecuación por el método de la Secante y una ecuación por el método de Newton-Raphson, las ecuaciones resueltas también tendrán su grafica correspondiente para visualizarlas de manera individual cada una de ellas, con todo hecho ya por último, interpretaremos los resultados para darles una explicación final. Como siempre se mandarán capturas de pantallas con descripciones para indicar lo que se esta explicando y dando a mostrar.

Descripción

Los métodos numéricos nos vuelven aptos para entender esquemas numéricos a fin de resolver problemas matemáticos, de ingeniería y científicos en una computadora, reducir esquemas numéricos básicos, escribir programas y resolverlos en una computadora, así como usar correctamente el software existente para dichos métodos (sintaxis correcta). No solo aumenta nuestra habilidad para el uso de computadoras, sino que también nos ayuda a tener más pericia matemática y comprensión de los principios científicos básicos. El mundo real tiene un número infinito de problemas y muchos de ellos no pueden resolverse de manera exacta, por lo que se necesita modelar matemáticamente una solución que se aproxime al resultado más exacto posible, soportando un error mínimo. El análisis numérico trata de diseñar métodos para “Aproximar de una manera eficiente las soluciones de problemas expresados matemáticamente, realizando cálculos puramente aritméticos y lógicos. Por ello es importante adaptarse a los diferentes cambios que la tecnología presenta, obteniendo las herramientas necesarias para lograr tener un aprendizaje que permita la resolución de problemas de manera efectiva.

Justificación

En esta actividad se centra específicamente en resolver nuestro problema en un lenguaje de programación que es “R”, uno se puede preguntar por qué no simplemente se resuelven estos problemas desde un cuaderno, la respuesta es simple, tenemos que adaptarnos a las tecnologías que tenemos en nuestra época, los diferentes softwares que tenemos son esenciales para resolver los problemas de una manera más rápida y que nos ayuda en nuestro desarrollo cognitivo y resolución de problemas, desde un simple Excel, hasta RStudio, son maneras en las que se pueden resolver diferentes maneras los problemas que tenemos, sobre todo en las actividades que tenemos en esta materia que nos plantean diferentes escenarios y es nuestro trabajo elegir que método es el correcto para ejecutarlo y que este sea correcto utilizando las diferentes herramientas que tenemos a nuestro alcance. En este caso nosotros por medio de los métodos necesarios logramos resolver varias ecuaciones gracias al software de RStudio.

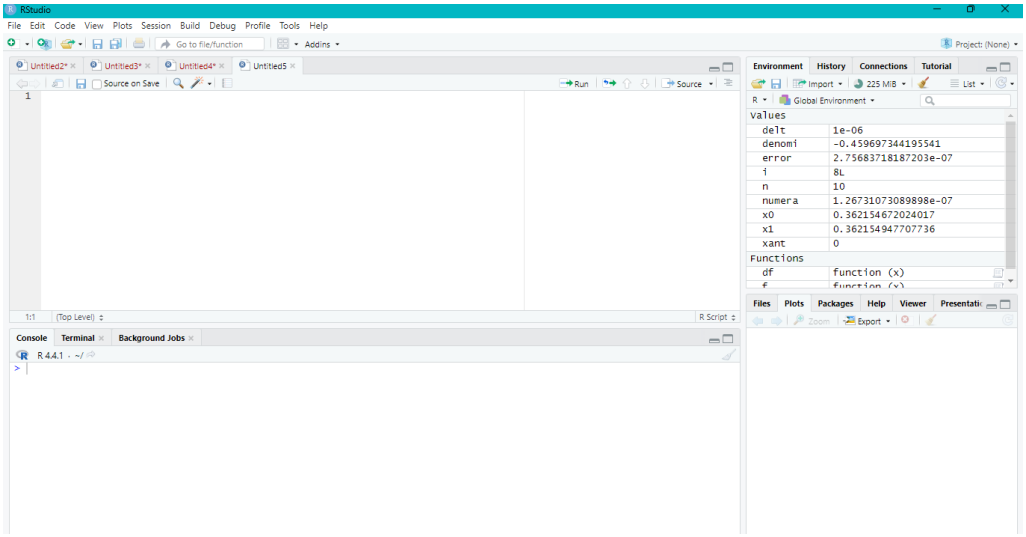
Desarrollo

Ecuación método Secante:

Tenemos la siguiente ecuación que está en nuestra actividad:

Ecuación 1: Mètodo SECANTE	$f(\theta) = \sin(\theta) + \cos(1 - \theta^2) - 1$

Para resolver la siguiente actividad nos dirigiremos a nuestro software de RStudio:



Colocamos el valor inicial de las diferentes variables, así como el error que vamos a permitir:

```
# Valor inicial
x0 = 1; xant = 0; error = 0;
```

El error:

```
#Error permitido
delt = 0.000001
```

Asignamos el número de iteraciones máximo que tendrá el comando:

```
#Numero de iteraciones
n = 10
```

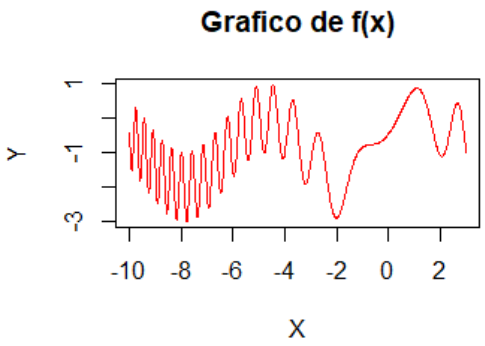
A continuación, colocaremos la ecuación en la sintaxis de RStudio vinculándolo al valor de una función:

```
#Función para encontrar la raíz
f = function(x) sin(x)+cos(1-x^2)-1
```

La Grafica es posible gracias a la siguiente sintaxis en el software, en donde podemos observar en grosor, el nombre, el color de la línea, las etiquetas, etc.:

```
plot(f,-10,3,
     lwd = 1,
     main = "Grafico de f(x)",
     col = "red",
     xlab = "X",
     ylab = "Y",
     axes = TRUE,
     n = 1000
)
```

Y la gráfica se nos muestra de la siguiente manera:



En el siguiente paso vamos a colocar los ciclos de las iteraciones junto con los resultados obtenidos, para ello empezaremos en orden, podemos observar el rango de iteraciones que nosotros queremos, así como la creación de varias variables para definirlas:

```
#Ciclo de iteraciones y resultados
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)
```

En este punto crearemos las diferentes medidas en las que convergería las iteraciones, junto con un comando de finalización del comando:

```
if(error<delt){
  cat("La solución converge en ", i, "Iteraciones. Raíz= ", x1)
  break()
}
x0= x1
```

Colocaremos la función “if” para mencionar que las iteraciones son iguales a “n” números para después imprimir un mensaje mencionando que el máximo número de iteraciones ha sido alcanzado:

```
 if(i==n){
   print("Máximo número de iteraciones alcanzada.")
 }
}
```

Si corremos el comando nos mostrara el siguiente resultado, nos muestra que el número total para llegar al error permitido es en la iteración numero 8:

```
[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. Raíz=  0.3621549
```

Mètodo Newton-Raphson:

La ecuación que se nos plantea es al siguiente:

Ecuación 2: Mètodo Newton-Raphson. $f(x) = 2x^3 - 8x^2 + 10x - 15$

Colocaremos un valor inicial:

```
# Valor inicial x0
x0 = 5
```

El error permitido o en valor de precisión, recordar que no es exacto, es solo la aproximación:

```
#Valor de precisión
delt = 0.000001
```

El número de iteraciones:

```
#Número de iteraciones
n = 15
```

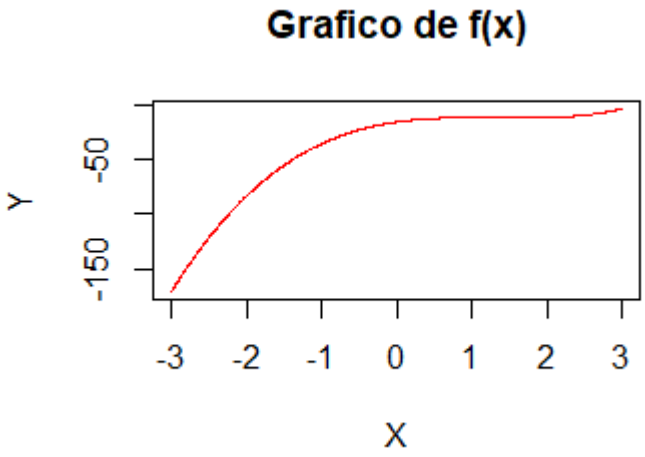
No hay que olvidarse de lo más importante que es la función trasladada a la sintaxis que nos pide el software:

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

Colocamos el comando para que nos muestre la gráfica de dicha función:

```
plot(f,-3,3,
     lwd = 1,
     main = "Grafico de f(x)",
     col = "red",
     xlab = "X",
     ylab = "Y",
     axes = TRUE,
     n = 1000
     )
```

La grafica quedaría de la siguiente manera



En este método es necesario sacar la derivada de la función para lograr seguir con la actividad que se nos pide:

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

Al igual que la actividad anterior vamos a colocar los ciclos de las iteraciones junto con sus resultados, en donde colocamos las funciones correspondientes para ejecutar correctamente la función, la impresión de dichas acciones, así como la finalización de la función:

```
#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, Raíz= ", x1);
    break()
  }
  x0= x1
```

Utilizamos la función “if” para definir que si una iteración es mayor a “n” numero nos imprimirá el mensaje de que se ha alcanzado el máximo número de iteraciones:

```
  if(i>n){
    print("Máximo Número de iteraciones alcanzado.");
    break()
  }
}
```

Si corremos la función por completo obtenemos el siguiente resultado en donde el error permitido se alcanza en la iteración número 6:

```
[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
```

Interpretación de Resultados:

Ecuación método de la Secante:

El resultado que obtuvimos se debe a que el error permitido que nosotros seleccionamos es de “6” decimales, es decir “0.000001” por ende comparar la conclusión obtenida en esta ecuación da por hecho lo siguiente “0.3621549” son los seis decimales que el nosotros forzamos a que se ejecutara.

```
[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. Raíz=  0.3621549
```

Ecuación método de Newton-Raphson:

El resultado de la siguiente ecuación es muy parecido a la anterior, el error permitido que nosotros seleccionamos es el mismo, es decir, “0.000001” para que este se detenga tiene que llegar a la iteración número “6” que en este caso seria “3.169038”, este es el punto en el que el programa se detendrá.

```
[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
```

Conclusión

En esta actividad aprendí que es necesario adaptarnos y aprovechar todas las herramientas que nos proporciona la tecnología actual. El documento demostró cómo se resolvían las actividades de acuerdo a los métodos necesarios, aprendimos a utilizar la sintaxis correcta para que el programa detectara todos los comandos de manera adecuada y lo ejecutara sin ningún problema, a simple vista puede resultar tedioso el aprender este tipo de cosas, ya que como sabemos no es lo mismo leer una información para resolver un problema que es realizar las instrucciones para realizar la solución correspondiente, y en mi caso fue la segunda, realizar la actividad por mano propia y experimentar en las diferentes resultados y maneras que se podía llegar a la conclusión , gracias a ello logre adentrarme más en este nuevo rubro que era la resolución de ecuaciones que uno podía hacerlas en el cuaderno pero que esta vez se realizaban en un software con los comandos correctos y de una manera más rápida e interesante.

Link para GitHub:

https://github.com/Leyzu-Ing/M-todos-Num-ricos.git

Referencias

Libreriaing. (2021, 19 noviembre). *¿Qué son y para qué sirven los métodos numéricos?* La Librería del Ingeniero. https://www.libreriaingeniero.com/2021/11/que-son-y-para-que-sirven-los-metodos-numericos.html