UNIDAD 3

MÉTODOS NUMÉRICOS

TEMA 1:
Raíces de una ecuación.

FR0018/ v1.0 / 18-11-2019

UNEMI ONLINE

ÍNDICE

1. Unidad 3: Raíces de Ecuaciones e Integración Numérica

Tema 1: Raíces de una ecuación

Objetivo:

Comprender la interpretación gráfica de una raíz, obtener la solución de ecuaciones no lineales aplicando métodos cerrados y abiertos, analizar la convergencia o divergencia de los métodos y calcular raíces de ecuaciones no lineales.

Introducción:

Resolver una ecuación, sea lineal o no lineal es común en la ingeniería a fin de obtener valores de la variable dependiente. Comúnmente estas ecuaciones se plantean para trabajos de diseño, con el propósito de predecir valores de los modelos matemáticos o funciones. Algunas de las aplicaciones de ingeniería son: Balance de masa y energía, balance de fuerzas, optimización y parámetros de circuitos eléctricos (Chapra & Canale, 2006).

Una raíz de una ecuación se define como el valor de x que hace que f(x)=0, si trabajamos con una ecuación lineal bastará despejar la variable independiente, normalmente denotada como x o t. Si el modelo es una ecuación cuadrática se recurre a la formula general para encontrar las raíces de la ecuación. Pero si se trata de funciones no lineales o que no son algebraicas como: exponenciales, trigonométricas o logarítmicas, despejar de forma analítica para encontrar la raíz puede ser complejo o en otros casos imposible. A continuación, se dan algunos ejemplos de funciones trascendentales en los que se necesita encontrar la raíz y no es sencillo despejar analíticamente la x.

$$f(x) = \ln\left(x^2 + 1\right) - 3$$
$$f(x) = e^{-0.4x} sen(4x - 0.25) \qquad (1)$$
$$f(x) = \ln(x^2 + 1) - e^{-\frac{x}{2}} \cos(\pi x)$$

Para resolver este tipo de problemas se utilizan los métodos gráficos y si se requiere calcular con precisión los valores de las raíces, se recurre a los métodos cerrados o abiertos.

Los métodos cerrados utilizan intervalos, estos intervalos encierran la raíz y se va reduciendo el tamaño del intervalo conforme avancen las iteraciones, los métodos cerrados que se describen en esta sección son: bisección y falsa posición.

Los métodos abiertos no necesitan un intervalo inicial, pero si de un valor inicial que este cerca de la raíz que se busca calcular, para obtener este punto inicial se utiliza el método gráfico. Algunos métodos abiertos son: Newton Raphson y Secante.

Los métodos cerrados requieren de más iteraciones, pero garantizan la convergencia de encontrar la raíz. Mientras que en ciertos casos los métodos abiertos convergen más rápido a la solución, pero si la ecuación presenta una pendiente muy alta y en estos casos no converge el método.

2. Información de los subtemas

2.1 Subtema 1: Método Gráfico

Un método sencillo y muy utilizado para hallar raíces de una ecuación de forma aproximada es el método gráfico, así se observa los puntos de corte con el eje x que son las rices o solución de la ecuación. Así se obtiene una aproximación inicial de la raíz o a su vez se establece el intervalo inicial para aplicar ya sea métodos cerrados o abiertos.

Ejercicio 1

Indique el número de raíces que tiene la función en el intervalo de [-2, 2]:

$$f(x) = e^{-0.2x} sen(2x - 0.25)$$

Solución

Código en Python para graficar la función

```
import matplotlib.pyplot as plt
import numpy as np
```

```
x=np.linspace(-2,2,100)
y=np.exp(-0.2*x)*np.sin(2*x-0.25)
plt.plot(x,y)
plt.title('Método gráfico')
plt.legend(['exp(-0.2x)sen(2x-0.25)'])
plt.grid()
plt.show()
```

Figura 1: Código en Python para graficar una función
Fuente: Elaboración propia

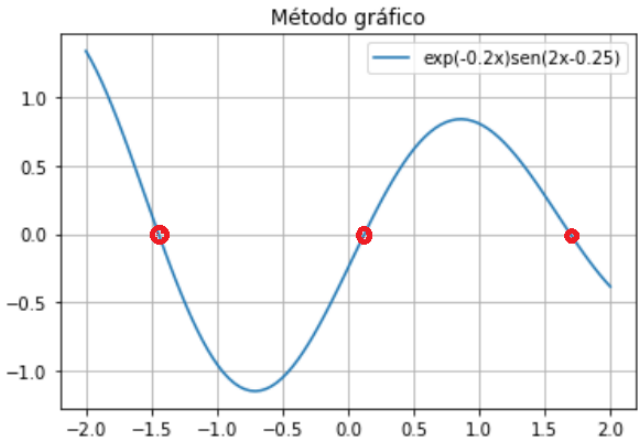


Figura 2: Gráfica de la ecuación en Python
Fuente: Elaboración propia

Con la gráfica determinamos que en el intervalo de [-2, 2] la función tiene 3 raíces y se puede aproximar los valores. Si se requiere mayor precisión en los valores por método gráfico, se pueden utilizar otros programas como por ejemplo Matlab, allí podemos poner el cursor sobre el grafico y obtener la coordenada (x,y).

Ejercicio 2

Obtenga la raíz más pequeña de f(x) en el intervalo de [0, 5]

$$f(x) = \ln(x^2+1) - e^{-\frac{x}{2}}\cos(\pi x)$$

Código en Matlab

```
fplot(@(x) log(x.^2+1)-exp(x./2).*cos(pi*x),[0 4],'b')
title('Método Gráfico')
legend('log(x.^2+1)-exp(x./2).*cos(pi*x)')
grid on
```

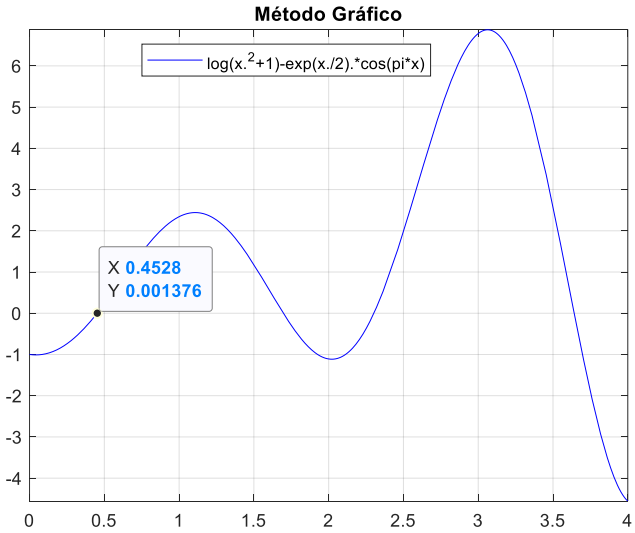


Figura 3: Gráfica de la ecuación ejercicio 2
Fuente: Elaboración propia

La raíz más pequeña de f(x) en el intervalo [0, 4] es de 0.4528 aproximadamente.

Ejercicio 3

Obtenga la raíz más grande de f(x) en el intervalo de [0, 5]

$$f(x) = sen(2x) + \cos(3x)$$

Código en Matlab

```
x=linspace(0,5,100)
y=sin(2*x)+cos(3*x)
plot(x,y,'r','LineWidth',1.5)
legend('sin(2*x)+cos(3*x)')
title('Método Gráfico')
grid on
```

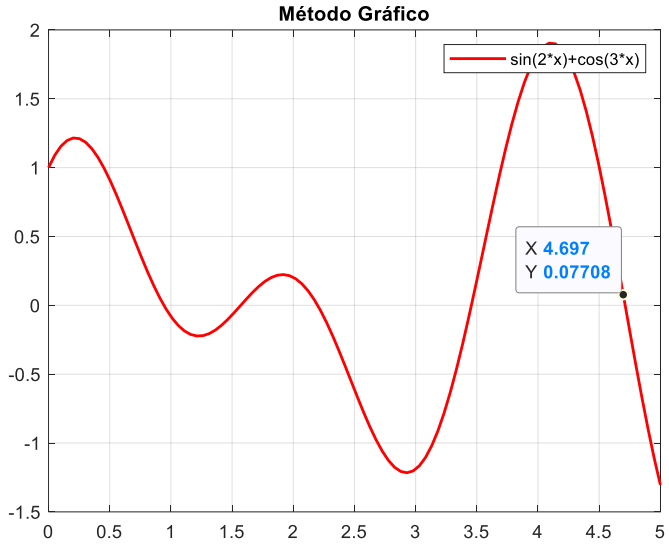


Figura 4: Gráfica de la ecuación ejercicio 3

Fuente: Elaboración propia

La raíz más grande de f(x) en el intervalo [0, 5] es de 4.697 aproximadamente.

Los métodos abiertos que se analizarán son bisección y un método que reduce el número de iteraciones cuando la solución converge llamado falsa posición, estos métodos permitirán determinar con mayor precisión las raíces de una ecuación comparado con el método gráfico.

2.2 Subtema 2: Método de la Bisección

Para aplicar el método de la bisección f(x) tiene que ser continua en el intervalo [a, b]. Para determinar el intervalo se evalúa la función en [f(a), f(b)], un valor debe ser positivo y el otro negativo, esto garantiza encontrar una raíz x*=r que hace f(r)=0 en el intervalo [a, b] (Sauer, 2012).

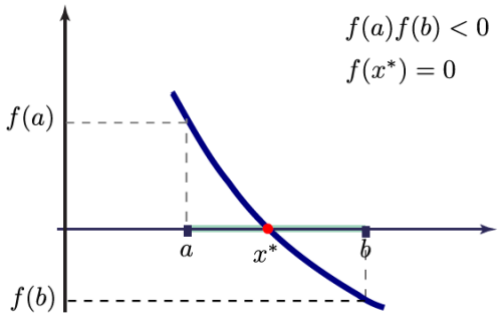


Figura 5: Grafica de una función f(x) que tiene una raíz en [a, b]
Fuente: https://bit.ly/3n1npQw

En cada iteración el intervalo va cambiando, haciéndose más pequeño hasta llegar a la raíz, con el criterio de parada del error de la estimación.

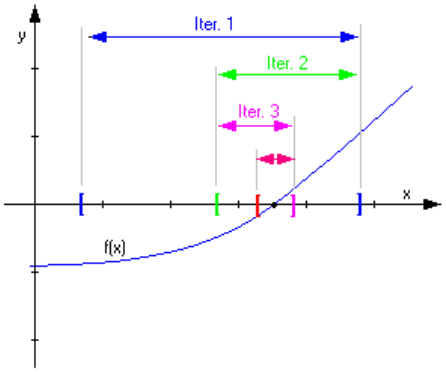


Figura 6: Cambio de intervalo por cada iteración en método de bisección
Fuente: https://bit.ly/38kMBgQ

Algoritmo de Bisección

Sea f(x) una función continua en el intervalo [a, b] y que f(a) y f(b) tienen signos diferentes, entonces existe un valor r que hace f(r)=0.

El método de la bisección consiste en dividir por la mitad el intervalo por cada iteración dependiendo de la ubicación de la raíz, el punto medio se calcula con r=(a+b)/2 y se sustituye por [r, b] o [a,r] para la primera iteración y así sucesivamente hasta obtener el valor de f(r)=<error. Como el intervalo va reduciéndose a la mitad en cada iteración, al final la distancia será muy pequeña y el valor calculado estará cerca de r (Nieves & Dominguez, 2014).

Para aplicar bisección siga los siguientes pasos:

a_i=a y b_i=b

Paso 1: Calcule r =(a_i+b_i)/2.

Paso 2: Calcule f(r) y si f(r)=<error, entonces DETÉNGASE.

Paso 3:

1. Si f(a_i)*f(r)<0, entonces haga a_{i+1}=a_i y b_{n+1} =r.
2. Si f(a_i)*f(r)>0, entonces haga a_{i+1}= r y b_{i+1} = b_i.

Repita los tres pasos hasta alcanzar el error deseado.

El error puede calcularse bajo dos criterios

1. El error de la raíz, para ello f(x*) <= error permitido
2. mediante el error relativo por cada iteración con:

$$\left|\frac{r_{i+1}-r_i}{r_i}\right| \leq e \qquad (2)$$

Ejercicio 4

La ecuación de una ola estacionaria reflejada en un puerto está dada por:

$$h = h_0\left[sen\left(\frac{2\pi x}{\lambda}\right)cos\left(\frac{2\pi tv}{\lambda}\right)+e^{-x}\right] \qquad (3)$$

Los parámetros de la ecuación de la ola son: la longitud de onda, las alturas respectivas, la velocidad y el tiempo.

Para λ =16 (longitud de onda), t=12 s (tiempo); v=48 m/s(velocidad); h=0.4 h_0 (altura), Determine el valor de x (distancia donde rompe la ola), con error de 10^{-6} cuando la altura es del 40% del valor inicial. Emplee el método de la bisección con el intervalo [5,8]

Solución en Python

```
def biseccion(f,a,b,e):
    while b-a>=e:
        r=(a+b)/2
        if abs(f(r))<=e:
            return r
        else:
            if f(a)*f(r)>0:
                a=r
            else:
                b=r
    return r
```

```
def f(x):
    lam=16
    t=12
    v=48
    return ((np.sin((2*np.pi*x)/lam)*np.cos((2*np.pi*t*v)/lam))+np.exp(-x))-0.4
```

```
x=biseccion(f,5,7,10**-6)
f(x)
```

```
1.0915178483283938e-07
```

```
x
```

```
6.954730987548828
```

Figura 7: Código en Python Método de Bisección
Fuente: Elaboración propia

El valor de la raíz de la función es x=6.95473

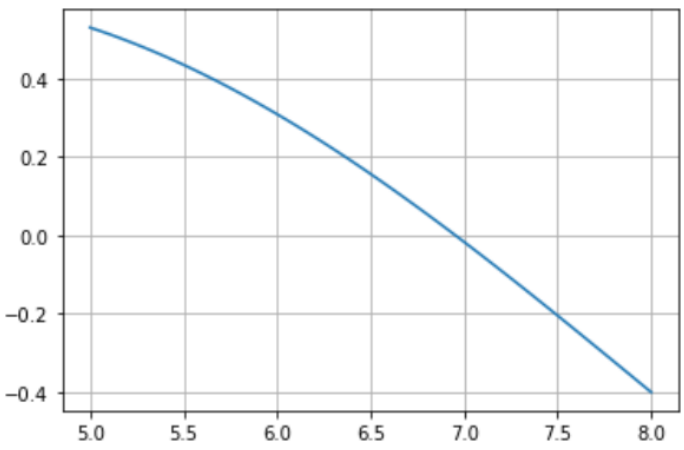


Figura 8: Grafica de la función ejercicio 4.
Fuente: Elaboración propia

Ejercicio 5

La ecuación para un canal de agua en forma de semicírculo con radio r y longitud L, está dado por:

$$V = L\left[0.5\pi r^2 - r^2\, arcsin(h/r) - h\left(r^2 - h^2\right)^{1/2}\right] \quad (4)$$

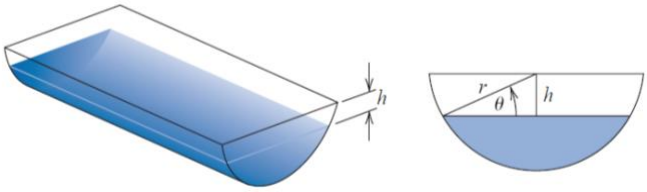


Figura 9: Canal de agua en forma de semicírculo.
Fuente: (Burden & Faires, 2011)

Suponga que longitud = 10 pies, radio = 1 pie y volumen = 10.4pie^3. Calcule la profundidad del agua (h) mediante el método de la bisección. Considere el intervalo [0, 1] y error de estimación de 10^{-6}
Fuente: (Burden & Faires, 2011).

```
def f(h):
    L=10
    r=1
    V=10.4
    return V-L*(0.5*np.pi*r**2-r**2*np.arcsin(h/r)-h*(r**2-h**2)**0.5)
```

```
x=biseccion(f,0,1,10**-6)
f(x)
```

```
9.731812351532199e-07
```

```
x
```

```
0.26866626739501953
```

```
n
```

```
20
```

Figura 10: Solución en Python ejercicio 5
Fuente: Elaboración propia

La profundidad del agua es de 0.2686 pies, el error de la estimación es de $9.73*10^{-7}$ y se requieren 20 iteraciones.

Convergencia del método de la Bisección

Suponga que $f \epsilon\ C[a,b]\ y\ f(a) * f(b) < 0$. El método de la Bisección aproxima f(r) a cero con una sucesión:

$$|r_n - r| \leq \frac{b-a}{2^n} \qquad (5)$$

Se puede determinar el número de iteraciones (n) para alcanzar un mínimo error permitido en la estimación despejando n de la ecuación 5

$$n > \frac{\log((b-a)/error)}{\log(2)} \qquad (6)$$

La razón de convergencia del método de Newton es:

$$r_n = r + O\left(\frac{1}{2^n}\right) \qquad (7)$$

Método Falsa posición

En el método de la bisección por cada iteración se divide por la mitad el intervalo, sin considerar las magnitudes de f(a) y f(b), por ejemplo, si f(a) está más próximo a cero que f(b), entonces la raíz r estará más cerca de a. El método de la falsa posición une mediante una recta los dos puntos y el cruce por cero (c) va a converger con un menor número de iteraciones que el método de la bisección.

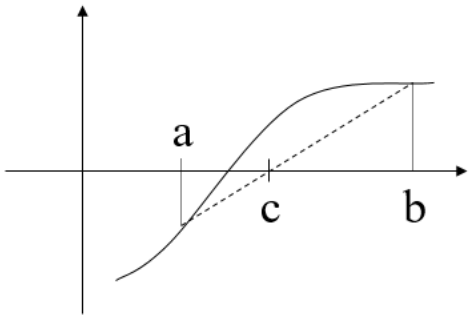


Figura 11: Método de la falsa posición
Fuente: Elaboración propia

La implementación del algoritmo es similar a bisección, la diferencia es en la forma en cómo se divide el intervalo, que está dado por una recta que une los puntos f(a) y f(b), el punto de corte es el nuevo valor de r. Para actualizar valores de [c,b] o [a, c] se utiliza la siguiente formula:

$$r = a - f(a)\frac{b-a}{f(b)-f(a)} \qquad (8)$$

Ejercicio 6

Resuelva el ejercicio 5 ahora por falsa posición y compare la cantidad de iteraciones con respecto al método bisección.

```
def falsapos(f,a,b,e):
    n=0
    while b-a>=e:
        r=a-f(a)*((b-a)/(f(b)-f(a)))
        n=n+1
        if abs(f(r))<=e:
            return r,n
        else:
            if f(a)*f(r)>0:
                a=r
            else:
                b=r
    return r,n
```

```
def f(h):
    L=10
    r=1
    V=10.4
    return V-L*(0.5*np.pi*r**2-r**2*np.arcsin(h/r)-h*(r**2-h**2)**0.5)
```

```
x, n=falsapos(f,0,1,10**-6)
f(x)
```

```
5.900919983758968e-07
```

```
x
```

```
0.26866624750943063
```

```
n
```

```
5
```

Figura 12: Solución en Python ejercicio 6
Fuente: Elaboración propia

En falsa posición para calcular la raíz con un error menor a 10^{-6} se necesitaron 5 iteraciones, mientras que para el método de bisección se requirieron de 20 iteraciones. Este algoritmo es más eficiente y reduce el cálculo computacional, siempre que la solución converja.

Ejercicio 7

Utilice el método de la falsa posición para determinar la raíz de:

$$f(x) = e^{6x} + 1.441e^{2x} - 2.079e^{4x} - 0.3330 \qquad (9)$$

Estime con error de 10^{-7} en el intervalo de [-1,0]

Fuente: (Sauer, 2012)

```
def f(x):
    return np.exp(6*x)+1.441*np.exp(2*x)-2.079*np.exp(4*x)-0.333
```

```
x, n=falsapos(f,-1,0,10**-7)
f(x)
```

```
9.90753460938798e-08
```

```
x
```

```
-0.1695560909870012
```

```
n
```

```
548
```

Figura 13: Solución en Python ejercicio 6
Fuente: Elaboración propia

La raíz para la función f(x) en el intervalo de [-1, 0] es de -0.169556 con error de $9.9*10^{-8}$

En este ejercicio se requirieron de 548 iteraciones, notemos como es la gráfica.

```
x=np.linspace(-1,0,100)
y=np.exp(6*x)+1.441*np.exp(2*x)-2.079*np.exp(4*x)-0.333
plt.plot(x,y)
plt.grid()
plt.show()
```

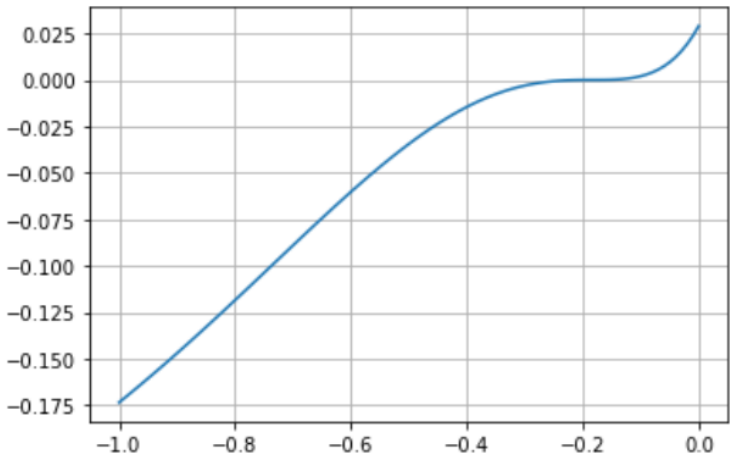


Figura 14: Código y grafica del ejercicio 7.
Fuente: Elaboración propia.

2.3 Subtema 3: Método de Newton Raphson

Los métodos descritos anteriormente son cerrados, debido a que la raíz se encuentre entre los limites inferior y superior del intervalo inicial. Los métodos abiertos que se describirán requieren de un solo valor de inicio para el método de Newton y dos puntos para el de la secante, estos valores no necesariamente encierran la raíz. Estos métodos puede que no converjan a la raíz, sobre todo si la pendiente de la función es muy vertical, pero para casos de convergencia requieren de menos iteraciones que los métodos cerrados.

El método de Newton o llamado Newton-Raphson, se utiliza para resolver ecuaciones de la forma f(x)=0, donde la función f(x) es continua y derivable. Este método es un caso particular del método del punto fijo, donde se pasa la ecuación f(x)=0 a la forma x=g(x), (Nieves & Dominguez, 2014).

Para encontrar la raíz se parte de un valor inicial x_0, se traza la recta tangente a f, el corte con el eje x dará el valor de x_1, que será utilizado en la siguiente iteración. Note en la figura 9 que, por cada valor de x calculado, se traza la recta tangente a f hasta encontrar la raíz.

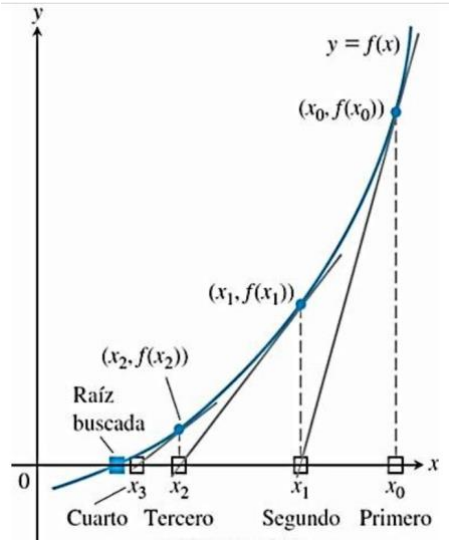


Figura 15: Visualización del método de Newton
Fuente: https://bit.ly/2JPfS9d

La fórmula de Newton Raphson se obtiene a partir de la serie de Taylor, enunciando lo siguiente:

Si $x_0 \epsilon$ [a,b] es una aproximación para x tal que $f'(x_0) \neq 0$ y $|x - x_0|$ es pequeño. El polinomio de Taylor expandido alrededor de x_0 y evaluado en x: (Burden & Faires, 2011)

$$f(x) = f(x_0) + (x - x_0) f'(x_0) + \frac{(x - x_0)^2}{2} f''(\xi(x)) \qquad (10)$$

El valor f(x)=0, y como $|x - x_0|$ es pequeño, el término $(x-x_0)^2$ es mucho más pequeño, se obtiene la aproximación:

$$0 \approx f(x_0) + (x - x_0) f'(x_0) \qquad (11)$$

Al despejar x de la ecuación 8 se tiene la ecuación de Newton Raphson

$$x = x_0 - \frac{f(x_0)}{f'(x_0)} \qquad (12)$$

En forma iterativa sería

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)} \quad \text{para } i = 0,1,2... \qquad (13)$$

La ecuación 1.2 puede ser generalizada como se aprecia en la expresión 1.3

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)} \qquad (14)$$

En resumen, el algoritmo funciona de la siguiente manera: (Lara, 2019)

1. Elegir un punto inicial x_0
2. Para i=0,1,… calcular x_{i+1} usando la ecuación 11 hasta que el error sea lo suficientemente pequeño.
3. El proceso iterativo puede terminar cuando el error sea lo suficientemente pequeño usando la ecuación 12

$$\left| \frac{x_{i+1} - x_i}{x_i} \right| \leq e \qquad (15)$$

Ejercicio 8

Aplique Newton Raphson para obtener la raíz de f(x) con x_0=36 y error de 10^{-6}

$$f(x) = x^3 \cos(x) - 5x^2 - 1$$

Solución en Python

```
from sympy import*
def newton(f,v,u,e,m):
    g=v-f/diff(f,v)
    r=u
    n=1
    for i in range(m):
        r=float(g.subs(v,u))
        if abs(r-u)<=e:
            return r,n
        u=r
        n=n+1
    return None
```

```
x=Symbol('x')
f=x**3*cos(x)-5*x**2-1
```

```
r,n=newton(f,x,36,10**-3,10)
r
```

```
36.26664494452215
```

```
e=float(f.subs(x,r))
e
```

```
8.549250196665525e-11
```

```
n
```

```
3
```

Figura 16: Código en Python para Newton Raphson
Fuente: Elaboración propia.

La raíz que se obtuvo como punto de partida x_0= 36 es r=36.266 con error de $8.849*10^{-11}$, para encontrar la raíz se necesitaron 3 iteraciones.
Si se grafica la función en el intervalo x de [0, 40] se observa que tiene varias raíces, pero el método de Newton solo da una raíz, la que esté más cercana al valor de x_0

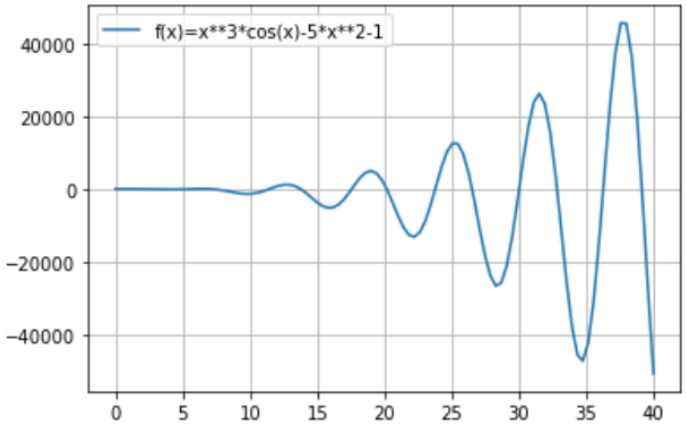
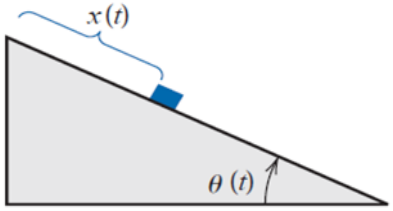


Figura 17: Grafica de la función f(x) ejercicio 8
Fuente: Elaboración propia

Ejercicio 9

Una partícula comienza en reposo en un plano inclinado suave cuyo ángulo cambia a una tasa constante.

Fuente: (Burden & Faires, 2011).



$$\frac{d\theta}{dt} = \omega < 0.$$

Al final t de los segundos, la posición del objeto viene dada por:

$$x(t) = \frac{g}{2\omega^2}\left(\frac{e^{wt} - e^{-wt}}{2} - \sin \omega t\right)$$

Suponga que la partícula se ha movido 1.7 ft (0.52m) en 0.5 s. Encuentre, dentro de 10^{-5}, la velocidad ω a la que cambia θ. Suponga que g= 9.8m/s2 y w_0=1 rad/s.

Solución en Python

```
x=Symbol('x')
t=0.5
g=9.8
k=0.52
f=-k+((g/(2*x**2))*(((exp(x*t)-exp(-x*t))/2)-sin(x*t)))
```

```
r,n=newton(f,x,1,10**-8,10)
r
```

```
2.5390840938472587
```

```
n
```

```
4
```

```
e=float(f.subs(x,r))
e
```

```
1.1102230246251565e-16
```

Figura 18: Solución en Python ejercicio 9
Fuente: Elaboración propia.

La velocidad angular de la partícula (w) cuando se ha movido 0.52 m es de 2.539 rad/seg con error de $1.11*10^{-16}$

Convergencia cuadrática del Método de Newton

El error de paso es $e_i = |x_i - r|$, la definición de convergencia cuadrática es: (Sauer, 2012)

$$\lim_{i\to\infty} \frac{e_{i+1}}{e_i^2} = \left|\frac{f''(r)}{2f'(r)}\right| \qquad (16)$$

La ecuación 16 puede ser escrita como:

$$e_{i+1} \approx Me_i^2 \qquad (17)$$

Esto siempre que $f'(r) \neq 0$, siendo $M = |f''(r)/(2f'(r)|$. La aproximación es mejor a medida que hay convergencia del método. La convergencia cuadrática aplica cuando se calcula una raíz simple.

Convergencia lineal del Método de Newton

Aplica cuando se calcula una raíz múltiple, esta convergencia se determina mediante la expresión: (Sauer, 2012)

$$e_{i+1} = Se_i \quad \text{donde} \quad S = (m-1)/m \qquad (18)$$

Siendo m la multiplicidad de la raíz, misma que debe ser mayor a 1 para aplicar la ecuación 18.

Para calcular el número de iteraciones del algoritmo de Newton, aplicamos la ecuación 19.

$$\left(\frac{m-1}{m}\right)^n < error \qquad (19)$$

Al despejar n:

$$n > \frac{\log(error)}{\log((m-1)/m)} \qquad (20)$$

Método de la Secante

Para este método se reemplaza la derivada por una ecuación de diferenciación numérica, que representa la recta tangente entre los puntos X_i y x_{i-1}. La intersección de la recta con el eje x, es el nuevo valor estimado x_{i+1} de forma iterativa hasta que x converge a la raíz.

$$f'(x_i) = \frac{f(x_i) - f(x_{i-1})}{x_i - x_{i-1}} \qquad (21)$$

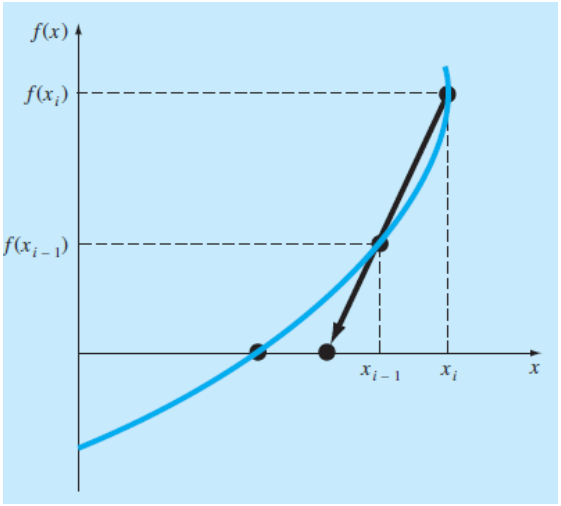


Figura 19: Representación gráfica del método de la Secante
Fuente: (Chapra & Canale, 2006)

La ecuación de secante se obtiene al reemplazar la fórmula de diferenciación numérica en la fórmula de Newton. Para aplicar esta fórmula inicialmente se requieren dos valores (Chapra & Canale, 2006).

$$x_{i+1} = x_i - \frac{f(x_i)(x_i - x_{i-1})}{f(x_i) - f(x_{i-1})} \qquad i = 1, 2, \ldots \qquad (22)$$

Una ventaja de implementar este método similar el de Newton, es que no se requiere calcular la derivada de f(x), que en algunos casos puede resultar compleja si se realiza de forma analítica.

Convergencia del método de la Secante

Suponiendo que el método de la Secante converge a r y $f'(r) \neq 0$, se cumple al aproximar el error (Sauer, 2012).

$$e_{i+1} \approx \left| \frac{f''(r)}{2f'(r)} \right| e_i e_{i-1} \qquad (23)$$

Ejercicio 10

Calcule la raíz de: (Chapra & Canale, 2006)

$$f(x) = 2x^3 - 11.7x^2 + 17.7x - 5$$

Aplicando método secante, x_0= 3, x_1 =4, el error deber ser menor a 10^{-7}

Solución en Python

```
def secante(f,x0,x1,e,m):
    xa=x0
    xb=x1
    n=1
    for i in range(m):
        x=xb-((f(xb)*(xb-xa))/(f(xb)-f(xa)))
        if abs(f(x))<e:
            return x,n
        xa=xb
        xb=x
        n=n+1
    return None
```

```
def f(x):
    return 2*x**3-11.7*x**2+17.7*x-5
```

```
x,n=secante(f,3,4,10**-7,10)
x
```

```
3.563160824859791
```

```
n
```

```
7
```

```
e=abs(f(x))
e
```

```
2.375344365646015e-11
```

Figura 20: Código en Python para método de la Secante
Fuente: Elaboración propia.

La raíz de la ecuación es 3.5631 con error de $2.375*10^{-11}$, para encontrar la raíz se necesitaron 7 iteraciones.

Ejercicio 11

Obtenga la raíz más pequeña de f(x) aplicando el método de la secante con $x_0=0$ y $x_1=1$, obtenga un error menor a 10^{-7}

$$f(x) = \ln(x^2+1) - e^{-\frac{x}{2}}\cos(\pi x)$$

```
import numpy as np
def f(x):
    return np.log(x**2+1)-np.exp(-x/2)*np.cos(np.pi*x)
```

```
x,n=secante(f,0,1,10**-7,10)
x
```

```
0.43191157151350673
```

```
n
```

```
4
```

```
e=abs(f(x))
e
```

```
3.93373389417917e-11
```

Figura 21: Solución por Secante ejercicio 11
Fuente: Elaboración propia.

La raíz más pequeña de f(x) es 0.431911 con error de $3.9337*10^{-11}$, se necesitaron 4 iteraciones.

3. Preguntas de Comprensión de la Unidad

¿En qué casos se recomienda utilizar el método gráfico para determinar la raíz de una ecuación?

El método gráfico se utiliza para casos donde no se requiere de alta precisión, también sirve para determinar intervalos iniciales y puntos para trabajar con los métodos abiertos o cerrados.

¿En qué consiste la diferencia de los métodos abiertos y cerrados?

Una diferencia es que los métodos cerrados requieren de un intervalo inicial que encierre la raíz y que haga f(a)*f(b)<0, mientras que los métodos abiertos requieren de un punto o dos que no necesariamente encierran la raíz.

¿Qué se pude decir de la convergencia de los métodos abiertos y cerrados?

En el método de bisección la convergencia es $O(1/2^n)$, una convergencia lenta que requiere de un mayor número de iteraciones, pero garantiza hallar una raíz sin importar la tasa de crecimiento o decrecimiento de la función. Los métodos abiertos convergen más rápido, por ejemplo, para una raíz de multiplicidad 1 la convergencia de Newton Raphson es cuadrática $e_{i+1} \approx Me_i^2$, pero el inconveniente es que en ocasiones diverge cuando se trabaja con funciones con pendientes pronunciadas verticalmente u horizontalmente.

¿Indique el número de iteraciones que se requiere para el método de Bisección para hallar la raíz de una ecuación en el intervalo [5,8] con error menor a 10^{-6}?

Utilizando la ecuación $n > \frac{\log((b-a)/error)}{\log(2)}$ se puede calcular el número de iteraciones aplicando bisección.

$$n > \frac{\log((8-5)/10^{-6})}{\log(2)}$$

$$n > 21.52$$

Se requieren al menos 22 iteraciones para alcanzar el error admisible, note que para el cálculo de n no se toma en cuenta la función f(x).

¿Cuál es la condición de stop de los algoritmos para encontrar raíces de una ecuación?

El criterio de stop de estos algoritmos es el error, hay varios tipos de error que se pueden calcular por ejemplo el error absoluto de las iteraciones:

Error absoluto: $|r_{i+1} - r_i| \leq e$

Error relativo: $\left|\frac{r_{i+1} - r_i}{r_i}\right| \leq e$

Error de la función evaluada en la raíz: $abs(f(r_i)) \leq e$

Para los ejemplos realizados en esta sección se ha utilizado el error de la función evaluada en r_i, el valor de $f(r_i)$ idealmente debe ser cero, se utiliza valor absoluto $abs(f(r_i)) \leq e$ porque al evaluar la función en r, puede tomar valores negativos y detener el algoritmo sin alcanzar la precisión deseada.

4. Material Complementario

Los siguientes recursos complementarios son sugerencias para que se pueda ampliar la información sobre el tema trabajado, como parte de su proceso de aprendizaje autónomo:

Videos de apoyo:

Método Gráfico para determinar la raíz: https://bit.ly/38n9IHr

Método de la Bisección: https://bit.ly/36bDKLI

https://bit.ly/357BaqE

Método de la Falsa Posición: https://bit.ly/2IlaGZN

Método de Newton Raphson: https://bit.ly/38mFynO

Método de la Secante: https://bit.ly/3p5YwoG

https://bit.ly/32o8rMF

Bibliografía de apoyo:

- Burden, R., & Faires, D. (2011). *Numerical Analysis*. (BOOKS/COLE, Ed.) (Ninth). Boston.
- Chapra, S., & Canale, R. (2006). *Métodos numéricos para ingenieros*. (McGrawHill, Ed.) (Quinta). México

Links de apoyo:

Link de libros: https://bit.ly/34WB0Td

5. Bibliografía

- Burden, R., & Faires, D. (2011). *Numerical Analysis*. (BOOKS/COLE, Ed.) (Ninth). Boston.
- Chapra, S., & Canale, R. (2006). *Métodos numéricos para ingenieros*. (McGrawHill, Ed.) (Quinta). México.
- Lara, O. (2019). Métodos Numéricos: Teoría y Aplicaciones. *2018*.
- Nieves, A., & Dominguez, F. (2014). *Métodos Numéricos aplicados a la ingeniería*. (Patria, Ed.) (Primera). México.
- Sauer, T. (2012). *Análisis Numérico*. (PEARSON, Ed.) (Segunda). México. Retrieved from http://librosysolucionarios.net/