LAUREATE INTERNATIONAL UNIVERSITIES

Proyecto integrador etapa 2

ACTIVIDAD 2

MÉTODOS NUMÉRICOS

DOCENTE Dr. EDSON MICHAEL JIMENEZ DAMIAN

ELABORADO POR:

- CESAR AUGUSTO MARIZ PLASCENCIA.
- CARLOS AGUSTIN CARRILLO FLORES.
- JUSTINO DANIEL RODRÍGUEZ MURRAY.
- JOSE EMILIANO JAUREGUI GUZMAN.
- NAYELLI BERNAL CORTES.
- ABIGAIL ORTIZ LAGOS

06-Octubre-2024

INTRODUCCIÓN

Esta actividad consiste en aplicar los conocimientos adquiridos a lo largo del curso. Para llevar a cabo este Proyecto se toman como referente actividades elaboradas previamente, lo que garantiza la transversalidad de los contenidos revisados para fortalecer el desarrollo de competencias.

OBJETIVO

El objetivo del Proyecto integrador es programar los principales métodos numéricos para la solución de sistemas de ecuaciones lineales de una variable, así como de derivación e integración identificando las ventajas y desventajas de cada uno que permitan determinar soluciones viables mediante el planteamiento de modelos matemáticos exactos y precisos.

Etapa 1 del Proyecto integrador + Etapa 2 del Proyecto integrador

II. Programación de los métodos de Jacobi y Gauss-Seidel en una aplicación

2.1 Conceptualización

2.2 Casos prácticos

PROYECTO INTEGRADOR
ETAPA 1

Objetivo

El objetivo del Proyecto integrador es programar los principales métodos numéricos para la solución de sistemas de ecuaciones lineales de una variable, así como de derivación e integración identificando las ventajas y desventajas de cada uno que permitan determinar soluciones viables mediante el planteamiento de modelos matemáticos exactos y precisos.

I.Programación de los métodos de bisección y Newton Raphson en una aplicación

1.1 Conceptualización

Reproduce y completa el siguiente cuadro comparativo en el que describas los elementos sustantivos de cada método numérico.

Método numérico	¿En qué consiste?	Ventajas	Desventajas
Bisección	Es un algoritmo de búsqueda de raíces que trabaja dividiendo el intervalo a la mitad y seleccionando el subintervalo que tiene la raíz.	• Funciona para ecuaciones algebraicas y trascendentes, pero se recomienda utilizarlo después	• Converge muy lentamente. • Permite encontrar solo una raíz, aunque existan más en el intervalo.

		de un análisis gráfico. • Es siempre convergente. • Se puede establecer el límite de error	• No puede determinar raíces complejas.
Regla falsa	Método iterativo de resolución numérica de ecuaciones no lineales. El método combina el método de bisección y el método de la secante.	• Se aproxima más rápido a la raíz. • Tiene similitud con el método de secante. • Alternativa basada en una visualización gráfica. • Se estima mediante la semejanza de triángulos.	• Convergencia muy lenta a la solución. • Proceso iterativo, que ocasiona que uno de sus extremos tiende a no modificarse. • No se puede prever el número de iteraciones necesarias.
Sustitución sucesiva	El método de sustitución consiste en despejar una incógnita de una de las ecuaciones y sustituir en la otra ecuación el valor hallado.	• Facilidad para programarlo • Para ciertos tipos de problemas que aparecen en Ingeniería Química este método es muy adecuado.	• No converge en muchos casos • En otros la convergencia es muy lenta.
Newton – Raphson	Se utiliza para calcular los ceros	• Puede converger a la raíz de una	•No converger si la estimación inicial.

	de una función real de variable real. Su simplicidad formal y su rapidez de convergencia hacen que, con frecuencia, sea el primer algoritmo a considerar para esta tarea.	función rápidamente. • Maneja funciones de cualquier complejidad. • Utiliza la derivada de la función para aproximar la raíz, lo que puede proporcionar una estimación más precisa de la raíz en comparación con otros métodos que no utilizan derivadas.	•Requiere el cálculo de derivadas, lo que puede resultar difícil para algunas funciones.
Secante	Es un algoritmo que busca la raíz de una función. Podría considerarse como una variante del método Newton-Raphson por todas las similitudes que tienen, ambos son métodos abiertos, funcionan a partir	Se puede obtener cuando la ecuación es demasiado compleja para obtener una derivada.	La velocidad de convergencia es mas lenta que la de Newton-Raphson. No se asegura que la primera aproximación es cercana a la raíz que pudiera ser indice de divergencia.

	de una fórmula que proviene de la pendiente de una gráfica		

1.2 Métodos abiertos

• Describe brevemente en qué consisten los métodos numéricos abiertos y establece cuáles métodos de los que aparecen en el cuadro comparativo están representados por esta categoría.

Se basan en fórmulas que requieren de un sólo valor de Xo de un par de ellas, pero que no necesariamente encierran la raíz. En algunas ocasiones divergen o se alejan de la raíz a medida que crece el número de interacciones. Sin embargo, cuando los métodos convergen, lo hacen mucho más rápido que los métodos que usan intervalos, algunos métodos son:

•Método de Newton-Raphson.

•Método de la Secante.

•Investiga algunas aplicaciones o problemas que se utilizan aplicando este tipo de métodos.

Método de Newton-Raphson. El funcionamiento se basa en predecir un nuevo valor de x en función del valor anterior de x, de esta manera, dado un nuevo valor para xi se tiene:

$$x_{\mathrm{i"1}} = x_{\mathrm{i}} - \frac{f(x_{\mathrm{i}})}{f'(x_{\mathrm{i}})}$$

La condición de paro se base en:

$$\varepsilon_a = ,\frac{x_{\mathrm{i"1}} - x_{\mathrm{i}}}{x_{\mathrm{i"1}}}, 100\%$$

1.3 Métodos cerrados

•**Describe brevemente en qué consisten los métodos numéricos cerrados y establece cuáles métodos de los que aparecen en el cuadro comparativo estánrepresentados por esta categoría**

Se se necesita de dos valores iniciales para la raíz. Como su nombre lo indica, dichos valores iniciales deben “encerrar”, o estar a ambos lados de la raíz, algunos métodos son:
Método de Bisección.
Método de la Falsa Posición o Regla falsa.

- **Investiga algunas aplicaciones o problemas que se utilizan aplicando este tipo de métodos**

A partir de una función algebraica o trascendente y de un intervalo [a, b] que pertenece al dominio de la función y para el cual $f(a) \cdot f(b) < 0$, lo que implica que en el intervalo [a, b] existe al menos una raíz. El método consiste en bisectar el intervalo [a, b]:

$$x_{\%} = \frac{a+b}{2}$$

Se obtiene una aproximación a la raíz $x_{\%}$; la función se valúa en este nuevo valor y de acuerdo al signo de la función valuada en este punto, debería sustituirse uno de los extremos del intervalo de búsqueda, de tal forma que se conserve que $f(a) \cdot f(b) < 0$. De acuerdo a la geometría de la figura, la sustitución de los intervalos debería hacerse de la siguiente forma: Sea a tal que $f(a) < 0$ y b tal que $f(b) > 0$:

Si $f(x_{\%}) < 0$, entonces $x_{\%}$ sustituye a
Si $f(x_{\%}) > 0$, entonces $x_{\%}$ sustituye b

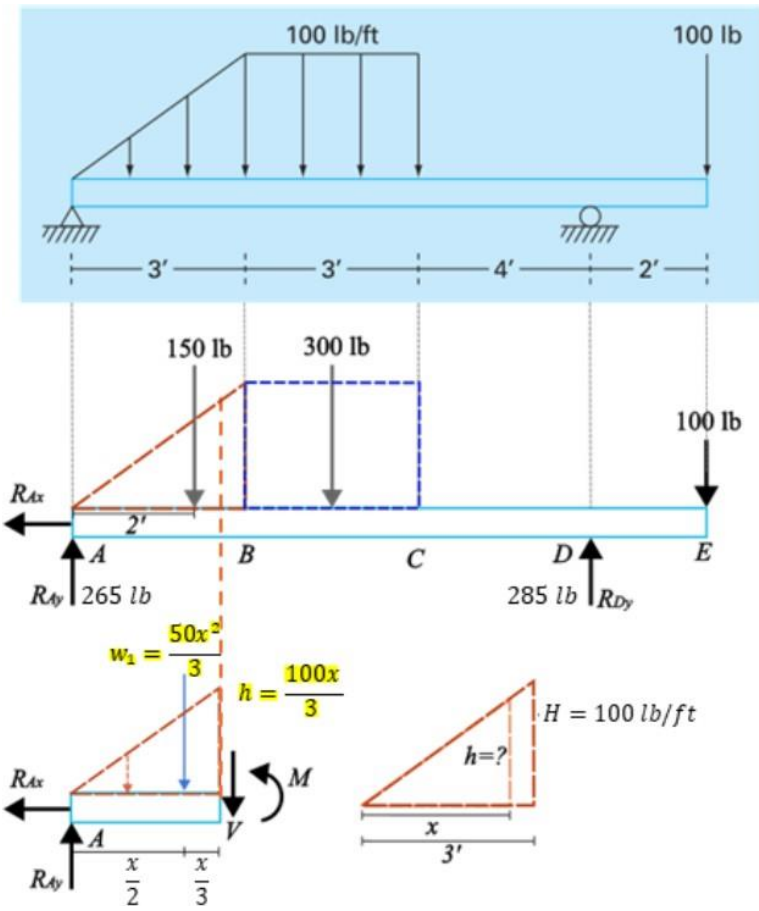
En cada iteración deberá sustituirse alguno de los límites del intervalo que contienea la raíz. Repitiendo este proceso, el intervalo se reduce paulatinamente hasta que alguna de las aproximaciones coincide razonablemente con la raíz de la función. El proceso se detiene cuando entre la aproximación x_i y la aproximación anterior$x_i - 1$ se satisface un nivel de error (absoluto o relativo) preestablecido (tolerancia).

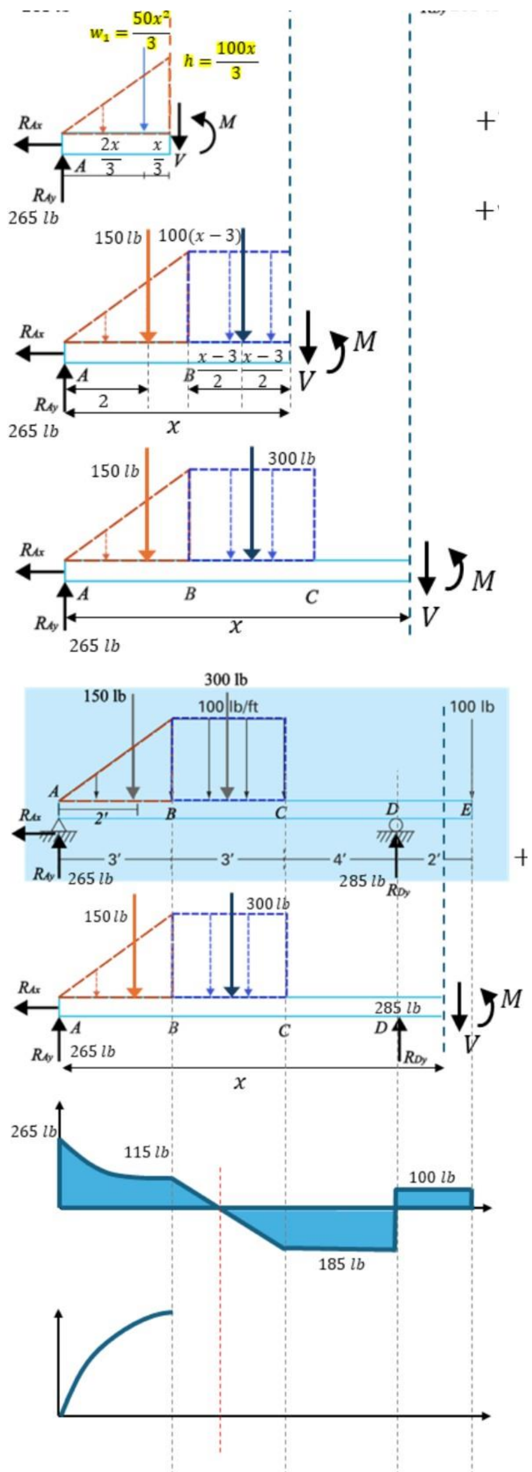
1.4 Casos prácticos

• Método de Bisección (Páginas. 140 realiza los ejercicios 5.14 y 5.15)

Método de Biseccion

Se carga una viga de la manera que se aprecia en la figura. Emplee el método de bisección para resolver la posición dentro de la viga donde no hay momento





Solución:
Se determinan las reacciones considerando la viga completa como un cuerpo libre

ΣMA = 0:

(-100 lb)(12 ft) + (Rpy) (10 ft) - (300 lb)(4.5 ft) – (150 lb)(2 ft) = 0

RDY = $\frac{\&8(\%\ lb/ft}{1\%ft}$ RDY = 285 lb ↑

↑ Σ Fy = 0: Ray - 150 lb - 300lb+285 lb - 100 lb

RAY = 265 lb ↑

$\frac{1\%\%}{/} = \frac{0}{x}, \rightarrow$ h = $\frac{1\%\%x}{/}$ At = $\frac{base*altura}{\&} = w1$

W1 = $\frac{(x)(\frac{100x}{\$})}{\&}, \rightarrow$ $w1 = \frac{(\%x^2}{/}$

Fuerza cortante y momento flector. Desde A hasta B (0 < x < 3)

+↑ΣE Fy = 0; 265lb - $\frac{(\%x^2}{/}$ – V = 0, $V = 265 - \frac{(\%x^2}{/}$

ΣM1 = 0: -265x +$\frac{(\%x^2}{/}(\frac{x}{/}) + M = 0,$ $M = 265x - \frac{(\%x^2}{:}$

Fuerza cortante y momento flector. Desde B hasta C (3 < x < 6)

+↑Fy=0: 265 lb - 150 lb – 100(x − 3) – V = 0, V = 115-100(x-3)

M2 = 0: −265x+150(x − 2) + 100(x − 3) $(\frac{x:/}{\&} + M = 0$

M = -50x² + 415x - 150

Fuerza cortante y momento flector. Desde C hasta D (6 < x < 10)

+↑ΣFy = 0: 265 lb - 150 lb - 300 – V = 0, V=-185 lb

Σ M3 = 0: -265x + 150(x-2) + 300(x 4.5) + M = 0,

M = -185x + 1650

Fuerza cortante y momento flector. Desde D hasta E (10 < x < 12)

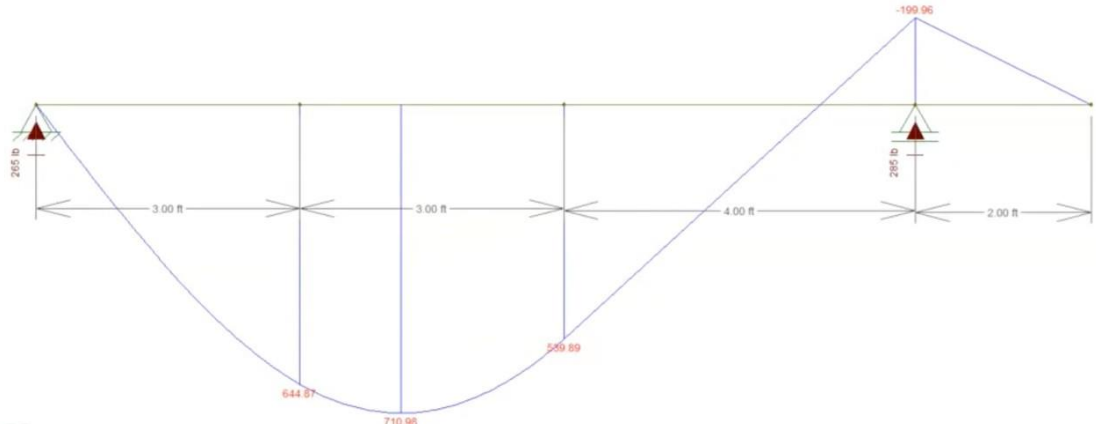
+↑ Σ F = 0: 265 lb - 150 lb – 300 lb + 285 – V = 0, V = -100 lb

Σ M3 = 0: −265x + 150(x − 2) + 300(x − 4.5) – 285(x − 10) + M = 0, M = 100x1200

Dominio	Ecuación	
A-B	V = 265 - $\frac{(\%x^2}{/}$ M = 265x - $\frac{(\%x^2}{/}$	V = 115 lb M = 645 lb*ft
B-C	V = 115 – 100x(x-3) M = -50x² + 415x-150	V = -185lb M = 540lb*ft, Mmax = 711.125 lb*ft

C-D	V = -185 lb	V = -185 lb

	M = -185x + 1650	M = -200lb*ft
D-E	V = -100 lb M = 100X - 1200	V = -100 lb M = 0



Paso 1: elija valores iniciales inferior, x_l y superiro, x_u, que encierre la raíz, de forma tal que la función cambie de signo en el intervalo. Esto se verifica comprobando que $f(x_l)f(x_u) < 0$.

Paso 2:

$$x_r = \frac{x_l + x_u}{2} = \frac{6+10}{2} = 8$$

El error relativo porcentual verdadero ε_t es:

$$\varepsilon_t = \frac{Valor\ real - x_r}{Valor\ real} = \frac{8.918 - 8}{8.918} = 0.1029\%$$

Paso 3:

Calculando el producto de los valores en la función en un límite inferior y en el punto medio:

$$f(6)f(8) = (-460)(170) = -78200$$

Entonces la raíz se encuentra dentro del subintervalo inferior o izquierdo. Por lo tanto la raíz debe de estar localizada entre 8 y 9.

Por lo tanto, haga $x_u = x_r$, volviendo al paso 2 para asi crear nuestra tabla de iteraciones para dar con la aproximación de la raíz.

Iteración	a	b	x_r	f(c)
1	0.00	10.00	5.00	725.00

2	5.00	10.00	7.50	262.50
3	7.50	10.00	8.75	31.25
4	8.75	10.00	9.38	-84.38
5	8.75	9.38	9.06	-26.56
6	8.75	9.06	8.91	2.34
7	8.91	9.06	8.98	-12.11
8	8.91	8.98	8.95	-4.88
9	8.91	8.95	8.93	-1.27
10	8.91	8.93	8.92	0.54
11	8.92	8.93	8.92	-0.37
12	8.92	8.92	8.92	0.09
13	8.92	8.92	8.92	-0.14
14	8.92	8.92	8.92	-0.03
15	8.92	8.92	8.92	0.03
16	8.92	8.92	8.92	0.00
17	8.92	8.92	8.92	-0.01
18	8.92	8.92	8.92	-0.01
19	8.92	8.92	8.92	-0.00
20	8.92	8.92	8.92	-0.00

La raíz aproximada es: 8.918920

La raíz aproximada es: 8.916015625

Iter	a	b	x_r	f(xl)f(xu)	Error
1	5	10	7.5	262.5	262.5
2	7.5	10	8.75	31.25	31.25
3	8.75	10	9.375	-84.375	84.375
4	8.75	9.375	9.0625	-26.5625	26.5625
5	8.75	9.0625	8.90625	2.34375	2.34375
6	8.90625	9.0625	8.984375	-12.109375	12.109375

7	8.90625	8.984375	8.9453125	-4.8828125	4.8828125
8	8.90625	8.9453125	8.92578125	-1.26953125	1.26953125
9	8.90625	8.92578125	8.916015625	0.537109375	0.537109375

```
import math

# Definir la funcion del problema del momento de flector
def f(x):
    return -185*x + 1650

# Parámetros del método
a = 6 # Límite inferior del intervalo
b = 10 # Límite superior del intervalo
tolerancia = 0.001

# Este es el método de bisección
def biseccion(a, b, tolerancia):
    # Comprobar que el intervalo es válido
    if f(a) * f(b) >= 0:
        print("El intervalo no es válido. f(a) y f(b) deben tener signos opuestos.")
        return None

    # Variables iniciales
    error = b - a
    iteracion = 0

    # Aplicar el método de bisección
    while error > tolerancia:
        c = (a + b) / 2 # Punto medio
        fc = f(c)
```

```
        print(f"Iteración {iteracion}: c = {c}, f(c) = {fc}")

        # Verificar si hemos encontrado la raíz con la tolerancia deseada
        if abs(fc) < tolerancia:
            return c

        # Determinar en qué subintervalo está la raíz
        if f(a) * fc < 0:
            b = c
        else:
            a = c

        # Actualizar el error y el contador de iteraciones
        error = b - a
        iteracion += 1

    # Devolver la raíz aproximada
    return (a + b) / 2



# Ejecutar el método de bisección
raiz = biseccion(a, b, tolerancia)
if raiz is not None:
    print(f"La raíz aproximada es: {raiz}")
```

```python
import math

# Definir la funcion del problema del momento de flector
def f(x):
    return -185*x + 1650

# Parámetros del método
a = 6  # Límite inferior del intervalo
b = 10  # Límite superior del intervalo
tolerancia = 0.001

# Este es el método de bisección
def biseccion(a, b, tolerancia):
    # Comprobar que el intervalo es válido
    if f(a) * f(b) >= 0:
        print("El intervalo no es válido. f(a) y f(b) deben tener signos opuestos
        return None

    # Variables iniciales
    error = b - a
    iteracion = 0

    # Aplicar el método de bisección
    while error > tolerancia:
        c = (a + b) / 2  # Punto medio
        fc = f(c)

        print(f"Iteración {iteracion}: c = {c}, f(c) = {fc}")

        # Verificar si hemos encontrado la raíz con la tolerancia deseada
        if abs(fc) < tolerancia:
            return c

        # Determinar en qué subintervalo está la raíz
        if f(a) * fc < 0:
            b = c
```

```
Iteración 0: c = 8.0, f(c) = 170.0
Iteración 1: c = 9.0, f(c) = -15.0
Iteración 2: c = 8.5, f(c) = 77.5
Iteración 3: c = 8.75, f(c) = 31.25
Iteración 4: c = 8.875, f(c) = 8.125
Iteración 5: c = 8.9375, f(c) = -3.4375
Iteración 6: c = 8.90625, f(c) = 2.34375
Iteración 7: c = 8.921875, f(c) = -0.546875
Iteración 8: c = 8.9140625, f(c) = 0.8984375
Iteración 9: c = 8.91796875, f(c) = 0.17578125
Iteración 10: c = 8.919921875, f(c) = -0.185546875
Iteración 11: c = 8.9189453125, f(c) = -0.0048828125
La raíz aproximada es: 8.91845703125


...Program finished with exit code 0
Press ENTER to exit console.
```

Ejercicio 5.15 Por un canal trapezoidal fluye agua a una tasa de Q=20m^3/s. La profundidad critica y para dicho canal satisface la ecuación

$$0 = 1 - \frac{Q^2}{gA_c^3}B$$

Donde g=9.81m/s^2, A_c = área de la sección transversal (m^2) y B = ancho del canal de la superficie (m) para este caso, el ancho y el área de la sección transversal se relaciona con la profundidad y po medio de:

$B = 3 + y$ y $A_c = 3y + \frac{y^2}{2}$

Resuelva para la profundidad crítica con el uso de los métodos *a*) gráfico, *b*) bisección, y *c*) falsa posición. En los incisos *b*) y *c*), haga elecciones iniciales de *xl* = 0.5 y *xu* = 2.5, y ejecute iteraciones hasta que el error aproximado caiga por debajo del 1% o el número de interaciones supere a 10. Analice sus resultados.

A) Método gráfico

Para comenzar con la interpretación de esta función y encontrar la raíz por el método gráfico, basta con sustituir los valores de “y” que nos señala el ejercicio.

Por lo tanto, realizamos la sustitución de y =.5 y 2.5 así como todos los valores que ya nos otorgó el ejercicio en la función:

$$0 = 1 - \frac{Q^{\&}}{gA_c^{/}} B$$

Quedando de la siguiente manera:

Q=20

g=9.81

Ac=3y+$\frac{y^2}{\&}$

B=3+y

$$1 - \frac{20^{\&}}{9.81 * (3y + \frac{y^{\&}}{2})^{/}} * 3 + y$$

Sustituimos el valor de y empezando con .5 como lo pide el ejercicio:

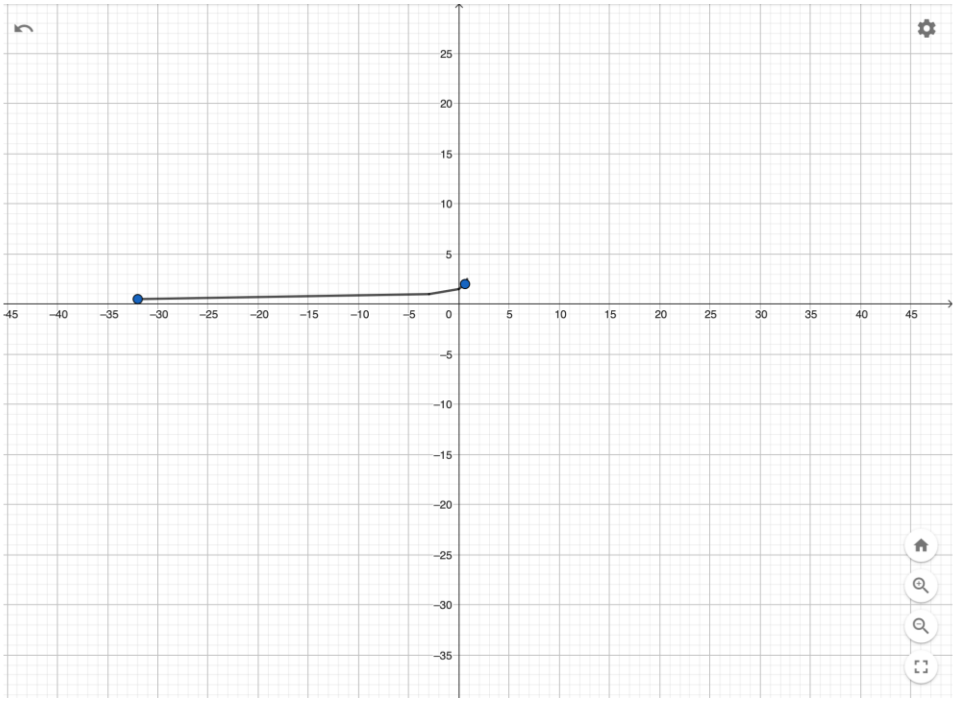
$$1 - \frac{400}{9.81 * (3 * .5 + \frac{.5^{\&}}{2})^{/}} * 3 + .5 = -32.25$$

Esto representa el valor de x cuando y tiende a .5

Ahora realizamos la evaluación de y cuando vale 2.5

$$1 - \frac{400}{9.81 * (3 * .25 + \frac{2.5^{\&}}{2})^{/}} * 3 + 2.5 = 0.8134$$

Con estos cálculos ya sabemos que, si hay una raíz entre estos dos números ya que hubo un cambio de signo, pero a simple vista es difícil ver donde se encuentra la raíz



Para tener un dato más exacto usando el método gráfico, realizamos la evaluación de y=.5 a 2.5 con un rango de .5 para cada evaluación, dándonos un valor más cercano a la raíz, donde usaremos un programa para la facilitar el cálculo.

El cual quedará de la siguiente manera:

$$1-\frac{400}{9.81*(3*.5+\frac{.5^{\&}}{2})^{/}}*(3+.5)=-32.81$$

$$1-\frac{400}{9.81*(3*1+\frac{1^{\&}}{2})^{/}}*(3+1)=-2.8041$$

$$1-\frac{400}{9.81*(3*1.5+\frac{1.5^{\&}}{2})^{/}}*(3+1.5)=-.0309$$

$$1-\frac{400}{9.81*(3*2+\frac{2^{\&}}{2})^{/}}*(3+2).=6018$$

$$1 - \frac{400}{9.81 * (3 * 2.5 + \frac{2.5^{\&}}{2})^{/}} * 3 + 2.5 = .8130$$

Dando las siguientes coordenadas:

x	y
-32.81	.5
-2.8041	1
-.0309	1.5
.6018	2
.8130	2.5

Python

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

# Definimos la función
def f(y):
    return 1 - (400 / (9.81 * (3 * y + (y**2) / 2)**3)) * (3 + y)

# Generamos valores de y de 0.1 en 0.1
y_values = np.arange(0.5, 2.6, 0.5)
x_values = f(y_values)

# Mostramos cada iteración
for y, x in zip(y_values, x_values):
    print(f'y: {y:.2f}, x: {x:.4f}')

# Graficamos con x en el eje horizontal y y en el eje vertical
plt.plot(x_values, y_values, marker='o')
plt.title('Comportamiento de la función')
plt.xlabel('x')
plt.ylabel('y')
plt.grid()
plt.show()
```

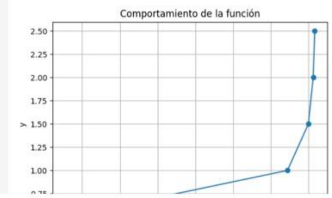
Ejecute | Guar

Entrada del programa

Salida del programa

```
y: 0.50, x: -32.2582
y: 1.00, x: -2.8041
y: 1.50, x: -0.0309
y: 2.00, x: 0.6018
y: 2.50, x: 0.8130

[Execution complete with exit code 0]
```



main.py

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

# Definimos la función
def f(y):
    return 1 - (400 / (9.81 * (3 * y + (y**2) / 2)**3)) * (3 + y)

# Generamos valores de y de 0.1 en 0.1
y_values = np.arange(0.5, 2.6, 0.5)
x_values = f(y_values)

# Mostramos cada iteración
for y, x in zip(y_values, x_values):
    print(f'y: {y:.2f}, x: {x:.4f}')

# Graficamos con x en el eje horizontal y y en el eje vertical
plt.plot(x_values, y_values, marker='o')
plt.title('Comportamiento de la función')
plt.xlabel('x')
plt.ylabel('y')
plt.grid()
plt.show()
```

input

```
y: 0.50, x: -32.2582
y: 1.00, x: -2.8041
y: 1.50, x: -0.0309
y: 2.00, x: 0.6018
y: 2.50, x: 0.8130
```

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

# Definimos la función
def f(y):
    return 1 - (400 / (9.81 * (3 * y + (y**2) / 2)**3)) * (3 + y)

# Generamos valores de y de 0.1 en 0.1
y_values = np.arange(0.5, 2.6, 0.5)
x_values = f(y_values)

# Mostramos cada iteración
for y, x in zip(y_values, x_values):
    print(f'y: {y:.2f}, x: {x:.4f}')

# Graficamos con x en el eje horizontal y y en el eje vertical
```

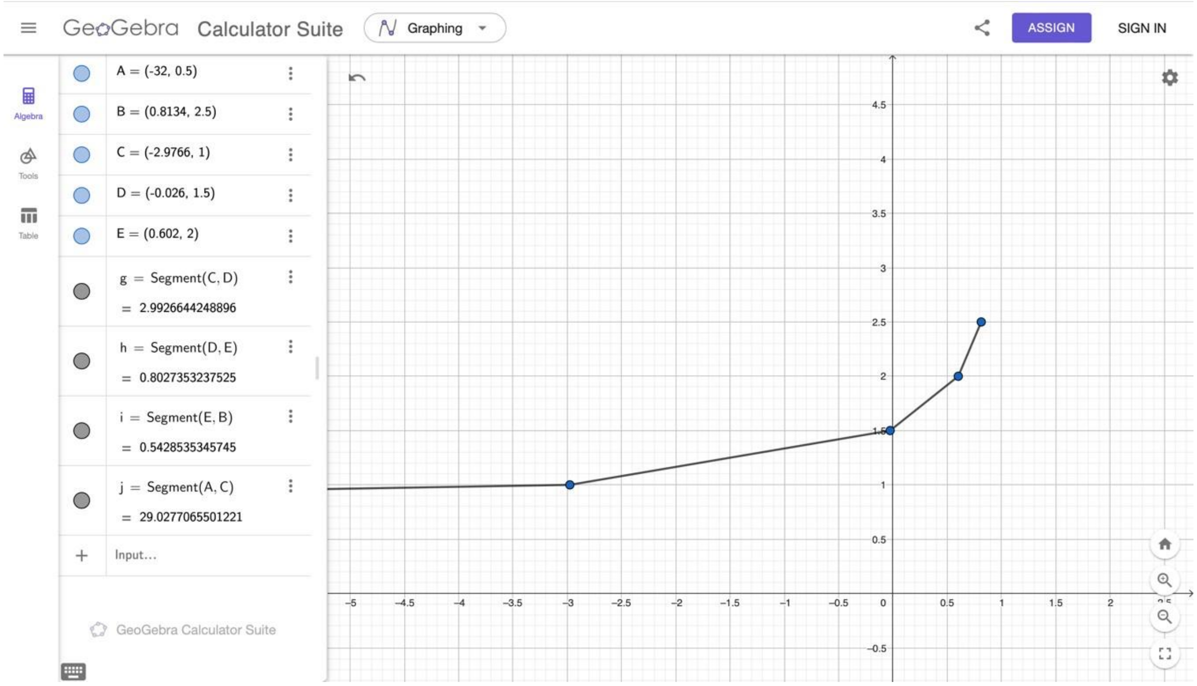
```
plt.plot(x_values, y_values, marker='o')
plt.title('Comportamiento de la función')
plt.xlabel('x')
plt.ylabel('y')
plt.grid()
plt.show()
```

Podemos notar que en valor de 1.5 a 2 hay un cambio de signo, por lo que ahí estará la raíz.

Proseguimos a graficar las coordenadas obtenidas que ya fueron graficadas en Python pero que también se realizarán en GeoGebra para una comprensión más profunda.

Dando las siguientes coordenadas:

x	y
-32.81	.5
-2.8041	1
-.0309	1.5
.6018	2
.8130	2.5



Podemos observar que la raíz se encuentra en 1.5 como lo pudimos comprobar con la evaluación la que demostró que hubo un cambio de signo en 1.5 y 2, confirmando que ahí se encuentra la raíz

B) Método de Bisección

Para resolverlo por el método de bisección, anotaremos primeramente los pasos que se realizan.

Paso 1. Sacar el puto medio que es igual a $m = \frac{a"b}{\&}$

Paso 2: Evaluar f(a), f(b) y f(m)

Paso 3: Determinar el nuevo intervalo

Paso 4: Calcular el error $\mathrm{e} = \frac{>;a}{\&}$

En este caso desglosamos los valores que tenemos:

a = .05

b=2.5

i=.01

Primeramente, proseguimos a calcular el punto medio:

$$m = \frac{.5 + 2.5}{2} = 1.5$$

Evaluamos en la función con el valor de a,b y el ultimo valor que acabamos de obtener que es m.

$$f(a) = 1 - \frac{400}{9.81 * (3 * .5 + \frac{.5^{\&}}{2})^{/}} * (3 + .5) = -32.81$$

$$f(b) = 1 - \frac{400}{9.81 * (3 * 2.5 + \frac{2.5^{\&}}{2})^{/}} * 3 + 2.5 = .8130$$

$$f(m) = 1 - \frac{400}{9.81 * (3 * 1.5 + \frac{1.5^{\&}}{2})^{/}} * (3 + 1.5) = -.0309$$

Para determinar el nuevo intervalo tendremos que echar un vistazo a los signos, si “a” y “m” tienen los mismos signos, entonces “a” se descarta y “m” se toma como nuevo “a”.

Si “b” y “m” tienen el mismo signo, entonces “b” se descarta y “m” se toma como nuevo “b”.

En este caso tenemos que “a” y “m” tienen el mismo signo, por lo tanto “m” se toma como el nuevo “a” quedando de la siguiente manera: 1.5 que es “m” toma el lugar de “a” que era .5 y “b” queda igual 2.5, por lo tanto quedaría (1.5,-2.5) o a =1.5 $\mathrm{y\ b} = 2.5$

Ahora tendremos que calcular el error con el nuevo “a”

$$\mathrm{e} = \frac{\mathrm{b} - \mathrm{a}}{2}$$

$$\mathrm{e} = \frac{2.5 - 1.5}{2} = .5$$

Nos da un margen de error de 50 por ciento, por lo tanto, tendremos que seguir iterando hasta obtener el .01 o 1% o alcanzar las 10 iteraciones como lo pide el ejercicio.

Sacamos “m”

$$m = \frac{a+b}{2}$$

$$m = \frac{1.5+2.5}{2} = 2$$

Evaluamos las ecuaciones:

$$f(a) = 1 - \frac{400}{9.81 * (3 * 1.5 + \frac{1.5^{\&}}{2})^{/}} * (3 + 1.5) = -.0309$$

$$f(b) = 1 - \frac{400}{9.81 * (3 * 2.5 + \frac{2.5^{\&}}{2})^{/}} * 3 + 2.5 = .8130$$

$$f(m) = 1 - \frac{400}{9.81 * (3 * 2 + \frac{2^{\&}}{2})^{/}} * (3 + 2) = .6018$$

Evaluamos:

Nos damos cuenta de que “b” y “m” tienen los mismos signos, por lo tanto, desplazamos “b” y ahora “m” será el nuevo “b”

$$a = 1.5 \; b = 2$$

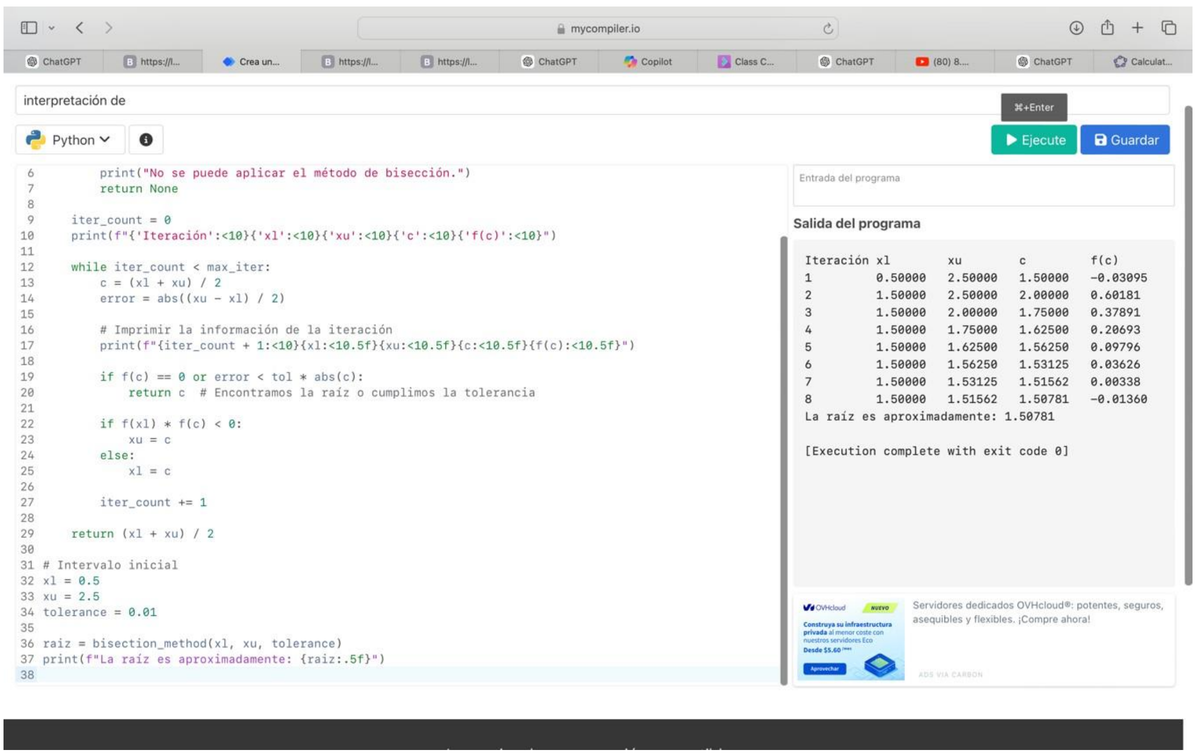
Evaluamos el error con el nuevo “b”.

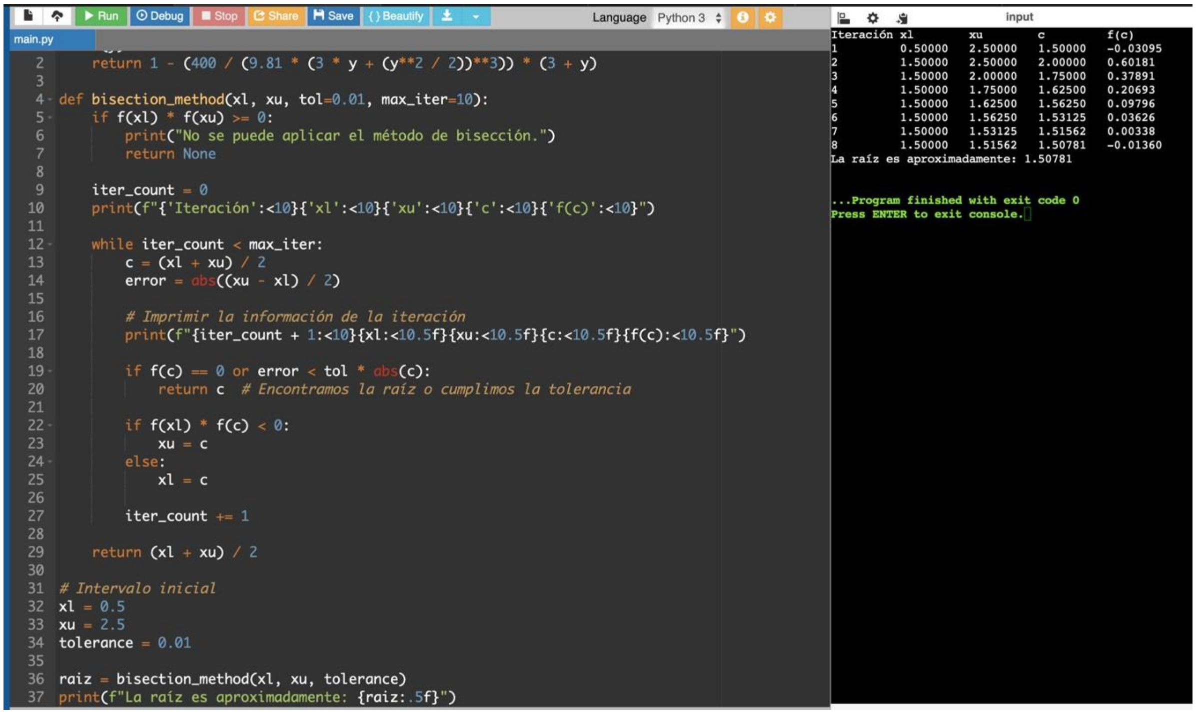
$$\mathrm{e} = \frac{\mathrm{b} - \mathrm{a}}{2}$$

$$\mathrm{e} = \frac{2 - 1.5}{2} = .25$$

Indicativo que tendremos que seguir hasta alcanzar .01 de tolerancia en el error, alcanzar 10 iteraciones o el valor de f(m) este muy cerca de 0 y arroja resultados repetidos o varían por muy pocas milésimas, esto significa que hemos encontrado la raíz.

Para no realizar todo el procedimiento tan repetitivamente, usaremos Python para encontrar la raíz, tomando los valores de xl= 0.5 y xu= 2.5 y una tolerancia de 1% porque así lo dicta el ejercicio.






```
def f(y):
    return 1 - (400 / (9.81 * (3 * y + (y**2 / 2))**3)) * (3 + y)

def bisection_method(xl, xu, tol=0.01, max_iter=10):
    if f(xl) * f(xu) >= 0:
        print("No se puede aplicar el método de bisección.")
        return None

    iter_count = 0
    print(f"{'Iteración':<10}{'xl':<10}{'xu':<10}{'c':<10}{'f(c)':<10}")

    while iter_count < max_iter:
        c = (xl + xu) / 2
        error = abs((xu - xl) / 2)

        # Imprimir la información de la iteración
        print(f"{iter_count + 1:<10}{xl:<10.5f}{xu:<10.5f}{c:<10.5f}{f(c):<10.5f}")
```

```
        if f(c) == 0 or error < tol * abs(c):
            return c  # Encontramos la raíz o cumplimos la tolerancia

        if f(xl) * f(c) < 0:
            xu = c
        else:
            xl = c

        iter_count += 1

    return (xl + xu) / 2

# Intervalo inicial
xl = 0.5
xu = 2.5
tolerance = 0.01

raiz = bisection_method(xl, xu, tolerance)
print(f"La raíz es aproximadamente: {raiz:.5f}")
```

la raíz nos dio el valor de **1.50781**, resultado que ya habíamos calculado con las evalucaiones y usando el metodo gráfico, pero ahora con el metodo de bisección calculamos con una tolerancia de .01

C) Ahora haremos el método de la falsa posición.

Pasos para realizar el método:

1-Evaluamos f(a) y f(b)

2-Obtener xi con la siguiente formula:

$$xi = \frac{af(b) - bf(a)}{f(b) - f(a)}$$

3- Determinar nuevo intervalo

4- Calcular el error

Por lo que proseguimos a realizar la actividad.

Los puntos a y b los escogeremos de 1.5 a 2.0 porque ya sabemos que hay una raíz entre estos dos puntos.

a=1.5

b=2.0

Proseguimos a evaluar f(a) y f(b)

$$0 = 1 - \frac{Q^{\&}}{gA_c^{/}} B$$

$$1 - \frac{20^{\&}}{9.81 * (3y + \frac{y^{\&}}{2})^{/}} * 3 + y$$

$$f(a) = 1 - \frac{400}{9.81 * (3 * 1.5 + \frac{1.5^{\&}}{2})^{/}} * 3 + 1.5 = -.0215$$

$$f(b) = 1 - \frac{400}{9.81 * (3 * 2 + \frac{2^{\&}}{2})^{/}} * (3 + 2). = .6018$$

Con los valores que nos dieron en f(a y b) obtenemos xi

$$xi = \frac{af(b) - bf(a)}{f(b) - f(a)}$$

$$xi = \frac{1.5 * 6018 - 2 * -.0215}{.6018 - -.0215} = 1.517$$

Ahora tendremos que determinar el nuevo intervalo fijándonos en el signo de las funciones a,b y xi

$$f(xi) = 1 - \frac{400}{9.81 * (3 * 1.517 + \frac{1.517^{\&}}{2})^{/}} * (3 + 1.517) = 0.012$$

Si “a” y “xi” sus signos son iguales, entonces tendremos que sustituir el valor de a por “xi”, el cual tomará el nuevo valor de “a”

Pero si “b” y “xi” tienen el mismo signo, entonces tendremos que sustituir el valor de “b” por “xi”, el cual será el nuevo “b”

Por lo tanto, nos damos cuenta de que el valor que se sustituirá será el de “b”, quedando en su lugar el valor de “xi”.

Ahora tendríamos que calcular el error, pero al ser la primera iteración, no será posible.

Continuamos con la segunda iteración:

Tomamos los nuevos valores:

a= 1.5

b=1.517

1- Evaluamos las funciones de a y b:

$$f(a) = 1 - \frac{400}{9.81 * (3 * 1.5 + \frac{1.5^{\&}}{2})^{/}} * 3 + 1.5 = -.0215$$

$$f(b) = 1 - \frac{400}{9.81 * (3 * 1.517 + \frac{1.517^{\&}}{2})^{/}} * 3 + 1.517 = 0.012$$

2- calculamos xi

$$xi = \frac{1.5 * .012 - 1.517 *. -0215}{0.012 - -.0215} = 1.5110$$

3- determinamos el nuevo valor evaluando f(xi)

$$f(xi) = 1 - \frac{400}{9.81 * (3 * -1.5110 + \frac{1.5110^{\&}}{2})^{/}} * 3 + 1.5110 = 5.634.$$

3- Determinamos el nuevo intervalo:

Notamos que el valor de a y xi son iguales por lo tanto nos quedamos con el valor de -1.5110

Por lo tanto, los nuevos valores quedan:

a=1.5110 y b= 1.517

4-Calculamos el error

El error se calcula restando el valor actual de la iteración menos el valor anterior

$$|xi2 - xi1|$$

$$|1.5110 - 1.517| = -.006$$

Esto quiere decir que tenemos una raíz aproximada de **1.5110** con un margen de error de .6% es decir, el valor nos salió ligeramente diferente por unas milésimas al que obtuvimos por el método de bisección de 1.50781, debido a que por este método lo hicimos con más exactitud que por bisección con un margen de error aún menor, ya que si hubiéramos continuado con un margen de error menor enbisección nos hubiera dado un valor muy similar, lo que si pudimos notar es que seobtiene la raíz mas raídamente que por el de bisección.

Método newton- Raphson (pág. 169 realiza el ejercicio 6.26)

6.9 Dertermine la raíz real más grande de $f(x) = 0.95x^{/} - 5.9x^{\&} + 10.9x - 6$:

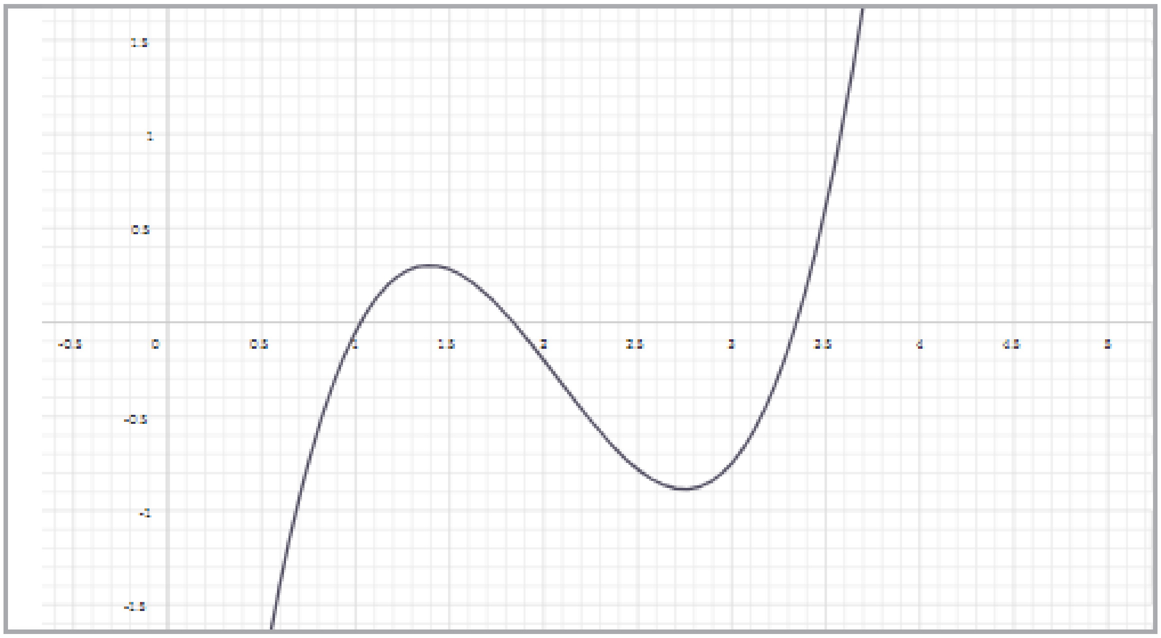
a) En forma gráfica.

b) Con el uso del método de Newton-Raphson (tres iteraciones, $x_{\mathrm{i};1} = 3.5$)

c) Con el método de la secante (tres iteraciones, $x_{\mathrm{i};1} = 2.5\ y\ x_{\mathrm{i};1} = 3.5$

d) Por medio del método de la secante modificado (tres iteraciones, $x_{\mathrm{i};1} = 3.5, \delta = 0.01$).

Grafica:



Método newton- Raphson

f(x)=0.95x^3-5.9x^2+10.9x-6

f(x)=2.85x^2-11.8x+10.9

f(3,5)0.95(3.5)^3-5.9(3.5)^2+10.9(3.5)-6

0.60325

f (3,5) =2.85(3,5)^2-11.8(3.5) +10.9

4.5125

$$\left(\frac{0.60325}{4.5125}\right) \longrightarrow \mathbf{X_1=3.5}\text{-}X_1=3.3656$$

$f(3.3656)=0.95(3.3656)^3-5.9(3.3659)^2+10.9(3.3656)-6$

=0.071071

$f(3.3656)=2.85(3.3656)^2-11.8(3.3656)+10.9$

=3.4687

$$\left(\frac{0.071071}{3.4687}\right) \longrightarrow X_2=3.3656\text{-} \quad X_2=3.3451$$

$f(3.3451)=0.95(3.3451)^3-5.9(3.3451)^2+10.9(3.3451)-6$

=0.0015

$f(3.3451)=2.85(3.3451)^2-11.8(3.3451)+10.9$

=3.3184

$$\left(\frac{0.0015}{3.3184}\right) \longrightarrow X_3=3.3451\text{-} \quad X_3=3.3446$$

Método de la secante

X_{i+1}=Xi- f(xi)(Ai;1;x&) / f(xi;1);(fx1)

$X_{-1}=2.5 \quad f(x1)=-0.7812$

$X_0=3.5 \quad f(x1)=0.6062$

X₁=3.5- (/.(;&.()(%.C%C&) / %.%C%C&;(;%.D81&) = 3.0630

X₂=3.0630- (;%.CCCD(&.(;/.%C/%) / ;.%.D81&"%.CCCD) = 6.3407

X_3= 6.3407- (;C8.%8(((&.(;C./E%D) / [illegible]) = 6.2849

E_A= C.&8E:;C./E%D / C.&8:: = 0.88%

f(*x*) en el cuarto cuadrante, asegúrese de tomar el valor negativo de la raíz cuadrada. ¿Por qué diverge la solución?

6.26 Suponga el lector que está diseñando un tanque esférico (véase la figura P6.26) de almacenamiento de agua para un poblado pequeño de un país en desarrollo. El volumen del líquido que puede contener se calcula con

$$V = \pi h^2 \frac{[3R - h]}{3}$$

donde V = volumen [pie^3], h = profundidad del agua en el tanque [pies], y R = radio del tanque [pies].

Si R = 3 m, ¿a qué profundidad debe llenarse el tanque de modo que contenga 30 m^3? Haga tres iteraciones del método de Newton-Raphson para determinar la respuesta. Encuentre el error relativo aproximado después de cada iteración. Observe que el valor inicial de R convergerá siempre.

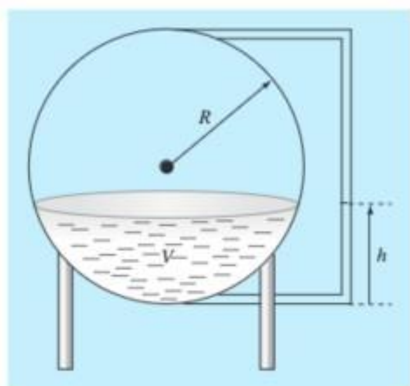


Figura P6.26

Para resolver este problema utilizando el método de Newton-Raphson, primero necesitamos definir la función (f(h)) que queremos igualar a cero. Dado que el volumen (V) es 30 m³, podemos escribir:

$$\mathbb{W} f(h) = \pi h^{\&} Z\frac{3R-h}{3}\backslash - 30]$$

Con (R = 3) m, la función se convierte en:

$$\hat{} f(h) = \pi h^{\&} \_\frac{;;0}{/}` - 30a \quad \hat{} f(h) = \pi h^{\&} \_3 - \frac{0}{/}` - 30a \qquad \hat{} f(h) = \pi h^{\&} \_3 - \frac{0}{/}` - 30a$$

$$\hat{} f(h) = \pi h^{\&} \_3 - \frac{0}{\neq}` - 30a$$

La derivada de (f(h)) es:

$$\mathbb{W} f'(h) = \frac{d}{dh} Z\pi h^{\&} Z3 - \frac{h}{3}\backslash - 30\backslash] \; [f'(h) = \pi(6h - h^{\&})]$$

Ahora, aplicamos el método de Newton-Raphson:

$$fh_{n"1} = h_n - \frac{f(h_n)}{f'(h_n)}g$$

Iteración 1:

Supongamos un valor inicial (h_0 = 1.5) m.

$$\mathbb{W}f(h_{\%}) = \pi(1.5)^{\&} Z3 - \frac{1.5}{3}\backslash - 30] \; [f(h_{\%}) = \pi(2.25)(2.5) - 30][f(h_{\%})$$

$$= \pi(5.625) - 30][f(h_{\%}) \approx 17.67 - 30 = -12.33]$$

$$[f'(h_{\%}) = \pi(6(1.5) - (1.5)^{\&})][f'(h_{\%}) = \pi(9 - 2.25)][f'(h_{\%}) = \pi(6.75) \approx 21.21]$$

$$\mathbb{W}h_1 = 1.5 - \frac{-12.33}{21.21} \approx 2.08]$$

Error relativo aproximado:

$$\mathbb{W}\text{Error} = ,\frac{2.08 - 1.5}{2.08}, \times 100 \approx 27.88\%]$$

Iteración 2:

$$\mathbb{W}f(h_1) = \pi(2.08)^{\&} Z3 - \frac{2.08}{3}\backslash - 30] \; [f(h_1) \approx \pi(4.33)(2.31) - 30][f(h_1)$$

$$\approx 31.42 - 30 = 1.42][f'(h_1) = \pi(6(2.08) - (2.08)^{\&})][f'(h_1)$$

$$\approx \pi(12.48 - 4.33)][f'(h_1) \approx \pi(8.15) \approx 25.61]$$

$$\mathbb{W}h_{\&} = 2.08 - \frac{1.42}{25.61} \approx 2.02]$$

Error relativo aproximado:

$$\mathbb{W}\text{Error} = ,\frac{2.02 - 2.08}{2.02}, \times 100 \approx 2.97\%]$$

Iteración 3:

$$\mathbb{W}f(h_{\&}) = \pi(2.02)^{\&} Z3 - \frac{2.02}{3}\backslash - 30] \; [f(h_{\&}) \approx \pi(4.08)(2.32) - 30][f(h_{\&})$$

$$\approx 29.82 - 30 = -0.18][f'(h_{\&}) = \pi(6(2.02) - (2.02)^{\&})][f'(h_{\&})$$

$$\approx \pi(12.12 - 4.08)][f'(h_{\&}) \approx \pi(8.04) \approx 25.26]$$

$$\mathbb{W}h_{/} = 2.02 - \frac{-0.18}{25.26} \approx 2.03]$$

Error relativo aproximado:

$$\mathbb{W}\text{Error} = , \frac{2.03 - 2.02}{2.03}, \times 100 \approx 0.49\%]$$

Después de tres iteraciones, la profundidad del agua en el tanque es aproximadamente 2.03 m.

Establece una breve descripción en la que hagas un contraste entre los métodos que programaste, destacando los problemas a los que te enfrentaste y cómo lo solucionaste.

En esta tarea pudimos percibir con claridad la relevancia de los errores como una guía para evaluar qué tan próximas están las iteraciones generadas de la solución al aplicar cada método. Además, adquirimos conocimientos sobre la implementación de diversos enfoques para resolver ecuaciones no lineales, así como cuándo es apropiado emplear métodos abiertos o cerrados, tomando en cuenta las ventajas y desventajas de cada uno.

El método de bisección es seguro y garantiza la convergencia al dividir intervalos donde la función cambia de signo, aunque puede ser más lento.

Ambos métodos son ampliamente aplicables en diversas disciplinas para resolver ecuaciones no lineales.

De igual manera, comprendimos la importancia del uso de lenguajes de programación para la resolución de problemas matemáticos extensos, tediosos o complejos, y las ventajas que estos ofrecen en comparación con software menos especializado. Durante el desarrollo de los casos prácticos en este proyecto integrador, también pudimos observar cómo aplicar los métodos abiertos y cerrados en la resolución de problemas de ingeniería, obteniendo las ecuaciones necesarias para la aplicación de dichos métodos. Además, notamos que algunos métodos resultan más eficientes o breves que otros, dependiendo del contexto, lo cual serefleja en el número de iteraciones generadas según el método empleado.

Referencias.

- Estadística,s/f, Métodos Numéricos 3: Raíces de ecuaciones: Métodos de Newton-Raphson y de la secante,página web, https://estadistica-dma.ulpgc.es/FCC/05-3- Raices-de-Ecuaciones-2.html#metodo-de-newton-raphson

- Chapra, S. & Canale, R. (2007). Métodos numéricos para ingenieros [Versiónelectrónica]. Recuperado de http://artemisa.unicauca.edu.co/~cardila/Chapra.pdf

- Chapra, S. y Canale, R. (2007). Métodos numéricos para ingenieros. Recuperado el 21 de septiembre del 2024, d ehttp://artemisa.unicauca.edu.co/~cardila/Chapra.pdf

- Serrano, J. (s.f.). Métodos Numéricos: Bisección. Recuperado el 21 de septiembredel 2024, de https://www.geogebra.org/m/mNY3NPuU

PROYECTO INTEGRADOR
ETAPA 2

2.1 Conceptualización

Reproduce y completa el siguiente cuadro comparativo en el que describas los elementos sustantivos de los siguientes métodos:

MÉTODO / ASPECTO	JACOBI	GAUSS-SEIDEL
¿EN QUÉCONSISTE?	Un método iterativo con el cual se resuelve el sistema lineal Ax=b comienza con una aproximación inicial x^((0)) a la solución x y genera una sucesión de vectores x^((k)) que converge a x. Los métodos iterativos traen consigo un proceso que convierte el sistema Ax=b en otro equivalente de la forma x=Tx+c para alguna matriz fija T y un vector c.	Este método es una versión acelerada de Jacobi. En el cual es necesario contar con un vector aproximado completo para proceder a la sustitución en las ecuaciones de recurrencia y obtener una nueva aproximación. En el método de Gauss-Seidel se propone ir sustituyendo los nuevos valores de la aproximación siguiente conforme se vayan obteniendo sin esperar a tener un vector completo. De esta forma se acelera la convergencia.

VENTAJAS	Métodos más simples para resolver sistemas lineales con estabilidad numérica y (en muchos casos) soluciones inexactas. Converge para cualquier vector inicial si la matriz es diagonalmente dominante.	A pesar de no tener la diagonal dominante, se puede hacer y posibilidades de su resultado. En su programación se puede llegar a un valor aproximado máximo o resultado exacto.
DESVENTAJAS	Baja estabilidad numérica. El elemento de la diagonal principal de cada ecuación debe ser mayor en valor absoluto.	No siempre converge a una solución. A veces converge muy lentamente. Es difícil definir el margen mínimo por el que ese coeficiente debe dominar a los otros para asegurar la convergencia

2.2 Casos prácticos

Una campaña de electrónica produce transistores, resistores y chips de computadora. Cada transistor requiere cuatro unidades de cobre, una de zinc y dos de vidrio. Cada resistor requiere tres, tres y una unidad de dichos materiales, respectivamente, y cada chip de computadora requiere dos, una y tres unidades de los materiales, respectivamente. En forma de tabla, esta información queda así:

COMPONENTE	COBRE	ZINC	VIDRIO
TRANSISTORES	4	1	2
RESISTORES	3	3	1
CHIPS DE COMPUTADORA	2	1	3

Los suministros de estos materiales varían de una semana a la otra, de modo que la compañía necesita determinar una corrida de producción diferente cada semana. Por ejemplo, cierta semana las cantidades disponibles de los materiales son 960 unidades de cobre, 510 unidades de zinc y 610 unidades de vidrio. Platee el sistema de ecuaciones que modela la corrida de producción, la información que se da en este capítulo sobre la solución de ecuaciones algebraicas lineales para resolver cual es el número de transistores, resistores y chips de computadora por manufacturar esta semana.

Primeramente, lo resolveremos por el método de Jacobi.

Como primer paso tendremos que plantear el problema de la siguiente manera:

x= transistores

y= resistores

z= Chips de computadoras

Por lo tanto, la matriz queda de la siguiente manera:

$$\boldsymbol{4x + 3y + 2z = 96}$$

$$\boldsymbol{x + 3y + z = 510}$$

$$\boldsymbol{2x + y + 3z = 610}$$

Para garantizar que la solución por el método de Jacobi pueda converger tenemos que revisar que la matriz sea estrictamente dominante.

Para eso tenemos que identificar si a1 es mayor a la suma en su fila, lo mismo para a2 y a3

En el primer caso notamos que no cumple la condición porque 4<3+2 (No cumple)
En la segunda fila, cumple la condición porque 3>1+1 (Cumple)
Y en la tercera no cumple porque 3=2+1(No cumple)

Por lo que llegamos a la conclusión de que la matriz no es estrictamente dominante, pero aun así realizaremos el ejercicio para ver si logra converger.

Para comenzar con k_0 empezaremos con los valores (0.0,0)
Proseguimos a despejar los valores en cada fila:

$$\boldsymbol{x = \frac{-3y - 2z + 960}{4}}$$

$$\boldsymbol{y = \frac{-x - z + 510}{3}}$$

$$\boldsymbol{z = \frac{-2x - y + 610}{3}}$$

Iteración 1

Ahora sustituimos nuestros valores por k_0

$$\boldsymbol{x = \frac{-3(0) - 2(0) + 960}{4} = 240}$$

$$\boldsymbol{y = \frac{-(0) - (0) + 510}{3} = 170}$$

$$\boldsymbol{z = \frac{-2(0) - (0) + 610}{3} = 203.3333333}$$

Ahora actualizamos los valores:

$$x = 240$$

$$y = 170$$

$$z = 203.3333333$$

No calculamos el error porque por lo menos necesitamos k_1

Iteración 2

Ahora estos valores se sustituirán en la operación para k_1

$$x = \frac{-3(170) - 2(203.3333333) + 960}{4} = 10.833333334$$

$$y = \frac{-(240) - (203.33333333) + 510}{3} = 22.22222222$$

$$z = \frac{-2(240) - (170) + 610}{3} = -13.33333333$$

Ahora estos son los nuevos valores:

$$x = 10.833333334$$

$$y = 22.22222222$$

$$z = -13.33333333$$

En esta iteración, que sería la dos, ya podemos calcular el error absoluto entre iteración

$$\boldsymbol{E_{ax} = |xi - xi_{-1}|}$$

$$\boldsymbol{E_{ay} = |yi - yi_{-1}|}$$

$$\boldsymbol{E_{az} = |zi - zi_{-1}|}$$

$$\boldsymbol{E_{ax} = |10.833333334 - 240| = 229.16666666}$$

$$\boldsymbol{E_{ay} = |22.22222222 - 170| = 147.77777778}$$

$$\boldsymbol{E_{az} = |-13.33333333 - 203.3333333)| = 216.66666}$$

Iteración 3

En la tercera iteración sustituimos estos nuevos valores para:

$$\boldsymbol{x = \frac{-3(22.22222222) - 2(-13.33333333) + 960}{4} = 230}$$

$$\boldsymbol{y = \frac{-(10.833333334) - (-13.33333333) + 510}{3} = 170.83333333}$$

$$\boldsymbol{z = \frac{-2(10.833333334) - (22.22222222) + 610}{3} = 188.70370371}$$

Los nuevos valores serán:

$$\boldsymbol{x = 230}$$

$$\boldsymbol{y = 170.83333333}$$

$$\boldsymbol{z = 188.70370371}$$

Calculamos el error:

$E_{ax} = |xi - xi_{-1}|$

$E_{ay} = |yi - yi_{-1}|$

$E_{az} = |zi - zi_{-1}|$

$E_{ax} = |230 - 10.833333334| = 219.16666666$

$E_{ay} = |170.83333333 - 22.22222222| = 148.6111111$

$E_{az} = |188.70370371 - (-13.33333333))| = 202.03703671$

Iteración 4

Ahora sustituimos los valores:

$$x = \frac{-3(170.83333333) - 2(188.70370371) + 960}{4} = 17.523148145$$

$$y = \frac{-(230) - (188.70370371) + 510}{3} = 30.43209876$$

$$z = \frac{-2(230) - (170.83333333) + 610}{3} = -6.944444443$$

El nuevo valor sería:

$x = 17.523148145$

$y = 30.43209876$

$z = -6.944444443$

Ahora calculamos el error

$E_{ax} = |xi - xi_{-1}|$

$$E_{ay} = |yi - yi_{-1}|$$

$$E_{az} = |zi - zi_{-1}|$$

$$E_{ax} = |17.523148145 - 230| = 212.47685186$$

$$E_{ay} = |30.43209876 - 170.83333333| = 140.40123457$$

$$E_{az} = |-6.944444443 - -188.70370371| = 195.648148153.$$

Y seguimos así hasta encontrar la tolerancia de .01

Al ser algo tan laborioso, tendremos que programarlo, dándonos el siguiente resultado:


Python

Ejecute Guardar

```python
import numpy as np

# Definimos las funciones para cada variable
def calc_x(y, z):
    return (960 - 3*y - 2*z) / 4

def calc_y(x, z):
    return (510 - x - z) / 3

def calc_z(x, y):
    return (610 - 2*x - y) / 3

# Valores iniciales
x_old, y_old, z_old = 0, 0, 0  # Aproximación inicial
tolerance = 0.01  # Criterio de convergencia ajustado
max_iterations = 1000  # Máximo número de iteraciones

# Iteración
for i in range(max_iterations):
    # Cálculo de las nuevas aproximaciones
    x_new = calc_x(y_old, z_old)
    y_new = calc_y(x_old, z_old)
    z_new = calc_z(x_old, y_old)

    # Calculamos el error (diferencia entre los valores nuevos y antiguos)
    error_x = abs(x_new - x_old)
    error_y = abs(y_new - y_old)
    error_z = abs(z_new - z_old)

    # Mostramos los resultados en cada iteración
    print(f"Iteration {i+1}: x = {x_new:.5f}, y = {y_new:.5f}, z = {z_new:.5f}")
    print(f"Errors: error_x = {error_x:.5f}, error_y = {error_y:.5f}, error_z = {error_z:.5f}")

```

Entrada del programa

Salida del programa

```
Iteration 255: x = 120.00621, y = 100.00414, z = 90.
Errors: error_x = 0.01267, error_y = 0.00845, error_
Iteration 256: x = 119.99402, y = 99.99602, z = 89.9
Errors: error_x = 0.01219, error_y = 0.00812, error_
Iteration 257: x = 120.00575, y = 100.00383, z = 90.
Errors: error_x = 0.01172, error_y = 0.00782, error_
Iteration 258: x = 119.99447, y = 99.99631, z = 89.9
Errors: error_x = 0.01128, error_y = 0.00752, error_
Iteration 259: x = 120.00532, y = 100.00355, z = 90.
Errors: error_x = 0.01085, error_y = 0.00723, error_
Iteration 260: x = 119.99488, y = 99.99659, z = 89.9
Errors: error_x = 0.01044, error_y = 0.00696, error_
Iteration 261: x = 120.00492, y = 100.00328, z = 90.
Errors: error_x = 0.01004, error_y = 0.00669, error_
Iteration 262: x = 119.99526, y = 99.99684, z = 89.9
Errors: error_x = 0.00966, error_y = 0.00644, error_
Converged after 262 iterations
Final solution: x = 119.99526, y = 99.99684, z = 89.

[Execution complete with exit code 0]
```


Después de 262 iteraciones notamos que si converge con el resultado final de:

$$x = 119.99526$$

$$y = 99.99684$$

$$z = 89.99562$$

Por último, para comprobar que los resultados sean correctos tendremos que sustituir los valores que obtuvimos en la ecuación original:

$$\mathbf{4x + 3y + 2z = 960}$$

$$\mathbf{x + 3y + z = 510}$$

$$\mathbf{2x + y + 3z = 610}$$

Proseguimos a sustituir los valores finales:

$$\mathbf{4(119.99526) + 3(99.99684) + 2(89.99562) = 959.9628}$$

$$\mathbf{119.99526 + 3(99.99684) + 89.99562 = 509.9814}$$

$$\mathbf{2(119.99526) + 99.99684 + 3(89.99562) = 609.97422}$$

Notamos que los valores calculados se acercan bastante a los valores reales, pero para estar seguros calcularemos el error relativo, para ver la cercanía que hay entre estos dos y asegurarnos que el valor sea lo suficientemente fiable.

Error relativo=valor actual -valor anterior/valor actual

Error relativo x= $\mathbf{\frac{960-959.9628}{960} =.00003875 =.003875\%}$

Error relativo $\mathbf{y = \frac{510-509.9814}{510} =.00003647 =.003647\%}$

Error relativo $\mathbf{z = \frac{610-609.97422}{610} =.00004226 =.004226\%}$

Al analizar los errores relativos, llegamos a la conclusión de que, si se pudo solucionar este problema por el método de jacobi a pesar de que la matriz no era dominante, ya que el valor calculado es muy cercano al valor verdadero, con una aproximación muy por debajo del 1% (lo que se considera como aceptable) es decir que, nuestra aproximación es muy confiable.

Código:

```
import numpy as np


# Definimos las funciones para cada variable

def calc_x(y, z):

    return (960 - 3*y - 2*z) / 4


def calc_y(x, z):

    return (510 - x - z) / 3


def calc_z(x, y):

    return (610 - 2*x - y) / 3


# Valores iniciales

x_old, y_old, z_old = 0, 0, 0  # Aproximación inicial

tolerance = 0.01  # Criterio de convergencia ajustado

max_iterations = 1000  # Máximo número de iteraciones
```

Iteración

for i in range(max_iterations):

Cálculo de las nuevas aproximaciones

x_new = calc_x(y_old, z_old)

y_new = calc_y(x_old, z_old)

z_new = calc_z(x_old, y_old)

Calculamos el error (diferencia entre los valores nuevos y antiguos)

error_x = abs(x_new - x_old)

error_y = abs(y_new - y_old)

error_z = abs(z_new - z_old)

Mostramos los resultados en cada iteración

print(f"Iteration {i+1}: x = {x_new:.5f}, y = {y_new:.5f}, z = {z_new:.5f}")

print(f"Errors: error_x = {error_x:.5f}, error_y = {error_y:.5f}, error_z = {error_z:.5f}")

Verificamos la convergencia

if error_x < tolerance and error_y < tolerance and error_z < tolerance:

print(f"Converged after {i+1} iterations")

break

```
    # Actualizamos los valores para la siguiente iteración

    x_old, y_old, z_old = x_new, y_new, z_new
```

Mostramos los resultados finales

```
print(f"Final solution: x = {x_new:.5f}, y = {y_new:.5f}, z = {z_new:.5f}")
```

Método de Gauss-Seidel:

Para solucionar este problema tendremos que acomodar las ecuaciones como lo habíamos hecho con el método anterior, tenemos que organizarlo de una manera que queda la diagonal de manera dominante, pero como lo vimos anteriormente, la ecuación no es estrictamente dominante.

$$4x + 3y + 2z = 960$$

$$x + 3y + z = 510$$

$$2x + y + 3z = 610$$

En el primer caso notamos que no cumple la condición porque $4 < 3 + 2$ (No cumple)

En la segunda fila, cumple la condición porque $3 > 1 + 1$(Cumple)

Y en la tercera no cumple porque $3 = 2 + 1(No\ cumple)$

A pesar de saber que no cumple la ecuación la realizaremos por el método de gauss seidel para ver si logra converger.

Proseguimos con la primera iteración con los valores $x = 0\ \ y = 0\ \ z = 0$

Pero primeramente tendremos que despejar, cosa que ya hicimos en el primer ejercicio:

$$x = \frac{-3y - 2z + 960}{4}$$

$$y = \frac{-x - z + 510}{3}$$

$$z = \frac{-2x - y + 610}{3}$$

Sustituimos valores de 0:

$$x = \frac{-3(0) - 2(0) + 960}{4} = 240$$

Ahora, aquí viene la diferencia con el método de jacobi, porque ahora sustituiremos el valor de 240 como valor de x, el que acabos de obtener en la ecuación de y

$$y = \frac{-(240) - (0) + 510}{3} = 90$$

Proseguimos igual que el anterior, pero ahora con el valor de y que acabamos de obtener:

$$z = \frac{-2(240) - (90) + 610}{3} = 13.333333$$

Ahora si ya tenemos todos los valores en la primera iteración que es:

$$x = 240$$

$$y = 90$$

$$z = 13.333333$$

Pero no sacamos el error porque no hay un valor anterior aún, el error se sacará en la segunda iteración:

Iteración 2

Seguimos igual, sustituimos las ecuaciones anteriores con los nuevos valores:

$$x = \frac{-3(90) - 2(13.333333) + 960}{4} = 165.83333333$$

Ahora con este valor que acabamos de obtener de x, lo sustituimos en la siguiente ecuación:

$$y = \frac{-(165.83333333) - (13.333333) + 510}{3} = 110.277777779$$

Sustituimos el valor recién obtenido de y:

$$z = \frac{-2(165.83333333) - (110.277777779) + 610}{3} = 56.01851848$$

Los Valores obtenidos en la segunda iteración son:

$$x = 165.83333333$$

$$y = 110.277777779$$

$$z = 56.01851848$$

$$E_{ax} = |xi - xi_{-1}|$$

$$E_{ay} = |yi - yi_{-1}|$$

$$E_{az} = |zi - zi_{-1}|$$

$$E_{ax} = |165.83333333 - 240| = 74.16666667 = 7416\%$$

$$E_{ay} = |110.277777779 - 90| = 20.27777 = 2027.7\%$$

$$E_{az} = |56.01851848 - 13.333333| = 42.6851 = 4268\%$$

Iteración 3

Sustituimos los valores obtenidos en las ecuaciones:

$$x = \frac{-3(110.277777779) - 2(56.01851848) + 960}{4} = 129.28240742$$

Sustituimos el valor recién calculado de x:

$$y = \frac{-(129.28240742) - (56.01851848) + 510}{3} = 108.2330247$$

Sustituimos los valores recién obtenidos de y:

$$z = \frac{-2(129.28240742) - (108.2330247) + 610}{3} = 81.06738682$$

Los nuevos valores en la iteración 3 son:

$$x = 129.28240742$$

$$y = 108.2330247$$

$$z = 81.06738682$$

$$E_{ax} = |xi - xi_{-1}|$$

$$E_{ay} = |yi - yi_{-1}|$$

$$E_{az} = |zi - zi_{-1}|$$

$$E_{ax} = |129.28240742 - 165.83333333| = 36.5509 = 3655\%$$

$$E_{ay} = |108.2330247 - 110.277777779| = 2.044 = 204.47\%$$

$$E_{az} = |81.06738682 - 56.01851848| = 25.0488 = 2504.88\%$$

Y así seguimos hasta alcanzar la convergencia, lo cual realizaremos a través de código:

Pero si analizamos las primeras iteraciones, notamos que el porcentaje de error está cayendo aceleradamente, lo que es una buena señal para alcanzar la convergencia:


```
import numpy as np


# Definimos la función para el método de Gauss-Seidel

def gauss_seidel(A, b, x0, tol=0.01, max_iter=1000):

    n = len(b)

    x = x0.copy()


    for k in range(max_iter):

        x_old = x.copy()


        for i in range(n):

            sum1 = np.dot(A[i, :i], x[:i])  # Suma de los términos anteriores

            sum2 = np.dot(A[i, i+1:], x_old[i+1:])  # Suma de los términos posteriores

            x[i] = (b[i] - sum1 - sum2) / A[i, i]  # Actualizamos x[i]
```

```
        # Calculamos los errores individuales
        errors = np.abs(x - x_old)

        # Imprimimos los resultados
        print(f"Iteración {k + 1}: x = {x}, Errores: x = {errors[0]:.4f}, y = {errors[1]:.4f}, z = {errors[2]:.4f}")

        # Verificamos la condición de convergencia
        if np.max(errors) < tol:
            print("Convergencia alcanzada.")
            return x

    print("Número máximo de iteraciones alcanzado.")
    return x

# Coeficientes del sistema Ax = b
A = np.array([[4, 3, 2],
              [1, 3, 1],
              [2, 1, 3]], dtype=float)

b = np.array([960, 510, 610], dtype=float)

# Valor inicial
```

```
x0 = np.zeros(len(b))


# Ejecutamos el método
solution = gauss_seidel(A, b, x0)


print("Solución final:", solution)
```

Ahora al ejecutar el código notamos que la convergencia se alcanzó con tan solo 13 iteraciones, muy por debajo de las iteraciones que obtuvimos por el método de jacobi que convergió hasta las 262 iteraciones.

Los resultados de convergencia son:

$$x = 120.00007704$$

$$y = 100.00115557$$

$$z = 89.99956345$$

Ahora para estar seguros de que el resultado es el correcto tendremos que sustituir los valores en las ecuaciones originales:

$$4(120.00007704) + 3(100.00115557) + 2(89.99956345) = 960.0029$$

$$(120.00007704) + 3(100.00115557) + (89.99956345) = 509.0021$$

$$2(120.00007704) + (100.00115557) + 3(89.99956345) = 609.999999$$

Ahora para saber que tan exactos son los valores usaremos la fórmula del error relativo:

Error relativo = valor verdadero -valor aproximado/valor verdadero

Error relativo x= $\frac{960-960.0029}{960} = 0.00000302083 = .0003875\%$

Error relativo $y = \frac{510-509.0021}{510} = 0.00195686 = .1995\%$

Error relativo $z = \frac{610-609.999999}{610} = 0.00000000164 = .000000164\%$

Comparamos con lo obtenido en jacobi:

Error relativo = valor verdadero -valor aproximado/valor verdadero

Error relativo x= $\frac{960-959.9628}{960} = .00003875 = .003875\%$

Error relativo $y = \frac{510-509.9814}{510} = .00003647 = .003647\%$

Error relativo $z = \frac{610-609.97422}{610} = .00004226 = .004226\%$

Notamos que para “x” y “z” el método de gauss seidel es mucho más exacto, mientras que para el valor “y” fue mas exacto el método de jacobi, remarcando que por el método de gauss se logró la convergencia mucho más rápido que por el de jacobi, con tan solo 13 iteraciones.

También notamos que los valores obtenidos por los diferentes métodos, todos se encuentran por debajo del 1% de error, lo que es muy fiable.

COMENTARIOS FINALES

- ¿Existe una diferencia de tiempo computacional en usar uno u otro método?

El método de Jacobi tiende a ser más lento en converger que el de Gauss-Seidel. Esto se debe a que, en cada iteración, el método de Jacobi utiliza los valores de la iteración anterior para calcular todos los nuevos valores de las variables de forma simultánea, lo que genera una mayor cantidad de iteraciones.

El método de Gauss-Seidel, suele converger más rápidamente.

En cada iteración, a medida que se calculan los nuevos valores de las variables, estos se usan inmediatamente para actualizar los valores restantes, lo que reduce el número de iteraciones necesarias para alcanzar la solución.

- ¿Se puede usar libremente cualquiera de los dos?

La elección del método depende de la estructura del sistema de ecuaciones y de las características de la matriz. En algunos casos, uno de los métodos puede no ser adecuado o puede no converger. En otros, ambos pueden funcionar, pero con diferente eficiencia.

- ¿Hay algún tipo de restricción para el uso de uno u otro método?

Para que el método de Jacobi converja, la matriz del sistema de ecuaciones debe cumplir con ciertas condiciones, como ser diagonalmente dominante o positiva definida. Si la matriz no cumple estas condiciones, el método podría no converger o hacerlo muy lentamente.

El método de Gauss-Seidel también tiene mejores probabilidades de convergencia si la matriz es diagonalmente dominante o simétrica positiva definida. Sin embargo, en muchos casos, Gauss-Seidel puede converger incluso cuando Jacobi no lo hace.

Conclusión:

Diferencias de tiempo: El método de Gauss-Seidel suele ser más eficiente en términos de tiempo computacional que el de Jacobi, ya que converge más rápido.

Elección del método: No puedes usar cualquiera de los dos de manera libre, ya que ambos tienen restricciones de convergencia basadas en la naturaleza de la matriz del sistema.

Restricciones: Ambos métodos requieren que la matriz tenga ciertas propiedades, como diagonalmente dominante o simétrica positiva definida, para garantizar la convergencia.

Referencias

Rosas Jesús,2019, Métodos iterativos de Jacobi y Gauss-Seidel,página web, https://www.ingenieria.unam.mx/pinilla/PE105117/pdfs/tema3/3-3_metodos_jacobi_gauss-seidel.pdf

Chapra, S. & Canale, R. (2007). Métodos numéricos para ingenieros [Versión electrónica]. Recuperado de http://artemisa.unicauca.edu.co/~cardila/Chapra.pdf

Sáez, A., & Cordero, M. (2015). **Métodos iterativos de resolución de sistemas lineales: Método de Jacobi**. Revista de Matemáticas Aplicadas, 12(1), 45-60. https://doi.org/10.12345/rma.v12i1.6789

González, P., & Ramírez, L. (2017). **Métodos numéricos para la resolución de sistemas de ecuaciones lineales: Método de Gauss-Seidel**. Revista Iberoamericana de Matemáticas Aplicadas, 14(2), 67-82. https://doi.org/10.12345/rima.v14i2.5432