Fundamentos de la Programación

y

Métodos Numéricos

Unidad Temática 3 (2024).

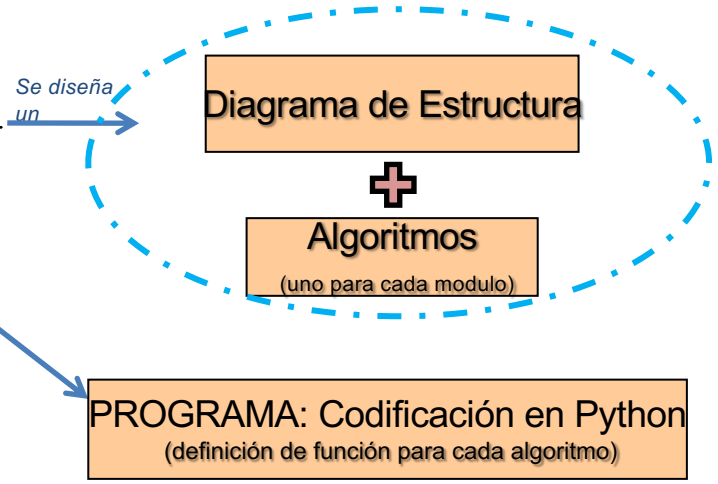
Ejercicio Practico de "Modelado de una pobracion de bacteias" aplicando los menuméricos para problemáticas de raíces de ecuaciones. Métodos cerrados (Método de bisección. Método de la Regla Falsa) y Métodos abiertos (Método de Newton-Raphson. Método de la Secante)

*Fases en la **resolución de un problema***

1. Definición del problema
2. Análisis del problema
3. Diseño del algoritmo
4. Codificación del algoritmo en un lenguaje
5. Análisis de los resultados

Análisis de Requerimientos:
¿**Qué** es lo que hay que hacer?

Diseño:
¿**Cómo** hacer lo especificado en la etapa de Análisis?

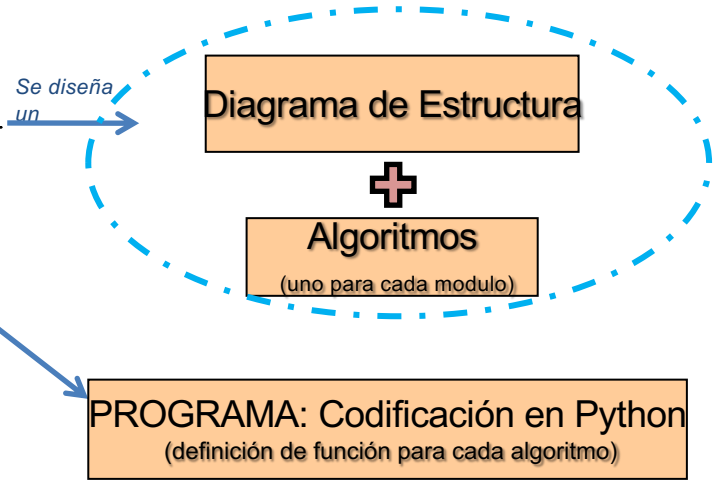


*Fases en la **resolución de un problema***

1. Definición del problema
2. Análisis del problema
3. Diseño del algoritmo
4. Codificación del algoritmo en un lenguaje
5. Análisis de los resultados

Análisis de Requerimientos:
¿**Qué** es lo que hay que hacer?

Diseño:
¿**Cómo** hacer lo especificado en la etapa de Análisis?



Modelado de una Población de Bacterias

Definición del Problema:

Se aplica un bactericida sobre un cultivo experimental. La efectividad del bactericida viene dada por la siguiente función:

$$B(t) = e^t - 3t$$

donde B(t) es el número de bacterias remanentes en el cultivo al instante t(s).

a) Resuelva el problema aplicando los métodos numéricos **de la bisección, regla falsa, Newton Raphson y de la Secante**. Para cada solución se requiere encontrar el instante de tiempo al cual **desaparecen totalmente** las bacterias con un **error aproximado menor que 0,001%.**

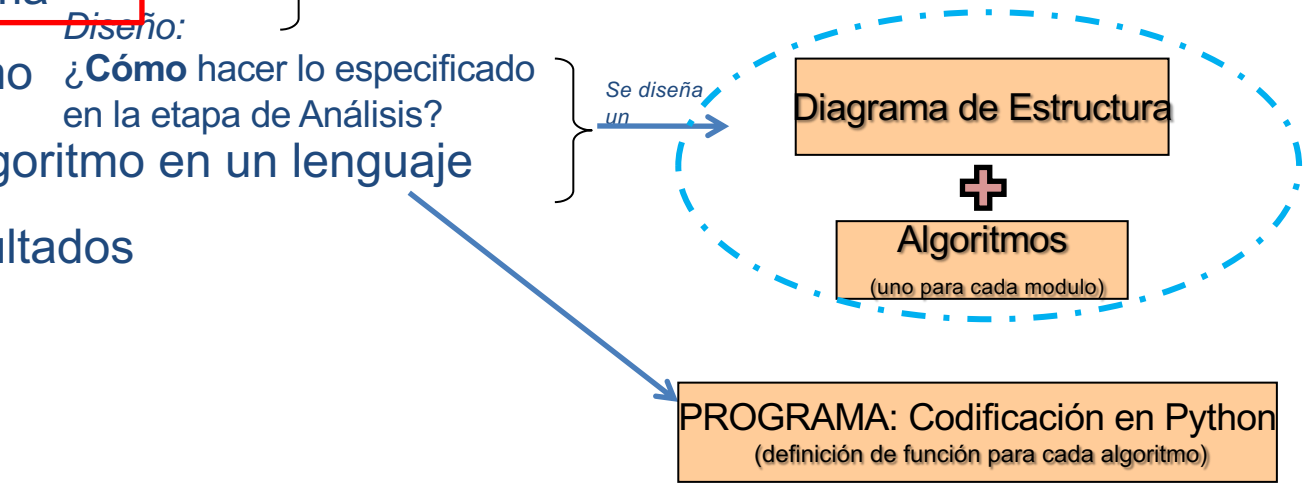
b) Con los resultados obtenidos en la aplicación de cada método evalúe la calidad de las soluciones obtenidas y compare el desempeño de cada método.

*Fases en la **resolución de un problema***

1. Definición del problema
2. Análisis del problema
3. Diseño del algoritmo
4. Codificación del algoritmo en un lenguaje
5. Análisis de los resultados

Análisis de Requerimientos:
¿**Qué** es lo que hay que hacer?

Diseño:
¿**Cómo** hacer lo especificado en la etapa de Análisis?



Analisis del Problema

La efectividad del bactericida viene dada por la siguiente función:

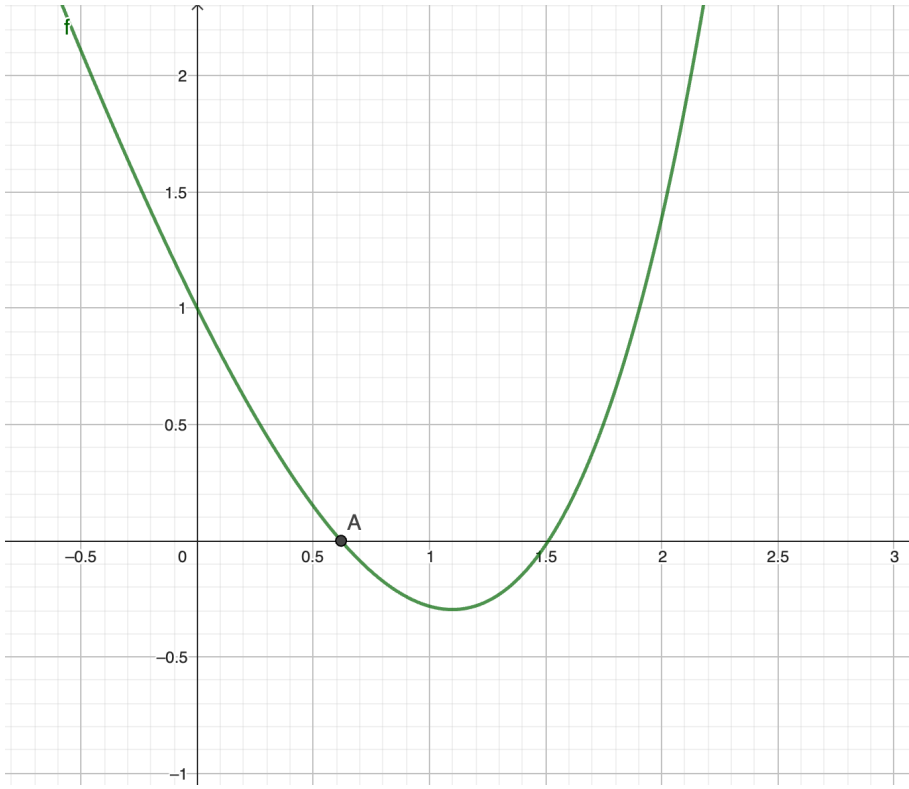
$$B(t) = e^t - 3t$$

Tiempo en el cual desaparecen totalmente

$$B(t) = 0$$

¿Cuál raiz tiene mas sentido con respect al contexto del problema?

Método Grafico



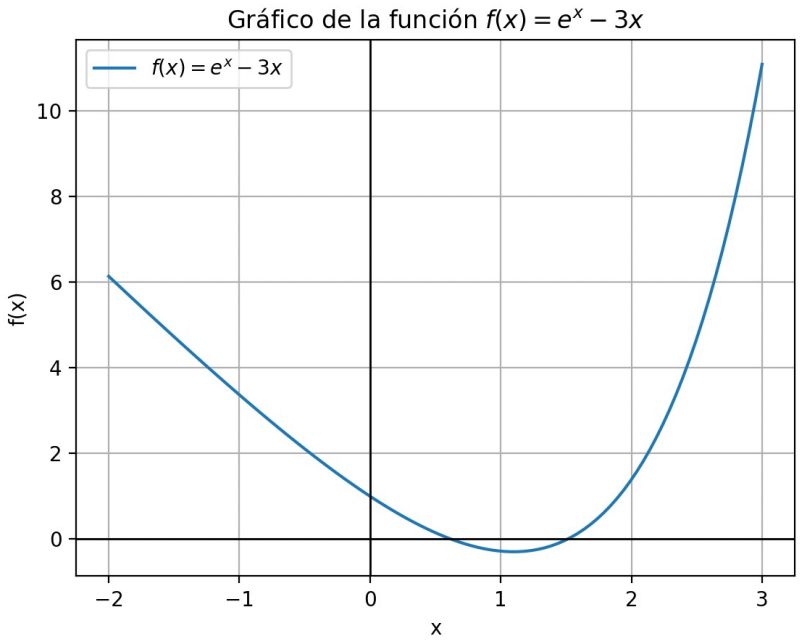
https://www.geogebra.org/classic?lang=es-AR

Analisis del Problema

Método Grafico utilizando Python:

```python
import numpy as np
import matplotlib.pyplot as plt
# ------------------------------------------------------------------------------
# FUNCION
# ------------------------------------------------------------------------------
def funcion(x):
    return np.exp(x) - 3 * x
# ------------------------------------------------------------------------------
# GRAFICAR
# ------------------------------------------------------------------------------
def graficar_funcion():
    # Crear un rango de valores para x
    x = np.linspace(-2, 3, 100)
    # Evaluar la función para cada valor de x
    y = funcion(x)
    # Crear la gráfica
    plt.plot(x, y, label=r'$f(x) = e^x - 3x$')
    plt.axhline(0, color='black',linewidth=1)  # Línea horizontal en y=0
    plt.axvline(0, color='black',linewidth=1)  # Línea vertical en x=0
    # Agregar etiquetas y título
    plt.xlabel('x')
    plt.ylabel('f(x)')
    plt.title('Gráfico de la función $f(x) = e^x - 3x$')
    # Mostrar la leyenda y la gráfica
    plt.legend()
    plt.grid(True)
    plt.show()

graficar_funcion()
```



Analisis del Problema

Criterio de corte de los métodos numéricos

"Error aproximado menor que 0,001%"
Donde $e_s = 0.001$ representa la tolerancia porcentual prefijada para el error.

Cifras significativas

$$e_s = (0.5\ x\ 10^{2-n})\%$$
$$0.001 = 0.5\ x\ 10^{2-n}$$
$$n \cong 5$$

Error relativo porcentual aproximado (e_a), donde x_r representa el valor aproximado actual y x_{r_prev} representa el valor aproximado previo:

$$e_a = \left|\frac{x_r - x_{r\_prev}}{x_r}\right| * 100\%$$

Vamos a seguir iterando o calcaludando un aproximado de una raiz mientras que: (cuando determinemos un e_a menor o igual a e_s debe dejar de iterar):

$$e_a > e_s \rightarrow e_a > 0.001$$

Analisis del Problema

Identificar con claridad las siguientes puntos:

Objetivo:

El objetivo es encontrar el instante de tiempo al cual desaparecen totalmente las bacterias.

¿Qué tratamiento (método) ha de realizarse a los datos de entrada para producir la salida deseada?

Se deben aplicar los métodos numéricos de la Bisección, Regla Falsa, Newton Raphson y de la Secante, comparando las soluciones obtenida.

Notar: que el enunciado especifica explícitamente que métodos a aplicar, a futuro esta decisión de que método numérico utilizar debe ser una decisión justificada en el análisis del problema.

Analisis del Problema

Elección del Método Numérico a utilizar:

Métodos Cerrados:

- Método de Bisección o de corte binario o de Bolzano
- Método de la Regla Falsa o Falsi o de la Falsa Posición

Métodos Abiertos:

- Método de Newton Raphson
- Método de la Secante

Analisis del Problema

Elección del Método Numérico a utilizar:

Métodos Cerrados:

- Método de Bisección o de corte binario o de Bolzano
- Método de la Regla Falsa o Falsi o de la Falsa Posición

Métodos Abiertos:

- Método de Newton Raphson
- Método de la Secante

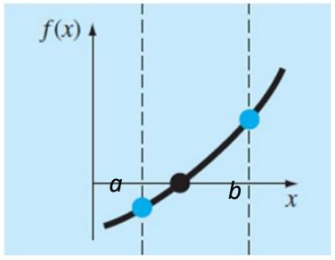
Diseño del Algoritmo

Metodo de la Biseccion

Consiste en dividir el intervalo a la mitad y localizar la mitad que contiene la raíz y volver a dividir el subintervalo hasta encontrar la raíz exacta o encontrar una raíz que se aproxime a la verdadera con un error por debajo de una cota de error establecida.

Se debe cumplir el Teorema de Bolzano

Teorema de Bolzano:
Si f:[a , b] es continua y f(a) y f(b) tienen signos diferentes, entonces f se anula en algún punto entre a y b. En este caso la unicidad de la solución puede garantizarse si la función es estrictamente creciente o decreciente.



f(x) continua en [a,b] y
el signo f(a) ≠ signo f(b)

$x_i = a,\ x_s = b,\ f(x_i) = f(a),\ f(x_s) = b$

$$x_r = \frac{x_i + x_s}{2}$$

- Si $f(x_i) \cdot f(x_r) < 0 \Rightarrow x_s = x_r$
- Si $f(x_i) \cdot f(x_r) > 0 \Rightarrow x_i = x_r$

Detenemos el proceso iterativo:

Si $f(x_i) \cdot f(x_r) = 0$ entonces x_r es la raíz

Si $\epsilon_a < \epsilon_s$, siendo ϵ_s conocido

Aprox. Actual

Diseño del Algoritmo (Metodo de la Bisección)

¿Qué entradas se requieren?

1. Error estimado (e_s)
2. Intervalo inicial (x_i)
3. Intervalo final (x_s)

¿Qué procesos se requieren?

1. Función ($B(t) = e^t - 3t$) --> calcular la función $B(t)$ para un valor de t dado
2. Calcular el proceso del algoritmo del método numérico de la Bisección

¿Cuál es la salida deseada?

1. El tiempo donde desaparecen las bacterias (x_r)
2. Error aproximado (e_a)
3. Numero de iteraciones (i)

$$\left|\frac{\text{Aprox Actual} - \text{Aprox Anterior}}{\text{Aprox. Actual}}\right| 100\%$$

Diseño del Algoritmo (Metodo de la Bisección)

¿Cuál es la algoritmo del método numérico de la Bisección?

1. **Entrada:**
   - Dos puntos iniciales xi (límite inferior) y xs (límite superior).
   - Un error estimado es (tolerancia).
   - Una función continua $f(x)$.
2. **Verificación Inicial del Intervalo:**
   - Si $f(xi) \times f(xs) > 0$, entonces el intervalo no contiene una raíz o contiene un número par de raíces, por lo que se debe seleccionar otro intervalo.
   - Si $f(xi) \times f(xs) < 0$, entonces el intervalo contiene al menos una raíz, y el proceso continúa.

Diseño del Algoritmo (Metodo de la Bisección)

¿Cuál es la algoritmo del método numérico de la Bisección?

3. **Inicialización:**
   - Establecer el error aproximado inicial $ea = 1$ (un valor grande).
   - Establecer el contador de iteraciones $i = 0$.
   - Inicializar $xr = xi$ (para calcular el punto medio del intervalo).
4. **Iteración (Repetir hasta que $ea \leq es$):**
   - a. Guardar el valor anterior de xr en xr_prev.
   - b. Calcular el nuevo punto medio xr como:

$$xr = \frac{xi + xs}{2}$$

   - c. Evaluar $f(xr)$:
     - Si $f(xr) = 0$, entonces xr es la raíz exacta, y el algoritmo termina.
     - Si $f(xi) \times f(xr) < 0$, la raíz está en el intervalo $[xi, xr]$, por lo que se establece $xs = xr$.
     - Si $f(xr) \times f(xs) < 0$, la raíz está en el intervalo $[xr, xs]$, por lo que se establece $xi = xr$.
   - d. **Calcular el error aproximado relativo ea si no es la primera iteración:**

$$ea = \left| \frac{xr - xr\_prev}{xr} \right| \times 100$$

   - e. Incrementar el contador de iteraciones i.

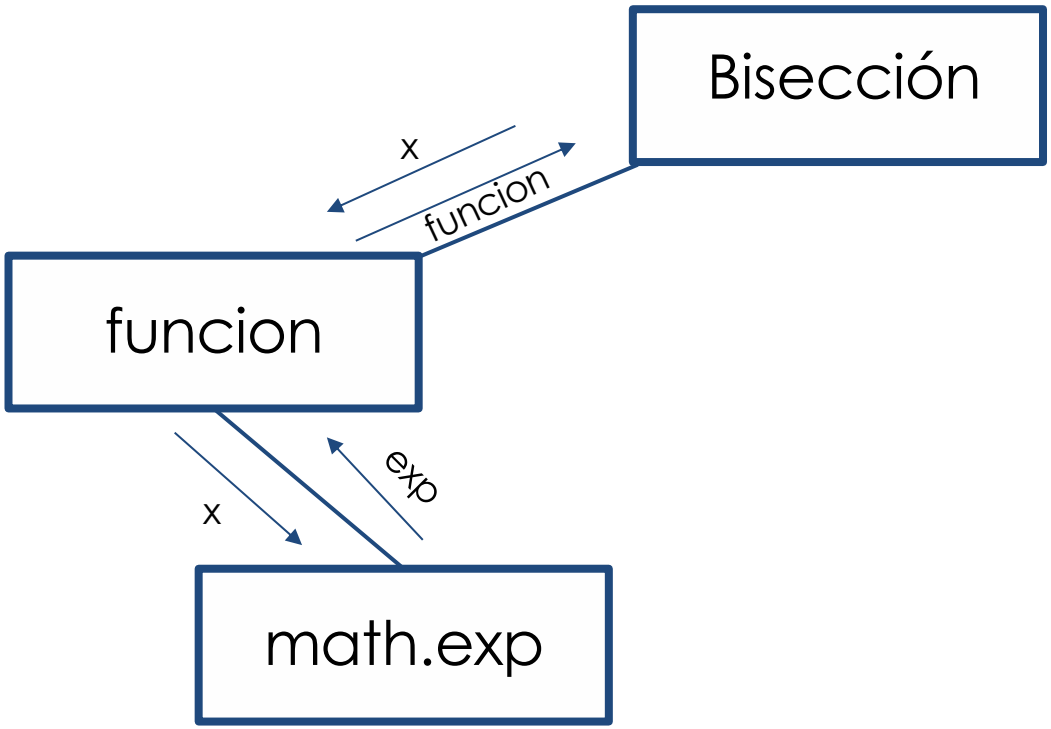
Diseño del Algoritmo (Metodo de la Bisección)

¿Cuál es la algoritmo del método numérico de la Bisección?

5. **Salida:**
   - El valor de xr, que es la aproximación de la raíz de $f(x)$.
   - El error aproximado ea.
   - El número de iteraciones i.

$$Ea = \frac{|\text{Aprox Actual} - \text{Aprox Anterior}|}{\text{Aprox. Actual}} 100\%$$

Diagrama de Estructura: (Método de la Bisección)



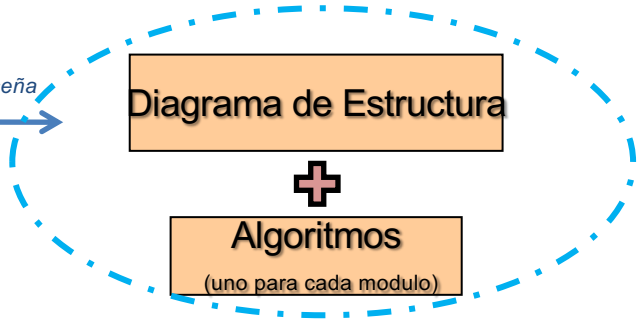
*Fases en la **resolución de un problema***

1. Definición del problema
2. Análisis del problema
3. Diseño del algoritmo
4. Codificación del algoritmo en un lenguaje
5. Análisis de los resultados

Análisis de Requerimientos:
¿**Qué** es lo que hay que hacer?

Diseño:
¿**Cómo** hacer lo especificado en la etapa de Análisis?

Se diseña un



PROGRAMA: Codificación en Python
(definición de función para cada algoritmo)

Codificación en Python (Método de la Bisección)

```
import math

# ------------------------------------------------------------------------
# ENTRADA
xi = float(input('Ingrese el valor del punto xi (límite inferior): '))
xs = float(input('Ingrese el valor del punto xs (límite superior): '))
es = float(input('Ingrese el error estimado: '))
# ------------------------------------------------------------------------
```

Codificación en Python (Método de la Bisección)

```
# ------------------------------------------------------------------------------------------
# PROCESO
# ------------------------------------------------------------------------------------------
def funcion(x):
    return math.exp(x) - 3 * x

if funcion(xi) * funcion(xs) > 0:
    print('Elegir otro intervalo, no contiene una raíz o contiene un número par de raíces')
else:
    print('El intervalo [' + str(xi) + ', ' + str(xs) + '] contiene por lo menos una raíz')

    ea = 1
    i = 0
    xr = xi
    while ea > es:
        xr_prev = xr
        xr = (xi + xs) / 2

        if funcion(xr) == 0:                # Si xr es la raíz exacta
            break
        elif funcion(xi) * funcion(xr) < 0:  # La raíz está en el intervalo [xi, xr]
            xs = xr
        else:                               # La raíz está en el intervalo [xr, xs]
            xi = xr

        # Calcular el error aproximado relativo si no es la primera iteración
        if i > 0:
            ea = abs((xr - xr_prev) / xr) * 100

        i += 1
# ------------------------------------------------------------------------------------------
```

Codificación en Python (Método de la Bisección)

```
# ------------------------------------------------------------------------------
# SALIDA
    print('xr: ', xr)
    print('Error aproximado: ', ea)
    print('Número de iteraciones: ', i)
# ------------------------------------------------------------------------------
```

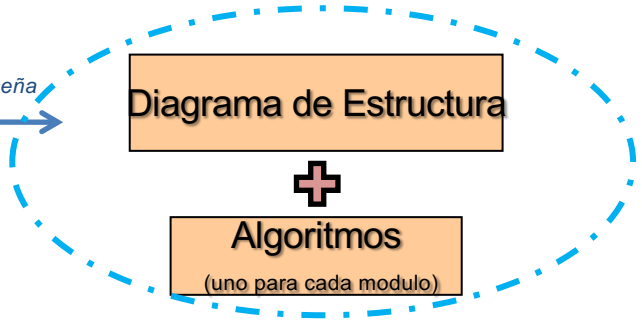
*Fases en la **resolución de un problema***

1. Definición del problema
2. Análisis del problema
3. Diseño del algoritmo
4. Codificación del algoritmo en un lenguaje
5. Análisis de los resultados

Análisis de Requerimientos:
¿**Qué** es lo que hay que hacer?

Diseño:
¿**Cómo** hacer lo especificado en la etapa de Análisis?

Se diseña un



PROGRAMA: Codificación en Python
(definición de función para cada algoritmo)

Analisis de Resultados (Metodo de la Bisección)

¿Es correcto informar lo resultados solo de esta forma?

```
Ingrese el valor del punto xi (límite inferior):
0
Ingrese el valor del punto xs (límite superior):
1
Ingrese el error estimado:
0.001
El intervalo [0.0, 1.0] contiene por lo menos una raíz
xr:  0.6190605163574219
Error aproximado:  0.0006162074893858259
Número de iteraciones:  18
```

Analisis de Resultados (Metodo de la Bisección)

¿Es correcto informar lo resultados solo de esta forma? --> NO

```
Ingrese el valor del punto xi (límite inferior):
0
Ingrese el valor del punto xs (límite superior):
1
Ingrese el error estimado:
0.001
El intervalo [0.0, 1.0] contiene por lo menos una raíz
xr:  0.6190605163574219
Error aproximado:  0.0006162074893858259
Número de iteraciones:  18
```

Ante la aplicación del método numérico de la Bisección se obtuvo el valor de la raíz aproximada 0.61906 (5 cifras significativas) en base a las entradas del intervalo [0.0, 1.0] y con una la tolerancia porcentual prefijada para el error 0.001. El método numérico necesito 18 iteraciones para obtener el valor de la raíz y con un error relativo porcentual aproximado final de 0.000616207. Por lo tanto, a los 0.61906 segundos desaparecen totalmente las bacterias.

Analisis del Problema

Elección del Método Numérico a utilizar:

Métodos Cerrados:

- Método de Bisección o de corte binario o de Bolzano
- Método de la Regla Falsa o Falsi o de la Falsa Posición

Métodos Abiertos:

- Método de Newton Raphson
- Método de la Secante

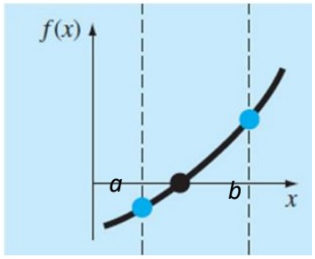
Diseño del Algoritmo

Método de la Regla Falsa

Igual que el método de la Bisección, este método aprovecha el Teorema de Bolzano y las magnitudes de la función en los extremos de los intervalos en los que se va dividiendo.

Consiste en dividir el intervalo y localizar la mitad que contiene la raíz y volver a dividir el subintervalo hasta encontrar la raíz exacta o encontrar una raíz que se aproxime a la verdadera con un error por debajo de una cota de error establecida.

Teorema de Bolzano:
Si f:[a , b] es continua y f(a) y f(b) tienen signos diferentes, entonces f se anula en algún punto entre a y b. En este caso la unicidad de la solución puede garantizarse si la función es estrictamente creciente o decreciente.



f(x) continua en $[x_i , x_s]$ y el signo $f(x_i) \neq$ signo $f(x_s)$

$$x_r = x_s - \frac{f(x_s)(x_i - x_s)}{f(x_i) - f(x_s)}$$

- Si $f(x_i) \cdot f(x_r) < 0 \Rightarrow x_s = x_r$
- Si $f(x_i) \cdot f(x_r) > 0 \Rightarrow x_i = x_r$

Detenemos el proceso iterativo:

Si $f(x_i) \cdot f(x_r) = 0$ entonces x_r es la raíz

ó Si $\epsilon_a < \epsilon_s$, siendo ϵ_s conocido

Diseño del Algoritmo (Método de la Regla Falsa)

¿Qué entradas se requieren?

1. Error estimado (e_s)
2. Intervalo inicial (x_i)
3. Intervalo final (x_s)

¿Qué procesos se requieren?

1. Función ($B(t) = e^t - 3t$) --> calcular la función $B(t)$ para un valor de t dado
2. Calcular el proceso del algoritmo del método numérico de la Regla Falsa

¿Cuál es la salida deseada?

1. El tiempo donde desaparecen las bacterias (x_r)
2. Error aproximado (e_a)
3. Numero de iteraciones (i)

$$\left|\frac{\text{Aprox Actual} - \text{Aprox Anterior}}{\text{Aprox. Actual}}\right| 100\%$$

Se tienen las misma entradas, estructura de proceso y salidas que el método de la Bisección.

Diseño del Algoritmo (Método de la Regla Falsa)

¿Cuál es la algoritmo del método numérico de la Regla Falsa?

1. **Entrada:**
   - Dos puntos iniciales xi (límite inferior) y xs (límite superior).
   - Un error estimado es (tolerancia).
   - Una función continua $f(x)$.
2. **Verificación Inicial del Intervalo:**
   - Si $f(xi) \times f(xs) > 0$, entonces el intervalo no contiene una raíz o contiene un número par de raíces, por lo que se debe seleccionar otro intervalo.
   - Si $f(xi) \times f(xs) < 0$, entonces el intervalo contiene al menos una raíz, y el proceso continúa.

$$Ea = \frac{|\text{Aprox. Actual} - \text{Aprox. Anterior}|}{\text{Aprox. Actual}} 100\%$$

Se tienen las misma entradas que el método de la Bisección.

Diseño del Algoritmo (Método de la Regla Falsa)

¿Cuál es la algoritmo del método numérico de la Regla Falsa?

3. **Inicialización:**
   - Establecer el error aproximado inicial $ea = 1$ (un valor grande).
   - Establecer el contador de iteraciones $i = 0$.
   - Inicializar $xr = xi$ (valor inicial para la raíz aproximada).
4. **Iteración (Repetir hasta que $ea \leq es$):**
   - a. Guardar el valor anterior de xr en xr_prev.
   - b. Calcular la nueva raíz aproximada xr usando la fórmula de la Regla Falsa:

$$xr = xs - \frac{f(xs) \times (xi - xs)}{f(xi) - f(xs)}$$

   - c. Evaluar $f(xr)$:
     - Si $f(xr) = 0$, entonces xr es la raíz exacta, y el algoritmo termina.
     - Si $f(xi) \times f(xr) < 0$, la raíz está en el intervalo $[xi, xr]$, por lo que se establece $xs = xr$.
     - Si $f(xr) \times f(xs) < 0$, la raíz está en el intervalo $[xr, xs]$, por lo que se establece $xi = xr$.
   - d. **Calcular el error aproximado relativo ea si no es la primera iteración:**

$$ea = \left| \frac{xr - xr\_prev}{xr} \right| \times 100$$

   - e. Incrementar el contador de iteraciones i.

Se tienen la misma estructura del método de la Bisección solo cambia:

Diseño del Algoritmo (Método de la Regla Falsa)

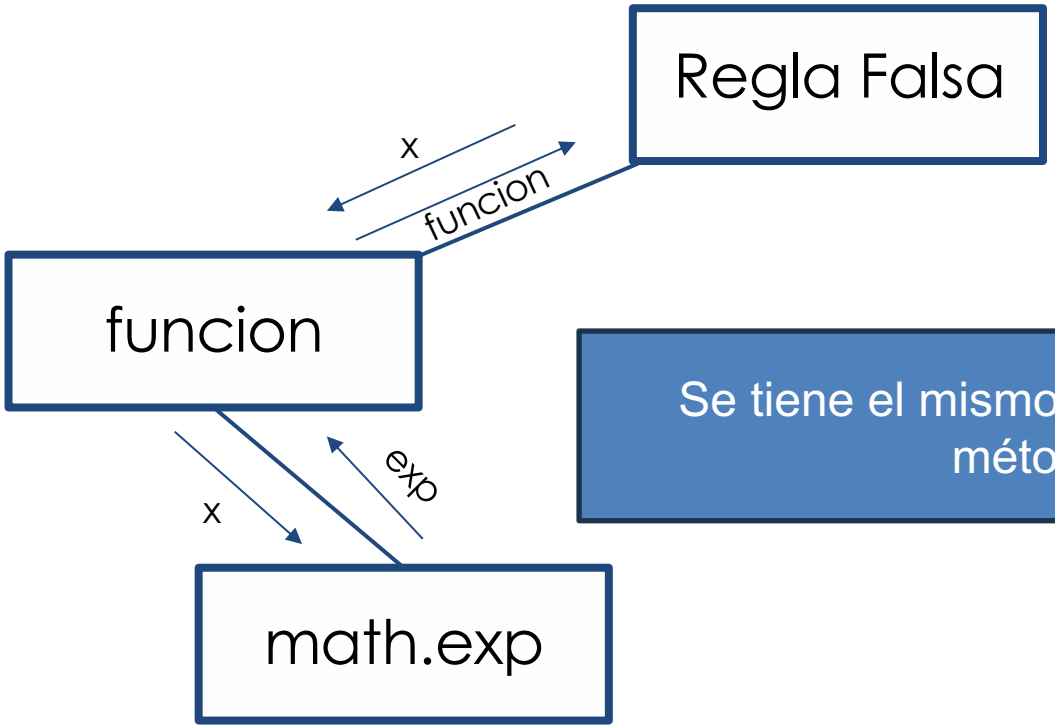
¿Cuál es la algoritmo del método numérico de la Regla Falsa?

5. **Salida:**
   - El valor de xr, que es la aproximación de la raíz de $f(x)$.
   - El error aproximado ea.
   - El número de iteraciones i.

$$Ea = \frac{|\text{Aprox Actual} - \text{Aprox Anterior}|}{\text{Aprox. Actual}} 100\%$$

Se tienen las misma salidas que el método de la Bisección.

Diagrama de Estructura: (Metodo de la Regla Falsa)



Se tiene el mismo diagrama de estructuras que el método de la Bisección.

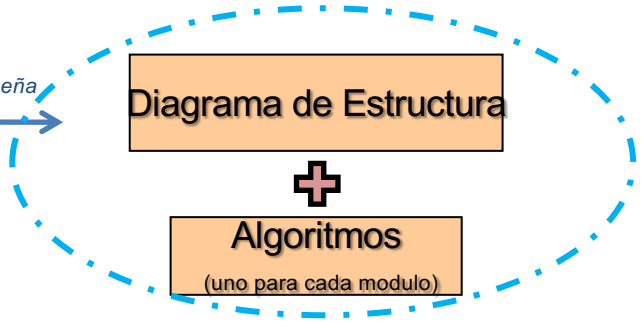
*Fases en la **resolución de un problema***

1. Definición del problema
2. Análisis del problema
3. Diseño del algoritmo
4. Codificación del algoritmo en un lenguaje
5. Análisis de los resultados

Análisis de Requerimientos:
¿**Qué** es lo que hay que hacer?

Diseño:
¿**Cómo** hacer lo especificado en la etapa de Análisis?

Se diseña un



PROGRAMA: Codificación en Python
(definición de función para cada algoritmo)

Codificación en Python : (Método de la Regla Falsa)

```python
import math

# ------------------------------------------------------------------------------
# ENTRADA
xi = float(input('Ingrese el valor del punto xi (límite inferior): '))
xs = float(input('Ingrese el valor del punto xs (límite superior): '))
es = float(input('Ingrese el error estimado: '))
# ------------------------------------------------------------------------------
```

Codificación en Python : (Método de la Regla Falsa)

```
# ------------------------------------------------------------------------------------------
# PROCESO
# ------------------------------------------------------------------------------------------
def funcion(x):
    return math.exp(x) - 3 * x

if funcion(xi) * funcion(xs) > 0:
    print('Elegir otro intervalo, no contiene una raíz o contiene un número par de raíces')
else:
    print('El intervalo [' + str(xi) + ', ' + str(xs) + '] contiene por lo menos una raíz')

    ea = 1
    i = 0
    xr = xi
    while ea > es:
        xr_prev = xr
        xr = xs - (funcion(xs) * (xi - xs)) / (funcion(xi) - funcion(xs))

        if funcion(xr) == 0:                # Si xr es la raíz exacta
            break
        elif funcion(xi) * funcion(xr) < 0:  # La raíz está en el intervalo [xi, xr]
            xs = xr
        else:                               # La raíz está en el intervalo [xr, xs]
            xi = xr

        # Calcular el error aproximado relativo si no es la primera iteración
        if i > 0:
            ea = abs((xr - xr_prev) / xr) * 100

        i += 1
# ------------------------------------------------------------------------------------------
```

Codificación en Python : (Método de la Regla Falsa)

```
# ------------------------------------------------------------------------------
# SALIDA
    print('xr: ', xr)
    print('Error aproximado: ', ea)
    print('Número de iteraciones: ', i)
# ------------------------------------------------------------------------------
```

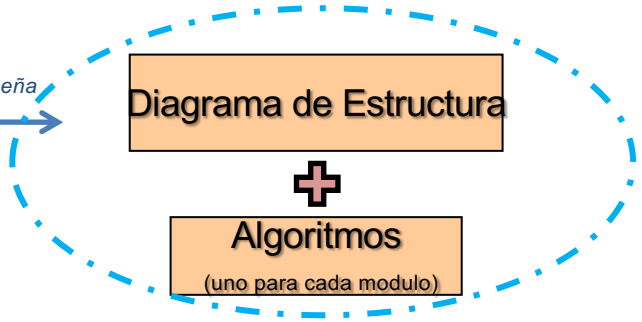
*Fases en la **resolución de un problema***

1. Definición del problema
2. Análisis del problema
3. Diseño del algoritmo
4. Codificación del algoritmo en un lenguaje
5. Análisis de los resultados

Análisis de Requerimientos:
¿**Qué** es lo que hay que hacer?

Diseño:
¿**Cómo** hacer lo especificado en la etapa de Análisis?

Se diseña un



PROGRAMA: Codificación en Python
(definición de función para cada algoritmo)

Analisis de Resultados (Método de la Regla Falsa)

```
Ingrese el valor del punto xi (límite inferior):
0
Ingrese el valor del punto xs (límite superior):
1
Ingrese el error estimado:
0.001
El intervalo [0.0, 1.0] contiene por lo menos una raíz
xr:  0.6190621954594939
Error aproximado:  0.00035500978729497 52
Número de iteraciones:  11
```

Ante la aplicación del método numérico de la Regla Falsa se obtuvo el valor de la raíz aproximada 0.61906 (5 cifras significativas) en base a las entradas del intervalo [0.0, 1.0] y con una la tolerancia porcentual prefijada para el error 0.001. El método numérico necesito 11 iteraciones para obtener el valor de la raíz y con un error relativo porcentual aproximado final de 0.00035500. Por lo tanto, a los 0.61906 segundos desaparecen totalmente las bacterias.

Analisis del Problema

Elección del Método Numérico a utilizar:

Métodos Cerrados:

- Método de Bisección o de corte binario o de Bolzano
- Método de la Regla Falsa o Falsi o de la Falsa Posición

Métodos Abiertos:

- Método de Newton Raphson
- Método de la Secante

Diseño del Algoritmo

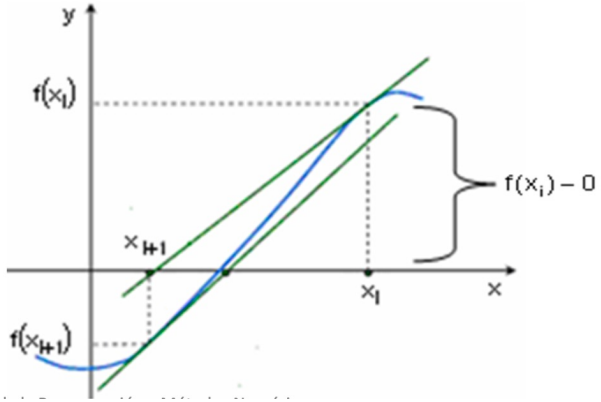
Método de Newton Raphson

Este método iterativo parte de una estimación inicial y utiliza la derivada de la función para mejorar esa estimación, convergiendo hacia la raíz con rapidez. Es especialmente útil cuando se dispone de la derivada de la función y se busca alta precisión.

A partir de: $f(x) = 0$ *y* $x_0 inicial$

Ecuación iterativa $x_{i+1} = f(x_i)$?

Se considera x_{i+1} como el punto en que la recta tangente a $f(x_i)$ corta al eje de las abscisas.



Diseño del Algoritmo (Método de Newton-Raphson)

¿Qué entradas se requieren?

1. Error estimado (e_s)
2. Intervalo inicial (x_i)

¿Qué procesos se requieren?

1. Función ($B(t) = e^t - 3t$) --> calcular la función $B(t)$ para un valor de t dado
2. Función de la derivada de ($B(t)$→ $e^t - 3$) → calcular la función $B(t)$ para un valor de t dado
3. Calcular el proceso del algoritmo del método numérico de Newton-Raphson

¿Cuál es la salida deseada?

1. El tiempo donde desaparecen las bacterias (x_r)
2. Error aproximado (e_a)
3. Numero de iteraciones (i)

Diseño del Algoritmo (Método de Newton-Raphson)

¿Cuál es la algoritmo del método numérico de la Newton-Raphson?

1. **Entrada:**
   - Un punto inicial xi (aproximación inicial de la raíz).
   - Un error estimado es (tolerancia).
   - Una función $f(x)$ y su derivada $f'(x)$.

Diseño del Algoritmo (Método de Newton-Raphson)

¿Cuál es la algoritmo del método numérico de la Newton-Raphson?

3. **Iteración (Repetir hasta que $ea \leq es$ o $f(xr) = 0$):**
   - a. Guardar el valor anterior de xr en xra.
   - b. Calcular la nueva raíz aproximada xr usando la fórmula de Newton-Raphson:

$$xr = xr - \frac{f(xr)}{f'(xr)}$$

   - c. Incrementar el contador de iteraciones i.
   - d. **Calcular el error aproximado relativo ea si no es la primera iteración:**

$$ea = \left| \frac{xr - xra}{xr} \right| \times 100$$

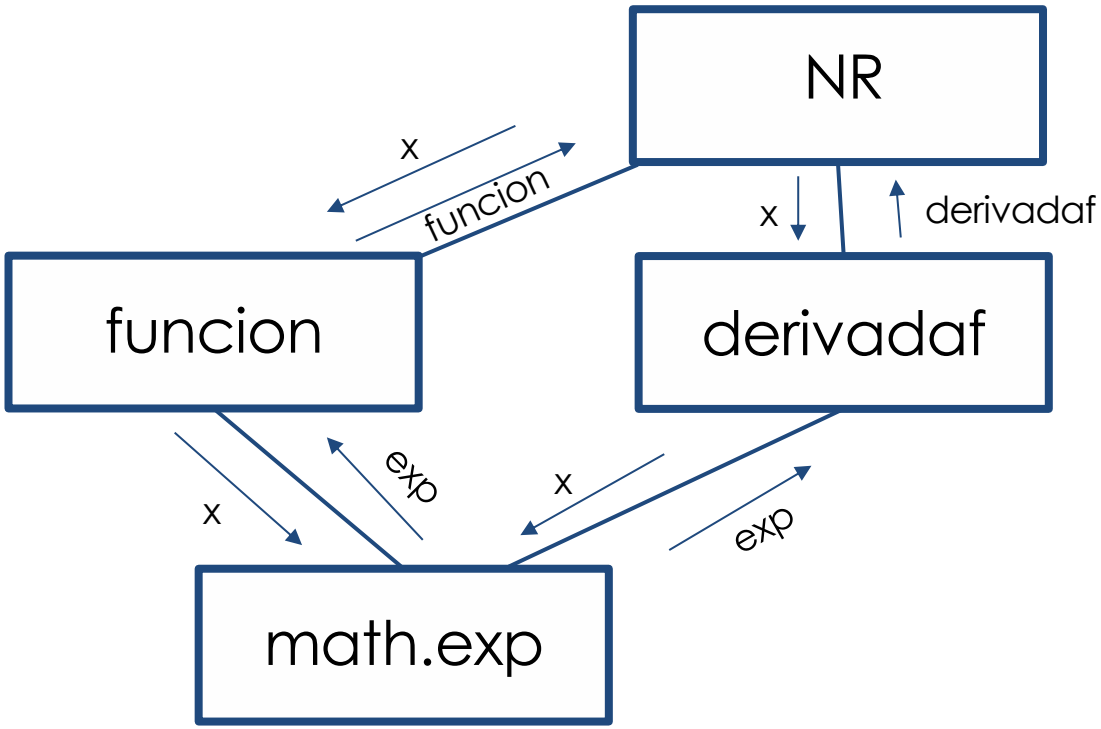
4. **Condición de Terminación:**
   - El proceso continúa hasta que el error ea sea menor o igual que la tolerancia es o hasta que $f(xr) = 0$, lo que indica que se ha encontrado la raíz exacta.

Diseño del Algoritmo (Método de Newton-Raphson)

¿Cuál es la algoritmo del método numérico de la Newton-Raphson?

5. **Salida:**
   - El valor de xr, que es la aproximación de la raíz de $f(x)$.
   - El error aproximado ea.
   - El número de iteraciones i.

Diagrama de Estructura: Newton-Raphson



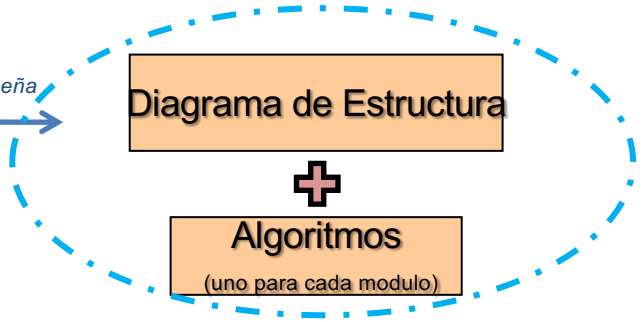
*Fases en la **resolución de un problema***

1. Definición del problema
2. Análisis del problema
3. Diseño del algoritmo
4. Codificación del algoritmo en un lenguaje
5. Análisis de los resultados

Análisis de Requerimientos:
¿**Qué** es lo que hay que hacer?

Diseño:
¿**Cómo** hacer lo especificado en la etapa de Análisis?

Se diseña un



PROGRAMA: Codificación en Python
(definición de función para cada algoritmo)

Codificación en Python : (Método de Newton-Raphson)

```
# ------------------------------------------------------------------------------
# ENTRADA
xi = float(input('Ingrese el valor del punto inicial xi: '))
es = float(input('Ingrese el error estimado: '))
# ------------------------------------------------------------------------------
```

Codificación en Python : (Método de Newton-Raphson)

```
# ------------------------------------------------------------------------------
# PROCESO
# ------------------------------------------------------------------------------
def funcion(x):
    return math.exp(x) - 3 * x

def derivadaf(x):
    return math.exp(x) - 3

ea = 1
i = 0
xr = xi

while funcion(xr) != 0 and ea > es:
    xra = xr

    xr = xr - funcion(xr) / derivadaf(xr)

    i += 1

    if i > 1:
        ea = abs((xr - xra) / xr) * 100
```

Codificación en Python : (Método de Newton-Raphson)

```
# ------------------------------------------------------------------------------
# SALIDA
print('xr: ', xr)
print('Error aproximado: ', ea)
print('Número de iteraciones: ', i)
# ------------------------------------------------------------------------------
```

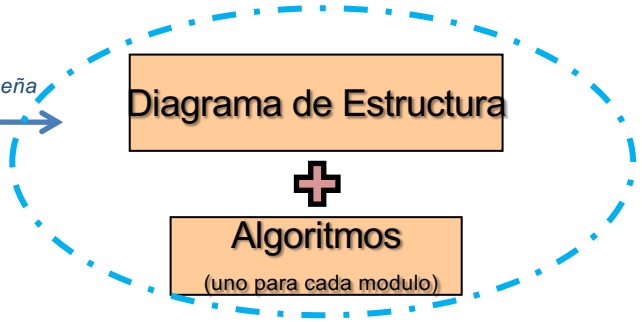
*Fases en la **resolución de un problema***

1. Definición del problema
2. Análisis del problema
3. Diseño del algoritmo
4. Codificación del algoritmo en un lenguaje
5. Análisis de los resultados

Análisis de Requerimientos:
¿**Qué** es lo que hay que hacer?

Diseño:
¿**Cómo** hacer lo especificado en la etapa de Análisis?

Se diseña un



PROGRAMA: Codificación en Python
(definición de función para cada algoritmo)

Analisis de Resultados (Método de Newton-Raphson)

```
Ingrese el valor del punto inicial xi:
1
Ingrese el error estimado:
0.001
xr:  0.6190612867359452
Error aproximado:  5.460901065995188e-07
Número de iteraciones:  6
```

Ante la aplicación del método numérico de la Newton-Raphson se obtuvo el valor de la raíz aproximada 0.61906 (5 cifras significativas) en base a la entrada del valor inicial igual a 1.0 y con una la tolerancia porcentual prefijada para el error 0.001. El método numérico necesito 6 iteraciones para obtener el valor de la raíz y con un error relativo porcentual aproximado final de 5.4609 x 10^-07. Por lo tanto, a los 0.61906 segundos desaparecen totalmente las bacterias.

Analisis del Problema

Elección del Método Numérico a utilizar:

Métodos Cerrados:

- Método de Bisección o de corte binario o de Bolzano
- Método de la Regla Falsa o Falsi o de la Falsa Posición

Métodos Abiertos:

- Método de Newton Raphson
- Método de la Secante

Diseño del Algoritmo

Método de la Secante

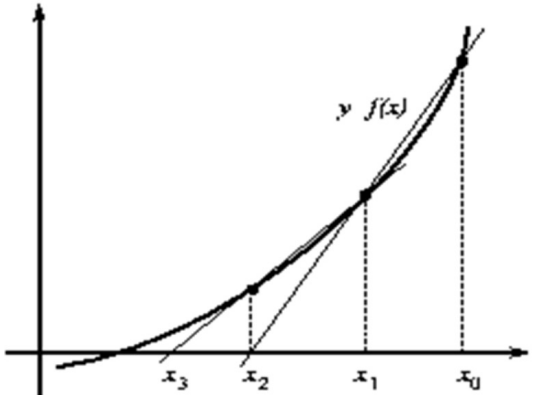
A diferencia del método de Newton-Raphson, no requiere el cálculo de la derivada, sino que utiliza dos aproximaciones iniciales para estimar la pendiente de la función. Con esas aproximaciones, mejora sucesivamente la estimación de la raíz.

A partir de: $f(x)=0$ *y* $x_0, x_1 \; inicial$

Ecuación iterativa

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

Se considera x_{i+1} como el punto en que la **recta secante** a f que pasa por los puntos $(x_0, f(x_0))$ y $(x_1, f(x_1))$ corta al eje de las abscisas



Diseño del Algoritmo (Método de la Secante)

¿Qué entradas se requieren?

1. Error estimado (e_s)
2. Intervalo inicial (x_i)
3. Intervalo final (x_s)

¿Qué procesos se requieren?

1. Función ($B(t) = e^t - 3t$) --> calcular la función $B(t)$ para un valor de t dado
2. Calcular el proceso del algoritmo del método numérico de la Secante

¿Cuál es la salida deseada?

1. El tiempo donde desaparecen las bacterias (x_r)
2. Error aproximado (e_a)
3. Numero de iteraciones (i)

$$\left|\frac{\text{Aprox Actual} - \text{Aprox Anterior}}{\text{Aprox Actual}}\right| 100\%$$

Se tienen las misma entradas, estructura de proceso y salidas que el método de la Bisección y de la Regla Falsa.

Diseño del Algoritmo (Método de la Secante)

¿Cuál es la algoritmo del método numérico de la Secante?

Entrada:

1. Leer xi: Primer punto inicial.
2. Leer xs: Segundo punto inicial.
3. Leer es: Error estimado.

$$x\,10^{2-n} = 0.001$$
$$n \cong 5$$

$$Ea = \frac{|\text{Aprox Actual} - \text{Aprox Anterior}|}{\text{Aprox. Actual}} 100\%$$

Se tienen las misma entradas que el método de la Bisección y de la Regla Falsa.

Diseño del Algoritmo (Método de la Secante)

¿Cuál es la algoritmo del método numérico de la Secante?

Proceso:

1. Definir la función $f(x)$ como $f(x) = e^x - 3x$.
2. Inicializar:
   - ea (error aproximado) en 1 (valor inicial alto).
   - i (contador de iteraciones) en 0.
   - $xr = xi$ (inicialización de xr).
3. Mientras $ea > es$:
   1. Guardar el valor anterior de xr en xra.
   2. Calcular la nueva aproximación de la raíz usando la fórmula de la secante:
      $$xr = xs - \frac{f(xs) \times (xi - xs)}{f(xi) - f(xs)}$$
   3. Incrementar i en 1.
   4. Si $i > 1$, calcular el error aproximado relativo:
      $$ea = \left| \frac{xr - xra}{xr} \right| \times 100$$
   5. Actualizar los valores de xi y xs:
      - Si $f(xi) \times f(xr) < 0$, la raíz está en el intervalo [xi, xr], por lo tanto, actualizar $xs = xr$.
      - De lo contrario, actualizar $xi = xr$.

Diseño del Algoritmo (Método de la Secante)

¿Cuál es la algoritmo del método numérico de la Secante?

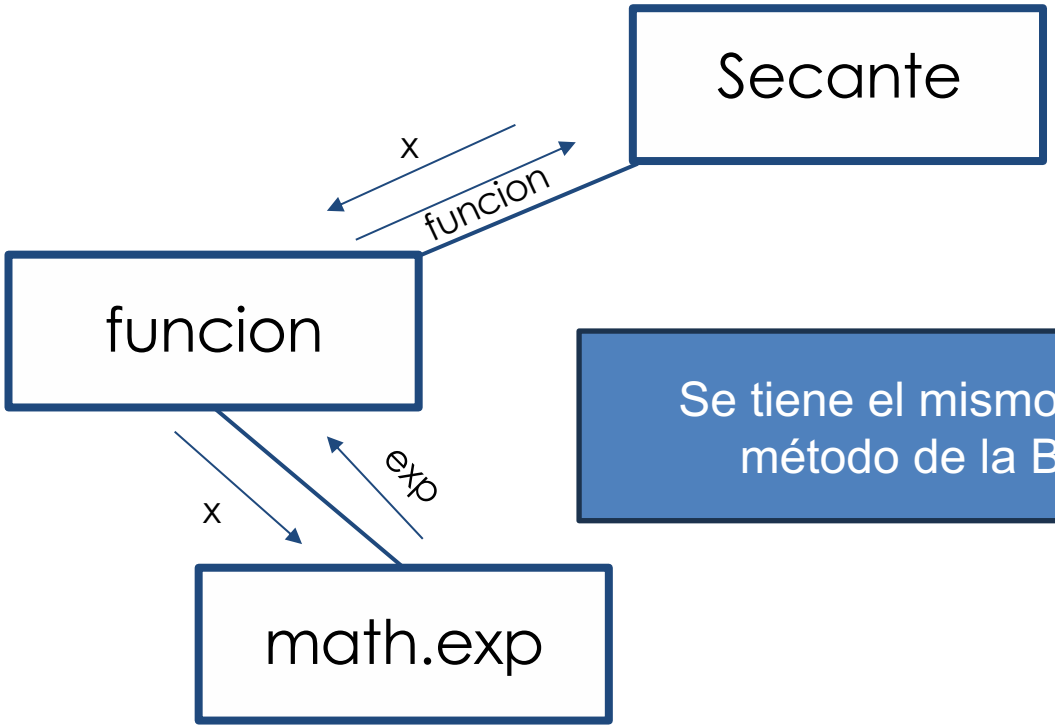
Salida:

1. Mostrar el valor de la raíz aproximada xr.
2. Mostrar el valor del error aproximado ea.
3. Mostrar el número de iteraciones i.

$$x\,10^{2-n} = 0.001$$
$$n \cong 5$$

Se tienen las misma salidas que el método de la Bisección y de la Regla Falsa.

Diagrama de Estructura: (Método de la Secante)



Se tiene el mismo diagrama de estructuras que el método de la Bisección y de la Regla Falsa.

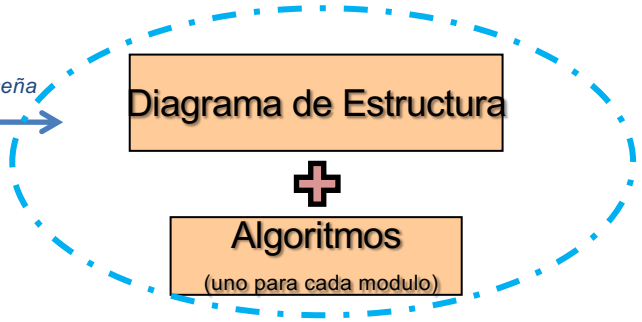
*Fases en la **resolución de un problema***

1. Definición del problema
2. Análisis del problema
3. Diseño del algoritmo
4. Codificación del algoritmo en un lenguaje
5. Análisis de los resultados

Análisis de Requerimientos:
¿**Qué** es lo que hay que hacer?

Diseño:
¿**Cómo** hacer lo especificado en la etapa de Análisis?

Se diseña un



PROGRAMA: Codificación en Python
(definición de función para cada algoritmo)

Codificación en Python : (Método de Secante)

```python
import math

# ------------------------------------------------------------------------------
# ENTRADA
xi = float(input('Ingrese el valor del primer punto xi: '))
xs = float(input('Ingrese el valor del segundo punto xs: '))
es = float(input('Ingrese el error estimado: '))
# ------------------------------------------------------------------------------
```

Codificación en Python : (Método de la Secante)

```
# ------------------------------------------------------------------------------
# PROCESO
# ------------------------------------------------------------------------------
def funcion(x):
    return math.exp(x) - 3 * x

ea = 1
i = 0
xr = xi

while ea > es:
    xra = xr

    xr = xs - (funcion(xs) * (xi - xs)) / (funcion(xi) - funcion(xs))

    i += 1

    if i > 1:
        ea = abs((xr - xra) / xr) * 100

    if funcion(xi) * funcion(xr) < 0:
        xs = xr
    else:
        xi = xr
```

Codificación en Python : (Método de la Secante)

```
# ------------------------------------------------------------------------------
# SALIDA
print('xr: ', xr)
print('Error aproximado: ', ea)
print('Número de iteraciones: ', i)
# ------------------------------------------------------------------------------
```

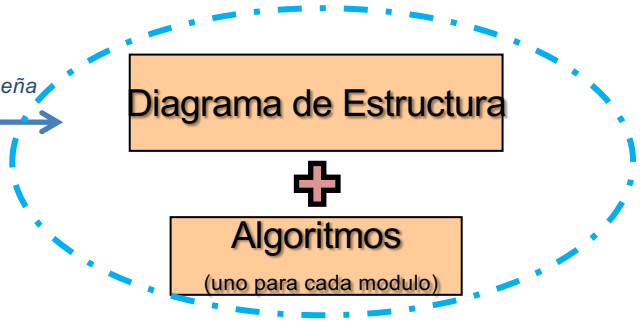
*Fases en la **resolución de un problema***

1. Definición del problema
2. Análisis del problema
3. Diseño del algoritmo
4. Codificación del algoritmo en un lenguaje
5. Análisis de los resultados

Análisis de Requerimientos:
¿**Qué** es lo que hay que hacer?

Diseño:
¿**Cómo** hacer lo especificado en la etapa de Análisis?

Se diseña un



PROGRAMA: Codificación en Python
(definición de función para cada algoritmo)

Analisis de Resultados (Método de la Secante)

```
Ingrese el valor del primer punto xi:
0
Ingrese el valor del segundo punto xs:
1
Ingrese el error estimado:
0.001
xr:  0.6190621954594939
Error aproximado:  0.00035500978729497 52
Número de iteraciones:  11
```

Ante la aplicación del método numérico de la Secante se obtuvo el valor de la raíz aproximada 0.61906 (5 cifras significativas) en base a la entrada del intervalo [0.0, 1.0] y con una la tolerancia porcentual prefijada para el error 0.001. El método numérico necesito 11 iteraciones para obtener el valor de la raíz y con un error relativo porcentual aproximado final de 0.00035500. Por lo tanto, a los 0.61906 segundos desaparecen totalmente las bacterias.