Tarea 1

Métodos Numéricos para Búsqueda de Raíces

Curso: Estructuras de Datos Abstractas y Algoritmos para Ingeniería
Profesor: Ignacio Gómez Chaverri

30 de agosto de 2025

Índice

Instrucciones Generales

1. **Fecha de Entrega:** La fecha límite para la entrega de esta tarea es el **viernes 5 de setiembre de 2025**, antes de las 23:59.
2. **Formato de Entrega:** Se debe entregar en forma de repositorio en GitHub unado el nombre EAAA-Tarea1-#deCarne, y se debe agregar al profesor como colaborador, nombre de usuario(IgnacioGomezCh).
3. **Calidad del Código:** El código debe ser claro, legible y estar bien documentado. Utilice nombres de variables descriptivos, comente las partes complejas de su lógica y mantenga una indentación consistente. La calidad del código es parte fundamental de la evaluación.
4. **Compilación:** Se espera que el código entregado compile sin errores. Un programa que no compile recibirá una **penalización significativa**, ya que no es posible verificar su correcta ejecución. No obstante, se realizará una revisión manual del código fuente para evaluar la lógica y el planteamiento de los algoritmos, permitiendo otorgar una calificación parcial basada en el trabajo conceptual demostrado.
5. **Originalidad:** El trabajo debe ser estrictamente individual. Cualquier caso de plagio o copia será sancionado según el reglamento de la universidad.
6. **Evaluación (100 puntos):** La tarea será calificada sobre una base de 100 puntos, distribuidos de la siguiente manera:
   - **40 puntos:** Implementación correcta y funcional del Método de Bisección, incluyendo el manejo de errores.
   - **40 puntos:** Implementación correcta y funcional del Método de Newton-Raphson, incluyendo el manejo de errores.
   - **10 puntos:** Diseño de la interfaz de usuario, validación de entradas y experiencia de uso.
   - **10 puntos:** Claridad, estilo y documentación del código fuente.

---

1. Fundamento Teórico

Antes de programar, es crucial entender el funcionamiento de los dos métodos numéricos que se implementarán para encontrar la raíz (el cero) de una función $f(x)$.

1.1. Método de Bisección

Este método se basa en el Teorema del Valor Intermedio. Si tenemos un intervalo $[a, b]$ tal que $f(a)$ y $f(b)$ tienen signos opuestos ($f(a) \cdot f(b) < 0$), garantizamos que existe al menos una raíz. El algoritmo reduce el intervalo a la mitad en cada paso.

Lógica para Redefinir el Intervalo

La clave del método es decidir con qué mitad del intervalo nos quedamos. El objetivo es simple: asegurar que la nueva, y más pequeña, sección del intervalo **siga conteniendo la raíz**. Esto se logra manteniendo siempre la condición de que los signos de la función en los extremos del intervalo sean opuestos.

Supongamos que iniciamos con $f(a)$ siendo **negativo (-)** y $f(b)$ siendo **positivo (+)**. Calculamos el punto medio c y evaluamos $f(c)$.

- **Caso A: Si $f(c)$ es negativo (-).** Ahora tenemos que $f(a)$ y $f(c)$ tienen el mismo signo, mientras que $f(c)$ y $f(b)$ tienen signos opuestos. Para mantener la raíz atrapada, debemos elegir el intervalo donde los signos son diferentes. Por lo tanto, nuestro nuevo intervalo se convierte en $[c, b]$. Descartamos la primera mitad.

- **Caso B: Si $f(c)$ es positivo (+).** Ahora $f(a)$ y $f(c)$ tienen signos opuestos, mientras que $f(c)$ y $f(b)$ tienen el mismo signo. La raíz debe estar en el intervalo donde los signos difieren. Nuestro nuevo intervalo se convierte en $[a, c]$. Descartamos la segunda mitad.

Regla para programar: Esta lógica se implementa eficientemente con una multiplicación. Si $f(a) \cdot f(c) < 0$, significa que tienen signos opuestos, y la nueva b será c. De lo contrario, la nueva a será c.

Ejemplo Matemático

Busquemos la raíz de $f(x) = x^3 - x - 2$ en el intervalo $[1, 2]$. ($f(1) = -2$ y $f(2) = 4$).

Iteración	a	b	c	f(c)	Nuevo Intervalo
1	1.0	2.0	1.5	-0.125	[1.5, 2.0]
2	1.5	2.0	1.75	1.609	[1.5, 1.75]
3	1.5	1.75	1.625	0.666	[1.5, 1.625]
4	1.5	1.625	1.5625	0.252	[1.5, 1.5625]

Cuadro 1: Cuatro iteraciones del Método de Bisección.

1.2. Método de Newton-Raphson

Este método utiliza la recta tangente a la función en un punto para aproximar la siguiente estimación de la raíz. Generalmente converge mucho más rápido, pero requiere la derivada y puede fallar si la estimación inicial es inadecuada.

Fórmula

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

Ejemplo Matemático

Busquemos la raíz de la misma función, $f(x) = x^3 - x - 2$, con su derivada $f'(x) = 3x^2 - 1$. Usemos una estimación inicial de $x_0 = 2$.

- **Iteración 1:** $f(2) = 4$, $f'(2) = 11$.

$$x_1 = 2 - \frac{4}{11} \approx 1{,}63636$$

- **Iteración 2:** $f(1{,}63636) \approx 0{,}763$, $f'(1{,}63636) \approx 7{,}033$.

$$x_2 = 1{,}63636 - \frac{0{,}763}{7{,}033} \approx 1{,}52788$$

- **Iteración 3:** $f(1{,}52788) \approx 0{,}051$, $f'(1{,}52788) \approx 5{,}996$.

$$x_3 = 1{,}52788 - \frac{0{,}051}{5{,}996} \approx 1{,}52138$$

- **Iteración 4:** $f(1{,}52138) \approx 0{,}00003$, $f'(1{,}52138) \approx 5{,}942$.

$$x_4 = 1{,}52138 - \frac{0{,}00003}{5{,}942} \approx 1{,}52138$$

Observa cómo en solo 4 iteraciones, el valor de x ya es extremadamente estable y $f(x)$ es prácticamente cero.

1.3. Criterios de Parada: ¿Cuándo dejamos de iterar?

Un algoritmo iterativo necesita reglas claras para detenerse. El bucle de iteración debe terminar tan pronto como se cumpla **cualquiera** de las siguientes condiciones.

1. Convergencia por Tolerancia (Caso de Éxito)

Consideramos que hemos encontrado la raíz cuando el valor de la función en nuestra estimación actual, x_n, está muy cerca de cero.

- **Condición:** $|f(x_n)| < \epsilon$

Donde ϵ (épsilon) es un número muy pequeño definido por nosotros, llamado la **tolerancia** (ej: 1×10^{-7}).

2. Límite de Iteraciones (Red de Seguridad)

Para evitar que el programa se quede en un bucle infinito, siempre debemos establecer un número máximo de iteraciones (ej: 100). Si el algoritmo alcanza este límite sin converger, se detiene.

3. Falla Numérica (Caso de Error)

A veces, el algoritmo no puede continuar por condiciones matemáticas.

- **Para Newton-Raphson:** Si la derivada $f'(x_n)$ es cero.
- **Para Bisección:** Si en el intervalo inicial $[a, b]$ no se cumple que $f(a) \cdot f(b) < 0$.

2. Descripción del Laboratorio

Objetivo

Implementar en C++ un programa que permita encontrar las raíces de funciones predefinidas utilizando los dos métodos numéricos descritos: Bisección y Newton-Raphson.

Funciones a Resolver

Tu programa debe poder encontrar las raíces de las siguientes dos funciones (y sus derivadas), las cuales deben estar implementadas en tu código:

1. **Función Polinómica:**
   - $f(x) = x^3 - x - 2$
   - $f'(x) = 3x^2 - 1$
2. **Función Trigonométrica:**
   - $f(x) = \cos(x) - x$
   - $f'(x) = -\sin(x) - 1$

Requisitos de Implementación

1. **Lenguaje:** El programa debe ser desarrollado en C++.
2. **Funciones de los Métodos:** Debes crear al menos dos funciones principales:
   - void metodoBiseccion();
   - void metodoNewton();
3. **Interfaz de Usuario:** En la función ‘main‘, crea un menú interactivo que le permita al usuario:
   a) Elegir el método a utilizar (Bisección o Newton-Raphson).
   b) Ingresar los valores iniciales requeridos: un intervalo $[a, b]$ para Bisección, o una estimación inicial x_0 para Newton-Raphson.
4. **Condiciones de Parada:** Ambos métodos deben implementar los criterios de parada discutidos en la sección teórica (tolerancia y límite de iteraciones).
5. **Manejo de Errores:**
   - Para Bisección, verifica la condición del intervalo inicial.
   - Para Newton-Raphson, verifica la división por cero en cada iteración.

Ejemplo de Ejecución

```
*** Calculadora de Raices Numericas ***
Seleccione el metodo:
1. Metodo de Biseccion
2. Metodo de Newton-Raphson
Opcion: 2

Ha seleccionado Newton-Raphson.
Ingrese la estimacion inicial x0: 2.0

Iteracion 1: x = 1.636364
Iteracion 2: x = 1.527885
Iteracion 3: x = 1.521380
Iteracion 4: x = 1.521380
Convergencia alcanzada.
La raiz es aproximadamente: 1.52138
```