Laboratorio 2

Se resolvieron ocho problemas usando programación lineal y métodos numéricos implementados en Python. Los modelos fueron desarrollados e implementados en Jupyter Notebook y archivos de reportes.md.

Integrantes

- Abby Donis
- Cindy Gualim
- Josué Say

Enlaces

- Repositorio

Problema 4

Implementar en Python los tres algoritmos vistos en clase para hallar los ceros de una función $f : [a, b] \to \mathbb{R}$

- método de bisección
- método de la secante
- método de Newton-Raphson

Como parámetros, sus algoritmos deben recibir la función f, la derivada df (en el caso de Newton), el intervalo $[a, b]$ o el punto inicial de búsqueda $x_0 \in \mathbb{R}$. Así como los criterios de paro maxIter y tol > 0.

Para la salida, sus funciones deben devolver la lista de aproximaciones realizadas y el valor de punto x^* donde se encontró el cero.

Método de Bisección

Función utilizada (ecuación de actualización):

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

Parámetros de entrada:

- f: función a evaluar
- a, b: extremos del intervalo inicial
- maxIter: número máximo de iteraciones

- tol: tolerancia para el criterio de paro
- verbose: indicador opcional para mostrar el proceso iterativo

Valor de retorno: Se devuelve un diccionario con las siguientes claves:

- converged: indica si el método alcanzó la tolerancia
- root: aproximación final de la raíz
- fAtRoot: valor de la función en la raíz
- iterations: número de iteraciones realizadas
- approximations: lista de todas las aproximaciones intermedias
- error: error estimado en la última iteración
- message: mensaje descriptivo del estado final

Validación: Se verifica que exista un cambio de signo en el intervalo inicial, es decir:

$$f(a) \cdot f(b) < 0$$

Si no se cumple, el método se detiene inmediatamente.

Criterio de paro: Se evalúa si el tamaño del intervalo es menor que la tolerancia:

$$|b - a| < \texttt{tol}$$

Esta condición es evaluada en cada iteración. Además se tiene el criterio de paro si se llega a la cantidad demaxIter enviado.

Método de la Secante

Función utilizada (ecuación de actualización):

$$x_{k+1} = \frac{x_{k-1} \cdot f(x_k) - x_k \cdot f(x_{k-1})}{f(x_k) - f(x_{k-1})}$$

Parámetros de entrada:

- f: función a evaluar
- x0, x1: dos puntos iniciales
- maxIter: número máximo de iteraciones
- tol: tolerancia para el criterio de paro
- verbose: indicador opcional para imprimir el proceso

Valor de retorno: Se retorna un diccionario con:

- converged: si se alcanzó la tolerancia
- root: aproximación final de la raíz
- fAtRoot: valor de la función en la raíz
- iterations: número de iteraciones realizadas
- approximations: lista de aproximaciones sucesivas
- error: diferencia entre las dos últimas aproximaciones
- message: descripción del resultado final

Validación: Se valida que la diferencia entre los valores funcionales no sea cero, es decir:

$$f(x_k) - f(x_{k-1}) \neq 0$$

Es decir que sea diferenciable $f'(x_k) \neq 0$

Criterio de paro: Se verifica si la diferencia entre dos iteraciones consecutivas es menor que la tolerancia:

$$|x_{k+1} - x_k| < \text{tol}$$

Además se tiene el criterio de paro si se llega a la cantidad demaxIter enviado.

Método de Newton-Raphson

Función utilizada (ecuación de actualización):

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}$$

Parámetros de entrada:

- f: función a evaluar
- df: derivada de la función
- x0: valor inicial
- maxIter: máximo número de iteraciones
- tol: tolerancia para el criterio de paro
- verbose: indicador opcional de impresión

Valor de retorno: El resultado es un diccionario con:

- converged: si se alcanzó la tolerancia
- root: valor aproximado de la raíz
- fAtRoot: valor de la función en la raíz

- iterations: número de iteraciones realizadas
- approximations: secuencia de valores generados
- error: diferencia entre las dos últimas aproximaciones
- message: texto explicativo del estado final

Validación: Se verifica que la derivada no se anule en el punto de evaluación:

$$f'(x_k) \neq 0$$

Esto asegura que la tangente esté bien definida.

Criterio de paro: Se detiene el proceso cuando la diferencia entre iteraciones es menor que la tolerancia:

$$|x_{k+1} - x_k| < \texttt{tol}$$

Además se tiene el criterio de paro si se llega a la cantidad demaxIter enviado.

Problema 5

Hallar todos los ceros de la función

$$g(x) = x^2 + \frac{1}{x-7}$$

con al menos **7 decimales de precisión.**

Compare las soluciones obtenidas con cada uno de los algoritmos anteriores en términos del número de iteraciones.

Resultados por método

1. Método de Bisección

Para cada raíz, se seleccionó un intervalo $[a, b]$ donde $f(a) \cdot f(b) < 0$.

Raíz encontrada	Intervalo usado	Iteraciones	Raíz (7 decimales)
-0.36839485401287675	[-2,0]	32	-0.3683949
0.38892324501648545	[0, 2]	32	0.3889232
6.979471608847382	[6.5, 6.98]	30	6.9794716

Este método encontró los ceros correctamente pero con más iteraciones.

2. Método de la Secante

Usando dos puntos iniciales bien elegidos en los alrededores de cada raíz:

Raíz encontrada	Puntos iniciales	Iteraciones	Raíz (7 decimales)
-0.368394853732975	(-2, 0)	12	-0.3683949
0.3889232446865156	(0, 2)	12	0.3889232
6.979471609046458	(6.5, 6.98)	8	6.9794716

Se utilizó los opuntos extremos del intervalo del método de bisección y se obtuvieorn los mismos resultados pero en menos iteraciones.

3. Método de Newton-Raphson

Usando un solo punto inicial x_0. Aquí se observa la sensibilidad del método.

Punto inicial x_0	Raíz encontrada	Iteraciones	Comentario
-2	-0.3683948537329749	7	Converge correctamente
0	-0.3683948537329749	10	Converge, aunque comenzó lejos
6.5	0.3889232446865155	8	No converge a la raíz cercana esperada
2	0.3889232446865155	7	Converge rápido a la raíz positiva
6.98	6.97947160904646	4	Muy buena convergencia

- El método de Newton-Raphson varió su comportamiento con distintos puntos iniciales.
- No es adecuado para encontrar todas las raíces sin análisis previo del dominio (sin embargo, para esta función se hizo análisis del dominio para saber previamente las raíces ya que para todos los métodos métodos no encuentra todos los ceros por lo que hay que ir por intervalo).
- Falló en distinguir entre las dos primeras raíces cuando se partía desde $x_0 = 6.5$ y $x_0 = 2$, conduciendo a la raíz equivocada.

Conclusión

- **Bisección**: garantiza convergencia, pero con mayor número de iteraciones (30–32).
- **Secante**: logra la mejor eficiencia cuando se escogen buenos puntos iniciales.
- **Newton-Raphson**: el más rápido en casos ideales, pero sensible a la elección del punto inicial; puede converger a la raíz equivocada.

Problema 6

Hallar todos los ceros del polinomio

$$f(x) = 2x^5 + 3x^4 - 3x^3 - 10x^2 - 4x + 4,$$

mediante los algoritmos numéricos.

Resultados por método

1. Método de Bisección

Para cada raíz se seleccionó un intervalo $[a, b]$ con cambio de signo: $f(a) \cdot f(b) < 0$.

Raíz encontrada	Intervalo usado	Iteraciones	Raíz (9 decimales)
-1.3037028114194982	[-5, 0]	34	-1.303702811
0.4546075598336756	[0, 1]	31	0.454607560
1.5937398741953075	[1, 5]	33	1.593739874

Este método encontró todos las raices, pero con mayor cantidad de iteraciones en cada caso.

2. Método de la Secante

Se probaron distintos pares de puntos iniciales, con resultados consistentes:

Raíz encontrada	Puntos iniciales	Iteraciones	Raíz (9 decimales)
-1.3037028112439697	(-5, -1)	9	-1.303702811
0.4546075601876466	(0, 1)	8	0.454607560
1.5937398739794297	(1.3, 5)	13	1.593739874

La secante logró resultados correctos en menos iteraciones que la bisección, pero fue sensible al escoger los puntos ya que al escoger los puntos extremos del intervalo del método de bisección se obtuvieron:

Raíz encontrada	Puntos iniciales	Iteraciones	Raíz (9 decimales)
0.4546075601876466	(-5, -1)	10	0.454607560
0.4546075601876466	(0, 1)	8	0.454607560
0.4546075601876466	(1.3, 5)	10	0.454607560

3. Método de Newton-Raphson

Este método fue probado con distintos puntos iniciales x_0:

Punto inicial x_0	Raíz encontrada	Iteraciones	Comentario
-5	-1.3037028112439695	12	Converge correctamente
0	0.4546075601876466	7	Converge rápido
1	0.4546075601876466	6	Converge, pero a raíz equivocada
5	1.5937398739794297	11	Converge correctamente

- Newton-Raphson converge rápidamente si el punto inicial está razonablemente cerca de una raíz.
- Con $x_0 = 1$, convergió a la raíz en $x \approx 0.4546$ en lugar de la cercana a 1.59.
- No detecta automáticamente todas las raíces, por lo que se necesita análisis previo del dominio o múltiples ejecuciones (al igual que en los otros métodos).

Conclusión

- **Bisección**: garantizó convergencia con precisión, pero tomó entre 31 y 34 iteraciones por raíz.
- **Secante**: fue más eficiente, convergiendo entre 8 y 13 iteraciones, dependiendo de los puntos iniciales.
- **Newton-Raphson**: el más rápido cuando parte cerca de la raíz (6 a 12 iteraciones), pero puede converger a una raíz no deseada si el punto inicial está mal ubicado.