Universidad de Buenos Aires

Facultad de Ingeniería

2do Cuatrimestre de 2023

Análisis Numérico I (75.12)

Búsqueda de Raíces

Curso:

Sassano

Integrantes:

Francisco Orquera Lorda	forqueral@fi.uba.ar	105554
Carolina Di Matteo	cdimatteo@fi.uba.ar	103963
Anita Vernieri	avernieri@fi.uba.ar	104734
María Zanatta	mzanatta@fi.uba.ar	108148

Lenguajes Elegidos: Rust & Python

Tabla de Contenidos

Objetivo del Trabajo

Con el objeto de completar las solicitudes que respecta al enunciado del Primer Trabajo Práctico de la materia Análisis Numérico I, estudiaremos y desarrollaremos

- la representación de números en formato punto flotante media precisión
- algoritmos para la búsqueda y resolución de distintos métodos numéricos de búsqueda de raíces
- algoritmos para la interpolación de datos

Introducción

El foco principal de este trabajo se encuentra en desarrollar y analizar métodos matemáticos asociados a la búsqueda de raíces utilizando ejemplos prácticos para comparar sus desempeños.

A lo largo del primer ítem desarrollaremos la lógica de representación de un número en punto flotante de media precisión y analizaremos qué sucede con la precisión del mismo según la herramienta con la que estemos trabajando. También mencionaremos qué ocurre con las mediciones de cualquier tipo de herramienta para cualquier tipo de cálculo y su respectivo error asociado.

En segundo lugar, desarrollaremos los resultados obtenidos para la búsqueda de raíces con la utilización de los métodos numéricos de Bisección, Punto Fijo, Secante, Newton Raphson y Newton Raphson Modificado. Finalmente, veremos en la sección de Resultados, cómo se comportan la convergencia y constante asintótica de las iteraciones sucesivas de cada uno de los métodos, como también el error absoluto y las diferencias sucesivas. Utilizamos para esta sección en particular, la funcionalidad que provee el lenguaje de programación Rust de la ejecución de los métodos numéricos en hilos paralelos del CPU con el objetivo de obtener la mayor capacidad de procesamiento posible.

Por último, elegiremos e implementaremos un algoritmo para obtener un polinomio interpolante en base a los datos de entrada brindados por el enunciado. Realizaremos los cálculos correspondientes para obtener el mismo y comentaremos qué ocurre respecto a los valores que resultan de la utilización de dicho polinomio interpolante, los diferentes polinomios que pueden obtenerse y qué tipo de ventajas propone su utilización en el ámbito profesional.

Desarrollo

A continuación, desarrollaremos el análisis realizado para cada uno de los puntos planteados en el enunciado propuesto.

No Usar Fracciones

Teniendo en cuenta que $total = \sum_{i=1}^{4} ultimo\_numero\_padron\left(integrante_i\right) = 19$:

a. Para representar el número $\frac{1}{total}$ en formato `Half(16 bits)`, teniendo en cuenta que se necesitan 1 bit para representar el signo, 10 para la mantisa y 5 para el exponente, seguimos los siguientes pasos:

   i. por ser un número positivo, ponemos el bit de signo en `0`

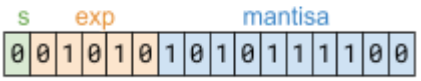
   ii. para encontrar la mantisa, buscamos en la calculadora el valor en decimal del número en cuestión, que equivale a `0.05263157895`, y lo dividimos sucesivamente -las suficientes para obtener la máxima representación posible- y nos quedamos con el primer dígito no decimal para armar la cadena de bits:

0,05263157895	* 2 =	0,1052631579	→	**0**
0,1052631579	* 2 =	0,2105263158	→	**0**
0,2105263158	* 2 =	0,4210526316	→	**0**
0,4210526316	* 2 =	0,8421052632	→	**0**
0,8421052632	* 2 =	1,684210526	→	**1**
0,6842105263	* 2 =	1,368421053	→	**1**
0,3684210526	* 2 =	0,7368421053	→	**0**
0,7368421053	* 2 =	1,473684211	→	**1**
0,4736842105	* 2 =	0,9473684211	→	**0**
0,9473684211	* 2 =	1,894736842	→	**1**
0,8947368421	* 2 =	1,789473684	→	**1**
0,7894736842	* 2 =	1,578947368	→	**1**
0,5789473684	* 2 =	1,157894737	→	**1**
0,1578947368	* 2 =	0,3157894737	→	**0**
0,3157894737	* 2 =	0,6315789474	→	**0**

   iii. a continuación, para obtener la mayor precisión posible, tomamos todos los bits calculados luego del primer `1` (dado que el mismo está implícito en representación de punto flotante), resultando la cadena: `1010111100`

   iv. luego, para obtener el exponente, puesto que nos movimos 5 lugares respecto de la posición inicial para armar la cadena de bits que conforman la mantisa, representamos en base 2 en notación en exceso el valor que resulta de $-5 + 15 = 10 \Rightarrow$ el exponente es `01010`

   v. finalmente, representamos el número $\frac{1}{19}$ en formato punto flotante media precisión como `bit(signo) + bit(exponente) + bit(mantisa)`:



Una vez calculado el valor en el formato pedido, para representar el valor siguiente y anterior posible simplemente sumamos un bit a la mantisa, resultando:

`0010101010111100` → `0.052612304687500` → valor actual

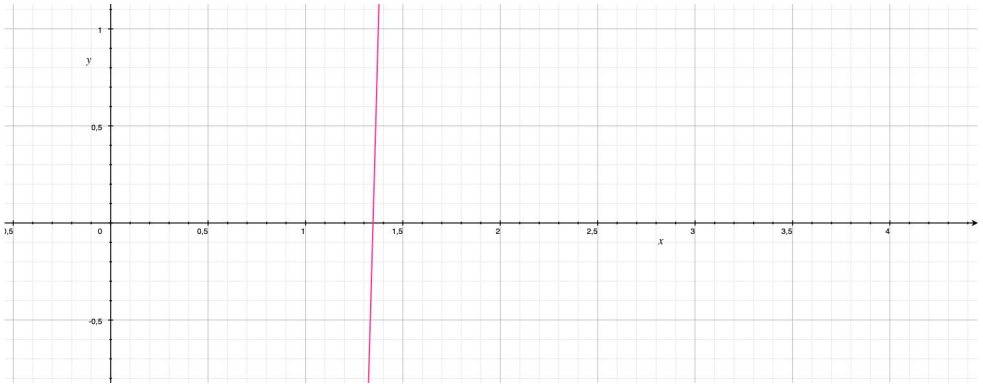
`0010101010111101` → `+ 1 bit` mantisa → `0.052642822265625` → valor siguiente

`0010101010111011` → `- 1 bit` mantisa → `0.052581787109375` → valor anterior

b. Veamos que al momento de calcular la propagación de errores debemos tener en cuenta que la fracción calculada es una variable más puesto que la computadora comete tanto errores de redondeo por el hecho de representar un número racional que podría tener decimales que escapan de la precisión con la que puede trabajar el equipo.
c. Respecto de los cálculos posibles que pudiera hacer cualquier herramienta, podemos decir que difícilmente la misma dé un resultado exacto ya que tendrá un error asociado de redondeo (en caso de no contar con la capacidad suficiente para representar todos los números necesarios) tratándose de computadoras; o bien inherentes si la herramienta es utilizada por el ojo humano; o bien algún error de truncamiento que corresponde a la discretización o aproximación que se pudiera cometer -por ejemplo- por el hecho de trabajar con una herramienta de cálculo que utilice métodos numéricos que descarten términos.

Búsqueda de Raíces

Para la función $f(x) = 2^x + 8^x - 19$ continua y con raíz única, realizamos un gráfico en el intervalo de interés:



Veamos entonces que la raíz se encuentra dentro del intervalo `[1,1.5]`. Para calcularla, utilizamos los métodos vistos en clase utilizando como criterio de paro la diferencia de $1 \cdot 10^{-5}$ y $1 \cdot 10^{-13}$ entre dos iteraciones sucesivas.

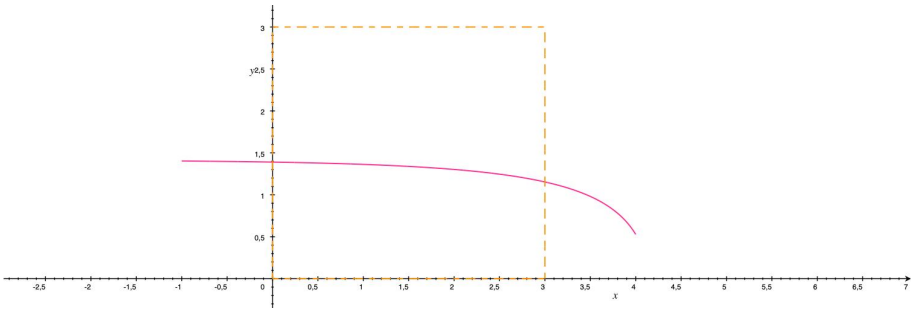
Bisección

Utilizando la fórmula de iteraciones sucesivas para el método de Bisección, en el intervalo de interés obtuvimos

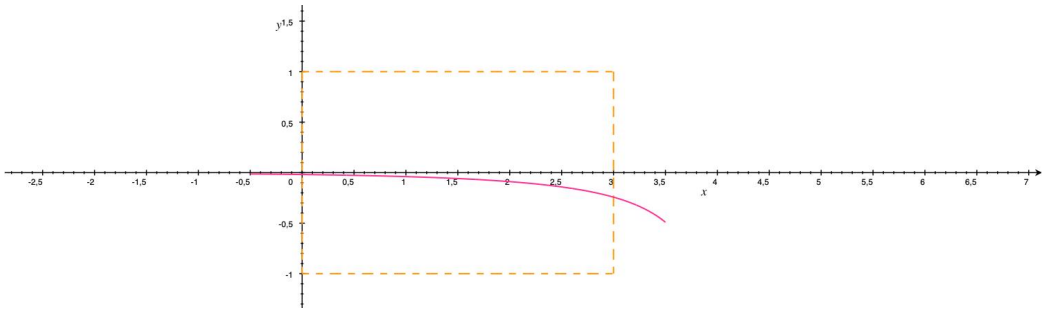
- Para $1 \cdot 10^{-5}$ iteraciones, se alcanza la raíz en la iteración 14 y resulta `1.3468475341796875`
- Para $1 \cdot 10^{-13}$ iteraciones, se alcanza la raíz en la iteración 41 y resulta `1.3468595839050295`

Punto Fijo

Para el análisis del método en cuestión, lo primero que realizamos fue buscar una $g(x)$ admisible, de la forma $g(x) = x - f(x)$. Luego, graficamos la función en el intervalo de interés para verificar las condición de existencia del punto fijo:



Luego, para verificar la unicidad del punto fijo, derivamos $g(x)$ y -graficándola en el intervalo de interés- vemos que efectivamente la misma está acotada entre $-1 < g(x) < 1$:



Finalmente, la función $g(x)$ cumple con las condiciones de existencia y unicidad de punto fijo por lo que resulta admisible para el cálculo de la raíz a partir del método de iteración de punto fijo.

Así, utilizando la fórmula de iteraciones sucesivas para el método de Punto Fijo, en el intervalo de interés obtuvimos

- Para $1 \cdot 10^{-5}$ iteraciones, se alcanza la raíz en la 5° iteración y resulta `1.3468596178060541`
- Para $1 \cdot 10^{-13}$ iteraciones, se alcanza la raíz en la 11° iteración y resulta `1.3468595839051418`

Secante

Utilizando la fórmula de iteraciones sucesivas para el método de Secante, en el intervalo de interés obtuvimos

- Para $1 \cdot 10^{-5}$ iteraciones, se alcanza la raíz en la 5° iteración y resulta `1.3468595820175167`
- Para $1 \cdot 10^{-13}$ iteraciones, se alcanza la raíz en la 7° iteración y resulta `1.346859583905141`

Newton Raphson

Sabiendo que $f \in C^2$ y utilizando la fórmula de iteraciones sucesivas para el método de Newton Raphson, en el intervalo de interés obtuvimos

- Para $1 \cdot 10^{-5}$ iteraciones, se alcanza la raíz en la 8° iteración y resulta `1.3468598054102443`
- Para $1 \cdot 10^{-13}$ iteraciones, se alcanza la raíz en la 9° iteración y resulta `1.3468595839051902`

Newton Raphson Modificado

Nuevamente, puesto que $f \in C^2$ y utilizando la fórmula de iteraciones sucesivas para el método de Newton Raphson Modificado, en el intervalo de interés obtuvimos

- Para $1 \cdot 10^{-5}$ iteraciones, se alcanza la raíz en la 6° iteración y resulta `1.3468515969591204`
- Para $1 \cdot 10^{-13}$ iteraciones, se alcanza la raíz en la 8° iteración y resulta `1.346859583905141`

Búsqueda de una "Raíz Real"

Para seguir con el análisis de la búsqueda de raíces, utilizamos una biblioteca con el fin de buscar la raíz que más se aproxime al valor real. En esta línea, utilizamos la biblioteca externa "roots" y obtuvimos que la misma resulta: `1.3468595839051407`.

Ventajas y Desventajas

Vemos una clara ventaja en el método de Newton Raphson y Newton Raphson Modificado contra el de Bisección en el número de iteraciones realizadas para hallar una aproximación a la raíz deseada. Esto se debe al orden de convergencia de los métodos, siendo para Newton Raphson y Newton Raphson Modificado de orden 2, y para Bisección de orden 1. Tanto el método de Iteración de Punto Fijo, como el de la Secante, logran aproximar la raíz en una cantidad razonable de iteraciones, siendo el orden de convergencia 1 y supra lineal ($1 < \alpha < 2$) respectivamente.

El orden de convergencia calculado, coincide con el modelo teórico, logrando para Bisección y Punto Fijo $\alpha = 1$, Secante $1 < \alpha < 2$ y Newton Raphson y Newton Raphson Modificado $\alpha = 2$.

Polinomio Interpolante

Dados los datos de entrada,

x	1	2	3	5
$f(x)$	4	3	4	8

optamos por obtener el polinomio interpolante de Newton. Para ello, programamos un algoritmo que en base a los datos y al grado del polinomio requerido, calcula las diferencias divididas y después define el polinomio según el método numérico en cuestión. Pediremos calcular el polinomio de grado 3. Las diferencias divididas obtenidas:

x_i	$f[x_i]$	$f[x_i, x_{i+1}]$	$f[x_i, x_{i+1}, x_{i+2}]$	$f[x_i, x_{i+1}, x_{i+2}, x_{i+3}]$
1	4			
		-1		
2	3		1	
		1		-0.166666667
3	4		0.333333333	
		2		
5	8			

Y el polinomio obtenido en base a la información presentada fue:

$$P_N(x) = 4 - (x-1) + (x-1)(x-2) - 0.16667(x-1)(x-2)(x-3)$$

a. Solo se puede calcular el valor estimado de puntos que estén dentro del intervalo de los puntos conocidos. Para estimar la cota del error en este caso, habría que sacrificar un grado del polinomio interpolante. Así quedaría definida como la diferencia entre el polinomio de Newton de grado 3 y el de grado 2 (ó el término de la última diferencia dividida). Esto se puede hacer siempre que el término que se descarte no comprometa la pertenencia del punto estimado al nuevo intervalo de puntos que definan el polinomio.
b. Con los distintos métodos es posible que se obtengan polinomios distintos, puesto que cada uno tiene distintas formas de encontrar un polinomio interpolante lo que hace que los polinomios resultantes difieran entre sí. Por otra parte, para un mismo método también pueden encontrarse polinomios distintos a partir de los mismos datos de entrada, puesto que podrían cambiarse el orden de los nodos de entrada y eso resultaría en un polinomio distinto efectivamente. En conclusión, sólo hay un polinomio interpolante para un mismo método y un mismo orden de nodos.
c. Depende de cuál sea el problema a modelizar podría utilizarse un polinomio interpolante como herramienta para la búsqueda de un resultado aproximado. Por ejemplo, podría utilizarse como medio para la estimación de la población en un año intermedio entre dos censos: si sólo tenemos datos de censos de los años 2010 y 2020, pero nos solicitan tener una estimación de la población en el año 2019, podemos calcular mediante un polinomio interpolador un valor aproximado.

Resultados

Veamos en detalle los resultados obtenidos, para cada una de las tolerancias propuestas por el enunciado, los diferentes valores que dieron la búsqueda de raíces a partir de los distintos métodos utilizados.

Tablas Busqueda de Raices

Bisección

Tolerancia: 1e-5	
Iteración	Valor
0	1.25
1	1.375
2	1.3125

Tolerancia: 1e-13	
Iteración	Valor
0	1.25
1	1.375
2	1.3125

...	...
13	1.346832275
14	**1.346847534**

...	...
40	1.346859584
41	**1.346859584**

Punto Fijo

Tolerancia: 1e-5	
Iteración	Valor
0	1.351662041
1	1.346611678
2	1.346872355
3	1.346858926
4	**1.346859618**

Tolerancia: 1e-13	
Iteración	Valor
0	1.351662041
1	1.346611678
2	1.346872355
...	...
10	**1.346859584**

Secante

Tolerancia: 1e-5	
Iteración	Valor
0	1.291152005
1	1.338571423
2	1.347333526
3	1.346855623
4	**1.346859582**

Tolerancia: 1e-13	
Iteración	Valor
0	1.291152005
1	1.338571423
2	1.347333526
...	...
6	**1.346859584**

Newton Raphson

Tolerancia: 1e-5	
Iteración	Valor
0	2.650652052
1	2.198282441
2	1.79545469
...	...
7	**1.346859584**

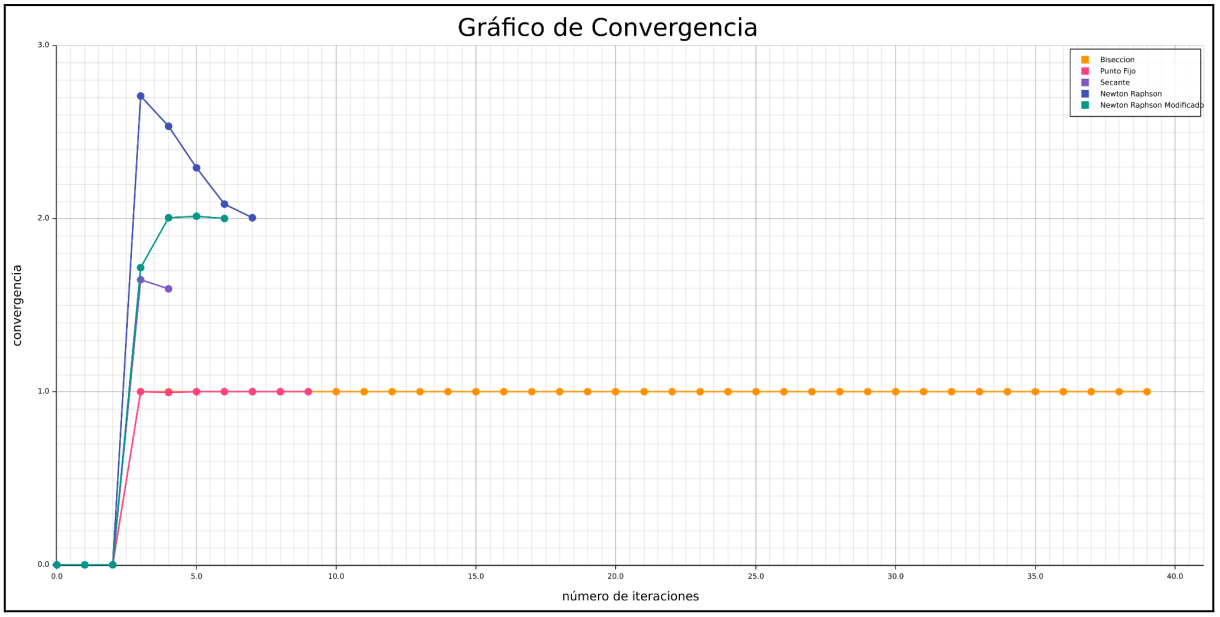
Tolerancia: 1e-13	
Iteración	Valor
0	2.650652052
1	2.198282441
2	1.79545469
...	...
8	**1.346859584**

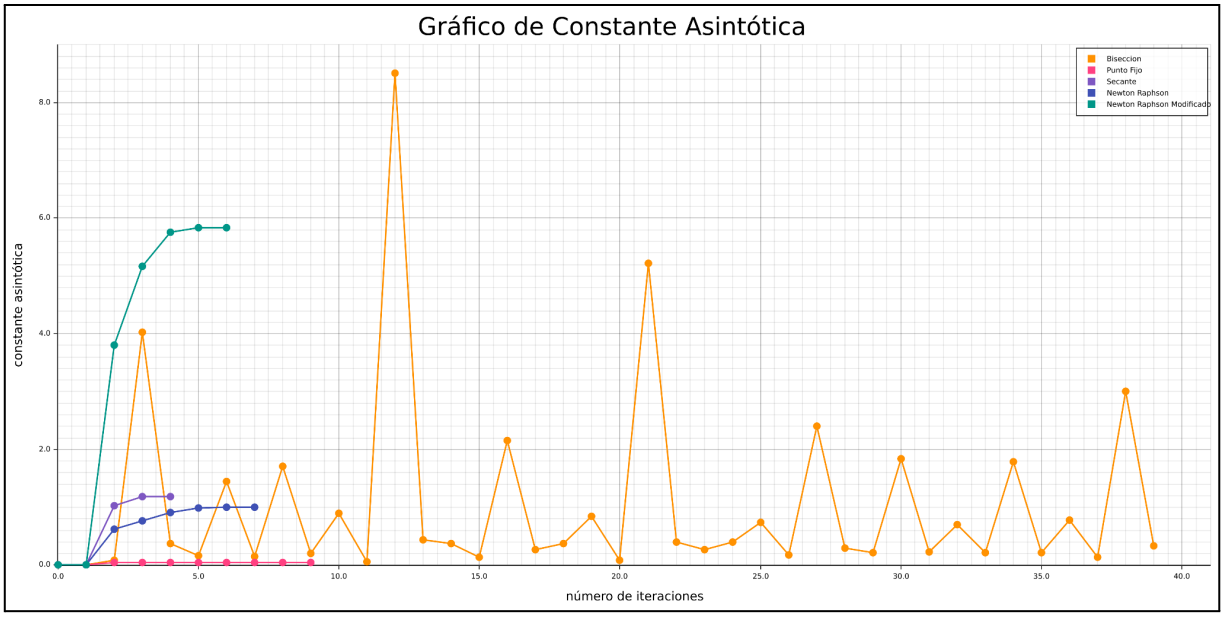
Newton Raphson Modificado

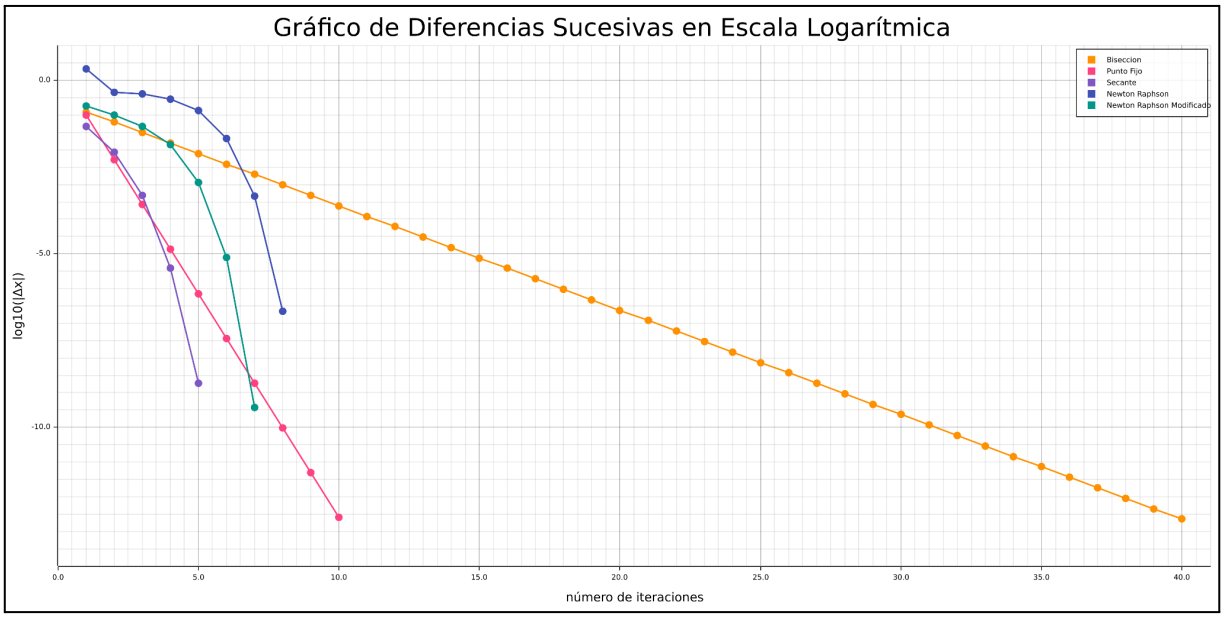
Tolerancia: 1e-5	
Iteración	Valor
0	1.18513196
1	1.283704805
2	1.331712978
3	1.345678589
4	1.346851597
5	**1.346859584**

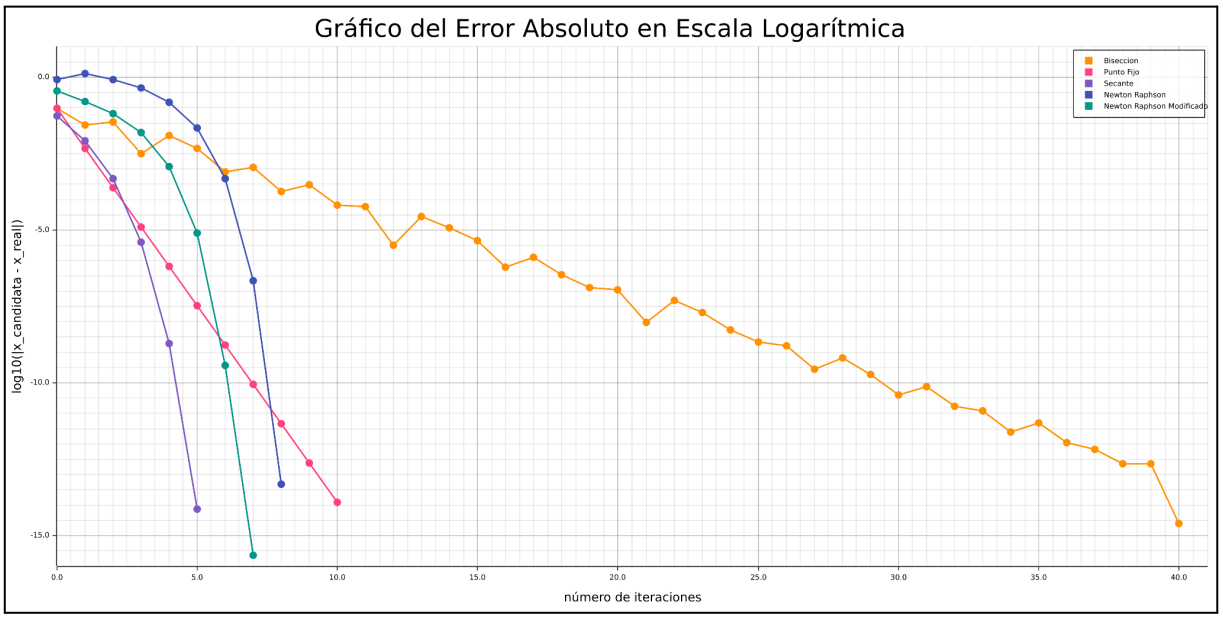
Tolerancia: 1e-13	
Iteración	Valor
0	1.18513196
1	1.283704805
2	1.331712978
3	1.345678589
...	...
7	**1.346859584**

Gráficos y Figuras









Ecuaciones

- Función de estudio: $f(x) = 2^x + 8^x - 19$
  - $f'(x) = 2^x \cdot \ln(2) + 3 \cdot \ln(2) \cdot 8^x$
  - $f''(x) = \left(2^x \cdot \log_{10}(2) + 8^x \cdot \log_{10}(8)\right)^2$
- Método Bisección: $p_n = \frac{a_n + b_n}{2}$
- Método Punto Fijo: $p_{n+1} = g(p_n)$
  - $g(x)$ propuesta: $g(x) = \log_8(19 - 2^x)$
  - $g'(x) = \frac{-2^x}{3 \cdot (19 - 2^x)}$
- Método Secante: $p_n = p_{n-1} - \frac{f(p_{n-1}) * (p_{n-1} - p_{n-2})}{f(p_{n-1}) - f(p_{n-2})}$
- Método Newton Raphson: $p_{n+1} = p_n - \frac{f(p_n)}{f'(p_n)}$
- Método Newton Raphson Modificado: $p_{n+1} = p_n - \frac{f(p_n) * f'(p_n)}{(f'(p_n))^2 - f(p_n) * f''(p_n)}$
- Polinomio Interpolante de Newton: $P_N(x) = f[x_0] + \sum_{k=1}^{n} f[x_0, \ldots, x_k](x - x_0) \ldots (x - x_{k-1})$
- Diferencias divididas: $f[x_i, \ldots, x_{i+k}] = \frac{f[x_{i+1}, x_{i+2}, \ldots, x_{i+k}] - f[x_i, x_{i+1}, \ldots, x_{i+k-1}]}{x_{i+k} - x_i}$

Conclusiones

Podemos observar que todos los métodos de búsqueda de raíces convergen a un valor similar al de la raíz "real" y la velocidad con la que estos lo logran se ajustan al modelo teórico estudiado. En el gráfico de convergencia notamos que el método de Newton Raphson es el más rápido de todos, seguido del método de la Secante, Punto Fijo y Bisección. En cuanto a la cantidad de iteraciones para lograr el valor deseado, Bisección es el peor de todos, con un total de 41 iteraciones con una tolerancia de 10^{-13}, el método de la secante solo necesito de 6 con la misma tolerancia, siendo este el mejor de todos, seguido de Newton Raphoson Modificado con 7. Cabe destacar que si bien el método de Newton Raphson resultó ser el más eficiente, es muy importante elegir una buena semilla para comenzar la iteración ya que el mismo puede diverger. En cuanto a Punto Fijo, si bien la convergencia es lineal, llega a un valor aproximado de la raíz en una cantidad razonable de iteraciones, sin embargo, encontrar una función $g(x)$ admisible puede traer inconvenientes. Como criterio de paro, utilizamos la diferencia de sucesiva entre iteraciones con una tolerancia de 10^{-5} y 10^{-13}. En el gráfico se puede observar como esta se vuelve más chica a medida que pasan las iteraciones en una escala logarítmica.

Para el cálculo del polinomio interpolante, optamos por el polinomio de Newton dada la información de entrada y que necesitamos guardar una menor cantidad de variables en memoria a la hora de calcular las diferencias divididas para llegar al resultado final. En base a los datos, es posible armar diferentes polinomios de grado 3, teniendo en cuenta que tendremos que sacrificar un grado del mismo para poder calcular la cota de error cuando intentemos interpolar un valor. Ningún polinomio interpolante estudiado (Lagrange, Newton, Hermite, Spline) nos sirven para estimar un valor el cual no se encuentra dentro del intervalo utilizado, teniendo que recurrir a otro tipo de métodos numéricos para esto.

Referencias

1. Análisis Numérico - Richard L. Burden
2. Diapositivas de Clase
3. Link al repositorio del código fuente del trabajo práctico: https://github.com/gcc-cdimatteo/Analisis-Numerico-I-75.12/tree/main/tp1