
PROYECTO 1: MÉTODO DE LA SECANTE, PUNTO FIJO, BISECCIÓN Y NEWTON-RAPSON

María José Gil Herrera; carné 20337

Alejandro Pallais García; carné 20093

17 de agosto de 2022

Índice

1. Introducción	3
2. Métodos numéricos	3
2.1. Punto Fijo	3
2.2. Bisección	3
2.3. Secante	3
2.4. Newton-Raphson	3
3. Resultados	4
4. Discusión de resultados	6
5. Descripción del código	6
5.1. Punto fijo	6
5.2. Bisección	7
5.3. Secante	8
5.4. Newton-Raphson	8
6. Conclusiones	8
7. Bibliografía	9

1. Introducción

Los métodos presentados a continuación son una manera versátil y práctica de obtener las raíces aproximadas de una función. Su versatilidad y facilidad de uso viene de la cantidad de pasos que hay que realizar en cada iteración, ya que son pocos y están establecidos de manera clara. Además de esto, los métodos se comportan casi de una manera binaria, de modo que es fácil saber hasta qué punto en la aproximación se puede llegar. Su mayor dificultad o desventaja es la exactitud, puesto que si se quiere conseguir mayor exactitud se deben ejecutar muchas iteraciones, además de utilizar una cantidad considerable de decimales. Consiguientemente, el hacer estos métodos a mano puede resultar confuso y bastante tedioso.

Se presenta entonces una ventaja al automatizar dichos métodos, pues de esa manera, el estudiante no solo puede entender cómo es que cada uno de los métodos funciona, pero puede también obtener una solución más exacta sin las dificultades expuestas anteriormente. Además de esto, al animar y graficar dichos métodos haciendo uso de los distintos lenguajes de programación, ayuda a visualizar las acotaciones y aproximaciones de cada uno, además de entender por qué existe cierto error en cada uno. En este caso, se decidió llevar a cabo la automatización en Python, aprovechando las diferentes librerías que nos proporciona este lenguaje.

2. Métodos numéricos

2.1. Punto Fijo

Este método también es conocido como *método de sustituciones sucesivas*. Y a grandes rasgos, consiste en obtener la aproximación de la raíz a través del reordenamiento de la función. Se sabe que $f(x) = 0$ puede ser reescrita como $f(x) = g(x) - x$ para algún $g(x)$. Debido a la dificultad para encontrar una $g(x)$ que cumpla con los requisitos, se ha desarrollado un método que fuerza a una función a cumplir dichas condiciones, en donde $g(x) = x - \lambda * f(x) : \lambda = \frac{1}{f'(x)_{MAX}} \iff f'(x)$ no cambia de signo dentro del intervalo que se está utilizando

2.2. Bisección

El método de bisección está basado en el [Teorema de Bolzano](#), creando un algoritmo que nos ayuda a encontrar la raíz de la función. Lo primero que se hace es suponer una función $f(x) = 0$ para algún $x \in [a, b]$. Luego, se toma el establecido y para encontrar la raíz, se divide a la mitad tal que $m = \frac{a+b}{2}$. Ahora, si $f(m) = 0 \implies m$ es la raíz buscada. Si $f(a)$ y $f(m)$ tienen signos diferentes \implies la raíz $\in [a, m]$, a cambio si $f(b)$ y $f(m)$ tienen signos diferentes \implies la raíz $\in [m, b]$. Continuamos dividiendo el intervalo tanto como se desee, a manera de encontrar una aproximación de la raíz que se está buscando. Finalmente, se tiene una fórmula para determinar el número máximo de iteraciones, la cual es $\frac{b-a}{2^n} < \epsilon$, para un valor ϵ establecido

2.3. Secante

El método de la secante utiliza una aproximación de la derivada de la función para obtener así la aproximación de la raíz. Se basa en el método de Newton-Raphson, con la diferencia de que no emplea la derivada de la función, sino que hace uso de la aproximación $f'(x_0) = \frac{f(x_1) - f(x_0)}{x_1 - x_0}$. Para este método se requieren dos puntos x_0, x_1 , a los cuales se les aplica la siguiente función $x_2 = x_0 - \frac{x_1 - x_0}{f(x_1) - f(x_0)} * f(x_0)$ para obtener el siguiente punto de la iteración. Este punto estará más próximo a la raíz que los anteriores, y así sucesivamente

2.4. Newton-Raphson

Este método nos permite encontrar la raíz de una ecuación siempre y cuando se inicie con una aproximación cercana a esta. El método deriva del [Teorema de Taylor](#), donde se utilizan una función $f(x)$, un punto x_0 y la derivada de la función $f'(x)$, de modo que se obtiene el siguiente punto por medio de la ecuación $x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}$, obteniendo cada vez un punto más cercano a la raíz.

3. Resultados

Luego de programar los métodos anteriores, se ingresó una función al programa y se imprimieron los resultados de cada uno como se ve en la figura 1. Se calculó la misma raíz, de modo que es fácil realizar la comparación entre cada uno de estos, como la exactitud de las respuestas y la cantidad de pasos que se requirieron para llegar a estas.

```

Función: -1 + 7*exp(-x)*sin(x)
Tolerancia: 1e-10
x_0: 0.1
x_1: 0.3
Método: secante
raíz: x=0.170179993751426,
f(x)=-1.16084919454806e-11,
cantidad de pasos: 5

Método: punto fijo
raíz: x=1.89305902947971,
f(x)=-8.58886295418415e-11,
cantidad de pasos: 161

Método: biyección
raíz: x=0.1701799937523901,
f(x)=-6.96420698886868e-12,
cantidad de pasos: 31

Método: Newton
raíz: x=0.170179993753835,
f(x)=0,
cantidad de pasos: 4

```

Figura 1: Resultados de la ejecución de los métodos

A continuación se muestra la raíz de la función calculada en geogebra de manera gráfica en la figura ??, y comparándola con esta, podemos ver que las aproximaciones realizadas por nuestros métodos son bastante certeras.

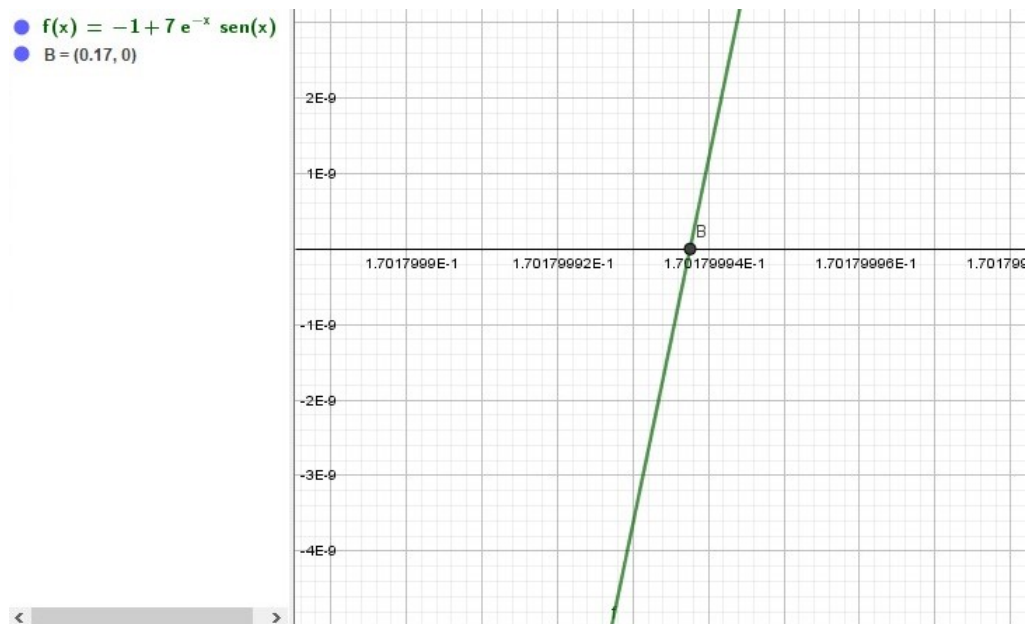


Figura 2: Aproximación gráfica en GeoGebra

Mostramos algunas ejecuciones más con diferentes funciones, para que se pueda apreciar la diferencia entre los métodos propuestos.

```

Función: (x - 1)*(x + 2)
Tolerancia: 1e-10

```

metodo	entrada/s	repeticiones	raiz	resultado	error
secante	x_0=0.1, x_1=1.1	5	0.999999999999949	-1.52544643583494E-13	5.08482145278322E-14
punto fijo	x=-1	2	1.00000000000000	0	0
bisección	x_0=0.1, x_1=1.1	33	1.0000000000232832	6.98494595502289e-11	-2.3283153183228933e-11
Newton	x=0.1	5	1.000000000000503	1.51005874471619e-11	-5.03352914904553e-12

```

Función: (x - 1)*(x + 2)
Tolerancia: 1e-10

```

metodo	entrada/s	repeticiones	raiz	resultado	error
secante	x_0=-3, x_1=-1	7	-1.999999999999998	-6.39488462184086E-14	1.06581410364015E-14
punto fijo	x=0	1	-2.00000000000000	0	0
bisección	x_0=-2.5, x_1=-1	34	-2.000000000029104	8.73114913710481e-11	-1.4551915228366852e-11
Newton	x=-3	5	-2.00000000000000	0	0

```

Función: x**3 + 3*x + 9
Tolerancia: 1e-10

```

metodo	entrada/s	repeticiones	raiz	resultado	error
secante	x_0=0.1, x_1=1.1	8	-1.60969549401664	3.22408766351145E-13	-1.03366204484701E-11
punto fijo	x=0.1	diverge	-	-	-
bisección	x_0=0.1, x_1=-2	37	-1.60969549401998	-3.56719098704161e-11	-1.241229341530925e-11
Newton	x=0.1	6	-1.60969549402241	-6.18900486415441e-11	-1.39241951302438e-11

Figura 3: Resultados de la ejecución de los métodos

```

Función: exp(x) - 2
Tolerancia: 1e-10

```

metodo	entrada/s	repeticiones	raiz	resultado	error
secante	x_0=0.1, x_1=1.1	6	0.693147180560056	2.21156426505331E-13	5.76271252938909E-11
punto fijo	x=0.1	diverge	-	-	-
bisección	x_0=0.1, x_1=1.1	34	0.69314718056703	1.41691103294761e-11	4.756572913322543e-11
Newton	x=0.1	5	0.693147180559946	8.88178419700125e-16	5.77861092310172e-11

```

Función: log(x + 0.1)
Tolerancia: 1e-10

```

metodo	entrada/s	repeticiones	raiz	resultado	error
secante	x_0=0.1, x_1=1.1	6	0.900000000000001	6.66133814775094E-16	-6.66133814775094E-16
punto fijo	x=0.1	diverge	-	-	-
bisección	x_0=0.1, x_1=1.1	32	0.9000000000465662	4.65663063653737e-11	-5.174016770581602e-11
Newton	x=0.1	6	0.900000000000000	0	0

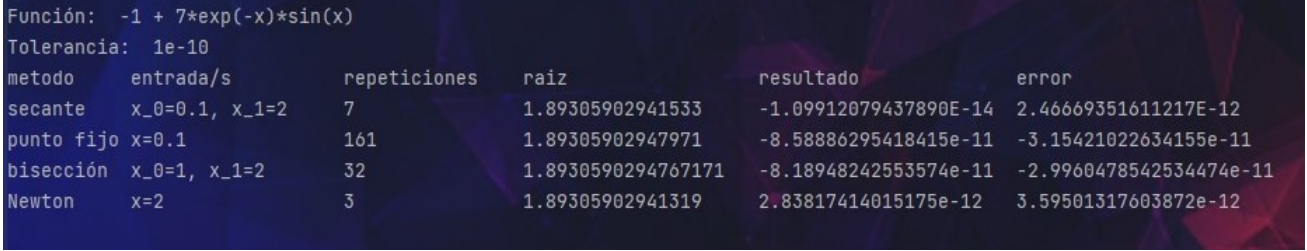
```

Función: cos(x)
Tolerancia: 1e-10

```

metodo	entrada/s	repeticiones	raiz	resultado	error
secante	x_0=0.1, x_1=1.1	5	1.57079632679490	6.12323399573677E-17	1.30572885836955E-10
punto fijo	x=0.1	4	1.57079632679490	6.12323399573677e-17	1.30572885836955e-10
bisección	x_0=0.1, x_1=2	33	1.5707963267457672	4.91294284734475e-11	1.6184964479748487e-10
Newton	x=1	3	1.57079632679549	-5.91243550575401e-13	1.30196520231607e-10

Figura 4: Resultados de la ejecución de los métodos



```

Función: -1 + 7*exp(-x)*sin(x)
Tolerancia: 1e-10

```

metodo	entrada/s	repeticiones	raiz	resultado	error
secante	x_0=0.1, x_1=2	7	1.89305902941533	-1.09912079437890E-14	2.46669351611217E-12
punto fijo	x=0.1	161	1.89305902947971	-8.58886295418415e-11	-3.15421022634155e-11
bisección	x_0=1, x_1=2	32	1.8930590294767171	-8.18948242553574e-11	-2.9960478542534474e-11
Newton	x=2	3	1.89305902941319	2.83817414015175e-12	3.59501317603872e-12

Figura 5: Resultados de la ejecución de los métodos

Finalmente se presentan los videos de cómo funcionan los diferentes métodos para obtener las raíces. Cabe resaltar que estas animaciones se hicieron utilizando $f(x)$ aleatorias

- [Método de Bisección](#)
- [Método Punto Fijo](#)
- [Método de la Secante](#)

4. Discusión de resultados

En este caso, podemos ver que el método que calculó la raíz más exacta fue el método de Newton-Raphson, así como el que menor cantidad de pasos utilizó, debido a que se comenzó con un valor bastante cercano a la raíz. Estos resultados nos demuestran que el método de Newton-Raphson es el más efectivo siempre y cuando la función sea derivable y se tenga una idea bastante cercana de cuál es la raíz. Esto es debido al tipo de iteraciones que realiza el método, ya que cada punto nuevo es más cercano al anterior. Es decir, la diferencia entre el punto nuevo y el punto anterior será menor que la diferencia de ese punto anterior y su respectivo precesor. Debido a esto es que también podemos determinar que en caso no se tenga una idea de cuál es la raíz, el método de Newton-Raphson es poco eficiente.

Guiándonos por estos resultados, podemos entonces argumentar que tanto el método de la secante, como el de bisección son los más efectivos en el caso en el que no se tenga una aproximación a la raíz de la función. Como podemos ver, ambos trabajan por medio de intervalos. Y a pesar de que en este caso el método de bisección haya utilizado una cantidad considerablemente mayor de pasos que el de la secante, esto no invalida la capacidad de la bisección para reducir intervalos de tamaño considerable. Debido a esto, en caso de que se tenga un intervalo muy grande, podría entonces en una primera instancia emplearse el método de bisección para reducir el intervalo de manera abrupta y continuar entonces con el método de la secante.

Finalmente, podemos determinar, tanto por el video presentado como por los resultados obtenidos por el programa, que el método menos efectivo para aproximar una raíz es el de punto fijo. A pesar de parecer este el método más sencillo, realmente es uno de los más trabajosos. Si nos damos cuenta, su poca eficiencia radica en que depende de $g(x)$ para encontrar su raíz, y a pesar de que es una función que forzosamente debe cumplir las condiciones establecidas, esto no asegura la cercanía de la función propuesta con la raíz. Además de esto, podemos ver que en varias de las funciones utilizadas como ejemplo, este método diverge, mientras que el resto de los métodos sí nos proporcionan una aproximación. Esto sucede siempre que la magnitud de la derivada sea mayor que uno en el punto fijo. Con esto nos damos cuenta que el método de punto fijo debería ser, en casi todos los casos, nuestra última opción al momento de calcular una raíz.

5. Descripción del código

El código utilizado para automatizar las funciones es bastante sencillo. A continuación presentamos una breve descripción de cada uno de las funciones definidas para los métodos. Sin embargo invitamos al lector a leer el [código](#) y la [interfaz](#) por sí mismos, para comprender mejor cómo funciona este.

Para todos los métodos, se define en primer lugar la función a trabajar, el punto inicial, y final si es que lo requieren y la tolerancia

5.1. Punto fijo

Tiene como parámetros la función, el punto inicial y la tolerancia. Pasos:

1. Definimos una variable iniciada en 0, la cual será nuestro contador para la cantidad de pasos a realizar.
2. Evaluamos la función en nuestro punto inicial
3. Verificamos si el valor absoluto de la función evaluada es menor a la tolerancia. Si lo cumple retornamos un array con el contador y el valor del punto
4. Si no cumple, abrimos un ciclo condicional en el cual verificamos si $|f(x) - x| \geq \text{tolerancia}$ y además que la cantidad de pasos $< 10^3$
5. Dentro del ciclo, sumamos uno a la cantidad de pasos, tomamos como nuevo valor inicial el valor de la función evaluada y evaluamos la función en este nuevo valor
6. Al finalizar el ciclo devolvemos un array con la cantidad de pasos y el valor inicial

5.2. Bisección

Tiene como parámetros la función, el primer punto del intervalo, el segundo punto del intervalo y la tolerancia. Pasos:

1. Evaluamos la función en el punto inicial y final del intervalo
2. Establecemos nuestra variable contador
3. Definimos nuestro punto m
4. Evaluamos la función en m
5. Dentro de una condicional
6. Si la función evaluada en el punto inicial multiplicada por la función evaluada en el punto final es mayor a 0
 - a) Imprimir mensaje de error
 - b) Devolver array con valores -1 y 0
7. Si la función evaluada en el punto inicial es menor que la tolerancia
 - a) Devolver array con la cantidad de pasos y el valor inicial
8. Si la función evaluada en el punto final es menor que la tolerancia
 - a) Devolver array con la cantidad de pasos y el valor final
9. De cualquier otra manera
 - a) Abrir un ciclo condicional
 - 1) Verificar si el valor absoluto de $f(m) \geq$ la tolerancia
 - 2) Sumar uno a los pasos
 - 3) Calcular m
 - 4) Evaluar la función en m
 - 5) En una condicional
 - 6) Verificar si $f(m) = 0$
 - a' Imprimir que m es una raíz de la función
 - 7) Verificar si la función evaluada en el valor inicial multiplicada por $f(m)$ es mayor a 0
 - a' Definir el punto inicial como m
 - b' Definir la función evaluada en el valor inicial como $f(m)$
 - 8) En cualquier otro caso
 - a' Establecer el valor final como m
 - b' Establecer la función evaluada en el valor final como $f(m)$
 - b) Devolver un array con la cantidad de pasos y m

5.3. *Secante*

Tiene como parámetros la función, el primer punto del intervalo, el segundo punto del intervalo y la tolerancia. Pasos:

1. Establecemos nuestra variable contador
2. Evaluamos nuestra función en el valor inicial y en el final del intervalo
3. En una condicional
4. Si el valor absoluto de la función evaluada en el punto inicial es menor que la tolerancia
 - a) Devolver un array con la cantidad de pasos y el valor inicial
5. En cualquier otro caso
 - a) En un ciclo condicional
 - 1) Verificar si el valor absoluto de la función evaluada en el punto final
 - 2) Sumar 1 a la variable contador
 - 3) Obtener el segundo punto por la fórmula de la secante
 - 4) Establecer el punto inicial como el punto final
 - 5) Establecer el punto final como el segundo punto
 - 6) Establecer la función evaluada en el punto inicial como la evaluada en el punto final
 - 7) Establecer la función evaluada en el punto final como la evaluada en el segundo punto
 - b) devolver un array con la cantidad de pasos y el valor final

5.4. *Newton-Raphson*

Tiene como parámetros la función, el primer punto del intervalo y la tolerancia. Pasos:

1. Definir la variable contador
2. Evaluar la función en el punto inicial
3. En una condicional
4. Verificar si el valor absoluto de la función evaluada es menor a la tolerancia
 - a) Devolver un array con la cantidad de pasos y el valor inicial
5. De cualquier otra manera
 - a) Abrir un ciclo condicional
 - 1) Verificar si el valor absoluto de la función evaluada es mayor que la tolerancia
 - 2) Calcular la derivada de la función en el punto inicial
 - 3) Calcular el nuevo punto utilizando la fórmula de Newton
 - 4) Establecer la función evaluada como la función evaluada en el punto nuevo
 - 5) Sumar uno al contador
 - b) Devolver un array con la cantidad de pasos y el valor inicial

6. Conclusiones

Concluimos en este trabajo que el método que es más exacto y más sencillo de automatizar es el método de Newton. Sin embargo es poco eficiente debido a los requerimientos que este presenta. Los métodos más confiables que, si no se sabe dónde está la raíz, nos darán menos pasos son el de la secante y el de bisección. Consideramos que el automatizar los métodos ayuda a la comprensión de los mismos, así como a evitar los errores crasos que se cometen al calcularlos a mano

7. Bibliografía

Lizarralde, F. A. (2003, 26 agosto). Punto Fijo. Análisis Numérico. [http://www3.fi.mdp.edu.ar/analisis/temas/no_lineales_1/puntoFijo.htm#:~:text=El%20m%C3%A9todo%20de%20Punto%20Fijo,la%20funci%C3%B3n%20f\(x\)](http://www3.fi.mdp.edu.ar/analisis/temas/no_lineales_1/puntoFijo.htm#:~:text=El%20m%C3%A9todo%20de%20Punto%20Fijo,la%20funci%C3%B3n%20f(x)).

Sánchez Mora, M., & Maroto Alfaro, S. (s. f.). Métodos Numéricos. Métodos numéricos para la enseñanza. https://multimedia.uned.ac.cr/pem/metodos_numericos_ensenanza/modulo2/descripcionmetodo.html#:~:text=El%20m%C3%A9todo%20de%20bisecci%C3%B3n%20es,hay%20que%20realizar%20aumenta%20considerablemente.

M., R., E., M., S., A., A., Quintero, L. H. Q., S., C., & S. (2022, 1 febrero). Teorema de Bolzano — Superprof. Material Didáctico - Superprof. <https://www.superprof.es/apuntes/escolar/matematicas/calculo/derivadas/teorema-de-bolzano.html>

Díaz Villanueva, W. (1998, 5 noviembre). 4.4 Método de la secante. Método de la secante. <https://www.uv.es/~7Ediaz/mn/node21.html>

Lizarralde, F. A. (2003, agosto 21). Newton Raphson. Análisis Numérico. http://www3.fi.mdp.edu.ar/analisis/temas/no_lineales_1/newtonRaphson

D. (2020, 11 junio). de Taylor. Teorema. <https://www.teorema.top/teorema-de-taylor/>