Unidad 1 / Escenario 1

Lectura fundamental

Método de la bisección y método de Newton

Contenido

Palabras Claves: métodos iterativos, algoritmo, bisección, ecuaciones no lineales.

1. Introducción

Los métodos numéricos, más precisamente el análisis numérico es una rama de las matemáticas que surge en necesidad de encontrar soluciones exactas o soluciones aproximadas a problemas matemáticos provenientes del modelamiento matemático de distintos contextos de ingeniería, física, biología, finanzas, etc.
Particularmente en muchos problemas de ingeniería, al intentar encontrar soluciones matemáticas exactas, no siempre esto es posible a través de los métodos analíticos tradicionales. Por tanto, habrá que recurrir a otras técnicas o cálculo numérico. En pocas palabras, el uso de herramientas computacionales para la búsqueda aproximada de tales soluciones.

Este módulo pretende brindarle herramientas, más que desde su fundamento teórico, el poder conocerlas, distinguir su mecánica algorítmica y su implementación para encontrar soluciones aproximadas a distintos y complejos problemas.

1.1. Ecuaciones no lineales

En esta sección pretendemos recordar algunos conceptos básicos sobre ecuaciones no lineales y sobre todo, encontrar soluciones aproximadas en cuanto no sea posible encontrar soluciones exactas. Para ilustrar lo anterior, veamos el siguiente ejemplo.

Ejemplo 1. *La ecuación polinómica* $x^2 - 4x + 3 = 0$ *puede ser factorizable y por tanto, es posible encontrar sus soluciones exactas. Esto es:*

$$x^2 - 4x - 1 = 0$$

$$(x - 3)(x - 1) = 0,$$

donde $x = 3$ *y* $x = 1$ *satisfacen la ecuación. Es decir, son los puntos de corte en la gráfica en la figura 1*

Generalmente puede surgir otro tipo de ecuaciones no lineales como $e^x - x = \cos x$ donde no hay forma de poder factorizar o despejar la variable x. Así que el objetivo de esta unidad es estudiar algunos métodos numéricos para encontrar soluciones aproximadas a este tipo de problema.

2. Método de la bisección

2.1. Raíces de ecuaciones no lineales

Definición 1. *Para una función* f *continua en un intervalo* $[a, b]$*, la ecuación* $f(x) = 0$ *tiene una raíz* p *en dicho intervalo si* $f(p) = 0$*. Geométricamente esto significa que la gráfica de la función* f *corta al eje* x *justo en* p*.*

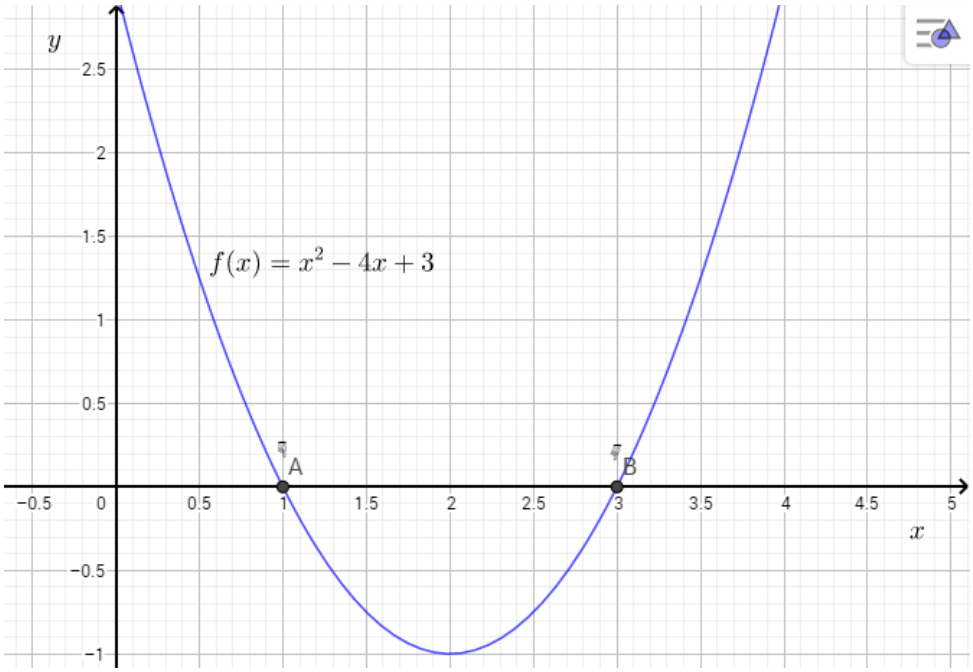

Figura 1: **Función con dos raíces exactas**
Fuente: elaboración propia

2.2. El teorema de Bolzano

Si queremos garantizar la existencia de una raíz en un intervalo, como mínimo la función deberá ser continua. Además, que tanto $f(a)$ como $f(b)$ tengan signos distintos. Es decir que, para los extremos del intervalo $[a, b]$ se cumpla que $f(a) \cdot f(b) < 0$, como se ilustra en la figura 2.

Con las anteriores condiciones, el siguiente teorema garantiza la existencia de raíces para estas funciones.

Teorema 1. ***Teorema de Bolzano.*** *Suponga que f es una función continua en el intervalo $[a, b]$ y que $f(a) \cdot f(b) < 0$, entonces existe un valor p en tal intervalo tal que $f(p) = 0$.*

Si bien el *teorema de Bolzano* garantiza la existencia de raíces en un intervalo, necesitaremos escoger intervalos suficientemente pequeños para que una raíz p sea única en dicho intervalo, de lo contrario los métodos numéricos podrían fallar.
La esencia de estos métodos iterativos es construir una sucesión $\{p_n\}_{n \in N}$ de números reales (o al menos de aritmética finita para el ordenador) con un dato inicial $p_0 \in [a, b]$ que converja a un valor p y tal que $f(p) = 0$. Es decir que si el método numérico es convergente, podremos obtener una sucesión de valores p_n que se aproximen tanto como se quiera a la solución p. Dicho en otros términos "si no podemos obtener una solución exacta p, entonces tendremos el consuelo de acercarnos a ella tanto como queramos".

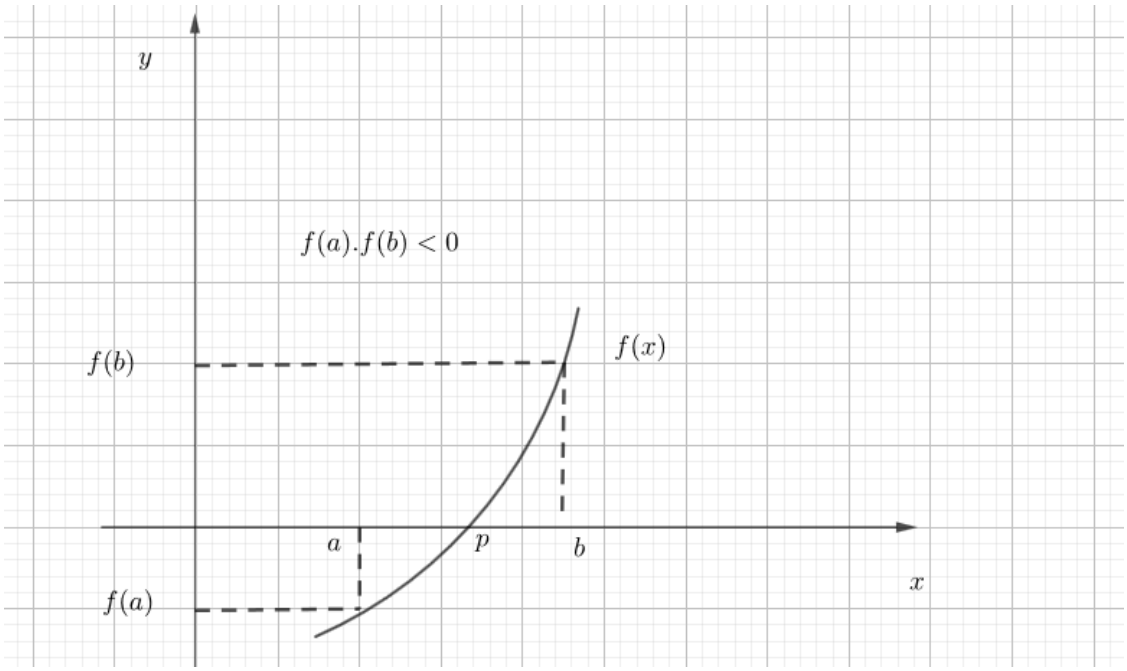

Figura 2. **Condiciones para el teorema de Bolzano**
Fuente: elaboración propia

2.3. El error como criterio de parada

Por lo anterior, si los métodos numéricos convergentes nos permiten acercarnos a la solución exacta, es de esperar que haya una pequeña diferencia (que consideramos como error) entre la solución exacta y la solución aproximada. De otro lado, es natural preguntar ¿en qué momento detener el proceso iterativo de la búsqueda de esa solución exacta?. Justamente basados en esa diferencia como error, podremos establecer algunos criterios de parada para tal proceso de búsqueda.

Definición 2. *Sea p_i y p_{i+1} dos iteraciones consecutivas. Por simplicidad, a la cantidad $|p_i - p_{i+1}|$ la llamaremos* ***error absoluto*** *y a la cantidad $\dfrac{|p_i - p_{i+1}|}{|p_{i+1}|}$* ***error relativo.***

Si tomamos algún valor E positivo suficientemente pequeño tal que

$$|p_i - p_{i+1}| < E$$

o

$$\frac{|p_i - p_{i+1}|}{|p_{i+1}|} < E,$$

entonces estamos exigiendo que a partir de la i-ésima iteración, los errores se hagan tan pequeños como queramos. Este valor E lo podemos pensar como nuestra "cota de satisfacción". Si el método iterativo la alcanza a cumplir, entonces la podremos usar como criterio de parada.

2.4. Método de la bisección y su interpretación geométrica

Bajo las condiciones del *teorema de Bolzano* y escogiendo un intervalo $[a, b]$ inicial moderadamente pequeño donde la función sea monótona (es decir, donde allí la gráfica de la función f sea creciente o al contrario sea decreciente) entonces es posible encontrar un valor p^* suficientemente cercano a la raíz p que lo consideraremos una solución aproximada de la ecuación $f(x) = 0$.

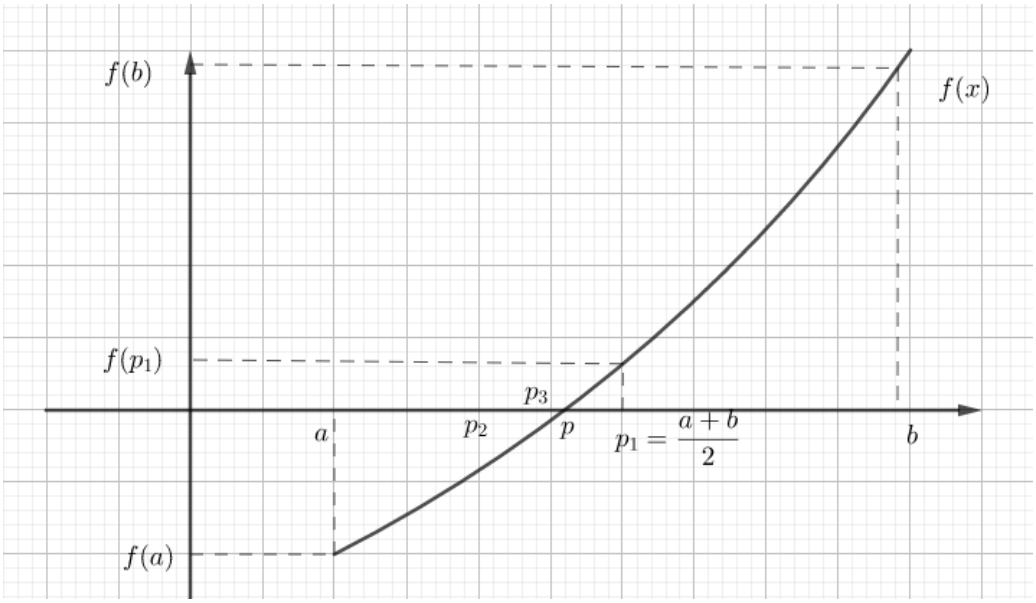

Figura 3. **Método de la bisección**
Fuente: elaboración propia

El método se basa en dividir el intervalo inicial $[a, b]$ en dos subintervalos tomando el punto medio $\frac{a+b}{2}$ para luego determinar en cual de ellos se encuentra la raíz buscada y luego repetir este procedimiento n-veces hasta "encerrar" a la raíz en un subintervalo $[a_n, b_n]$ suficientemente pequeño, incluso más pequeño que la cota de error E.

La mecánica del método consiste en tomar los extremos del intervalo inicial como $a_1 = a$ y $b_1 = b$, calcular el punto medio del intervalo y definirlo como $p_1 = \frac{a+b}{2}$, luego calcular las imágenes de estos puntos $f(a), f(b)$ y $f(p_1)$ (ver figura 3):

- Si $f(a) \cdot f(p_1) < 0$ entonces $p \in [a_1, p_1]$ y hacemos $a_2 = a$, $b_2 = p_1$. Es decir, que si la imagen del extremo izquierdo del intervalo a_1 tiene signo distinto a la imagen del punto medio del intervalo p_1, entonces la raíz p deberá estar en el subintervalo de la izquierda $[a_1, p_1]$ y no en el subintervalo de la derecha $[p_1, b_1]$. Por tanto nos quedaremos con el subintervalo $[a_1, p_1]$, dejaremos el mismo extremo izquierdo y el punto medio p_1 será el nuevo extremo derecho b_2 en el nuevo subintervalo $[a_2, b_2]$.
- Si $f(a) \cdot f(p_1) > 0$ entonces $p \in [p_1, b_1]$ y hacemos $a_2 = p_1$, $b_2 = b1$. *Se recomienda como ejercicio, dar una explicación análoga a la anterior para asimilar la mecánica del método.*

- Volvemos a aplicar lo anterior, tomando el punto medio del nuevo subintervalo donde se encuentra la raíz como $p_2 = \dfrac{a_2 + b_2}{2}$.

Con esto, obtenemos una sucesión de puntos $\{p_0, p_1, p_1, \cdots, \}$ los cuales convergen a p. Ver la siguiente figura 4

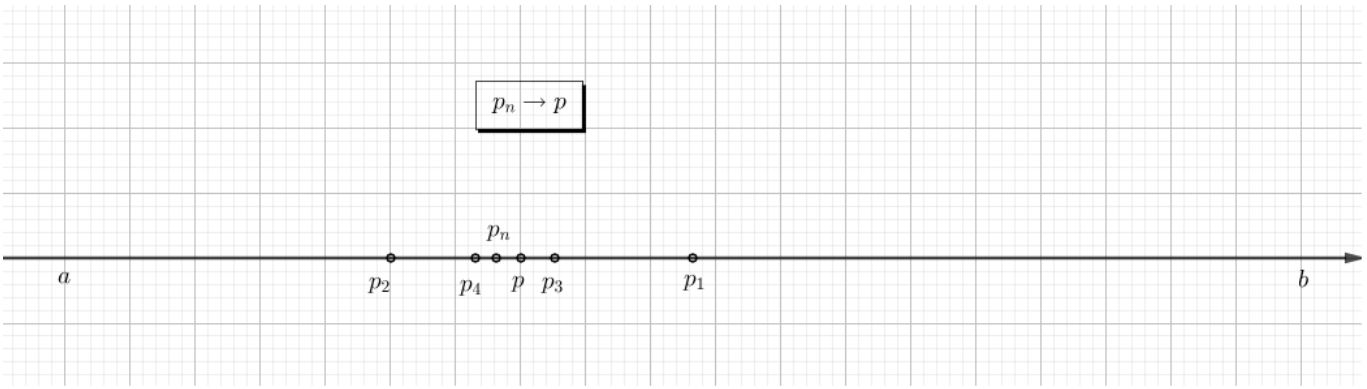

Figura 4. **Sucesión de puntos que convergen a** p
Fuente: elaboración propia

2.5. Ejemplos y tabla de valores

Ejemplo 2. *Calcular las primeras 10 iteraciones del método de la bisección para la ecuación* $10x^4 - 3xe^x - 3e^x = 0$ *en el intervalo* $[-1, -0.25]$. *En la figura 5 se muestra la función de forma general y en la figura 6 se puede apreciar la raíz de la ecuación dada.*

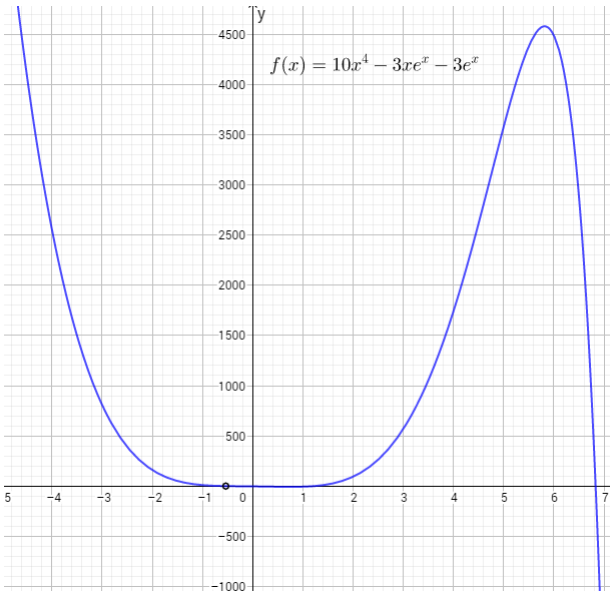

Figura 5. **Vista global de la función** $f(x) = 10x^4 - 3xe^x - 3e^x$
Fuente: elaboración propia

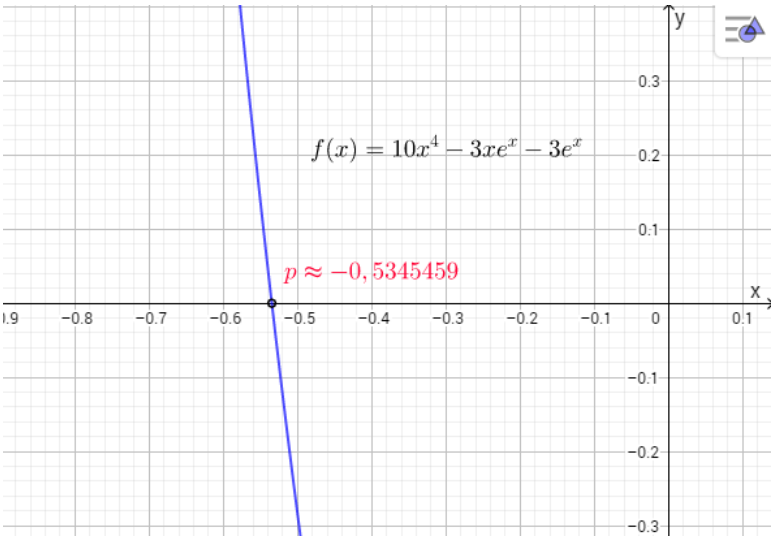

Figura 6. **Vista ampliada y local de la función** $f(x) = 10x^4 - 3xe^x - 3e^x$
Fuente: elaboración propia

A continuación se muestra en tabla 1 los valores generados por el algoritmo de la bisección

Tabla 1. **Método de la bisección para** $f(x) = 10x^4 - 3xe^x - 3e^x$

n	a_n	p_n	b_n	$f(p_n)$	Error relativo
0	-1	-0,625	-0,25	0,9237098	0,428571429
1	-0,625	-0,4375	-0,25	-0,72316836	0,176470588
2	-0,625	-0,53125	-0,4375	-0,0301734	0,081081081
3	-0,625	-0,578125	-0,53125	0,40713572	0,042253521
4	-0,578125	-0,5546875	-0,53125	0,17949452	0,021582734
5	-0,5546875	-0,54296875	-0,53125	0,07252525	0,010909091
6	-0,54296875	-0,53710938	-0,53125	0,02065578	0,005484461
7	-0,53710938	-0,53417969	-0,53125	-0,00488716	0,002734731
8	-0,53710938	-0,53564453	-0,53417969	0,00785201	0,001369238
9	-0,53564453	-0,53491211	-0,53417969	0,00147438	0,000685088
10	-0,53491211	-0,5345459	-0,53417969	-0,0017084	0,000342427

Fuente: elaboración propia

Si quisiéramos saber un número estimado d e iteraciones para lograr cierta cota de error E utilizando como criterio de parada el error absoluto, podemos usar la fórmula

$$N = \left[\left| \frac{ln\left(\frac{b-a}{E}\right)}{ln(2)} \right|\right] \tag{1}$$

Donde N es el número de iteraciones estimadas, [| |] denota el mayor entero y ln la función logaritmo natural.

Ejemplo 3. *En el ejemplo anterior usamos como criterio de parada el error relativo, sin embargo la fórmula anterior nos brinda una estimación para el número de iteraciones necesarias si queremos obtener una raíz aproximada*

con una precisión de 10^{-2}. *Al sustituir los valores dados en la fórmula*

$$N = \left\lceil \left| \frac{ln\left(\frac{-0.25+1}{10^{-2}}\right)}{ln(2)} \right| \right\rceil = [|6.22|] \approxeq 6,$$

obtenemos una estimación de 6 iteraciones.

Algunas consideraciones sobre el método de la bisección y su algoritmo

1. La fórmula (1) brinda solo una cota estimada del número de iteraciones necesarias, sin embargo esta estimación podría ser mucho mayor a la que en verdad se necesite para algunos casos.
2. El método es siempre convergente. Esto significa que si la raíz que buscamos está dentro del intervalo inicial de donde parte el método, siempre es posible aproximarnos a esta con la precisión que se desee. Esta es una gran característica a favor del método de la bisección.
3. La elección en el tamaño del intervalo inicial, ha de tenerse en cuenta en la eficiencia del método, pues si es suficientemente pequeño la aproximación buscada se logrará en menos iteraciones.
4. La consistencia del método de la bisección (y por tanto de su algoritmo) se puede ver seriamente afectada ante la presencia de distintas raíces en el mismo intervalo considerado.
5. Con respecto a la "velocidad de convergencia" es decir, que tan rápido se alcanza la precisión deseada en por cada iteración que se realice, en general el método de la bisección es lenta. Esta noción de velocidad de convergencia será tratada con más detalle en el próximo escenario como el concepto de *orden de convergencia* el cual pretende comparar la eficiencia de los métodos iterativos.
6. Motivados por encontrar algún método que converja mucho más rápido que el método de la bisección a la solución buscada, se presenta el siguiente, llamado el *método de Newton*.

2.6. Algoritmo para el método de la bisección

La figura (7) muestra el algoritmo del método de la bisección ejecutado en el lenguaje python[1] para el ejemplo 2 con los parámetros $a = -1, b = -0.25, \epsilon = 10^{-4}, n = 100$. La figura (8) muestra los valores de salida para dicho ejemplo, en el cual, se aprecia que la raíz buscada para la ecuación dada con los parámetros descritos es $p = -0.5348$.

El cuadro (2) contiene el pseudoalgoritmo para el método de la bisección.

[1] Algoritmo tomado del libro [1]

Tabla 2. **Pseudocódigo para el método de la bisección**

DATOS:	a, b, precisión *eps*, máximo de iteraciones N_0.
RESULTADOS:	Raíz aproximada p o mensaje de error.
	Paso 1: iniciar $i = 1$;
	Paso 2: mientras que $i \leq N_0$ haga los pasos 3-6;
	Paso 3: defina $p = \frac{b-a}{2}$;
	Paso 4: si $f(p) = 0$ o $\frac{b-a}{2} < eps$ entonces RESULTADOS p; PARAR;
	Paso 5: redefinir $i = i + 1$;
	Paso 6: si $f(a) \cdot f(p) > 0$ entonces hacer $a = p$, caso contrario hacer $b = p$;
	Paso 7: SALIDA "Procemimiento sin éxito"; PARAR;

Fuente: elaboración propia

```
from math import*
def f(x):
    return 10*x**4-3*x*exp(x)-3*exp(x)
def biseccion(f, a, b, tol, n):
    i=1
    while i<=n:
        p=a+(b-a)/2.0
        print ("Iter=", "%03d" % i, " ; p=", "%.14f" % p)
        if abs(f(p))<= 1e-15 or (b-a)/2.0 < tol:
            return p
        i+=1
        if f(a)*f(p) > 0:
            a=p
        else:
            b=p
    print("Iteraciones agotadas: Error")
    return
print("\n"+r"Método de la bisección:"+"\n")
biseccion(f, -1.0, -0.25, 1e-4, 100)
```

Figura 7. **Algoritmo en lenguaje python para la ecuación** $10x^4 - 3xe^x - 3e^x = 0$
Fuente: elaboración propia

3. Método de Newton

El método de Newton es una de las técnicas más eficientes para nuestro propósito de calcular raíces de ecuaciones no lineales.

3.1. Interpretación geométrica

Describiremos su mecánica de forma geométrica en la figura 9. Bajo ciertas condiciones de diferenciabilidad para la función f, la idea básica consiste en tomar un punto inicial p_0 bastante cerca a la raíz p buscada. Luego calcular

```
Método de la bisección:

Iter= 001  ; p= -0.62500000000000
Iter= 002  ; p= -0.43750000000000
Iter= 003  ; p= -0.53125000000000
Iter= 004  ; p= -0.57812500000000
Iter= 005  ; p= -0.55468750000000
Iter= 006  ; p= -0.54296875000000
Iter= 007  ; p= -0.53710937500000
Iter= 008  ; p= -0.53417968750000
Iter= 009  ; p= -0.53564453125000
Iter= 010  ; p= -0.53491210937500
Iter= 011  ; p= -0.53454589843750
Iter= 012  ; p= -0.53472900390625
Iter= 013  ; p= -0.53482055664062
```

Figura 8. **Salidas en lenguaje python para la ecuación** $10x^4 - 3xe^x - 3e^x = 0$
Fuente: elaboración propia

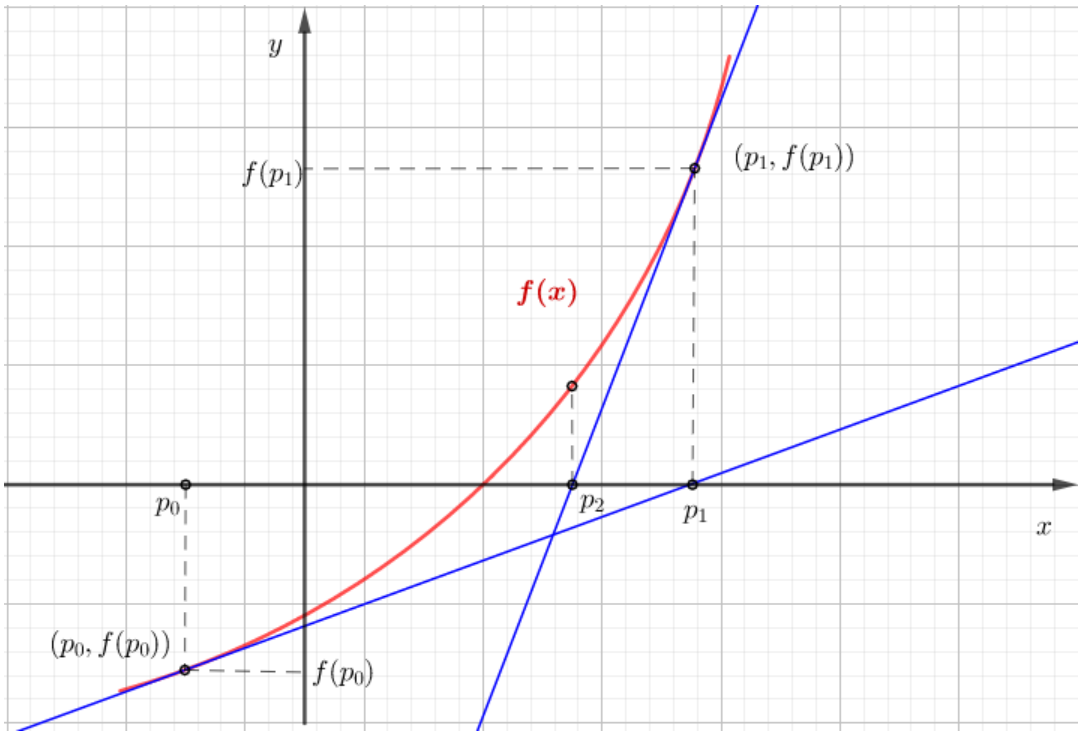

Figura 9. **Método de Newton**
Fuente: elaboración propia

su imagen $f(p_0)$ y tomar el punto $(p_0, f(p_0))$ sobre la curva. Luego, trazar la recta tangente que pasa por tal punto, la cual cortará el eje x. En este corte de la recta tangente con el eje x lo llamaremos p_1. Ahora con p_1 repetimos el proceso iterativamente. Esto generará una sucesión de puntos $p_0, p_1, p_2, \ldots, p_n$ los cuales convergen a p.
Esta sucesión de puntos puede ser obtenida y representada por la fórmula recurrente

$$p_n = p_{n-1} - \frac{f(p_{n-1}}{f'(p_{n-1})},$$

tomando como dato inicial p_0 y donde p_{n-1} es el valor obtenido en la iteración anterior a la n-ésima iteración.

Antes de enunciar el teorema que establece las condiciones para que el método de Newton sea convergente, motivemos tal método a partir del siguiente ejemplo.

3.2. Ejemplo y tabla de valores

Ejemplo 4. *Apliquemos el método de Newton a la ecuación* $2sen(x) - x = 0$ *para obtener una raíz aproximada en el intervalo* $[-3, -1.5]$ *con una exactitud de* 10^{-9}.

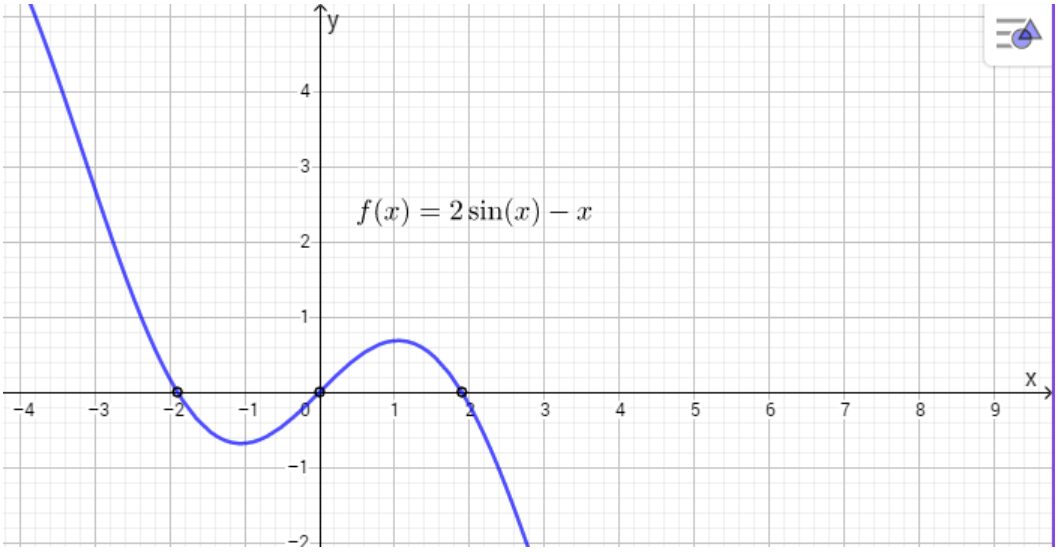

Figura 10. **Vista global de la función** $f(x) = 2sen(x) - x$
Fuente: elaboración propia

Tenemos que $f(x) = 2sen(x) - x$ *y* $f'(x) = 2\cos(x) - x$ *así,*

$$p_n = p_{n-1} - \frac{2sen(p_{n-1}) - p_{n-1}}{2\cos(p_{n-1}) - 1}.$$

Tomando como dato inicial $p_0 = -2.5$ *obtenemos nuestro primer valor* p_1

$$p_1 = p_0 - \frac{2sen(p_0) - p_0}{2\cos(p_0) - 1} = -2.5 - \frac{1.303055712}{-2.60228723} = -1.99926522.$$

Los demás valores se resumen a continuación en la tabla 3.

Tal precisión se logra con la quinta iteración, produciendo un valor de la raíz $p_5 = -1,89549427$. *De hecho, a partir de la quinta iteración se obtiene la raíz exacta.*

3.3. Teorema de convergencia para el método de Newton

Teorema 2. *Sea* f *una función con segunda derivada continua en el intervalo* (a, b), *tal que* f *tenga una raíz* p *en el intervalo* $[a, b]$ *y cuya derivada* f' *no sea nula en* p. *Entonces, el método de Newton genera una sucesión de valores* $\{p_n\}_{n\in N}$ *que converge a* p *en algún intervalo, si* p_0 *está lo suficientemente cerca de la raíz* p.

Tabla 3. **Método de Newton para** $f(x) = 2sen(x) - x$

n	p_n	$f(p_n)$	Error absoluto
0	-2,5	1,30305571	0,500734775
1	-1,99926522	0,18005931	0,098341629
2	-1,9009236	0,0089214	0,0054124
3	-1,8955112	2,7729E-05	1,6928E-05
4	-1,89549427	2,7158E-10	1,65796E-10
5	-1,89549427	0	0

Fuente: elaboración propia

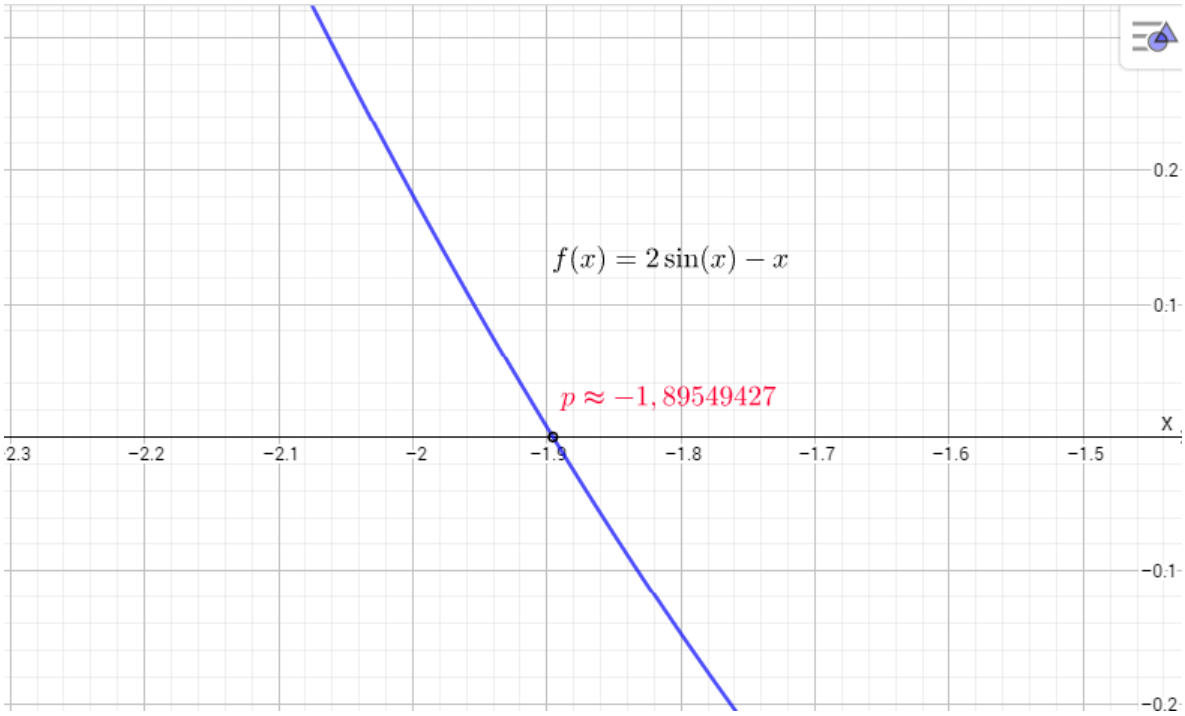

Figura 11. **Vista local y ampliada de la función** $f(x) = 2sen(x) - x$
Fuente: elaboración propia

En el ejemplo anterior $f(x) = 2sen(x) - x$ tiene de hecho infinitas derivadas, aunque nos basta con la segunda. Para verificar que la derivada no se anula en la raíz p, es suficiente mostrar que no se anula para ningún valor del intervalo dado $[-3, -1.5]$.
El análisis algebraico consiste en igualar la derivada a cero y despejar la variable. Esto es,

$$\begin{aligned} f'(x) &= 0 \\ 2\cos x - 1 &= 0 \\ \cos x &= \frac{1}{2} \\ x &\approx \pm 1.047. \end{aligned}$$

Es decir que en los valores $x = -1.047$ y $x = 1.047$ (ver figura 12) la recta tangente es paralela al eje x, pero nuestro intervalo $[-3, -1.5]$ no contiene estos valores. Por tanto $f(x) = 2sen(x) - x$ cumple las condiciones del teorema y es de esperarse que converja el método de Newton.

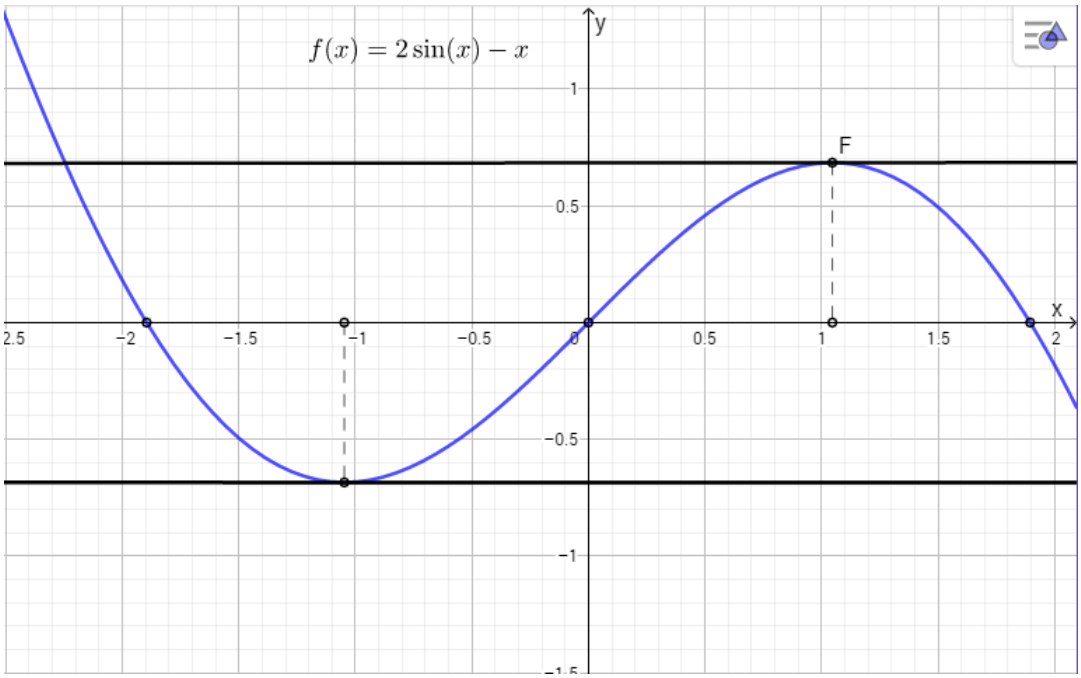

Figura 12. **Puntos donde la derivada f' se anula para función $f(x) = 2sen(x) - x$.**
Fuente: elaboración propia

Algunas consideraciones sobre el método de Newton y su algoritmo

1. El método de Newton bajo las condiciones del teorema (2) es muy efectivo, siempre y cuando el valor inicial p_0 sea bastante cercano a la raíz p. Esto, en términos algorítmicos se traduce en la precondición de elegir un buen valor inicial p_0 para alcanzar rápidamente la convergencia del método. Caso contrario, el método de Newton podría converger lentamente o incluso no hacerlo.

2. ¿Es posible combinar los métodos numéricos para aumentar su eficiencia? La respuesta es sí. De hecho, es posible iniciar la búsqueda de una raíz con el método de la bisección (que aunque lento su convergencia está asegurada) hasta hasta cierto valor p_k lo suficientemente cercano de la raíz para que a partir de allí, este valor sirva de valor inicial para implementar el método de Newton y así acelerar su convergencia.

3. Un punto débil del método de Newton es que puede verse afectado ante la presencia de distintas raíces en un mismo intervalo. Aunque el método de Newton se considera un método abierto en el sentido de no requerir ser aplicado dentro de un intervalo específico (como el método de la bisección), este podría fallar si existen distintas raíces cercanas.

4. Otra posible desventaja de este método es la necesidad de saber la derivada de la función y evaluarla en cada iteración lo cual requiere mayor número de operaciones y cálculo numérico para el ordenador. En el caso de los polinomios su derivada no representa mayor esfuerzo, sin embargo, en funciones más complicadas de derivar se requiere mayor trabajo computacional.

5. Ante la desventaja del método de Newton por exigir la derivada de f, existe una alternativa y es el método de la secante de la siguiente lectura.

3.4. Algoritmo para el método de Newton

Analicemos el comportamiento del algoritmo del método de Newton, aplicado al mismo ejemplo 4 y representado en la figura (10), cuando tomamos un valor inicial muy cercano a algún punto critico de la función dada $f(x) = 2\,\text{sen}(x) - x$.

Supongamos que se desea estimar la raíz que vale aproximadamente $p = 1.8$ partiendo del valor inicial $p_0 = 1.0$. Por el análisis hecho anteriormente, esta función tiene un punto crítico en $x = 1.047$, es decir, ahí la derivada vale cero, lo que puede afectar el rendimiento del método de Newton.

Usando el algoritmo respectivo en el lenguaje python[2], la figura (13) muestra el algoritmo con los siguientes parámetros: valor inicial $p_0 = 1.0$, valor de tolerancia $\epsilon = 10^{-10}$ y un máximo de iteraciones $n = 100$.

La figura (14) muestra las salidas de las estimaciones de la raíz deseada. Debemos notar, que en las primeras 9 iteraciones los valores oscilan fuertemente con el ánimo de divergencia. Luego, se consigue entrar en alguna vecindad cerca de la raíz buscada donde se logra la convergencia. Esto evidencia el riesgo de tomar el valor inicial o muy lejos de la raíz o muy cercano a valores críticos de la función.

```
 8 from math import*
 9 def trig(x):
10     return 2*sin(x)-x
11 def trigprima(x):
12     return 2*cos(x)-1
13 def newton(f, fprima, p0, tol, n):
14     i=1
15     while i<=n:
16         p=p0-f(p0)/fprima(p0)
17         print ("Iter=", "%03d" % i, " ; p=", "%.14f" % p)
18         if abs(p-p0)<tol:
19             return p
20         p0=p
21         i+=1
22     print("Iteraciones agotadas: Error")
23     return
24 print("\n"+r"Método de Newton para la ecuación 2sen(x)-x=0:"+"\n")
25 newton(trig, trigprima, 1, 1e-10, 100)
```

Figura 13. **Algoritmo en lenguaje python para la ecuación** $2sen(x) - x = 0$
Fuente: elaboración propia

Finalmente, el cuadro (4) proporciona el pseudocódigo del método de Newton.

[2] Algoritmo tomado y modificado del libro [1].

```
Método de Newton para la ecuación 2sen(x)-x=0:

Iter= 001  ; p= -7.47274063983125
Iter= 002  ; p= 14.47852098287016
Iter= 003  ; p= 6.93511540801481
Iter= 004  ; p= 16.63568412080635
Iter= 005  ; p= 8.34393755308490
Iter= 006  ; p= 4.95463273595273
Iter= 007  ; p= -8.30131857221435
Iter= 008  ; p= -4.81732186894687
Iter= 009  ; p= 3.79261640758188
Iter= 010  ; p= 1.86102306735813
Iter= 011  ; p= 1.89621583503402
Iter= 012  ; p= 1.89549456797983
Iter= 013  ; p= 1.89549426703403
Iter= 014  ; p= 1.89549426703398
```

Figura 14. **Salidas en lenguaje python para la ecuación** $2\,\mathrm{sen}(x) - x = 0$
Fuente: elaboración propia

Tabla 4. **Pseudocódigo para el método de Newton**

DATOS:	Valor inicial p_0, precisión eps, máximo de iteraciones N_0.
RESULTADOS:	Raíz aproximada p o mensaje de error.
	Paso 1: iniciar $i = 1$;
	Paso 2: mientras que $i \leq N_0$ haga los pasos 3-6;
	Paso 3: defina $p = p_0 - \frac{f(p_0)}{f'(p_0)}$;
	Paso 4: si $\lvert p - p_0 \rvert < 0$ entonces RESULTADOS p; PARAR;
	Paso 5: redefinir $i = i + 1$;
	Paso 6: hacer $p_0 = p$;
	Paso 7: SALIDA "Procemimiento sin éxito"; PARAR;

Fuente: elaboración propia

Referencias

[1] Arévalo, O. D., Bernal, Y. M. Á. y Posada, R. J. A. (2017). *Matemáticas para ingeniería: métodos numéricos con python.* Recuperado de https://ebookcentral-proquest-com.loginbiblio.poligran.edu.co

[2] Chapra, S. C., y Canale, R. P. (2007). *Métodos numéricos para ingenieros* (5a. ed.). Recuperado de https://ebookcentral-proquest-com.loginbiblio.poligran.edu.co

[3] Nieves, H. A. (2014). *Métodos numéricos: aplicados a la ingeniería.* Recuperado de https://ebookcentral-proquest-com.loginbiblio.poligran.edu.co

[4] Burden, R. L., Faires, J. D. (2007). *Análisis numérico* (7a. ed.). Thomsom Learning.

[5] Vázquez, L., Jiménez, S. (2009). *Métodos numéricos para la física y la ingeniería.* Recuperado de https://ebookcentral-proquest-com.loginbiblio.poligran.edu.co

INFORMACIÓN TÉCNICA

Módulo: Métodos Numéricos
Unidad 1: Solución de ecuaciones no lineales
Escenario 1: Método de la bisección y método de Newton

Autor: Joselin Montealegre Martínez

Asesor Pedagógico: Heidy LiLiana Moncada
Diseñador Gráfico: Santiago Rodriguez
Corrector de estilo: María Alejandra Romero Isaza
Asistente: Ginna Paola Quiroga Espinosa