

Pontificia Universidad  
**JAVERIANA**

FACULTAD DE INGENIERÍA

DEPARTAMENTO DE INGENIERÍA DE SISTEMAS

## **Reto 1**

Andres Garcia  
William Rodriguez  
Felipe Ariza

---

Análisis numérico

Eddy Herrera Daza

Bogotá, Marzo 2021

# Índice

|                                                                   |           |
|-------------------------------------------------------------------|-----------|
| <b>1. Algoritmo Brent</b>                                         | <b>2</b>  |
| 1.1. Explicación del método de Brent . . . . .                    | 2         |
| 1.2. Validaciones y precision. . . . .                            | 3         |
| 1.3. Aplicación del algoritmo. . . . .                            | 4         |
| 1.4. Analisis de resultados. . . . .                              | 5         |
| 1.5. Figuras brent. . . . .                                       | 5         |
| 1.6. Ejecución del algoritmo. . . . .                             | 6         |
| 1.7. Conclusión. . . . .                                          | 7         |
| <b>2. Metodo de la secante para la intersección entre curvas.</b> | <b>8</b>  |
| 2.1. Primera ecuación. . . . .                                    | 8         |
| 2.2. Segunda ecuación. . . . .                                    | 9         |
| 2.3. Prueba. . . . .                                              | 9         |
| 2.4. Analisis del resultado. . . . .                              | 9         |
| <b>3. Librerías en R y/o Python.</b>                              | <b>9</b>  |
| <b>4. Referencias.</b>                                            | <b>10</b> |

## 1. Algoritmo Brent

El algoritmo de Brent [1], utiliza en cada punto lo mas conveniente de las estrategias de el algoritmo de la bisección y de la secante (o Muller). Cuenta con una confiabilidad parecida a la bisección, sin embargo es bastante rápido en el momento de su ejecución. Este método se debe a Richard Brent y se basa a un algoritmo anterior de Theodorus Dekker. Por lo cual también se le conoce como método Brent-Dekker. El método es muy popular para encontrar ceros en las funciones desde su desarrollo. El algoritmo suele converger rápidamente a 0.

### 1.1. Explicación del método de Brent

El algoritmo de Brent, como se fue mencionado anteriormente, combina el método de la bisección con el método de la secante, para el calculo de una raíz de forma eficiente. El algoritmo verifica si los limites del intervalo tienen el mismo signo, si llegan a ser diferentes hace una iteración con el método de la secante, y si no evalúa si el resultado arrojado esta entre el siguiente intervalo, y si se encuentra aplica nuevamente el método de la secante, si no es así aplica el método de la bisección. El algoritmo reproduce el mismo procedimiento hasta encontrare el valor de la raíz o llegar al máximo de las iteraciones.

La condición utilizada por el método para encontrar los valores intermedios es la secante en cada iteración y su siguiente iteración evaluados en la función sean distintos. Tengase en cuenta que  $b_k$  y  $a_k$  son contrapuntos:

$$b_k - \frac{(b_k - b_{k-1})}{(f(b_k) - f(b_{k-1}))} * f(b_k)$$

Y también el condicional con el método de la bisección:

$$m = \frac{(a_k + b_k)}{2}$$

## 1.2. Validaciones y precision.

Conociendo los resultados esperados de un decimal periodico, se agrega una excepción al algoritmo de Brent, buscando llenar la posible tendencia de un numero a ser periodico, como parametro se compara un resultado y si este en su parte decimal lleva mas de tres numeros seguidos iguales buscando una fracción correspondiente a un decimal periodico y se evalua en la funcion con el objetivo de obtener un resultado igual a 0 y terminar con el algoritmo. Para lograr esta fracción se usan las librerias "Decimalz "fractions" para llegar a la precision que el algoritmo no llega a dar con su simple ejecución. En caso

de que no llegue a suceder se garantiza la precisión usando el tipo de dato float128 encontrado en la libreria numpy.

### 1.3. Aplicación del algoritmo.

Siguiendo con el orden, debemos evaluar la siguiente ecuación para solucionar la problematica de:

$$f(x) = x^3 - 2x^2 + 4x/3 - 8/27$$

Al implementar la ecuación, podemos notar que cumple con los resultados esperados, al mejorar la precision de los resultados modificando su tolerancia podemos observar los resultados expresados en las figuras 1, figura 2 y figura 3. Donde podemos observar con detalle que a mayor tolerancia la grafica obtiene mas detalles. Los resultados pueden ser comparables y podemos verlos a continuación:

Raíces con tolerancia 1e-8

Método de brent: 0.668673449222, iteraciones: 27

Raíces con tolerancia 1e-16

Método de Brent: 0.666671439616, iteraciones: 48

Raíces con tolerancia 1e-32

Método de Brent: 0.666669872646, iteraciones: 74

#### 1.4. Analisis de resultados.

Como podemos observar entre mas iteraciones realice el metodo, el resultado aproximado se ve afectado llegando a ser cada vez mas preciso. Comparandolo con las figuras el cambio se hace cada vez mas presente, dandonos un esperado bastante bueno en su precision. Comparandolo con otros metodos, al aumentar su tolerancia se observa que el numero de iteraciones aumenta considerablemente al punto de ser mucho mas eficiente en sus resultados que otros metodos.

#### 1.5. Figuras brent.

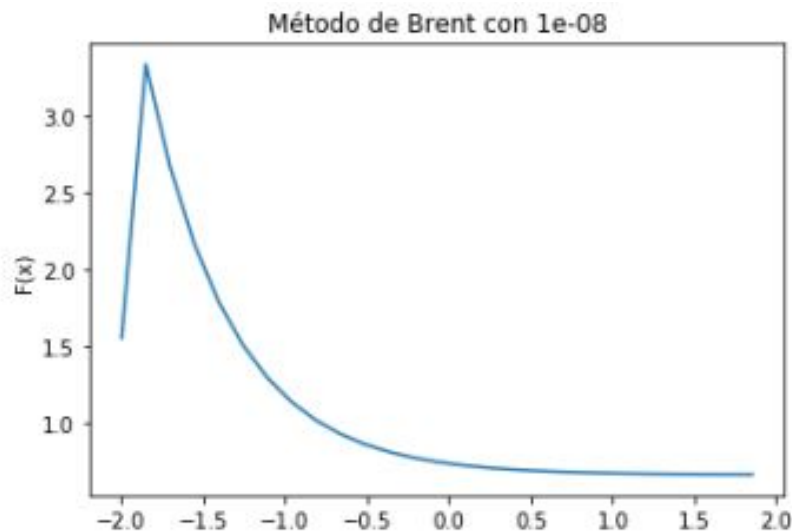


Figura 1

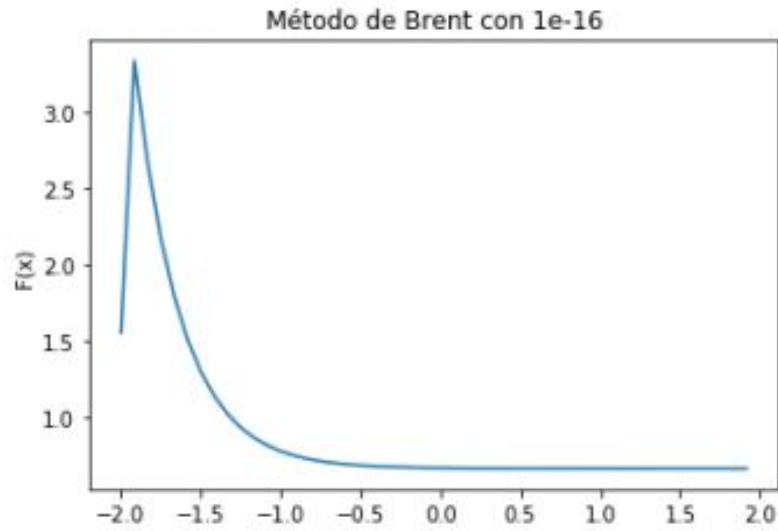


Figura 2

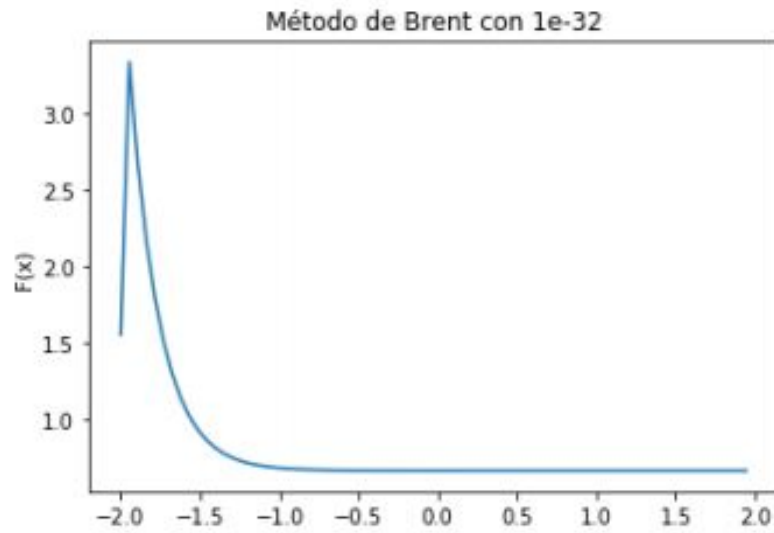


Figura 3

### 1.6. Ejecución del algoritmo.

El algoritmo recibe como parametro la funcion evaluar, limite inicial, limite final, una tolerancia y un maximo de iteraciones. A continuación intenta

cumplir con algunas condiciones:

- Verificar que la multiplicación resultante de evaluar la función en un punto  $a$  y un punto  $b$  sea menor que 0.
- 1. Se buscan unos nuevos puntos  $a$  y  $b$  con la función secante
- 1.1 De no ser así se comprueba si se puede aplicar nuevamente al función de secante
- 1.2 De no ser así se aplica la función de bisección para unos puntos  $c$  y  $b$
- 2. Se repite el paso de la verificación sin sus caminos alternos y si se cumple nuevamente la condición, se guarda como resultado
- 3. Si al evaluar la función en un punto  $b$ , el valor resultante es menor a la tolerancia establecida se termina el algoritmo de Brent.

### **1.7. Conclusión.**

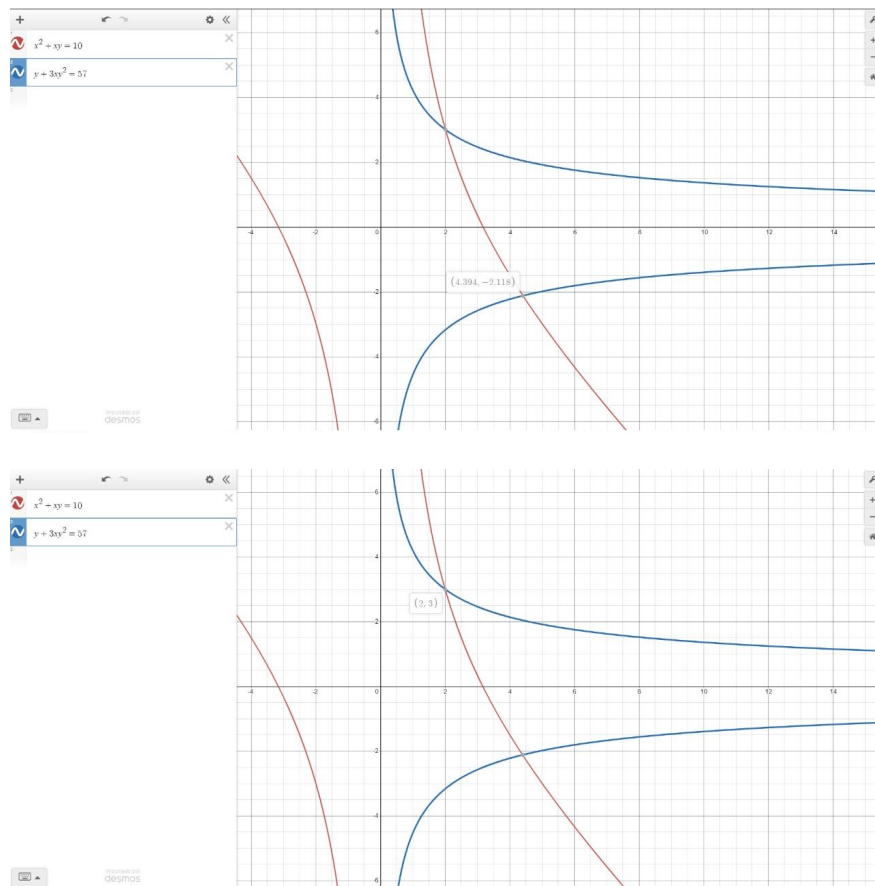
Para concluir, se puede decir que el algoritmo de brent implementado resulta ser mas rapido que otros metodos, pero no es lo suficiente para llegar a dejar algoritmos como el de la bisección descartados.

Por otra parte, si la tolerancia utilizada en la implementación del metodo es demasiado baja es recomendable recurrir a otro metodo.



## 2. Metodo de la secante para la intersección entre curvas.

Para la solución de este punto usamos el método de la secante que lo aplicamos a la simplificación de las dos ecuaciones, donde en la primera despejamos  $y$  en términos de  $x$  y en la segunda ecuación despejamos  $x$  en términos de  $y$ .



### 2.1. Primera ecuación.

$$x^2 + xy = 10 \rightarrow y = (-x^2 + 10)/x$$

$$y + 3xy^2 = 57 \rightarrow y + 3x((-x^2 + 10)/x)^2 - 57 = 0$$

## 2.2. Segunda ecuación.

$$y + (3xy^2 = 57).$$

$$x = (57-y/3y^2).$$

$$(57-y/3y^2) + (57-y/3y^2)y - 10 = 0.$$

## 2.3. Prueba.

```
Tolerancia: 1.52587890625e-05
Iteracion-1, x2 = 7.500000 Y f(x2) = 792.458333
Iteracion-2, x2 = 4.738661 Y f(x2) = 38.579567
Iteracion-3, x2 = 4.597350 Y f(x2) = 21.495414
Iteracion-4, x2 = 4.419551 Y f(x2) = 2.524032
Iteracion-5, x2 = 4.395896 Y f(x2) = 0.208291
Iteracion-6, x2 = 4.393769 Y f(x2) = 0.002373
Iteracion-7, x2 = 4.393744 Y f(x2) = 0.000002

Resultado de la raiz: 4.39374421686911809104003623360768
Punto Interseccion x2 = 4.39374421686911809104003623360768 , -2.1177810505094316
6176580234605353
>>> |
```

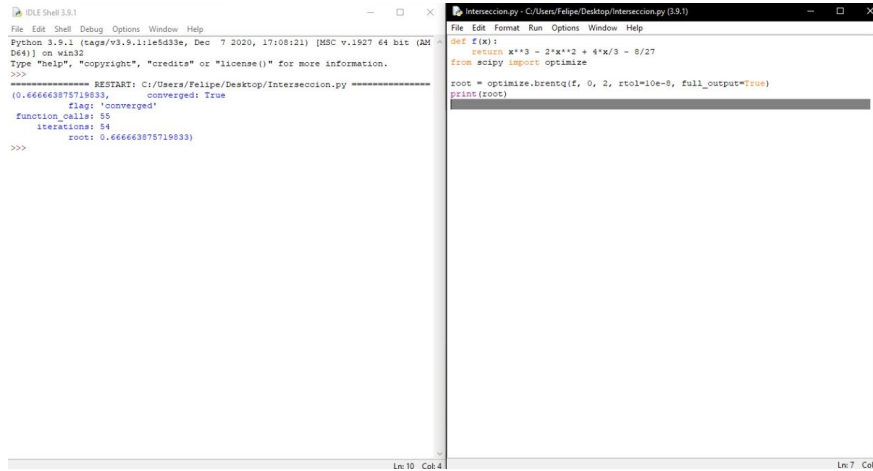
## 2.4. Analisis del resultado.

En conclusión, el algoritmo realizado nos mostro la aproximación correcta donde se puede observar que el método de la secante sirve para hallar la intersección, pero en este caso no fue posible hallar el resultado de y ya que se cometieron errores al momento de intentar implementar.

## 3. Librerías en R y/o Python.

[1]En las librerías numpy y scipy se pueden encontrar múltiples funciones que pueden ayudar a la solución de estos problemas, para el primer punto con el método de brent encontramos una función que nos ayuda a solucionar la función para poder así hallar el valor de la raíz, esta función es optimize.brentq() siendo está bastante útil ya que recibe múltiples parámetros que se acomodan a lo demandado, entre estos parámetros se encuentra la función, los límites, la tolerancia y además de mostrar toda la solución donde se puede

observar si la función converge o no y el número de iteraciones que tomo para hallar la solución.



```
Python Shell 3.8.1
File Edit Shell Debug Options Window Help
Python 3.8.1 (tags/v3.8.1:11e5d33e, Dec 7 2020, 17:08:12) [MSC v.1927 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:/Users/Felipe/Desktop/Interseccion.py =====
(0.666663875719833,
 flags: 'converged'
 function_calls: 55
 iterations: 54
 roots: 0.666663875719833)
>>>
```

```
Interseccion.py - C:/Users/Felipe/Desktop/Interseccion.py (3.8.1)
File Edit Format Run Options Window Help
def f(x):
    return x**3 - 2*x**2 + 4*x/3 - 5/27
from scipy import optimize
root = optimize.brentq(f, 0, 2, rtol=10e-6, full_output=True)
print(root)
```

Para la solución del segundo punto, encontramos [2] en la librería de numpy las funciones `numpy.argwhere()`, `numpy.diff()`, `numpy.sign()`, primero la función `np.sign()` otorga los símbolos correspondientes al momento de calcular la función, luego se usa la función `np.diff()` que revela todas las posiciones donde el signo cambia y por último la función `np.argwhere()` da los índices exactos donde los elementos son no cero, con esto podemos hallar las intersecciones entre la curvas.

## 4. Referencias.

[1] SciPy, «SciPy.org», [En línea]. Available:  
<https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.brentq.html>.

[2] Numpy.org, «Numpy», [En línea]. Available:  
<https://numpy.org/doc/stable/reference/generated/numpy.argwhere.html>.