

RETO 1: ANALISIS NUMERICO

Juan Alejandro Diaz, Cristian Camilo Benitez, Andres Ricardo Porras

14 de Marzo 2021

1. Algoritmo Brent

Es un algoritmo híbrido de búsqueda de raíces que combina el metodo de biseccion, el metodo de la secante y la interpolacion cuadratica inversa. Tiene la confiabilidad de la biseccion, pero puede ser tan rapido como algunos de los metodos menos confiables. El algoritmo intenta utilizar el método de la secante de convergencia rápida potencialmente o la interpolacion cuadrática inversa si es posible, pero recurre al método de bisección más robusto si es necesario, El metodo de brent se debe a Richard brent y se basa en un algoritmo anterior de Theodorus Dekker. En consecuencia, el método tambien se conoce como método de brent dekker. Se deben satisfacer dos desigualdades simultáneamente: Dada una tolerancia numérica especifica.

Algoritmo

```
input a, b, and (a pointer to) a function for f
calculate f(a)
calculate f(b)
if f(a)f(b) ≥ 0 then
  exit function because the root is not bracketed.
end if
if |f(a)| < |f(b)| then
  swap (a,b)
end if
c := a
set mflag
repeat until f(b or s) = 0 or |b - a| is small enough (convergence)
  if f(a) ≠ f(c) and f(b) ≠ f(c) then
    
$$s := \frac{af(b)f(c)}{(f(a) - f(b))(f(a) - f(c))} + \frac{bf(a)f(c)}{(f(b) - f(a))(f(b) - f(c))} + \frac{cf(a)f(b)}{(f(c) - f(a))(f(c) - f(b))}$$

    (inverse quadratic interpolation)
  else
    
$$s := b - f(b) \frac{b - a}{f(b) - f(a)}$$

    (secant method)
  end if
  if (condition 1) s is not between (3a + b)/4 and b or
    (condition 2) (mflag is set and |s - b| ≥ |b - c|/2) or
    (condition 3) (mflag is cleared and |s - b| ≥ |c - d|/2) or
    (condition 4) (mflag is set and |b - c| < |δ|) or
    (condition 5) (mflag is cleared and |c - d| < |δ|) then
    
$$s := \frac{a + b}{2}$$

    (bisection method)
  else
    set mflag
  end if
  clear mflag
end if
calculate f(s)
d := c (d is assigned for the first time here; it won't be used above on the
first iteration because mflag is set)
c := b
if f(a)f(s) < 0 then
  b := s
else
  a := s
end if
if |f(a)| < |f(b)| then
  swap (a,b)
end if
end repeat
output b or s (return the root)
```

Figura 1: Funcionamiento general del algoritmo

- Graficado con python

Primero se utilizo las librerias de *matplotlib.pyplot*, esto con el fin de graficar la funcion dada en el enunciado del ejercicio. Se hizo de esta manera por que permitio realizar un acercamiento al punto donde se obervo que estaba la raiz. Luego de hacer la grafica de esta funcion se pudo observar el interbalo en el cual esta la raiz, el cual fue enviado como paramatro a la funcion que usara del metodo de brent para calular la raiz.

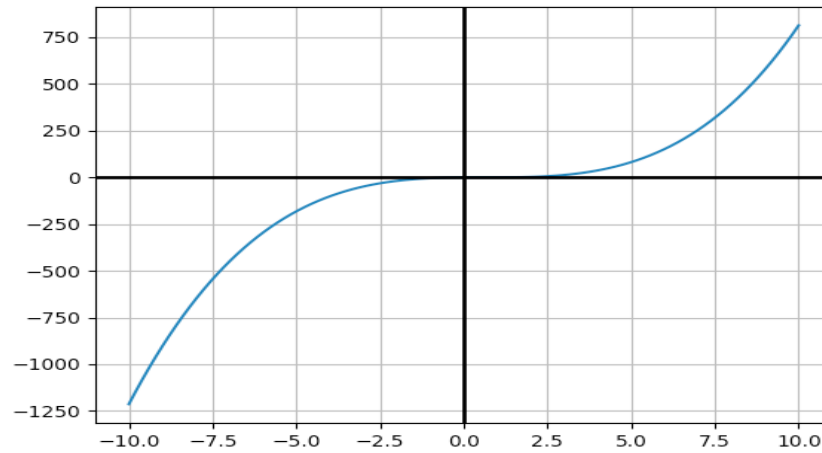


Figura 2: Funcion graficada para brent con python

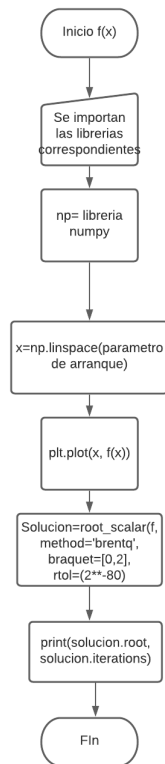


Figura 3: Diagrama de flujo con librerias

■ Código en Python

En el código realizado en python como ya se mencionó anteriormente, lo primero que se hizo fue hacer la gráfica para así obtener el intervalo en el cual definida la raíz. Luego con la ayuda de las librerías encontradas en el punto número tres del reto, se decidió usar de la librería ***scipy.optimize***, de donde se importó la función ***root_scalar***, en la cual como parámetros son enviados:

1. La función a usar
2. El método a usar (en este caso brent)
3. El intervalo
4. La tolerancia

Donde la función aplica el método de brent y nos devuelve la raíz y el número de iteraciones. A continuación se encuentra el código descrito anteriormente y sumado a esto todas las pruebas realizadas.

```

import numpy as np
from scipy.optimize import root_scalar
import math
import matplotlib.pyplot as plt

def f(x):
    return ((x**3) - (2*(x**2)) + ((4*x)/3) - (8/27))

# graficar para reconocer el intervalo
x = np.linspace(start=-10, stop=10, num=100)
plt.plot(x, f(x))
plt.grid()
plt.axhline(y=0, linewidth=2, c='k')
plt.axvline(x=0, linewidth=2, c='k')
plt.show()

solucion = root_scalar(f, method='brentq', bracket=[0, 2], rtol=(2**-50))
print(f"Metodo de Brent:\n\
      - raiz= {solucion.root}\n\
      - Iteraciones ={solucion.iterations}\n")

```

■ Pruebas y comprobación

En las pruebas se pudo observar en la figura 4 donde se uso la tolerancia de 2^{-50} pedida en el problema, que el metodo devuelve en 54 iteraciones el valor de la raiz con una buena precision.

Ademas en la figura 5 donde se uso la tolerancia de 2^{-15} se observa como el metodo devuelve en 49 iteraciones el valor de la raiz pero en este caso con una menor precisión.

En la ultim prueba realizada, que se puede observar en la figura 6 donde se uso la tolerancia de 2^{-51} el metodo devuelve que con esta tolerancia la precision aumenta pero se queda en un numero tan pequeño que la maquina entre en un ciclo infinito.

Por ultimo se utilizo el software de Geobra para poder comprobar las respuestas arrojadas por el metodo de brent, donde se observa que es acertada la respuesta dada por el algoritmo de brent.

```
Reto_punto1.py X
Users > alejandrodiaz > Library > Mobile Documents > com~apple~CloudDocs > UNIVERSIDAD > V SEME
1 import numpy as np
2 from scipy.optimize import root_scalar
3 import math
4 import matplotlib.pyplot as plt
5
6
7 def f(x):
8     return ((x**3) - (2*(x**2)) + ((4*x)/3) - (8/27))
9
10
11 # graficar parca reconocer el intervalo
12 x = np.linspace(start=-10, stop=10, num=100)
13 plt.plot(x, f(x))
14 plt.grid()
15 plt.axhline(y=0, linewidth=2, c='k')
16 plt.axvline(x=0, linewidth=2, c='k')
17 #plt.show()
18
19 solucion = root_scalar(f, method='brentq', bracket=[0, 2], rtol=(2**-50))
20 print(f"Metodo de Brent:\n\
21     - raiz= {solucion.root}\n\
22     - Iteraciones={solucion.iterations}\n")
23
PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE
alejandrodiaz@Alejos-MacBook-Pro ~ % /Library/Developer/CommandLineTools/usr/bin/python3 "/Use
RSIDAD/V SEMESTRE/ANALISIS NUMERICO/PRIMER CORTE/Reto_punto1.py"
Metodo de Brent:
- raiz= 0.666663875719833
- Iteraciones =54
```

Figura 4: Primera prueba con Brent

```
Reto_punto1.py X
Users > alejandrodiaz > Library > Mobile Documents > com~apple~CloudDocs > UNIVERSIDAD > V
1 import numpy as np
2 from scipy.optimize import root_scalar
3 import math
4 import matplotlib.pyplot as plt
5
6
7 def f(x):
8     return ((x**3) - (2*(x**2)) + ((4*x)/3) - (8/27))
9
10
11 # graficar parca reconocer el intervalo
12 x = np.linspace(start=-10, stop=10, num=100)
13 plt.plot(x, f(x))
14 plt.grid()
15 plt.axhline(y=0, linewidth=2, c='k')
16 plt.axvline(x=0, linewidth=2, c='k')
17 #plt.show()
18
19 solucion = root_scalar(f, method='brentq', bracket=[0, 2], rtol=(2**-15))
20 print(f"Metodo de Brent:\n\
21     - raiz= {solucion.root}\n\
22     - Iteraciones={solucion.iterations}\n")
23
PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE
alejandrodiaz@Alejos-MacBook-Pro ~ % /Library/Developer/CommandLineTools/usr/bin/python3
RSIDAD/V SEMESTRE/ANALISIS NUMERICO/PRIMER CORTE/Reto_punto1.py"
Metodo de Brent:
- raiz= 0.6666693310275446
- Iteraciones =49
```

Figura 5: Segunda prueba con Brent

```

Reto_punto1.py
Users > alejandrodiaz > Library > Mobile Documents > com~apple~CloudDocs > UNIVERSIDAD > V SEMESTRE > ANALIS
1 import numpy as np
2 from scipy.optimize import root_scalar
3 import math
4 import matplotlib.pyplot as plt
5
6
7 def f(x):
8     return ((x**3) - (2*(x**2)) + ((4*x)/3) - (8/27))
9
10
11 # graficar parca reconocer el intervalo
12 x = np.linspace(start=-10, stop=10, num=100)
13 plt.plot(x, f(x))
14 plt.grid()
15 plt.axhline(y=0, linewidth=2, c='k')
16 plt.axvline(x=0, linewidth=2, c='k')
17 #plt.show()
18
19 solucion = root_scalar(f, method='brentq', bracket=[0, 2], rtol=(2**-51))
20 print(f'Metodo de Brent:\n\
21       - raiz= {solucion.root}\n\
22       - Iteraciones = {solucion.iterations}\n")
23
PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

alejandrodiaz@Alejos-MacBook-Pro ~ % /Library/Developer/CommandLineTools/usr/bin/python3 "/Users/alejandrodiaz/
RSIDAD/V SEMESTRE/ANALISIS NUMERICO/PRIMER CORTE/Reto_punto1.py"
Traceback (most recent call last):
  File "/Users/alejandrodiaz/Library/Mobile Documents/com~apple~CloudDocs/UNIVERSIDAD/V SEMESTRE/ANALISIS NU
ule>
  solucion = root_scalar(f, method='brentq', bracket=[0, 2], rtol=(2**-51))
  File "/Library/Python/3.7/site-packages/scipy/optimize/_root_scalar.py", line 249, in root_scalar
    r, sol = methodc(f, a, b, args=args, **kwargs)
  File "/Library/Python/3.7/site-packages/scipy/optimize/zeros.py", line 775, in brentq
    raise ValueError("rtol too small (%g < %g)" % (rtol, _rtol))
ValueError: rtol too small (4.44089e-16 < 8.88178e-16)
alejandrodiaz@Alejos-MacBook-Pro ~ %

```

Figura 6: Tercera prueba con Brent

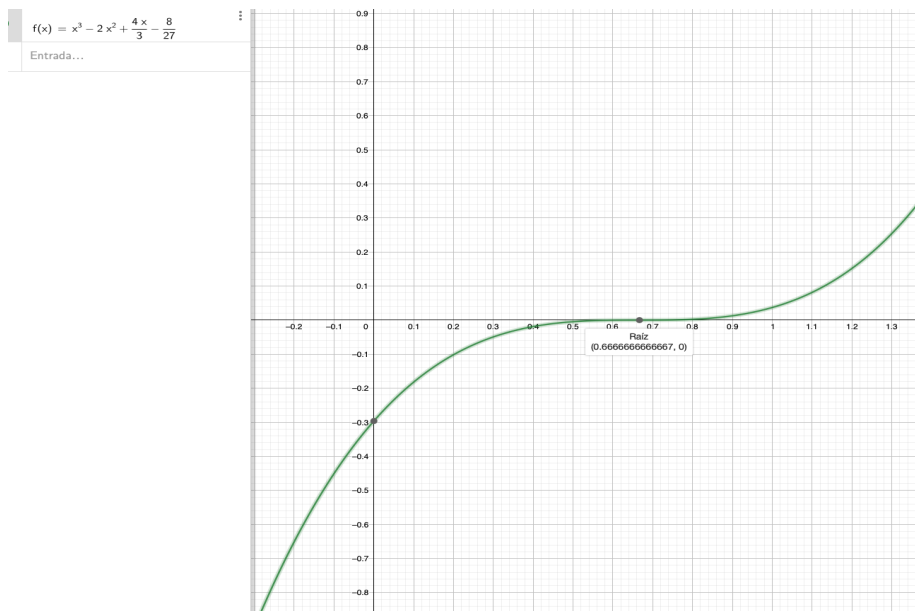


Figura 7: Raiz calculada con geobra

2. Intersección entre curvas

El metodo por el cual vamos a llevar a cabo la interseccion entre las dos funciones es el de punto fijo. Para resumir, el metodo de punto fijo, tambien conocido como método de iteracion funcional, es el fundamento matematico para construir métodos eficientes para el cálculo de raices reales de ecuaciones no lineales. El metodo consiste en rescribir la ecuación $f(x)=0$ en la forma $x=g(x)$. Esta nueva ecuacion debe ser equivalente a la ecuacion original en el sentido que debe satisfacerse con la misma raiz, es decir, la existencia de un punto fijo r de la ecuacion $x=g(x)$ es equivalente a encontrar una raiz real r de la ecuacion $f(x)=0$; $r=g(r)$ y esto de bicondicional $f(r)=0$.

El procedimiento empieza con una estimacion o conjetura inicial x , que es mejorada por iteracion hasta alcanzar la convergencia. Para que converja, la derivada (dg/dx) debe ser menor que 1 en magnitud (por lo menos para los valores de x que se encuentran durante las iteraciones). La convergencia será establecida mediante el requisito de que el cambio en x de una iteracion a la siguiente no sea mayor en magnitud que alguna pequeña cantidad e .

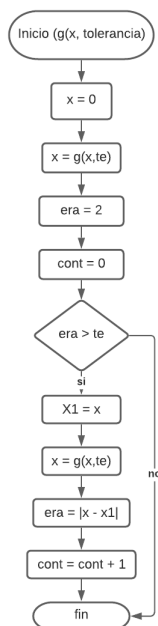


Figura 8: Diagrama de flujo metodo de punto fijo

El algoritmo que habitual mente se utilizaba era para calcular la reiz de una funcion determinando la cantidad de iteraciones y especificando una tolerancia y un error, en este caso, se el reto establecia dos funciones la cuales se generaba una interseccion entre estas dos. Para evaluar las expresiones, se tuvo que determinar

el valor de y de la primera y luego remplazar estos valores en la segunda.

La libreria `scipy` nos permitia declarar el metodo de punto fijo para determinar la raiz de la expresion ya descomprimida, como se muestra en el algoritmo a continuacion:

- Codigo en Python Punto fijo con librerias

```
from scipy import optimize
import numpy as np
from array import array
import matplotlib.pyplot as plt

def f(x):
    return ((3*(x**4))-(61*(x**2))-(57*x)+310)

x = np.linspace(start=-10, stop=10, num=100)
plt.plot(x, f(x))
plt.grid()
plt.axhline(y=0, linewidth=2, c='k')
plt.axvline(x=0, linewidth=2, c='k')
plt.show()

respuesta = optimize.fixed_point(f, x0=[2,4.5], xtol=2**-16)
print(respuesta)
```

Ahora en el algoritmo a continuacion, realizamos la implementacion del metodo pero mediante un algoritmo contruido por nosotros mismos.

- Codigo en python de punto fijo Nuestro

```
import numpy as np
import math

def puntofijo(gx, a, tolera, iteramax=100):
    i = 1
    b = gx(a)
    tramo = abs(b-a)
    while(tramo >= tolera and i <= iteramax):
        a = b
        b = gx(a)
        tramo = abs(b-a)
        i = i + 1
    print(" iteracion:" , i)
```



```

        print("raiz:" , b)

    respuesta = b

# INGRESO

#def gx(x): return np.exp(-x)
def gx(x):return ((3*(x**4))-(61*(x**2))-(57*x)+310)

a = 2      # intervalo
b = 4,5
tolera = 2**-16

# PROCEDIMIENTO
puntofijo(gx, a, tolera)

```

Especificamos estas implementaciones con el fin de demostrar cual pseudocódigo es más eficiente y ver cual nos arroja resultados más precisos. A continuación, en la figura 7 y 8 se mostrará la ejecución de las dos implementaciones mencionadas anteriormente:

```
Users > alejandroddiaz > Library > Mobile Documents > com~apple~CloudDocs > UNIVERSIDAD >  
1  from scipy import optimize  
2  import numpy as np  
3  from array import array  
4  import matplotlib.pyplot as plt  
5  
6  
7  def f(x):  
8      |   return ((3*(x**4))-(61*(x**2))-(57*x)+310)  
9  
10 x = np.linspace(start=-10, stop=10, num=100)  
11 plt.plot(x, f(x))  
12 plt.grid()  
13 plt.axhline(y=0, linewidth=2, c='k')  
14 plt.axvline(x=0, linewidth=2, c='k')  
15 #plt.show()  
16  
17 respuesta = optimize.fixed_point(f, x0=[2,4.5], xtol=2**-16)  
18 print(respuesta)  
19  
PROBLEMS  OUTPUT  TERMINAL  ...  1: Python  
alejandrodiaz@Alejos-MacBook-Pro ~ % /Library/Developer/CommandLineTools/usr/bin/python3  
ejandrodiaz/Library/Mobile Documents/com~apple~CloudDocs/UNIVERSIDAD/V SEMESTRE/ANALIS  
RIMER CORTE/Reto_punto2_ConLibrerias.py"  
[1.99029618 4.40403953]  
alejandrodiaz@Alejos-MacBook-Pro ~ %
```

Figura 9: Prueba uno punto fijo

```

1  # Algoritmo de punto fijo
2  # [a,b] intervalo de búsqueda
3  # error = tolera
4
5  import numpy as np
6  import math
7
8
9  def puntofijo(gx, a, tolera, iteramax=100):
10     i = 1 # iteración
11     b = gx(a)
12     tramo = abs(b-a)
13     while(tramo >= tolera and i <= iteramax):
14         a = b
15         b = gx(a)
16         tramo = abs(b-a)
17         i = i + 1
18         print("iteracion:" , i)
19         print("raiz:" , b)
20
21     respuesta = b
22
23 # INGRESO
24
25 #def gx(x): return np.exp(-x)
26 def gx(x):return ((3*(x**4))-(61*(x**2))-(57*x)+310)
27
28 a = 2 # intervalo
29 b = 4,5
30 tolera = 2**-16
31
32 # PROCEDIMIENTO
33 puntofijo(gx, a, tolera)
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100

```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE 1: Python

```

16745485313932709741046723897934362664079245016716324268441243726844944829422303198120724176649098583386085222050133620386397
6262670659885825666901228399325436971612462178988810517739303851951341757075403798284831386014406554029418009213610961452613
10024838116106718446050501332530465079042229117564116839810543895902519082893914853345868135885407602891136035326427623506620
62242094854328721745304922371422166487145847758528522463622644953682541176547245595813950499536984443252081438504460739979191
21995852312648877990604436430777234637308858874702682500761667849748264445992305588600396158597996239342681004200928113442090
52319803886346058962571999169351771461806041702206038991044888177777270025989751826251872858005994736636528865039709638212514
74000470384882640556058194222018078306362151184809859803968489881598885677831822257195678633418329620850193808150018314124
1712549305229216614700675879623349544989644251315498918026324692811092540612372848838411890643636276364777858193200601814022
53137652353197597022335235192067478058126271294646909527954491685861085548262096200253209284630903775544172864305592010456393
25805776169524108348777488107478308294828055002965066483778527037519008905864357134120803966185848332042547166810214855777264
821949886481898127923903216276929576860759369603937248628873698039954305894292623244000212450022171718671797913849482721438
94457285128471462192375203214911833486937347612807577314214474329254398298392598736172909115355785166780911829577383638315730
2800085256680066830383507459864908406063366883837973192738461547299125847865360735714779110183017371773597026925984413685239
4947519066605075462540378224624424679303337467882428786066789416785859909021733471434956461862439311108897941108942610010718
07648522738671718382843463033538482947371950728159132226741865712161539449332444486654472611712570259023975354000163519
4424833689282930440131587682420106804800407873389858306055916336449823874277849453938866259254073997849020708866148065118388
5857291442497842121247455255794308388906608827029319458262341073716959932677453658469860515563462450633806718303838656235424
50996586437412121613887378310257691143174949578599785419122974277817241315867813232228189700579727174593930742870062361923015
82829296061342077850236288334459579401222450393544053560328374982268075024000768316964602300667601493491702879784796031787822
09798754063746865398707445080760961150463948706419633274104066268045876236710
iteration: 11
^Craiz: Traceback (most recent call last):
  File "/Users/alejandrodiaz/Library/Mobile Documents/com~apple~CloudDocs/UNIVERSIDAD/V SEMESTRE/ANALISIS NUMERICO/PRIMER COR
TE/reto_punto2_ConNuestroMetodo.py", line 33, in <module>
    puntofijo(gx, a, tolera)
  File "/Users/alejandrodiaz/Library/Mobile Documents/com~apple~CloudDocs/UNIVERSIDAD/V SEMESTRE/ANALISIS NUMERICO/PRIMER COR

```

Figura 10: Prueba uno punto fijo

Evaluando los resultados en entre las dos implementacion, obtuvimos resultados completamente diferentes entre las dos; En la prueba de punto fijo aplicado con nuestro algoritmo, las iteraciones no eran precisas, eran erroneas. En la implementacion al metodo con la libreria scipy, arroja un resultados sin determinar la cantidad de iteraciones efectuadas, pero las raices son las correctas. Más adelante, en la parte de conclusiones, determinaremos el porque sucedio esto.

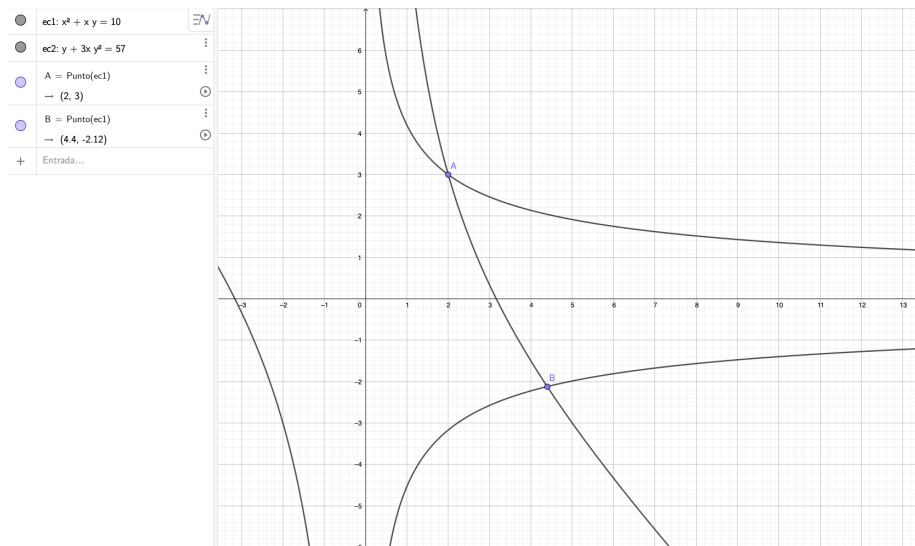


Figura 11: Intersección entre las curvas

3. Librerías en Python

Se realizó una búsqueda de la documentación de las librerías numpy y scipy para la resolución de los puntos anteriores. Para cada librería se encontraron diferentes funciones con su documentación para el manejo de parámetros que fueron de gran ayuda. A continuación encontrará las funciones con sus diferentes parámetros y la explicación de su funcionamiento usadas en cada librería en los puntos anteriores.

- Numpy: Es una biblioteca del lenguaje de programación python que da soporte para crear matrices y vectores grandes multidimensionales

Desde numpy se utilizaron para mejorar temas como la precisión, poder manejar números más pequeños y hacer gráficas para determinar intervalos de las funciones

- Scipy: Es una biblioteca libre y de código abierto para Python. Se compone de herramientas y algoritmos matemáticos

- Root-scalar: En la cual se nos da el prototipo de la función para jugar con los parámetros que se pueden introducir como lo son la función a usar, la tolerancia, los intervalos, las iteraciones y también usar funciones como secante, newton, brent, etc.

```
scipy.optimize.root_scalar(f, args=(), method=None, bracket=None, fprime=None, fprime2=None, x0=None, x1=None, xtol=None, rtol=None, maxiter=None, options=None)
```

- Fixed-Point: Para esta funcion utilizada de optimize nos permite aplicar el metodo de punto fijo para hallar la raiz en un intervalo de una funcion dada con una tolerancia especifica. Ademas de eso permite acelerar el metodo con aiken mandando por comandos que method aplicado se "del2"
`scipy.optimize.fixedpoint(func, x0, args=(), xtol=1e-08, maxiter=500, method='del2')`

4. Conclusiones

- Las librerias que nos ofrecen algunos leguajes de programacion nos permite llegar a una solucion de una manera mas eficiente; En los metodos iterativos de brent y de punto fijo, las librerias numpy y scipy nos permitio integrar las funciones directamente para llegar a la solucion y sus raices junto a la cantidad de iteraciones realizadas.
- Para nuestra solucion de interseccion entre curvas aplicado al metodo de punto fijo, utilizando el codigo desarrollado por nosotros; si logra determinar un numero de iteraciones pero la expresion al ser tan grande y compleja como lectura pasada como parametro dentro de las especificaciones del algoritmo, se queda iterando y no proporciona una solucion precisa. Teniendo en cuenta lo anterior, tuvimos que aplicar el metodo utilizando librerias, mandando parametros como la tolerancia, los intervalos y la funcion, se nos proporciona una solucion precisa ya que determina las raices de las funciones para la interseccion.

5. Referencias

Online . Disponible: docs.scipy.org/doc/scipy/reference/optimize.rootscalarbrentq

Online . Disponible: docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.fixedpoint

Online . Disponible: es.qaz.wiki/wiki/Brent