

## 1. Algoritmo Brent

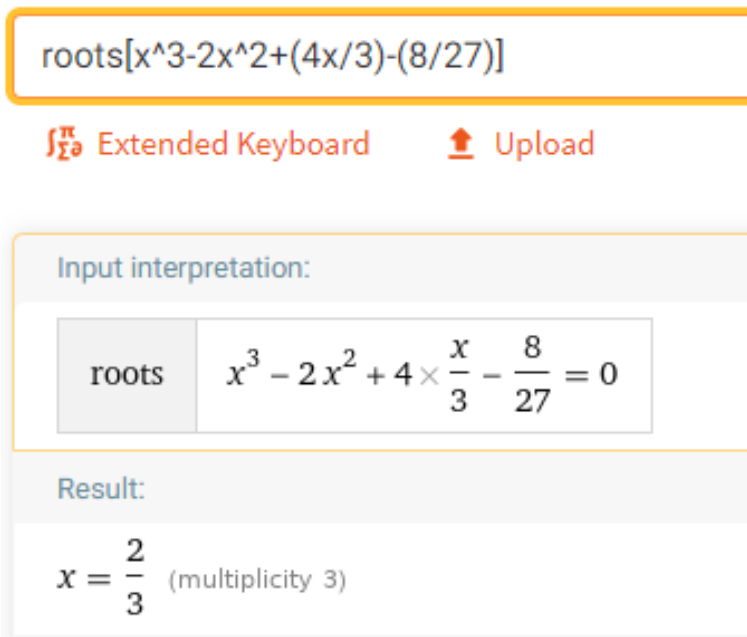
Este algoritmo de Brent, utiliza en cada punto lo más conveniente de las estrategias del de la bisección y del de la secante (o Muller). Este método suele converger muy rápidamente a cero; para las funciones difíciles ocasionales que se encuentran en la práctica.

**Problema:** Aplicar el algoritmo de Brent para encontrar las raíces del polinomio, con un error menor de  $2^{-50}$ :

$$f(x) = x^3 - 2x^2 + 4x/3 - 8/27 \quad (1)$$

### 1.1. Solucion

Para asegurarnos del valor de la raíz lo comprobamos en WolframAlpha.



roots[x^3-2x^2+(4x/3)-(8/27)]

Extended Keyboard Upload

Input interpretation:

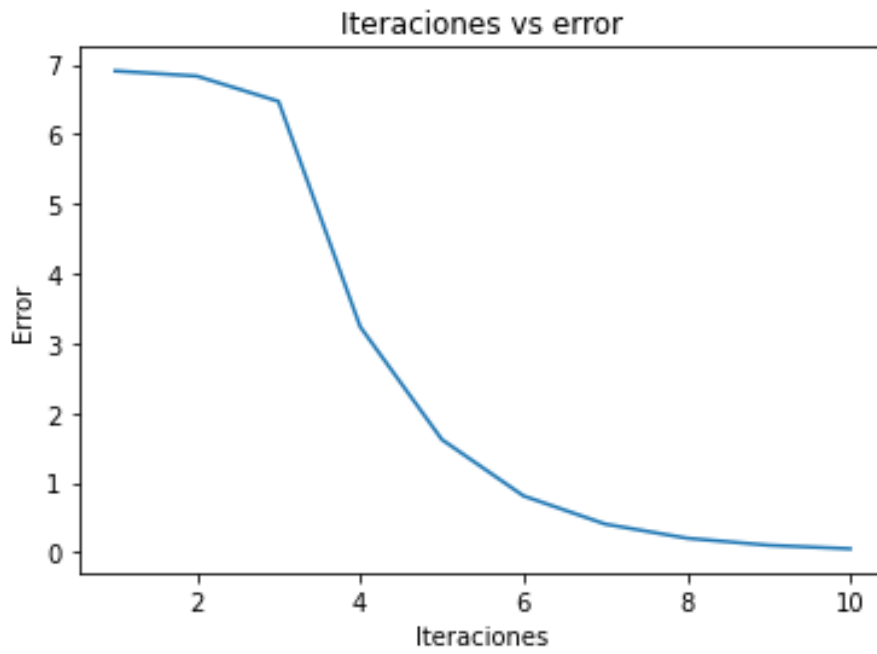
roots	$x^3 - 2x^2 + 4 \times \frac{x}{3} - \frac{8}{27} = 0$
-------	--

Result:

$x = \frac{2}{3}$  (multiplicity 3)

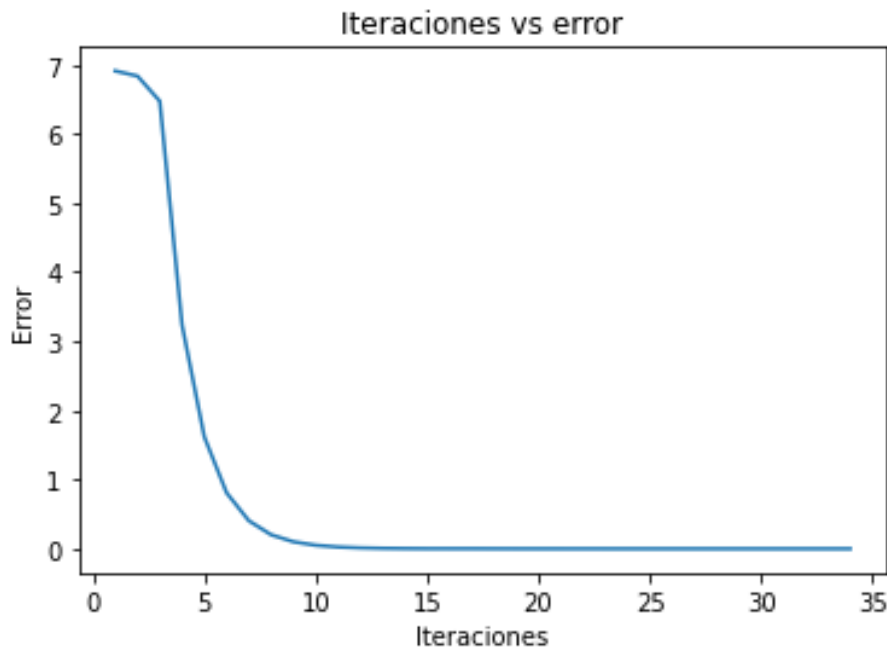
Se implemento en python el algoritmo de brent donde primero se trabajo con una tolerancia muy baja( $10^{-2}$ ) para que se apreciara mejor como a mayor numero de iteraciones se iba disminuyendo el error. Dicha implementacion se encuentra en la misma carpeta donde se subio este documento.

```
Iteraciones realizadas: 10  
La raíz encontrada es: 0.6639503272813216
```



Despues se trabajo con una tolerancia de  $(10^{-12})$  y se vio como el algoritmo se acerca mucho mas a la raiz teniendo en cuenta que esta es periodica, a continuacion vemos los resultados obtenidos por el codigo en phyton.

```
Iteraciones realizadas: 34  
La raiz encontrada es: 0.6666666479440323
```



Ademas de esta implementacion, se usaron las librerias de phyton las cuales se explicaran en el punto 3 y se logro llegar a la solucion de una manera mucho mas sencilla ya que el codigo sale mas corto al ya existir en

---

las librerías. A continuación se mostrara el código y sus resultados, este segundo código tiene como nombre `brentLibrerias` y se encuentra disponible en la carpeta donde se encuentra ese documento.

```
def f(x):  
    return (x**3 - 2*x**2+(4*x)/3- (8/27))  
  
from scipy import optimize  
  
root = optimize.brentq(f, 0, 1,)  
print("La raíz encontrada es: ",root)  
  
La raíz encontrada es: 0.6666685251964675
```

## 2. Intersección entre curvas

La intersección entre dos curvas es un problema común del cálculo y que enfrenta el desafío de solucionar un sistema de ecuaciones no lineales. No obstante, la solución se puede encontrar reduciendo el problema a determinar una aproximación adecuada de los ceros de una ecuación, con un error menor de  $2^{-16}$ .

**Problema:** Aplicar la técnica de aproximación a la raíz, que desarrollo en el trabajo en grupo, para encontrar la intersección entre

$$x^2 + xy = 10; y + 3xy^2 = 57 \quad (2)$$

Para este numeral se creó un algoritmo en R que nos termina dando la solución en donde las dos curvas anteriormente mencionadas se interceptan.

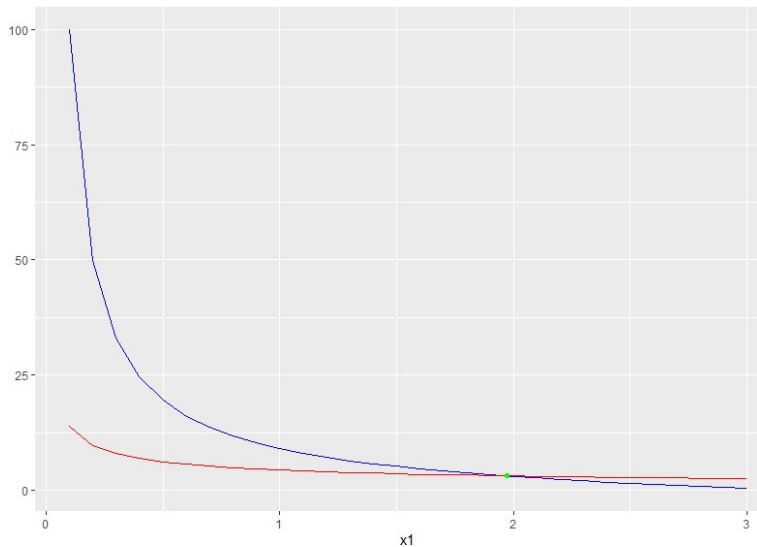
El punto de intersección según el algoritmo es:

Las funciones cortan en X: 1.970199 Y: 3.10543

Se usaron 59 iteraciones para llegar a este con el error recomendado.

Resultado: 1.970199 iteración: 59

En la siguiente gráfica en R usando `ggplot2` encontramos nuestro punto en verde y las dos gráficas en rojo y azul respectivamente.



### 3. Librerías en R y/o Python

El uso adecuado de las herramientas numéricas en Python o R es importante en cualquier problema. Es por esto, que es necesario revisar su uso e implementación para la solución de un problema.

**Problema:** Revisar las librerías numpy y SciPy en Python para resolver los problemas anteriores.

**Problema:** En el caso de utilizar R, revisar la función base `polyroot`, la función `uniroot` basado en el algoritmo de Brent en la base de R y los paquetes `pracma` y `rootSolve`, para resolver los problemas anteriores.

#### 3.1. Descripción general librerías Numpy y Scipy de Python

**Numpy:** NumPy nos agrega apoyo para el cálculo de vectores y matrices con funciones matemáticas de alto nivel. Arreglos:

- `ndarray.ndim`: Proporciona el número de dimensiones de nuestro array.
- `ndarray.dtype`: Es un objeto que describe el tipo de elementos del array.
- `ndarray.shape`: Devuelve la dimensión del array, es decir, una tupla de enteros indicando el tamaño del array en cada dimensión. Para una matriz de  $n$  filas y  $m$  columnas obtendremos  $(n, m)$ .
- `ndarray.data`: El buffer contiene los elementos actuales del array.
- `ndarray.itemsize`: devuelve el tamaño del array en bytes.
- `ndarray.size`: Es el número total de elementos del array.
- `identity(n, dtype)`: Devuelve la matriz identidad, es decir, una matriz cuadrada nula excepto en su diagonal principal que es unitaria.  $n$  es el número de filas (y columnas) que tendrá la matriz y `dtype` es el tipo de dato. Este argumento es opcional. Si no se establece, se toma por defecto como flotante.

- 
- `ones(shape, dtype)`: Crea un array de 1 compuesto de `shape` elementos.
  - `zeros(shape, dtype)`: Crea un array de 0 compuesto de “`shape`” elementos”.
  - `linspace(start, stop, num, endpoint=True, retstep=False)`: Crea un array con valor inicial `start`, valor final `stop` y `num` elementos.
  - `empty(shape, dtype)`: Crea un array de ceros compuesto de “`shape`” elementos” sin entradas.
  - `meshgrid(x, y)`: Genera una malla a partir de dos los arrays `x`, `y`.
  - `eye(N, M, k, dtype)`: Crea un array bidimensional con unos en la diagonal `k` y ceros en el resto. Es similar a `identity`. Todos los argumentos son opcionales. `N` es el número de filas, `M` el de columnas y `k` es el índice de la diagonal. Cuando `k=0` nos referimos a la diagonal principal y por tanto `eye` es similar a `identity`.
  - `arange([start,]stop[,step,], dtype=None)`: Crea un array con valores distanciados `step` entre el valor inicial `start` y el valor final `stop`. Si no se establece `step` python establecerá uno por defecto.

**SciPy**: Esta es una librería más compleja ya que realmente funciona por subpaquetes de funciones, se utilizan para realizar operaciones, análisis de datos más específicos; los más importantes son:

- Álgebra lineal: `linalg`
- Procesamiento de señales: `signal`
- Funciones estadísticas: `stats`
- Funciones especiales: `special`
- Integración: `integrate`
- Herramientas de interpolación: `interpolate`
- Herramientas de optimización: `optimize`
- Algoritmos de transformada de Fourier: `fftpack`
- Entrada y salida de datos: `io`
- Wrappers a la librería LAPACK: `lib.lapack`
- Wrappers a la librería BLAS: `lib.blas`
- Wrappers a librerías externas: `lib`
- Matrices sparse: `sparse`
- otras utilidades: `misc`
- Vector Quantization / Kmeans: `cluster`
- Ajuste a modelos con máxima entropía: `maxentropy`

---

## 3.2. Descripción general librerías en R

**Polyroot:** Esta paquetería sirve para encontrar ceros de un polinomio, sea real o complejo mediante el algoritmo de Jenkins-Traub

**Uniroot:** Para aproximar los ceros de una función  $f$  está la función `uniroot()` en la base de R. La documentación indica que esta función usa el método de Brent, este método es un método híbrido que combina bisección e interpolación cuadrática inversa.

**Pracma:** Este paquete proporciona un gran número de funciones de análisis numérico, álgebra lineal, optimización, ecuaciones diferenciales, series temporales (Se podría comparar con ScyPy) se pueden encontrar métodos como "bisección", "secante", "newtonraphson", "bren"; algunos de ellos son:

- `brentDekker`: Algoritmo de búsqueda de raíces Brent-Dekker
- `eigjacobi`: Método Jacobi Eigenvalue
- `akimaInterp`: Interpolación akima univariada
- `aitken`: Método de Aitken
- `diag`: Matriz Diagonal
- `cubispline`: Interpolación de spline cúbica
- `curvefit`: Ajuste de curva paramétrica
- `horner`: Regla de Horner

**rootSolve:** Se utiliza para hallar raíces no lineales, analizar ecuaciones diferenciales ordinarias mediante diferentes funciones, las cuales son:

- `jacobian.band`: Matriz jacobiana con bandas para un sistema de ODI (ecuaciones diferenciales ordinarias)
- `runsteady`: Ejecuta dinámicamente un sistema de ecuaciones diferenciales ordinarias (ODE) a
- `gradient`: Estima la matriz de gradiente para una función simple
- `hessian`: Estima la matriz hessiana
- `uniroot.all`: Encuentra muchas (todas) raíces de una ecuación dentro de un intervalo
- `jacobiano.full`: Matriz jacobiana cuadrada completa para un sistema de ODI (ecuaciones diferenciales ordinarias)
- `steady`: Solucionador general de estado estacionario para un conjunto de ecuaciones diferenciales ordinarias.
- `Stode`: Solucionador iterativo de estado estacionario para ecuaciones diferenciales ordinarias (ODE) y un jacobiano completo o con bandas.

- 
- `plot.steady1D`: Método de trazado y resumen para objetos `steady1D`, `steady2D` y `steady3D`
  - `rootSolve-package`: Raíces y estados estables
  - `multiroot.1D`: Resuelve  $n$  raíces de ecuaciones  $n$  (no lineales), creadas mediante la discretización de ecuaciones diferenciales ordinarias.
  - `Multiroot`: Resuelve  $n$  raíces de ecuaciones  $n$  (no lineales).
  - `steady.2D`: Solucionador de estado estacionario para ecuaciones diferenciales ordinarias de 2 dimensiones
  - `steady.3D`: Solucionador de estado estacionario para ecuaciones diferenciales ordinarias de 3 dimensiones
  - `steady.1D`: Solucionador de estado estacionario para ecuaciones diferenciales ordinarias 1D multicomponentes
  - `stodes`: Solucionador de estado estacionario para ecuaciones diferenciales ordinarias (ODE) con un jacobiano escaso.
  - `steady.band`: Solucionador de estado estacionario para ecuaciones diferenciales ordinarias; asume un jacobiano en banda

### 3.3. Librería utilizada para el desarrollo del primer ejercicio

Para desarrollar el algoritmo de brent en el primer problema utilizamos la librería SciPy, como se comentaba anteriormente esta librería funciona con sublibrerías, puntualmente utilizamos la subpaquetería de “Optimize” con una función llamada `brentq`.

**scipy.optimize.brentq:** Busca la raíz de una función en un intervalo utilizando el método de Brent.

**Parámetros para su uso:**

- `f`: Función a evaluar
- `a`: Un extremo del intervalo
- `b`: El otro extremo del intervalo

### 3.4. Librería utilizada para el desarrollo del segundo ejercicio

Se utilizaron principalmente dos librerías para realizar las gráficas, mientras que para solucionar el ejercicio se dio uso a operaciones matemáticas básicas y ciclos.

**Las paqueterías para graficar fueron dos, `plotly` y `ggplot2`:**

**Plotly:** Esta librería crea gráficos web interactivos a través de la biblioteca de gráficos JavaScript de código abierto `trazadamente.js`.

**ggplot2:** Un sistema para crear gráficos “declarativamente”, basado en “La gramática de los gráficos” cómo mapear variables estáticas, gráficos a utilizar, y se encarga de los detalles.

**Utilizamos:**

- 
- ggplot: Crear un nuevo ggplot
  - geom line: Líneas de referencia: horizontales, verticales y diagonales
  - geom point: Graficar los puntos