

Pontificia Universidad Javeriana



ANÁLISIS NUMÉRICO

## RETO 1

Autores:  
John Stiven  
Jimmy Alejandro  
Pablo Veintemilla  
Camilo Maldonado

Septiembre 2020

## RETO 1

EDDY HERRERA

RESUMEN. Presentamos aca el metodo de blah blah

### Parte 1. Metodo de Newton-Horner y Metodo de Laguerre

#### 1. Metodo de Newton-Horner

El método de Newton-Horner y el método de Laguerre son métodos iterativos usados para dar con la solución de ecuaciones polinómicas, son métodos que aseguran una convergencia rápida hacia las soluciones aplicando teorías de polinomios.

Método de Newton-Horner:

Dado un polinomio de grado  $n$

$$P(x) = \sum_{j=0}^n a_j x^j$$

se puede utilizar el método de Horner para factorizar en un valor  $x_0$  usando la división sintética así obtener un factor de la forma

$$P(x) = (x - x_0) \sum_{j=0}^{n-1} b_j x^j + h_1$$

Donde  $h_1$  es un sumando adicional para completar la factorización. Así pues se puede afirmar que

$$P(x_0) = h_1$$

Y de esta forma igualmente en  $P'(x)$

$$P'(x) = \frac{d}{dx} \left( (x - x_0) \sum_{j=0}^{n-1} b_j x^j + h_1 \right)$$

Se sabe que la derivada de  $h_1$  es 0, la derivada de  $(x - x_0)$  es 1 y la derivada de  $(j = 0)^{(n-1)}b_j x^j$  sería un polinomio de grado  $n-2$  al que se puede llamar  $G(x)$  aplicando la derivada del producto en  $P(x)$  se tiene que

$$P'(x) = \sum_{j=0}^{n-1} b_j x^j + (x - x_0)G(x)$$

Se puede realizar la misma división sintética con centro en  $x_0$  sobre el polinomio

$$(x - x_0) \sum_{j=0}^{n-2} c_j x^j + h_2$$

Entonces al evaluar  $P'(x)$  en  $x_0$

$$P'(x_0) = h_2$$

Lo cual sería igual que solo aplicar la división en el polinomio de grado  $n-1$  y extraer el nuevo sumando  $h_2$ . Conociendo el valor de  $P(x)$  y  $P'(x)$  se puede usar el método iterativo de Newton-Raphson teniendo el valor de  $x_0$

$$x_{k+1} = x_k - \frac{h_1}{h_2}, k = \{0, 1, 2, \dots\}$$

Se tiene un nuevo valor de  $x$  con el que se puede hacer una nueva iteración hasta hallar la solución.

Método de Laguerre:

Dado un polinomio

$$P(x) = \sum_{j=0}^n a_j x^j$$

se debe proponer una solución  $x = r$  y unas  $n-1$  soluciones en  $x = q$  de tal forma que

$$P(x) = (x - r)(x - q)^{n-1}$$

De esto podemos derivar  $P(x)$  y obtener

$$P'(x) = (x - q)^{n-1} + (x - r)(n - 1)(x - q)^{n-2}$$

Y se puede decir que

$$P'(x) = P(x) \left( \frac{1}{x - r} + \frac{n - 1}{x - q} \right)$$

$$\frac{P'(x)}{P(x)} = \frac{1}{x - r} + \frac{n - 1}{x - q}$$

Si derivamos esta expresión de nuevo

$$\frac{P''(x)}{P(x)} - \left( \frac{P'(x)}{P(x)} \right)^2 = -\frac{1}{(x - r)^2} - \frac{n - 1}{(x - q)^2}$$

Podemos sustituir

$$G(x) = \frac{1}{x - r} + \frac{n - 1}{x - q}$$

$$H(x) = \frac{1}{(x - r)^2} + \frac{n - 1}{(x - q)^2}$$

Podemos despejar  $(x - q)$  de las dos ecuaciones y poderlo igualar, a lo cual se obtiene una ecuación cuadrática para  $(x - r)$  la cual se soluciona

$$x - r = \frac{n}{G \pm \sqrt{(n - 1)(nH - G^2)}}$$

Para obtener el método iterativo de Laguerre

$$x_{k+1} = x_k - \frac{n}{G \pm \sqrt{(n - 1)(nH - G^2)}}, \quad k = \{0, 1, 2, \dots\}$$

$$G = \frac{P'(x)}{P(x)}, \quad H = G^2 - \frac{P''(x)}{P(x)}$$

Los métodos están enfocados en asegurar la menor pérdida de significancia y poder encontrar las diferentes raíces de un polinomio con convergencia teórica cubica.

Raíces Reales y Complejas:

Tanto el método de Newton-Horner como el Método de Laguerre son capaces de estimar de forma precisa una solución tanto dentro de los números reales como en los números complejos, más aún en la implementación se puede ver que se necesita una aproximación al cero en la parte imaginaria para conseguir estos resultados.

Pruebas:

Se realizaron pruebas tanto con diferentes polinomios como para diferentes valores de tolerancia y cambiando los valores iniciales se puede dar un comportamiento diferente en los resultados en numero de iteraciones como en el comportamiento de los errores graficados.

1. Prueba base:  $P(x) = x^4 - 5x^3 - 9x^2 + 155x - 250$

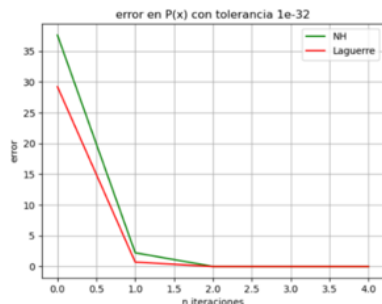
```
polinomios = [[1,-5,-9,155,-250]]
```

Soluciones teoricas

The solutions are

$x = 2, x = -5, x = 4 + 3i, x = 4 - 3i$

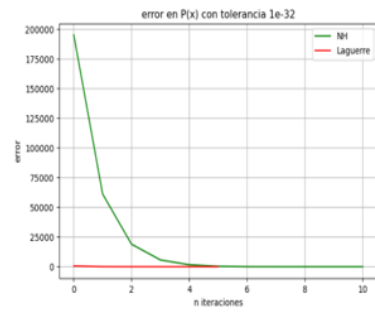
$X_0 = 0$ , se encuentra la raíz real  $x = 2$



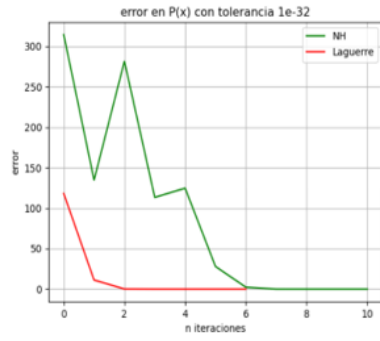
```
Metodo de Newton Horner, tolerancia 1e-32
1 x = 1.61290322580645151262581713063102+0.000000000000000000000000000000j
2 x = 1.97562579494804335666913175373338+0.000000000000000000000000000000j
3 x = 1.9999019217837257229250181408133+0.000000000000000000000000000000j
4 x = 1.99999999841438413916705485462444+0.000000000000000000000000000000j
5 x = 2.000000000000000000000000000000+0.000000000000000000000000000000j
P(x*) = 0.000000000000000000000000000000+0.000000000000000000000000000000j

Metodo de Laguerre, tolerancia 1e-32
1 x = 1.69491525423728783827925781224621+0.000000000000000000000000000000j
2 x = 1.99219296555683644811551857856102+0.000000000000000000000000000000j
3 x = 1.99999497420935234970329474890605+0.000000000000000000000000000000j
4 x = 1.999999999791810978422290645540+0.000000000000000000000000000000j
5 x = 2.000000000000000000000000000000+0.000000000000000000000000000000j
P(x*) = 0.000000000000000000000000000000+0.000000000000000000000000000000j
```

$X_0 = -3$ , se encuentra la raíz real  $x = -5$

[illegible]

$X_0 = 3+2.3i$ , se encuentra la raíz compleja  $4+3i$

[illegible]

2. Otras pruebas:  $P1(x) = x^2 + 4$ ,  $P2(x) = x^2 - 5x + 6$

```
polinomios = [[1,0,4],[1,-5,6]]
```

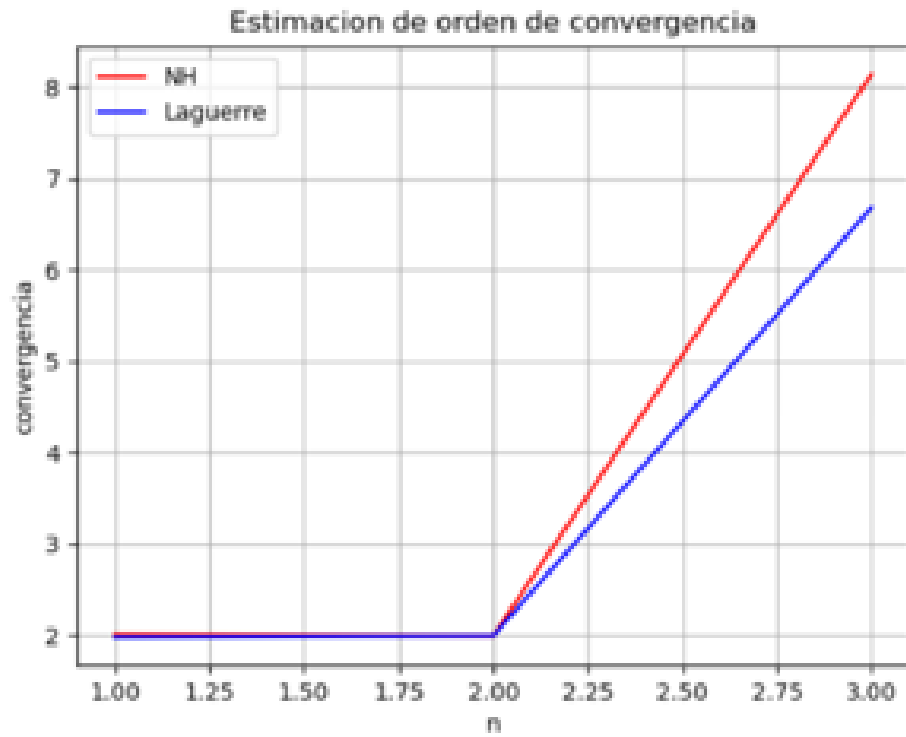
$$P1(x): x^2 + 4tieneunarazcomplejayconjugada x = 2i$$

$X_0 = 0$ , se obtiene el valor de la raíz, pero se necesitan muchas iteraciones para la tolerancia mínima

[illegible]

X0 = i se obtiene la solución con un número menor de iteraciones

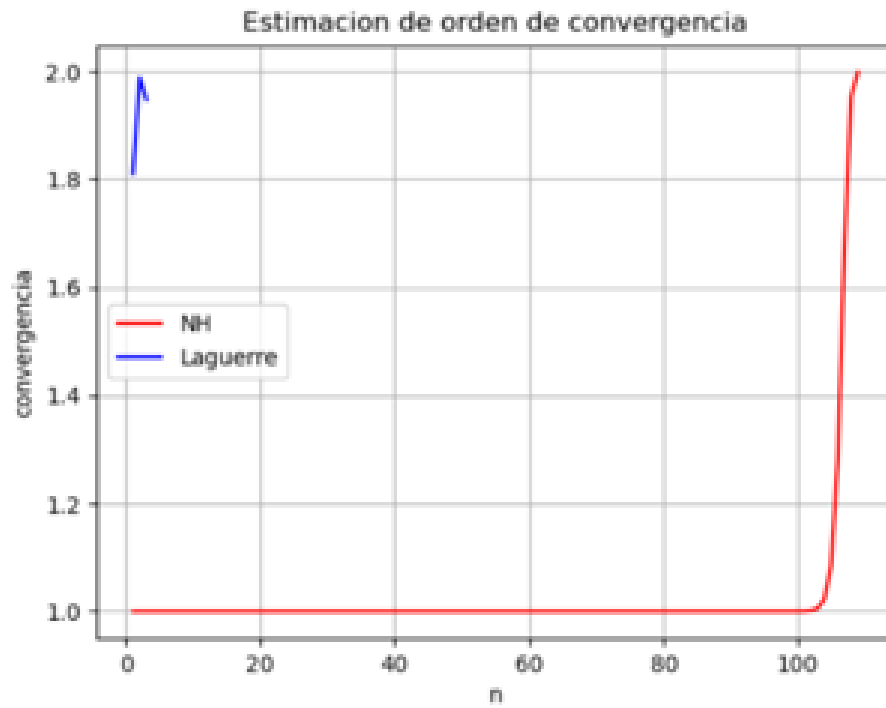




$$P(x) = x^2 + 4$$

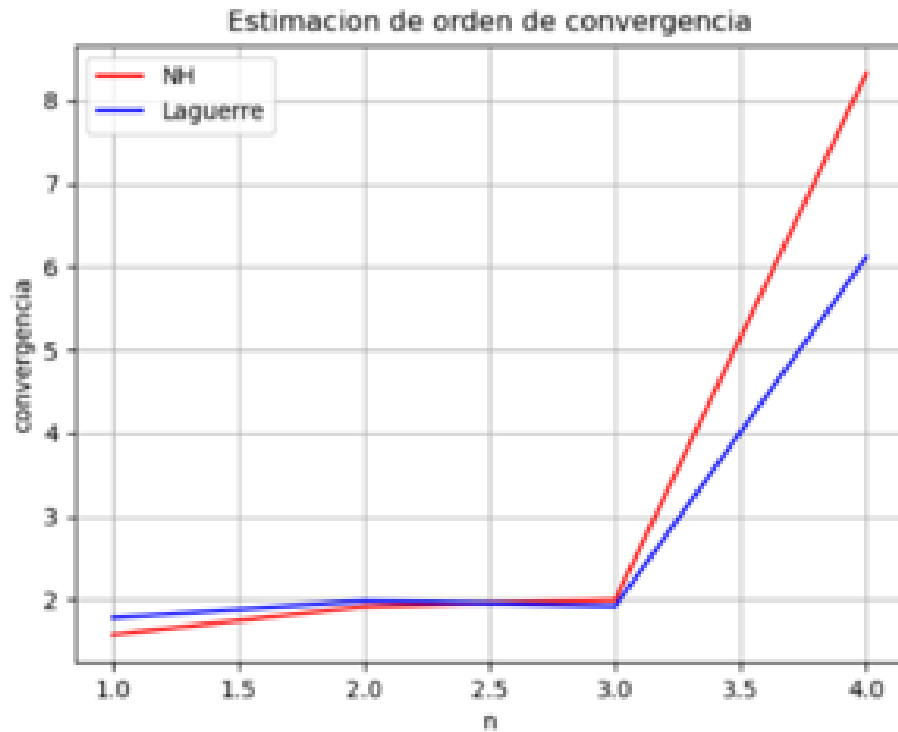
Teniendo en cuenta el proceso donde el algoritmo realiza el mayor número de iteraciones ( $x_0 = 0$ ) en el método de Newton-Horner, se puede realizar más claramente la estimación del orden de convergencia. En el método de Horner se puede ver claramente una constante en  $p = 1$  hasta el final donde se produce la irregularidad, mientras que





$$P(x) = x^2 - 5x + 6$$

En el cálculo de las dos raíces de este polinomio se puede observar el mismo comportamiento de la gráfica, primero se manifiesta



*Comparativa con otros métodos numéricos y Conclusiones :*

*La convergencia de los métodos de Newton–Hornery y Laguerre nos asegura que los métodos se comportan de manera adecuada, con la restricción de solo poder ser aplicados a funciones polinomiales. Tal vez si se intentara realizar una estimación de la convergencia de los métodos de Newton–Raphson, con la restricción de solo poder ser aplicados a funciones polinomiales.*

## Parte 2. Método de Brent

### Parametros Métodos Brent y Taylor:

Los parámetros para estos algoritmos son:

1. Brent
  - a) Función: debe ser continua y tener signos diferentes en el rango AB
  - b) A: inicio de intervalo
  - c) B: fin de intervalo
  - d) Tolerancia
  - e) Full Output: True, para obtener los datos de la convergencia
  - f) Disp: True para mostrar el orden de convergencia
2. Taylor
  - a) Función
  - b) X: donde se quiere aproximar el polinomio

c) N: el orden del polinomio a aproximar

### Significancia

El algoritmo de Brent que nos proporciona Python nos permite hallar resultados con un error mínimo, para el ejemplo, encontramos que nos da una solución, en pocas iteraciones, con un error de hasta  $2^{*-90}$ .

```

brent (0.6666685251964675,          converged: True
      flag: 'converged'
      function_calls: 50
      iterations: 49
      root: 0.6666685251964675)
taylor x/3 - 0.296296296296296
[0.66667125]
>>>
===== RESTART: C:\Users\pbr98\Desktop\reto1\bren
brent (0.6666678608664627,          converged: True
      flag: 'converged'
      function_calls: 47
      iterations: 46
      root: 0.6666678608664627)
taylor x/3 - 0.296296296296296
[0.66667125]
>>> |

```

En la imagen anterior vemos dos resultados, donde en el primero se ha utilizado un error de  $2^{*-90}$  mientras que, en el segundo, la tolerancia fue de  $2^{*-16}$ . Esto no afecta el orden de convergencia a grosso modo, pues la cantidad de iteraciones ha disminuido muy poco para lo que aumento el error.

De esto podemos concluir además que el método nos ofrece significancia hasta el 5to o 6to decimal, de ahí en adelante, el error podría aumentar.

En este caso, para hallar la raíz del polinomio de Taylor, se utilizó la función root, la cual no es precisa, por lo que se optó por hallar las raíces con brentq para su comparación.

```

brent (0.6666685251964675,          converged: True
      flag: 'converged'
      function_calls: 50
      iterations: 49
      root: 0.6666685251964675)
taylor 1.33333344365383e-10*x - 8.8888229399231e-11
[ 0.666661665334398 ]
>>>
===== RESTART: C:\Users\pbr98\Desktop\reto1\bren
brent (0.6666685251964675,          converged: True
      flag: 'converged'
      function_calls: 50
      iterations: 49
      root: 0.6666685251964675)
taylor 1.33333344365383e-10*x - 8.8888229399231e-11
[ 0.666661665334398 ]

```

En el caso de Taylor, evaluando el polinomio arrojado con brentq, nos da resultados iguales con error tanto de  $2^{*}90$  como de  $2^{*-16}$  y el valor, nos presta las mismas 5 cifras significativas, hasta volverse completamente diferente a las dos respuestas arrojadas por brentq.

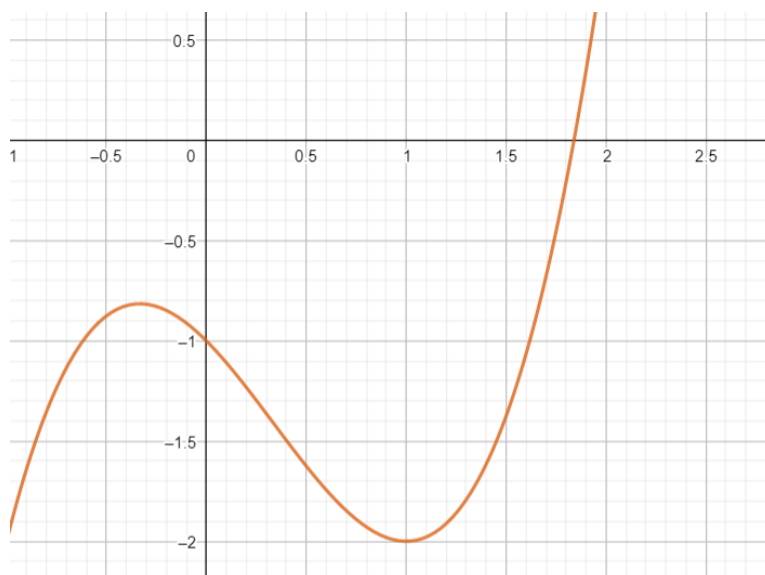
El algoritmo de Taylor presenta valores distintos desde el 5 decimal, incluso cambiando el grado del polinomio, el mejor parece ser con el polinomio de grado 4.

### Algoritmo de Brent

Créditos a Oscar Veliz, con un video que me explico el método.

Para la siguiente función  $f(x)$  la grafica es la que se muestra.

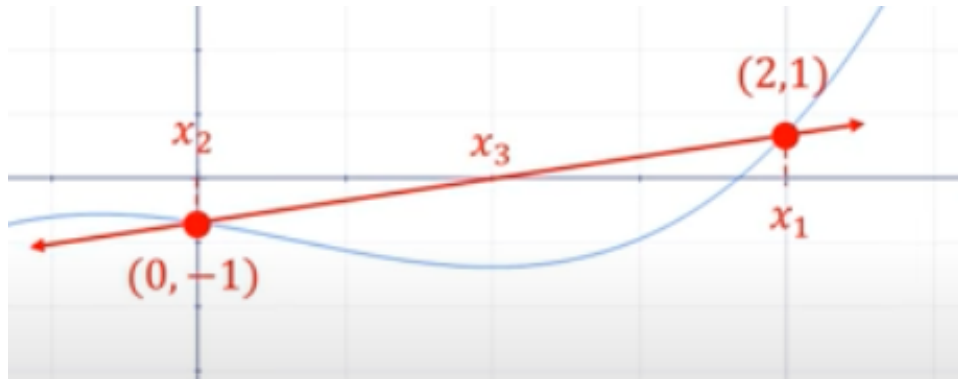
$$f(x) = x^3 - x^2 - x - 1$$



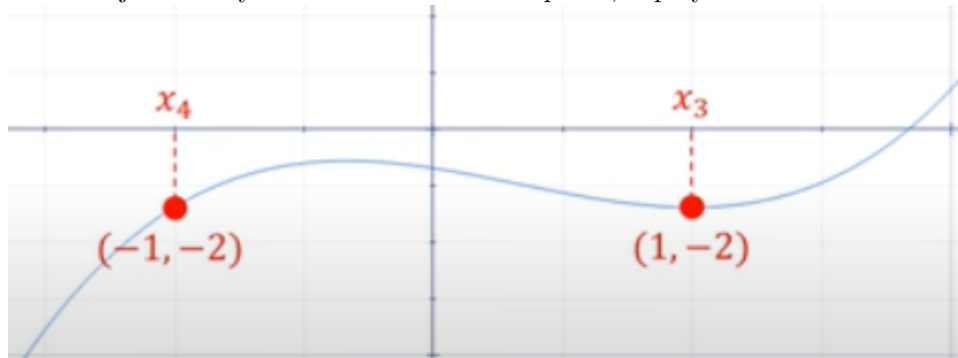
Para poder aplicar el algoritmo, es necesario que la función sea continua y que en el intervalo donde vamos a evaluar la función,  $f(x)$  tenga signos contrarios, como vemos en la función, entre 1.5 y 2, hay un cambio de signo, por lo que se puede aplicar este método.

Para explicar este método entonces debemos también explicar los tres métodos que lo componen.

Método de la secante.

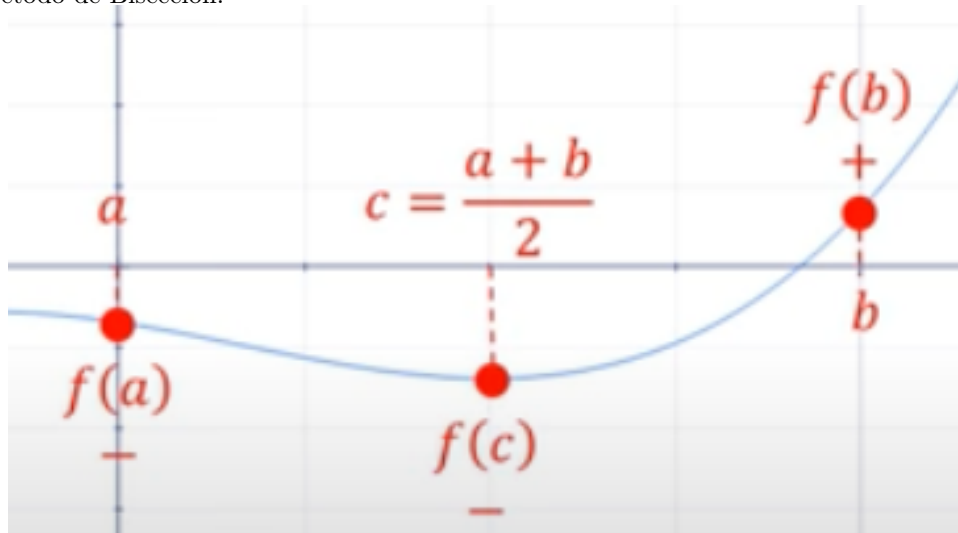


En una función, tomamos dos puntos y dibujamos una secante entre ellos, vemos que toca el eje x en  $x_3$  y este se convierte en otro punto, al proyectarlo a la función.



Utilizando  $x_2$  y  $x_3$ , dibujamos una secante también y vemos que se interseca en  $x_4$ , realizamos el mismo procedimiento y vemos que la secante no se interseca con el eje x por lo que ahí el método tiene un problema y es que en este caso, el método diverge.

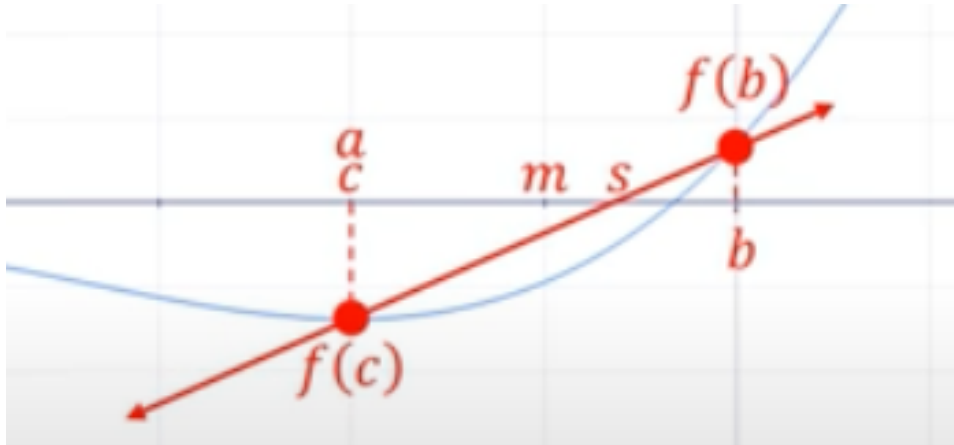
Método de Bisección.



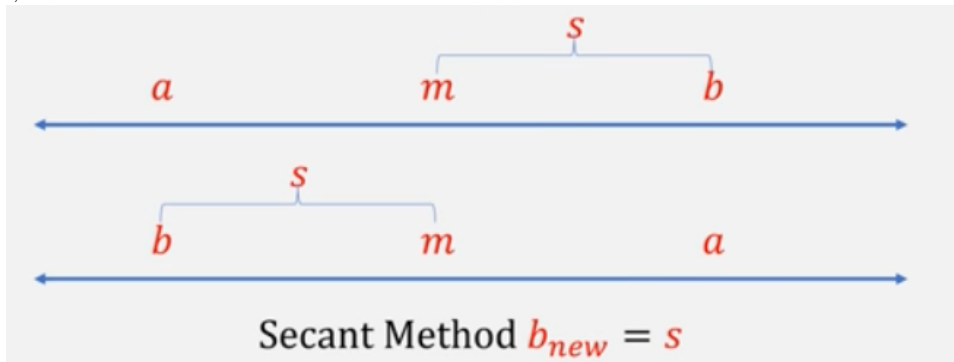
En este método, tomamos de nuevo dos puntos,  $a$  y  $b$  y sacamos un tercer punto en la mitad. Se repite el procedimiento utilizando  $c$  y  $b$  ya que el signo eso puesto. Repetimos el proceso hasta llegar a la raíz.

Método de Dekker.

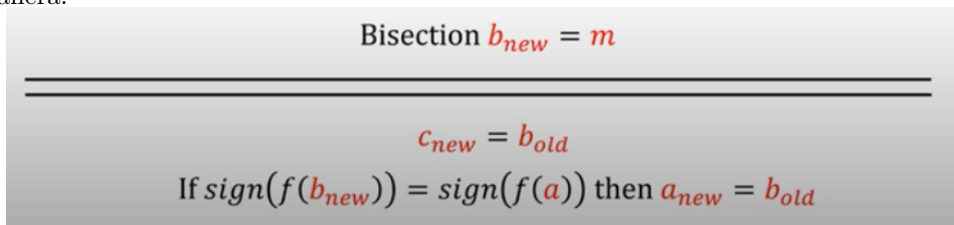
El propone que combinar estos dos últimos métodos, ya que el método de bisección es muy lento pero converge y el de la secante puede diverger. Para esto, dekker dice que entonces, se usara el método de la secante y se cambiara al método de bisección cuando se necesite. Teniendo  $a$  y  $b$ , con signos distintos, se obtiene  $m$ , como  $a+b/2$ , se utiliza el método de secante, pero si se divide por 0 entonces  $s$  permanecerá siendo  $m$ .



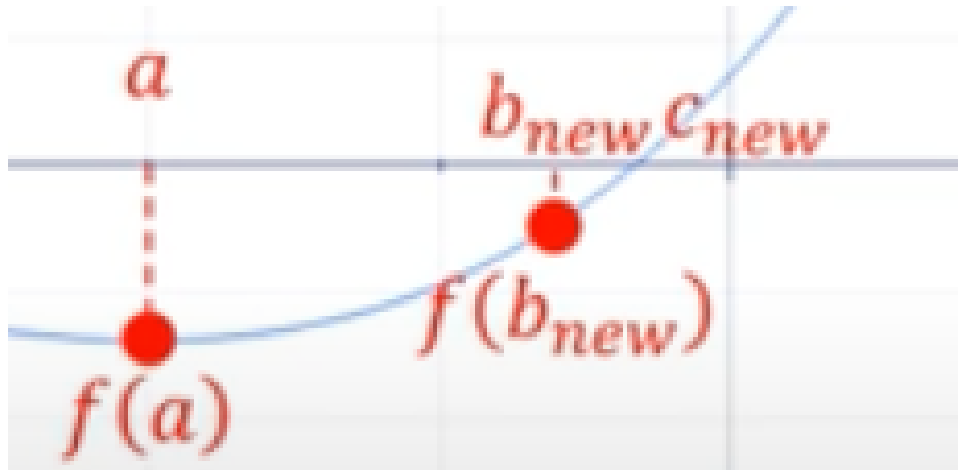
Existen entonces diferentes casos posibles. El primero, es cuando  $s$  está entre  $m$  y  $b$ , donde el nuevo  $b$  será.



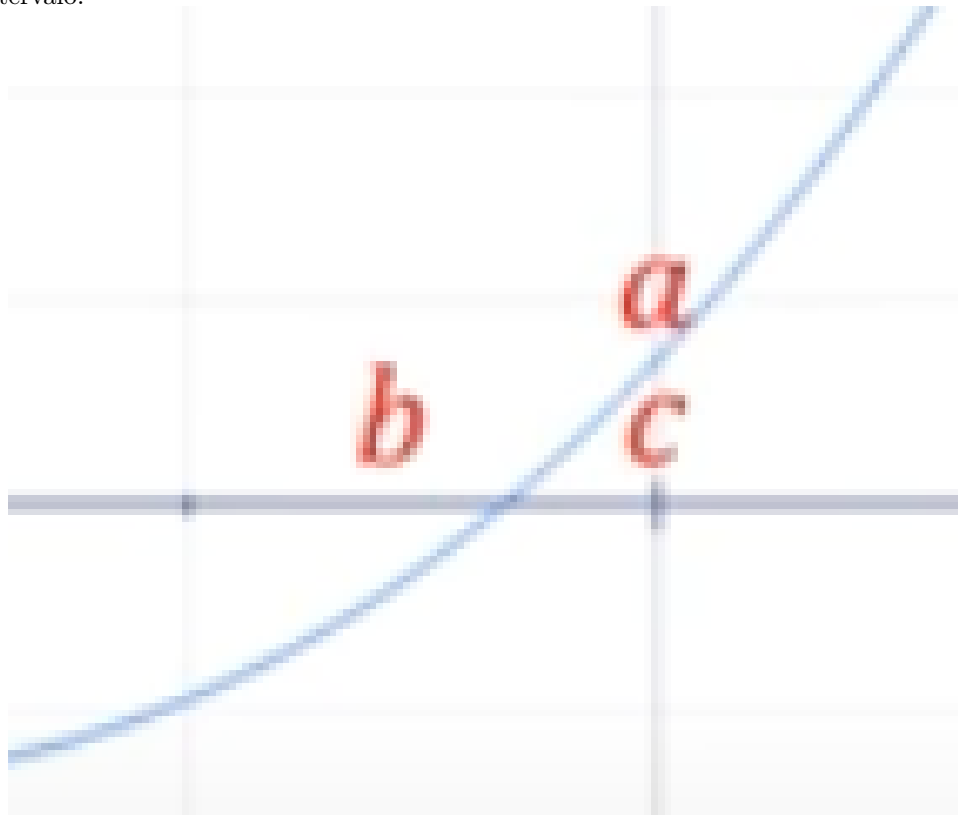
De no ser así, se utilizará bisección y se actualizarán las variables de la siguiente manera.



En nuestro ejemplo, tenemos que  $s$  está entre  $m$  y  $b$  por lo que  $b$  va a ser ahora  $s$  y  $c$  va a ser nuestro  $b$  anterior.



Pero como vemos, cuando validamos los signos vemos que la raíz no está en el intervalo.



Por ello, decimos que  $a$  va a tener el último valor de  $b$ , los cuales se van a invertir ( $a$  y  $b$ ) y se va a repetir el proceso.

Aquí va un ejemplo numérico.

Donde utilizamos la función  $x^3 - x^2 - x - 1$ , empezando en 1 y 2 con un error

de  $10^{-7}$  e iteramos hasta que la resta de  $a-b$  sea menor que el error, por lo que acaba el algoritmo y  $b$  es nuestra respuesta final.

$a$	$b$	$c$	$m$	$s$	$f(a)$	$f(b)$	$f(b_{new})$
1	2	1	1.5	1.666666667	-2	1	-0.814814815
2	1.666666667	2	1.833333333	1.81632653061	1	-0.814814815	-0.123230967
2	1.81632653061	1.666666667	1.90816326531	1.84299391123	1	-0.123230967	0.020341595
1.81632653061	1.84299391123	1.81632653061	1.82966022092	1.83921563321	-0.123230967	0.020341595	-0.00038904
1.84299391123	1.83921563321	1.84299391123	1.84110477222	1.83928653794	0.020341595	-0.00038904	-0.000001189
1.84299391123	1.83928653794	1.83921563321	1.84114022459	1.83928675523	0.020341595	-0.000001189	0.00000000006
1.83928653794	1.83928675523	1.83928653794	1.83928664658	1.83928675521	-0.000001189	0.00000000006	0

Método de interpolación inversa.

Su método consiste, o añade, una interpolación cuadrática que, de no lograrse, usa secante, y, si aún no así no se consigue, usa bisección. Para no caer con raíces imaginarias, Brent usa cuadrática inversa. Para la interpolación utiliza polinomios de Lagrange, lo que produce una parábola.

- $x_1 = 0, x_2 = 1, x_3 = 2$
- $y_1 = -1, y_2 = -2, y_3 = 1$
- Swap  $x$ 's for  $y$ 's
- $L_2(y) = x_1 \frac{y-y_2}{y_1-y_2} \cdot \frac{y-y_3}{y_1-y_3} + x_2 \frac{y-y_1}{y_2-y_1} \cdot \frac{y-y_3}{y_2-y_3} + x_3 \frac{y-y_1}{y_3-y_1} \cdot \frac{y-y_2}{y_3-y_2}$
- $L_2(y) = 0 \frac{y-(-2)}{(-1)-(-2)} \cdot \frac{y-1}{(-1)-1} + 1 \frac{y-(-1)}{(-2)-(-1)} \cdot \frac{y-1}{(-2)-1} + 2 \frac{y-(-1)}{1-(-1)} \cdot \frac{y-(-2)}{1-(-2)}$
- $L_2(y) = \frac{2y^2}{3} + y + \frac{1}{3}$

Como interpolación, tiene varios problemas también a la hora de hallar las raíces, se usa de manera prudente en el método de Brent.

Método de Brent.

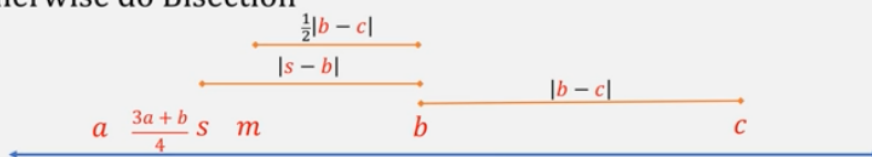
El método trabaja los mismos valores  $a, b, c, m$  y  $s$  que venimos hablando. Primero, intenta calcular  $s$  con el método de interpolación, si falla, saca la secante, si no, vuelve a bisección. Actualiza  $A, b, c$  mirando si  $s$  esta entre  $b$  y  $(3*(a+b))/4$  y repite validando el error y que  $F(b)$  o  $s$  sean 0. A demás, el método guarda un nuevo valor  $d$ , que guarda el valor de  $c$ .

La idea del algoritmo es forzar bisección cuando no nos den resultados los demás métodos. Para esto, Brent usa algunas pruebas.

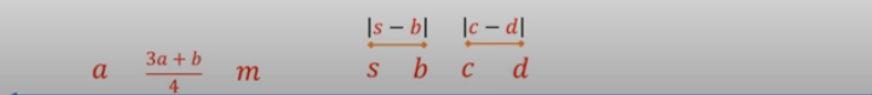
Si usamos bisección y  $s-b < (b-c)$  entonces se va a interpolar, si no, se continua con bisección. Esto es porque bisección nos daría un mejor salto que la interpolación.



If used Bisection and  $|s - b| < \frac{1}{2}|b - c|$  then interpolate,  
otherwise do Bisection

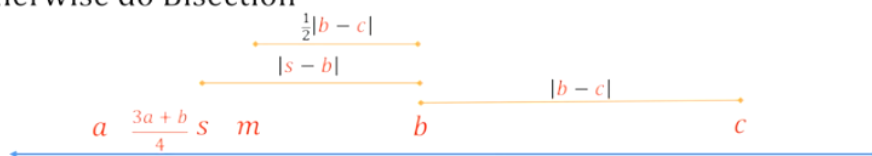


If used (Secant or IQI) and  $|s - b| < \frac{1}{2}|c - d|$  then interpolate,  
otherwise do Bisection

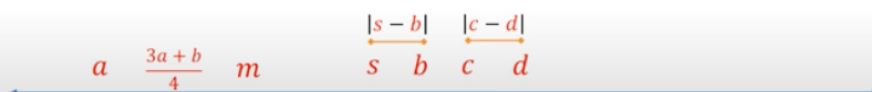


Ahora, si usamos secante o interpolación y  $s-b < 1/2c-d$  entonces también se sigue con interpolación, si no, se fuerza de nuevo bisección. Esto quiere decir que si la interpolación da pasos muy pequeños, será mejor utilizar bisección.

If used Bisection and  $|s - b| < \frac{1}{2}|b - c|$  then interpolate,  
otherwise do Bisection



If used (Secant or IQI) and  $|s - b| < \frac{1}{2}|c - d|$  then interpolate,  
otherwise do Bisection

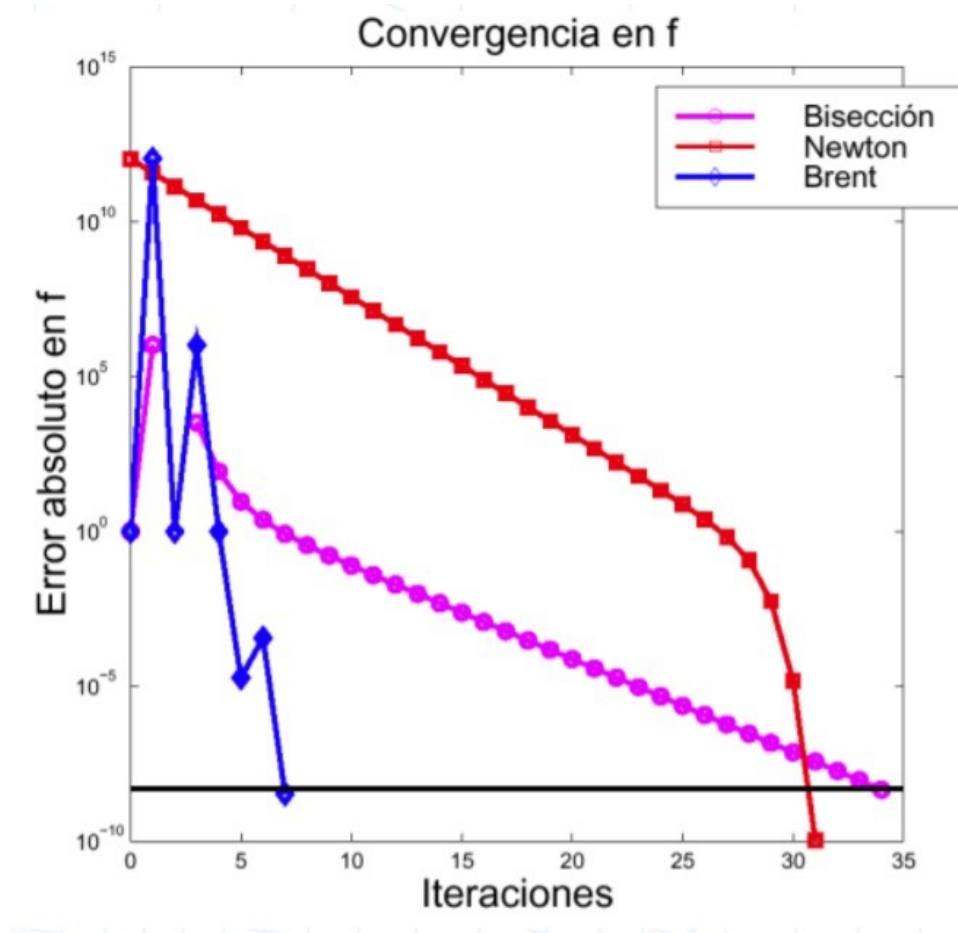


## Comparaciones

Method	Steps	Root Found
Bisection	24	1.8392867743968964
Secant	6	1.839286755226924
Dekker	7	1.8392867552141612
IQI - $1, \frac{3}{2}, 2$	5	1.8392867552230825
Brent	6	1.8392867552141612

He encontrado estas comparaciones y he llegado a las siguientes conclusiones:

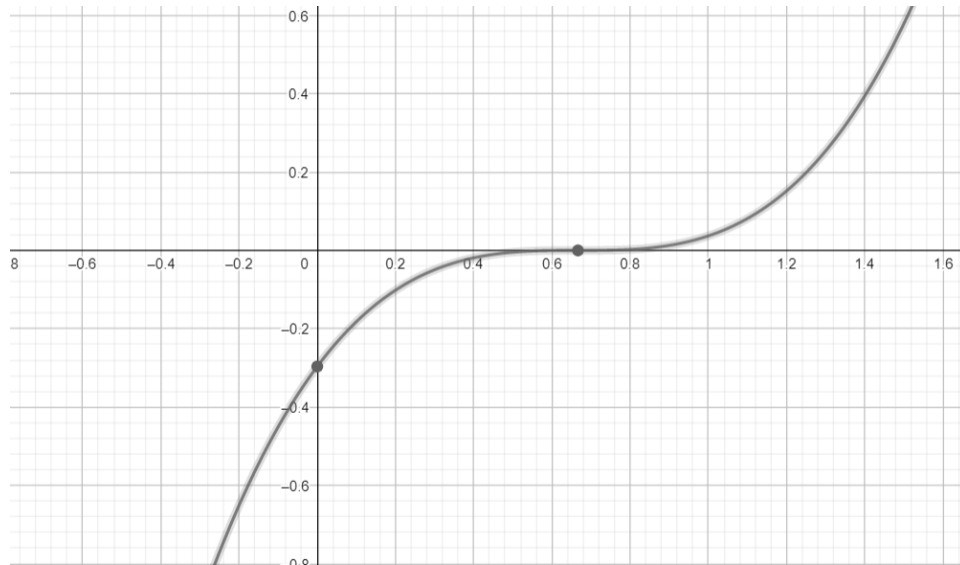
1. El algoritmo de Dekker es tan rápido como el de la secante, pero dependen del comportamiento de la función.
2. Interpolación Es tan rápido como Müller cuando funciona, aunque falla en repetidas ocasiones.
3. Brent es igual de rápido que interpolación cuando esta cerca, Dekker y secante cuando se comportan.
4. Brent dice que su algoritmo parece ser tan rápido como el de Dekker pero que además garantiza la convergencia para cualquier función con una velocidad razonable.



### Pruebas

Para estas pruebas, Taylor se mantendrá fijo en 0.6 con polinomio de grado 4.

$$f(x) = x^3 - 2x^2 + \frac{4x}{3} - \frac{8}{27}$$



1. En esta prueba, notamos que el método converge en 61 iteraciones y presenta un error de  $2.66\text{e-}06$  lo que es un error absoluto relativamente alto.

```
a: -3  b: 3
```

```
Brent
```

```
    converged: True
```

```
        flag: 'converged'
```

```
function_calls: 61
```

```
iterations: 60
```

```
        root: 0.6666680000766564
```

```
Taylor  0.6666706695951692
```

```
error:  2.669518512798952e-06
```

2. Para tratar de reducir el error, se han cambiado los límites del algoritmo

```

a: -2  b: 2
Brent
    converged: True
    flag: 'converged'
    function_calls: 61
    iterations: 60
    root: 0.6666660657635841
Taylor  0.6666706695951692
error:  4.603831585070495e-06

```

El error aumenta

3. Se cambian de nuevo los límites, esta vez, acercándolo mas a la solución

```

a: 0  b: 1
Brent
    converged: True
    flag: 'converged'
    function_calls: 50
    iterations: 49
    root: 0.6666685251964675
Taylor  0.6666706695951692
error:  2.1443987017200072e-06

```

No solo disminuye el error si no también el número de iteraciones.

4. Se reduce una ultima vez los límites, esta vez entre 0.5 y 1.

```

a: 0.5  b: 1
Brent
    converged: True
    flag: 'converged'
    function_calls: 48
    iterations: 47
    root: 0.6666628518987887
Taylor  0.6666706695951692
error:  7.81769638047436e-06

```

Logramos reducir aun mas el número de iteraciones, pero el error aumenta levemente. Aquí nos percatamos de que lo que ocurrió es que el valor del algoritmo de Brent se acercó aun más a la solución y

5. En esta prueba cambiamos la tolerancia del error de  $2^{*-90}$  a  $2^{*-16}$

```

a: 0.5  b: 1
Brent
    converged: True
    flag: 'converged'
    function_calls: 44
    iterations: 43
    root: 0.6666671560678465
Taylor  0.666670621360187
error:  3.465292340409931e-06

```

Se reducen las iteraciones nuevamente. Concluimos que el error con respecto a taylor siempre estará alrededor de  $10^{-6}$  ya que esta es la cantidad de cifras significativas que estudiamos en la parte de significancia.

### Parte 3. Metodo de Remez

#### Parametros Métodos Remez y Taylor:

Los parámetros para estos algoritmos son:

1. Remez
  - a) Función: debe ser continua
  - b) A: inicio de intervalo
  - c) B: fin de intervalo

- d) Tolerancia
- e) x: El valor inicial de para Chebyshev
- 2. Taylor
  - a) Función
  - b) X: donde se quiere aproximar el polinomio
  - c) N: el orden del polinomio a aproximar

El algoritmo de intercambio de Remez o el algoritmo de Remez es un algoritmo iterativo cuyo objetivo es el de encontrar aproximaciones simples a funciones. Para esto, el algoritmo de Remez comienza con la función a ser aproximada y un conjunto  $x$  de  $n+2$ . El algoritmo construye un polinomio de grado  $n$ -ésimo que lleva a valores de error alternos tomando como referencia  $n+2$ . Las ecuaciones del método se resuelven a continuación:

Dados los puntos de referencia se puede resolver el sistema para encontrar el polinomio  $P$  y el número  $e$ .

#### 1. Primer Paso:

El primer paso para desarrollar el metodo es seleccionar el valor de un  $x_0$  que debe estar dentro del intervalo de la funcion, con esto se genera un grupo de equis  $x_0, x_1, \dots, x_{n+1}$  en  $[a, b]$ , esta cantidad de numeros va desde 0 hasta el valor de  $n+1$ . En el cual  $n$  es el grado que fue enviado a la funcion. A continuacion la formula desarrollada para generar el valor de las equis:

$$(0.1) \quad x_i = \frac{1}{2}(a+b) + \frac{1}{2}(b-a) \cdot \cos\left(\frac{2i-1}{2n}\pi\right)$$

#### 2. Segundo Paso:

Vamos a considerar el siguiente sistema de ecuaciones:

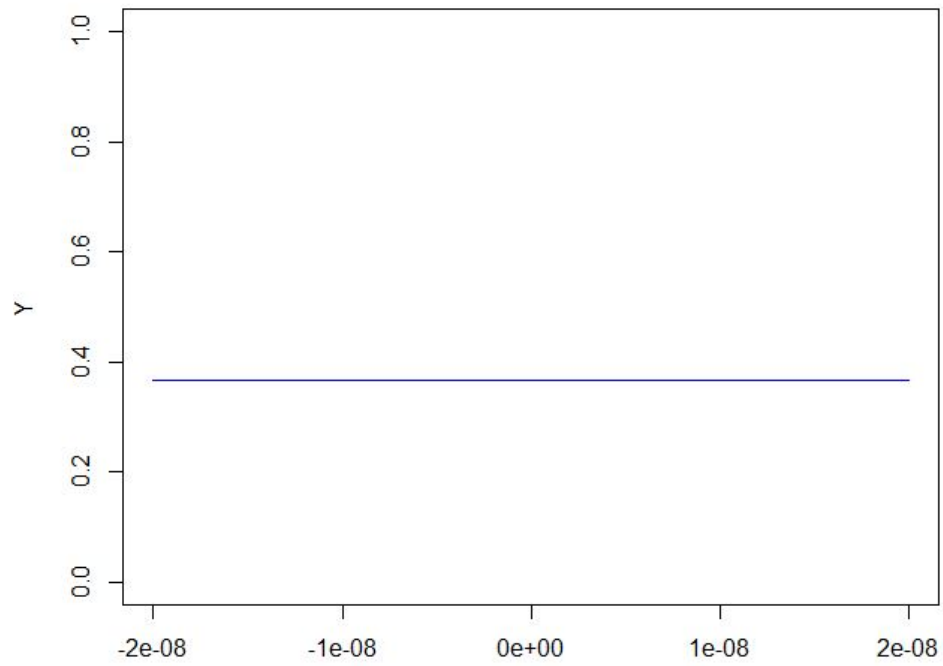
$$\begin{cases} p_0 + p_1x_0 + p_2x_0^2 + \dots + p_nx_0^n - f(x_0) & = +\epsilon \\ p_0 + p_1x_1 + p_2x_1^2 + \dots + p_nx_1^n - f(x_1) & = -\epsilon \\ p_0 + p_1x_2 + p_2x_2^2 + \dots + p_nx_2^n - f(x_2) & = +\epsilon \\ \dots & \dots \\ p_0 + p_1x_{n+1} + p_2x_{n+1}^2 + \dots + p_nx_{n+1}^n - f(x_{n+1}) & = (-1)^{n+1}\epsilon \end{cases}$$

Entonces cada una de las columnas del sistema de ecuaciones excepto la ultima, corresponden a cada uno de los grados que fueron enviados a la funcion. Y para cada una de las filas se considerara el valor de las equis generadas en el primer paso. La ultima columna del sistema corresponde al error que se recibe en la funcion

#### 3. Tercer Paso:

Solucionando el sistema de ecuaciones encontraremos un polinomio que corresponde a la aproximacion que esperamos que el metodo de remez nos proporciona y así compararlo con la funcion  $f(x)$

El metodo de remez

**Remez****Remez vs  $f(x)$** 