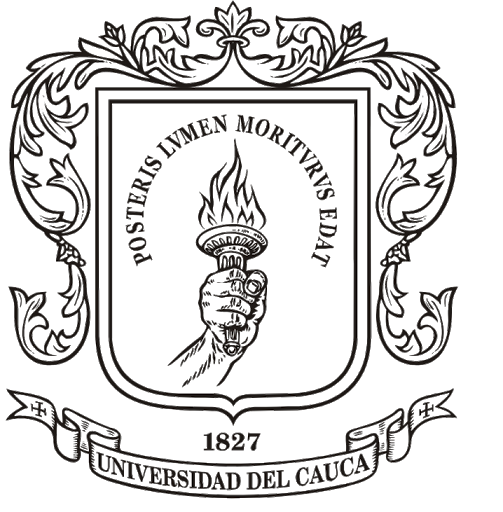
**UNIVERSIDAD DEL CAUCA**



Presentado por:

**JUAN CAMILO ZARTA CAMPO**

Presentado a:

**CARLOS ALBERTO ARDILA**

Materia:

**ANALISIS NUMERICO**

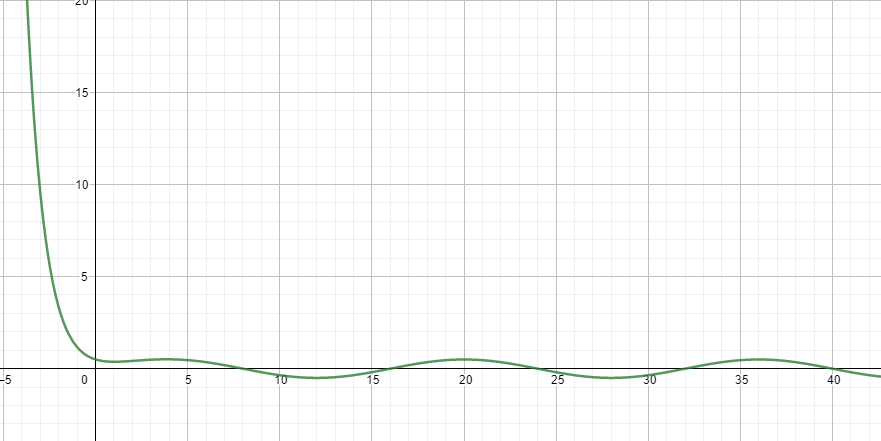
Trabajo:

**PRIMER PARCIAL**

**Primer Polinomio**

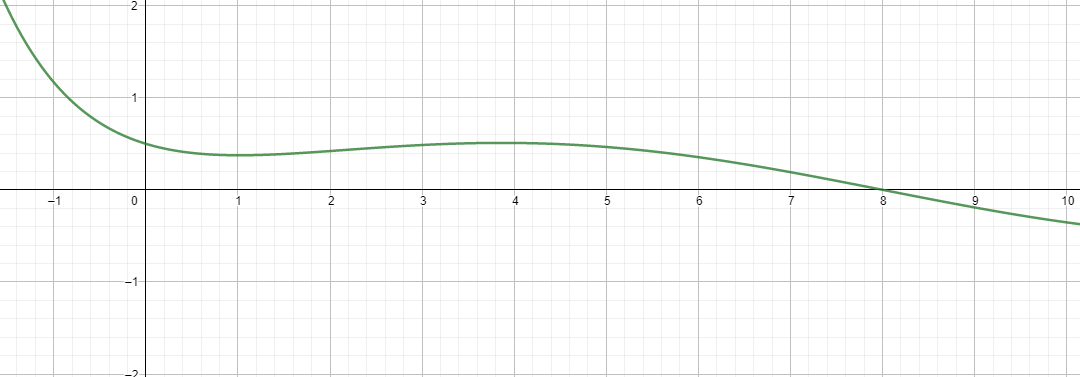
*h = h0 [sen ((2 π x)/ λ) cos ((2π t v)/ λ) + e^ (-x)]*

λ = 16 t = 12 v = 48 h = 0.5 h0.



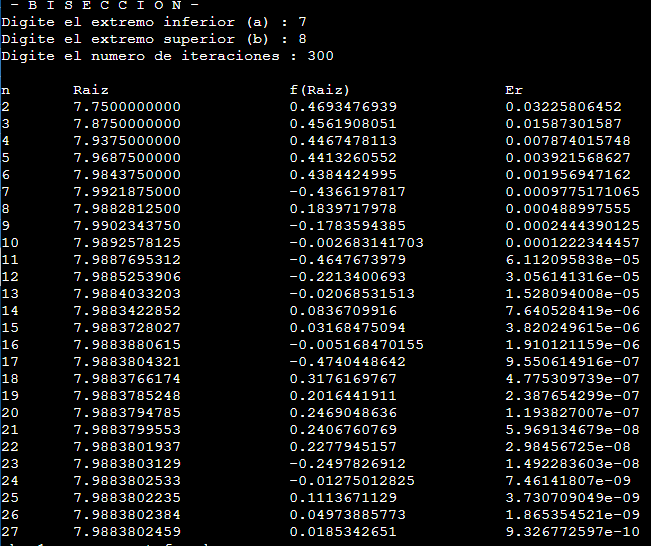
Para este polinomio el ejercicio nos pide que resolvamos el valor positivo más bajo para x  
así que tomaremos la primera raíz.

**Raíz 1**



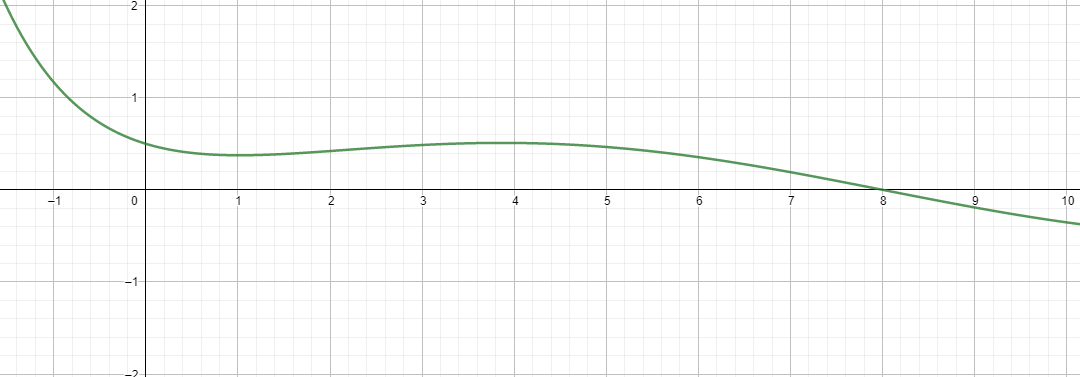
* **Bisección**

Usaremos el método de bisección con los intervalos (7, 8) y le daremos un cupo de 300 iteraciones.



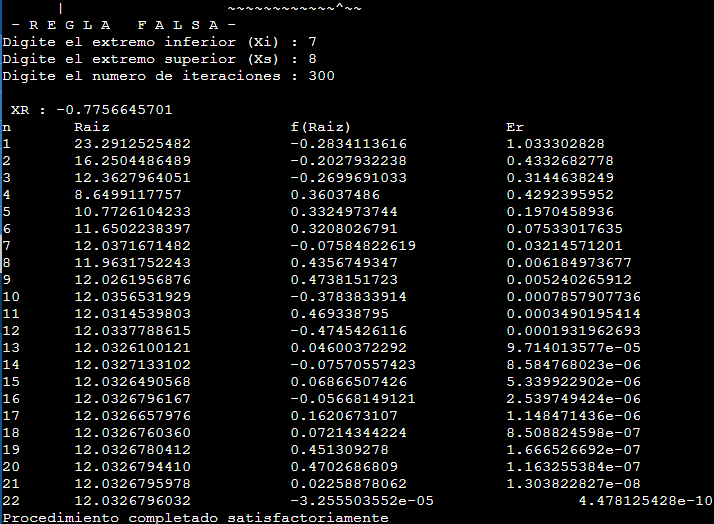
Vemos que nuestra raíz aproximada es 7.9883802459 con un error porcentual de 9.326772597e-10 después de 27 iteraciones, el método fue efectivo.

**Raíz 1**

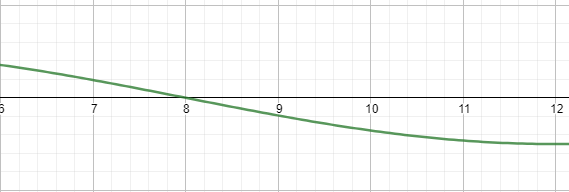


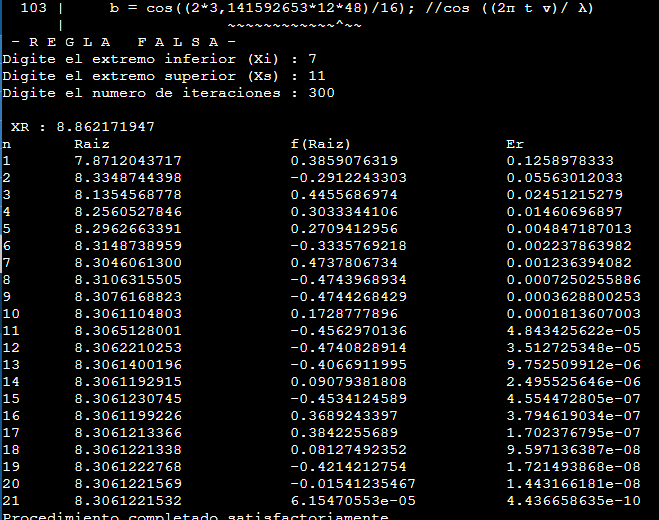
* **Regla falsa**

Usaremos el método de regla falsa para este tendremos los intervalos de (7 y 8) nuevamente a ver si funcionan



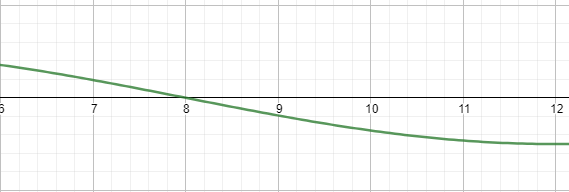
Vemos que se realizaron 22 iteraciones pero visualmente podemos deducir que la raíz aproximada no es esa, parece ser que es un valor muy cercano a la raíz, así que usaremos otro intervalo para probar este método, usaremos el intervalo de (7, 11)





En este caso nos da una raíz que si vemos en la gráfica coincide más que el resultado anterior, dándole un poco más de espacio para que se mueva este método, con 21 iteraciones y un error porcentual de 4.436658635e-10 tenemos una raíz aproximada de 8.3061221532

**Raíz 1**

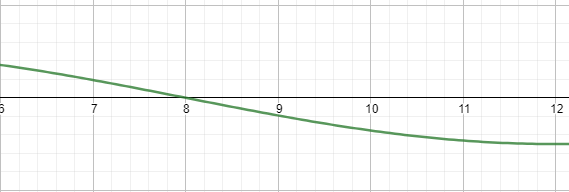


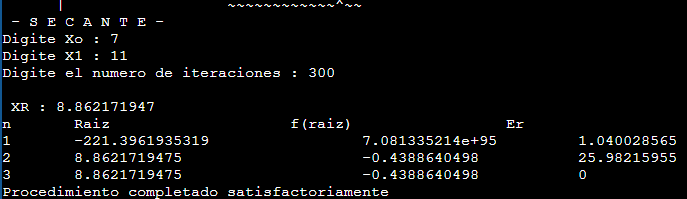
* **Secante**

Usaremos el método de Secante para hallar nuestra aproximación a la primera raíz de nuestra ecuación, para ello usaremos el intervalo de (7, 8) nuevamente



Claramente vemos que pese a que después de 64 iteraciones nos diga que funciono y encontró la raíz, no es así, este fracaso, así que intentaremos de nuevo con otro intervalo, (7, 11)

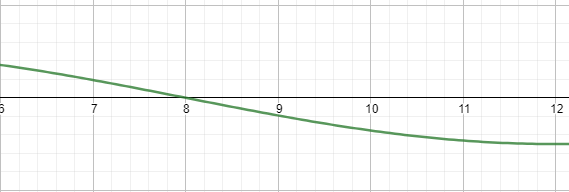


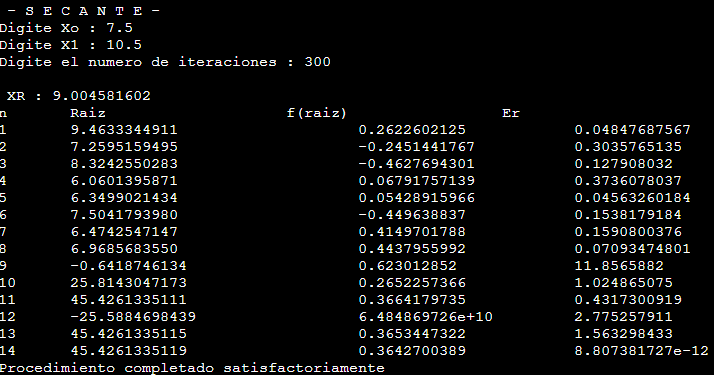


Ahora por el contrario, vemos que el método funciono muy eficazmente, también fue necesario darle un espacio un poco más grande a nuestro intervalo para que el método funcionara

Nos da un error porcentual de 0, con una raíz aproximada de 8.8621719475, lo cual visualmente en la gráfica vemos es un poco lejano, en términos de decimales. Probaremos con otros valores a ver qué resultados nos arroja esta vez

Usaremos el intervalo de (7.5 , 10.5)





Nuevamente no nos dio la primera raíz.

**Tabla De Resultados**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Método | Valor(es)  Inicial(es) | Raíz | Iteraciones | Cota de Error |
| Bisección | xi = 7  xs = 8 | 7.9883802459 | 27 | 9.326772597e-10 |
| Regla Falsa | xi = 7  xs = 11 | 8.3061221532 | 21 | 4.436658635e-10 |
| Secante | xi = 7  xs = 11 | 8.8621719475 | 3 | 0 |

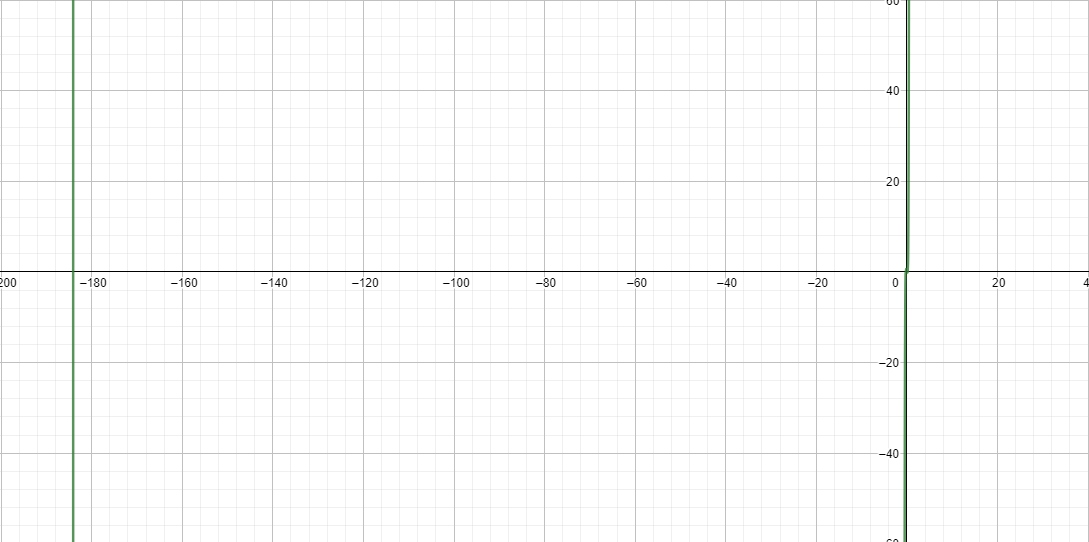
**Conclusiones:** podemos observar en cuestión de velocidad de procedimientos dados para encontrar la raíz (iteraciones) la secantes es mucho más rápida para esta ecuación.

Por otra parte observemos que también hay algo curioso dentro de la secante, ya que este tiene un erro porcentual de 0, es decir que la raíz en teoría debería ser 8.8621719475, lo cual en la gráfica se aprecia de otra forma, pese a que sea algo mínimo es notorio, siendo así entonces este método no funciono del todo bien

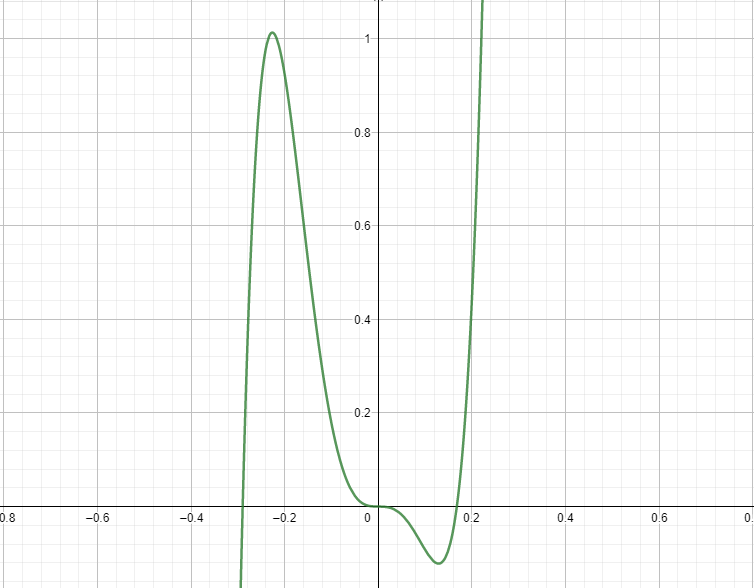
Respecto a los valores vemos que son bastante similares, la diferencia que hay entre ellos no es muy grande que digamos, y si comparamos entre bisección y regla falsa podemos decir que regla falsa nos da un resultado con menor error porcentual que la de bisección, esto es algo también poco notorio ya que varía en decimales.  
En cuanto iteraciones vemos que la regla falsa también gata menor cupo de iteraciones que el método de bisección. Ahora cabe mencionar que el lo que es el método de la regla falsa y el método de secante debemos tener cuidado con los valores que le damos ya que si son valores muy cercanos o muy lejanos la raíz no se encontrar como es deseado, sino que lo hará mas mejor, lo hará mal, o no lo hará.

**Segundo Polinomio**

*f(x) = 19x^6 + 3500x^5 + 426x^4 − 170x^3*



Visto desde un poco más lejos.



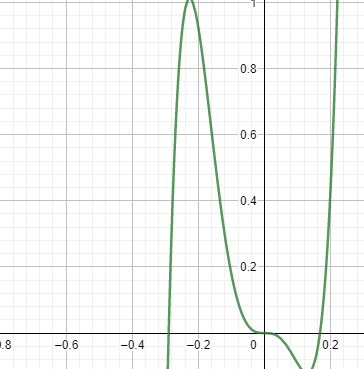
Visto desde más cerca

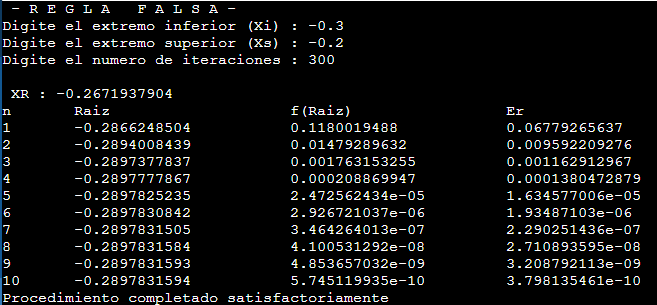
De este polinomio de grado 6 podemos observar en la dos imágenes anteriores que tenemos 4 raíces reales

* **REGLA FALSA**

Empezaremos por encontrar o aproximarnos a la raíz usando el método de bisección, para este caso usaremos los siguiente valores, xi = -0.3 xs = -0.2 teniendo en cuenta que tenemos 4 raíces reales y un cupo de 300 interacciones para encontrar la raíz.

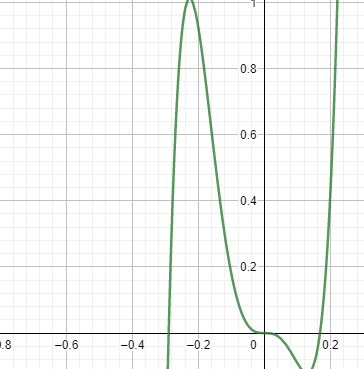
**RAIZ 1**

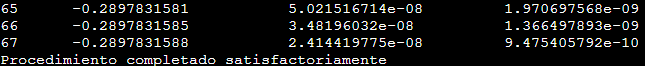




Notamos que se tarda poco, con un resultado dado después de 10 iteraciones. Con un error porcentual de 3.798135461e-10

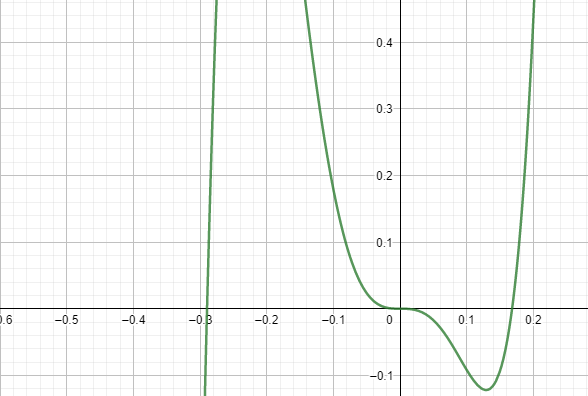
Veamos que sucede si cambiamos estos valores por xi = -0.4; xs = -0.1

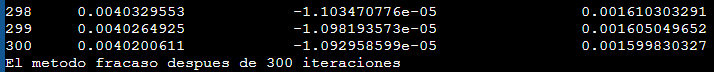




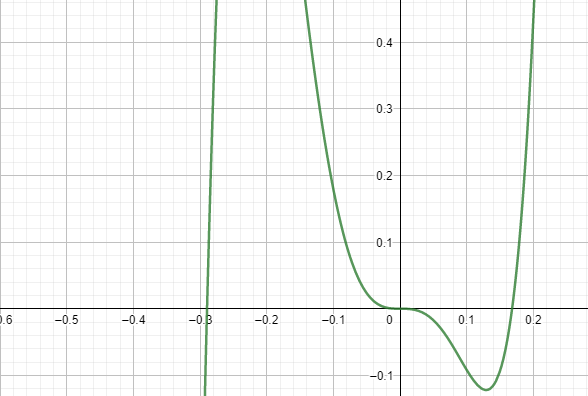
Notemos que el método funciono con un error porcentual de 9.475405792e-10, un poco mayor que el anterior y con muchas más iteraciones, 57 iteraciones más que el anterior

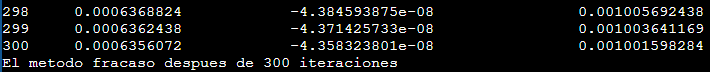
**RAIZ 2**

  
Para la segunda raíz usaremos los valores xi = -0.1 y xs = 0.1.



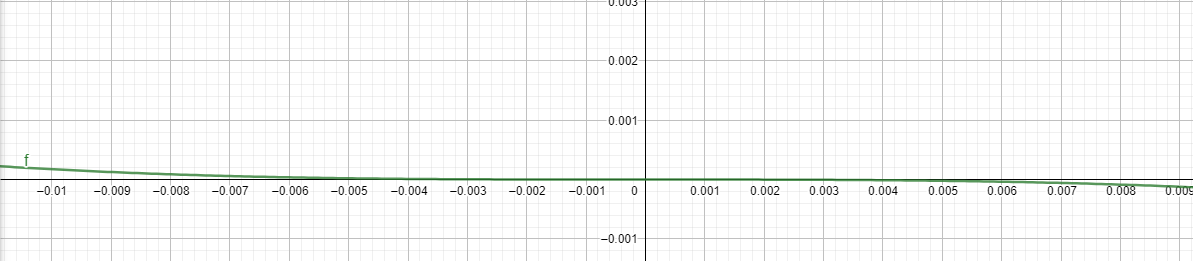
Podemos observar que el método ha fracasado después de 300 iteraciones, la raíz aproximada no pudo ser encontrada, así que intentaremos con otros valores xi = -0.02 y   
xs = 0.02

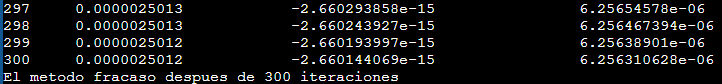




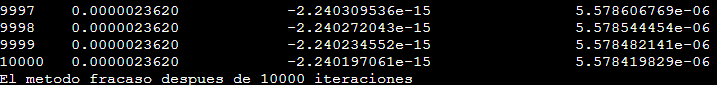
El método no fue satisfactorio, el cupo de las 300 iteraciones se terminó antes de poder encontrar la raíz aproximada.

Haremos un último intento con números más pequeños cercanos a la raíz tales como xi = -0.001 y xs = 0.001



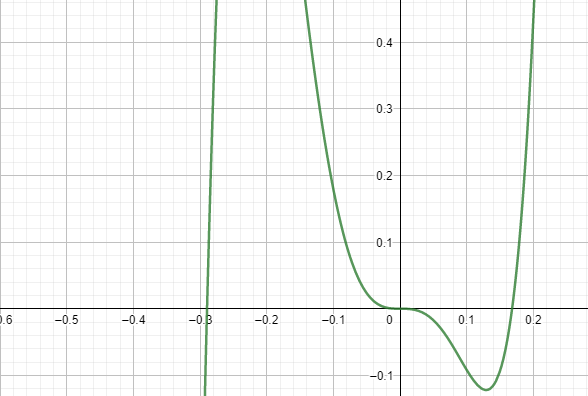


El método fracaso después 300 iteraciones nuevamente, haremos un intento dándole más cupo de iteraciones a ver si es posible encontrarlo, le daremos 10000 iteraciones

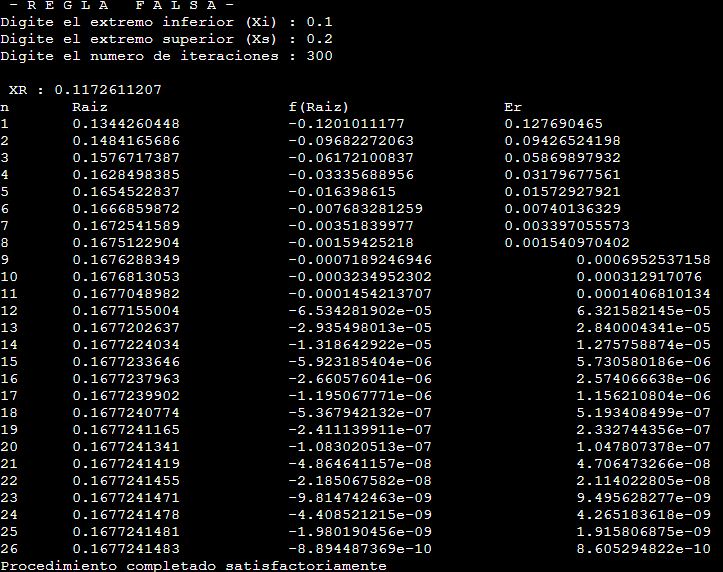


Podemos ver que el método fracaso nuevamente aun después de haberle dado 10.000 iteraciones. Su error porcentual disminuyo muy poco a comparación del anterior.

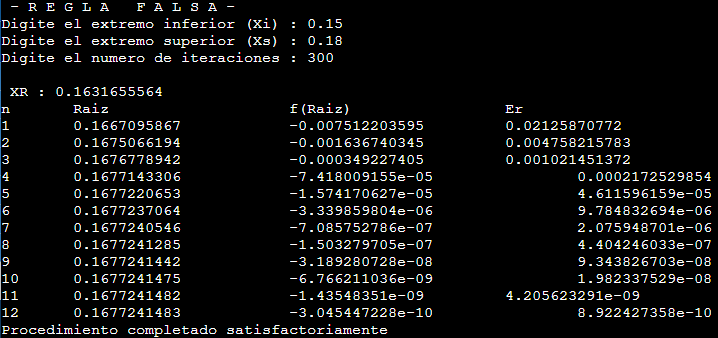
**RAIZ 3**



Para la raíz 3 usaremos xi = 0.1 y xs = 0.2 con un cupo de 300 iteraciones nuevamente



Podemos ver que la raíz fue encontrada después de 26 iteraciones con un error porcentual de 8.605294822-10 ahora intentemos cambiar estos valores por xi = 0.15 y xs 0.18

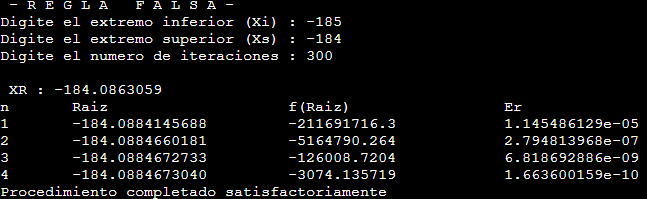


El proceso para este fue mucho más rápido que el anterior, con un total de 12 iteraciones y un error porcentual de 8.922427358e-10, un error porcentual por mínimo un poco más alto pero con menos iteraciones

**RAIZ 4**

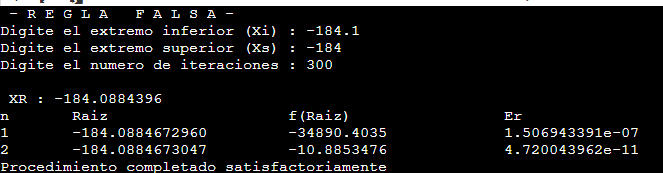
****

Para esta cuarta y última raíz real usaremos los intervalos xi = -185 y xs = -184



Podemos ver que el método para esta raíz fue bastante eficaz, en tan solo 4 iteraciones encontró la aproximación de la cuarta raíz con error porcentual de 1.663600159e-10

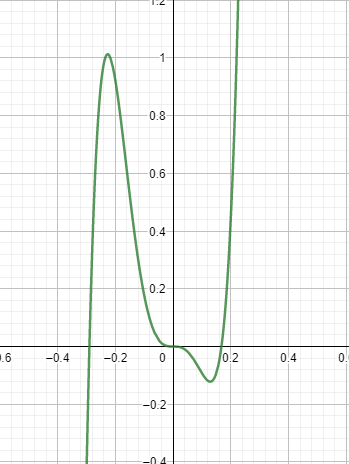
Ahora haremos una prueba con valores más cercanos a la raíz. Tales como   
xi = -184.1 y xs = -184

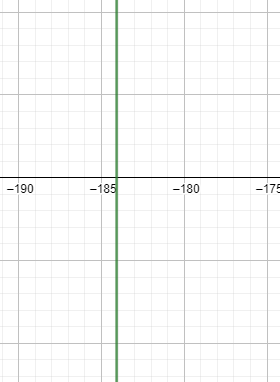


Podemos ver que el valor aproximado de la raíz aparece después de 2 iteraciones con un error porcentual más pequeño de 4.720043962e-11

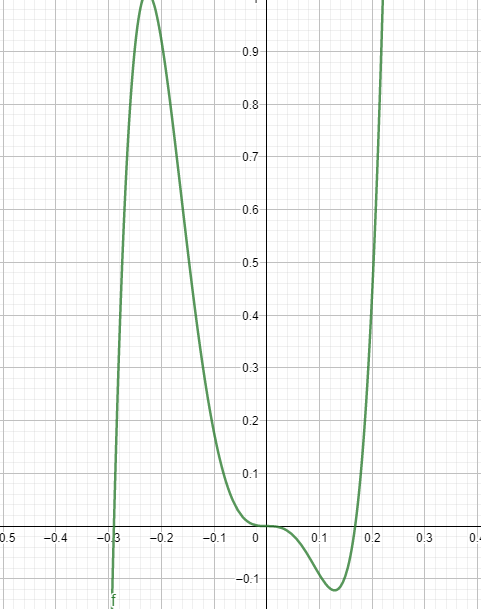
**Conclusión:** para el método de Regla falsa fue bastante efectivo en su mayoría, cabe resaltar que el número de iteraciones varia dependiendo la distancia de los valores de xi y xs que se le den, entre más cerca de la raíz estén estos valores consume menos cantidad de iteraciones, también vemos que fracaso al querer encontrar la segunda raíz, incluso dándole 10.000 iteraciones no fue posible que funcionara el método de regla falsa para este caso

* **MULLER**



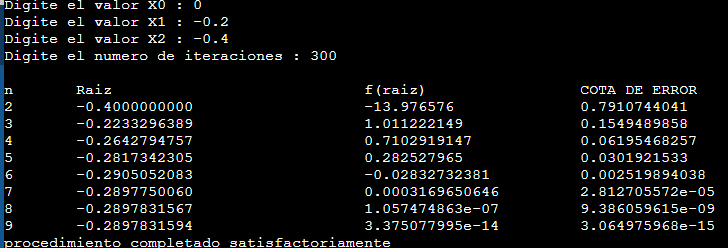


**Parábola 1**



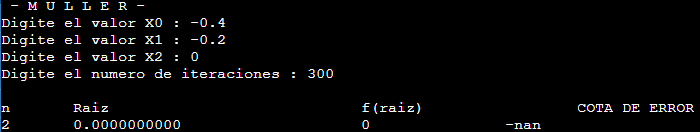
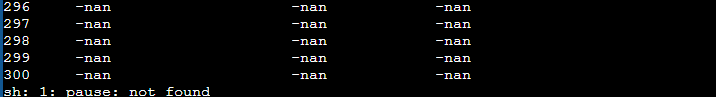
Usando el método de Müller asignaremos los valores para la primera parábola:  
x0 = 0  
x1 = -0.2  
x2 = -0.4

Le daremos Un cupo de 300 interacciones al igual que en bisección.

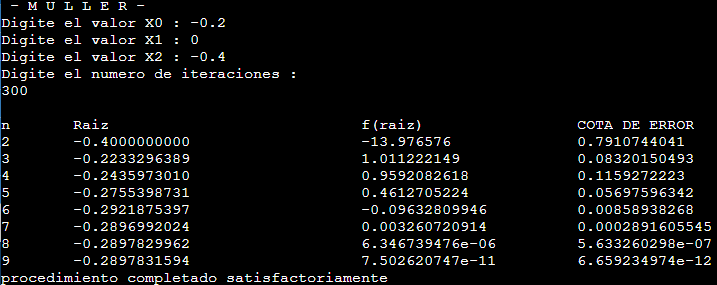


Tenemos el resultado en 8 iteraciones con un error porcentual de 3.064975968e-15

Ahora veamos que sucede si intercambiamos valores de x.

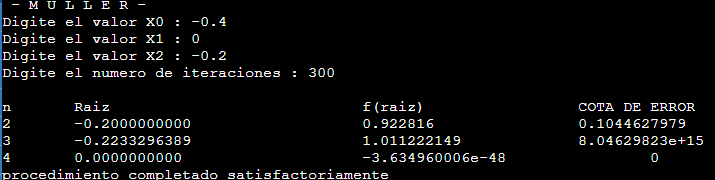
  


Podemos observar que después de gastar las 300 iteraciones que le asignamos este no pudo encontrar la raíz, ahora haremos un cambio más a ver qué sucede.



Podemos encontrar un mejor resultado que el primero con una iteración más, pero con un error porcentual más alto, pasamos de 3.064975968e-15 a 6.659234974e-12

Si cambiamos una última ves los mismos valores de x podemos encontrar el siguiente resultado



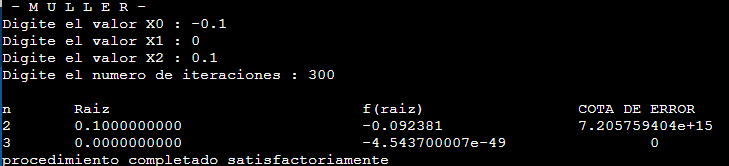
Vemos que el método no fue efectivo con el orden de los mismos valores de x que le dimos, terminando a las 3 iteraciones, pero sin encontrar la raíz.

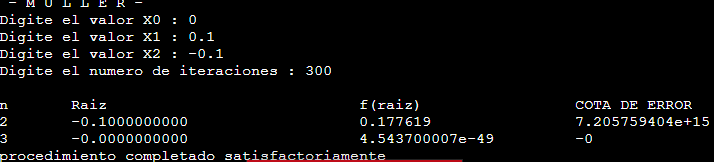
**Parábola 2**

Para la segunda parábola vamos a asignarle los siguientes valores a x:  
x0 = -0.1  
x1 = 0  
x2 = 0.1

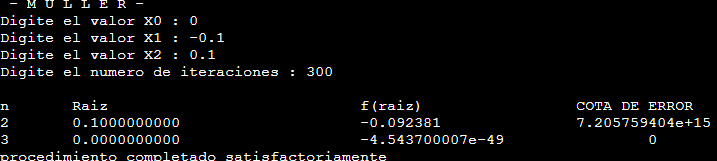
Le daremos Un cupo de 300 interacciones.



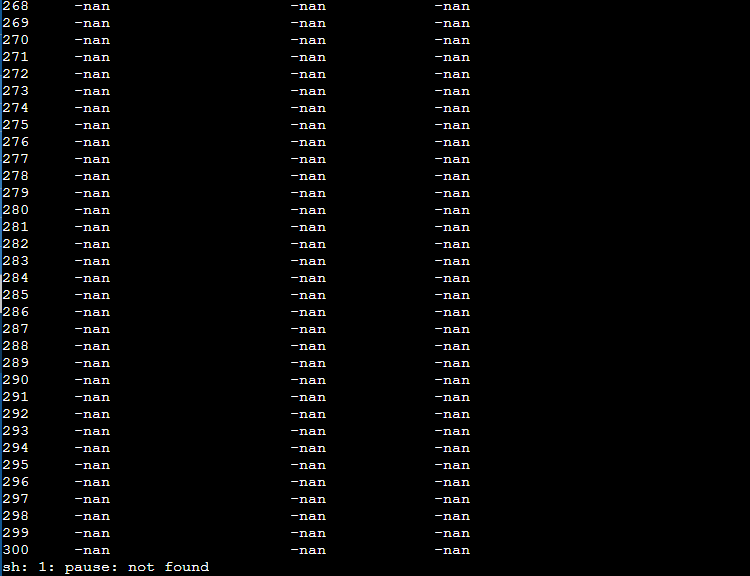
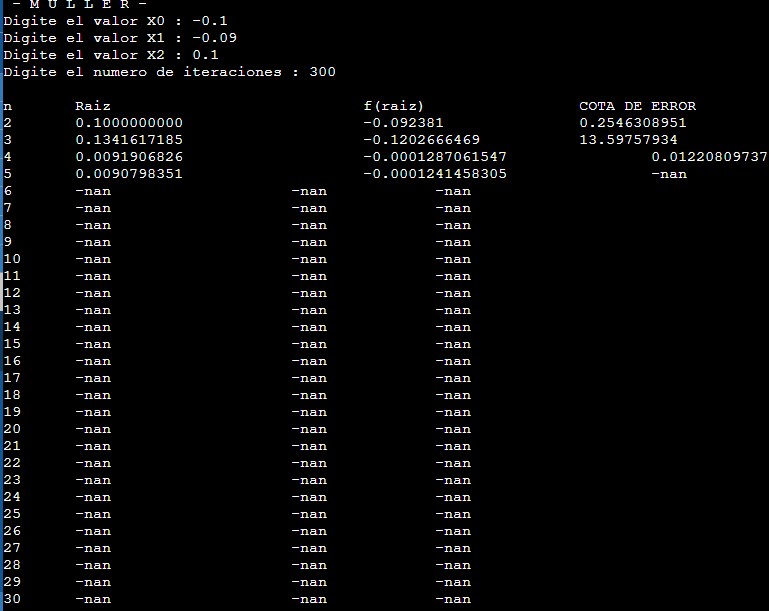


Intentaremos cambiando el orden de las x.

Intentaremos cambiando el orden de las x nuevamente

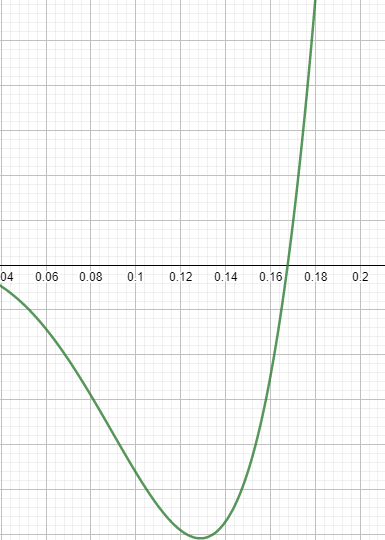


Intentemos cambiar los valores de x:



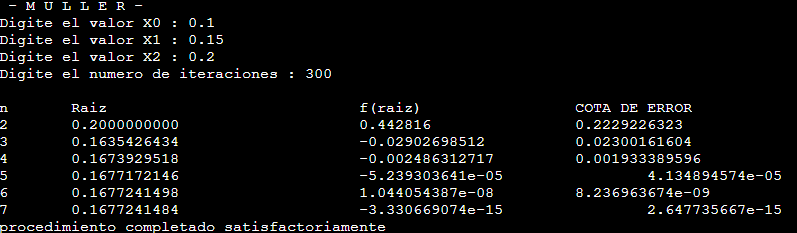
Fallo.

**Parábola 3**

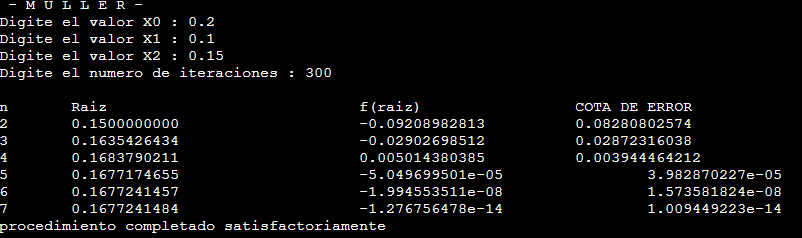


Para la tercera parábola vamos a asignarle los siguientes valores a x:  
x0 = 0.1  
x1 = 0.15  
x2 = 0.2

Le daremos Un cupo de 300 interacciones.

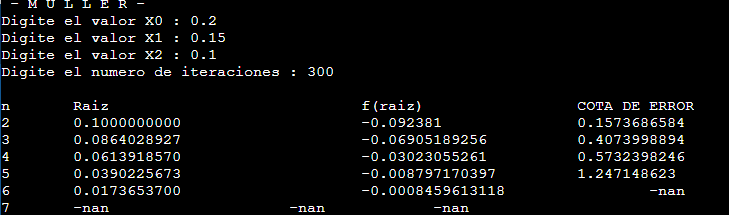


Podemos notar que el método funciono con 6 iteraciones con un error porcentual de 2.647735667e-15, ahora cambiaremos el orden de las x a ver qué sucede.

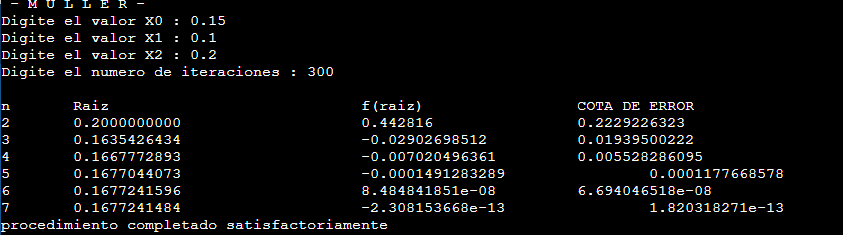


También se da en 6 iteraciones pero con un error un poco más grande, 1.009449223e-14

Podemos intentar cambiarle los valores de nuevo:

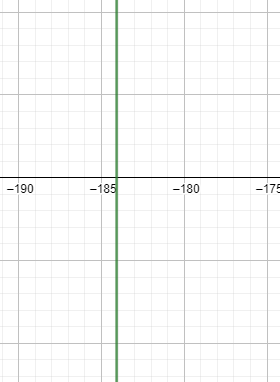


En este orden no funciono, intentemos una última vez.



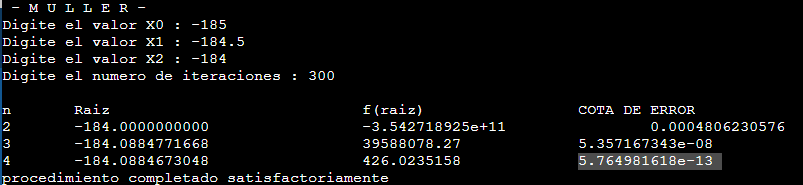
Tenemos que de esta forma hallamos un error porcentual de 1.820318271e-13 un poco mayor que los dos anteriores

**Parábola 4**



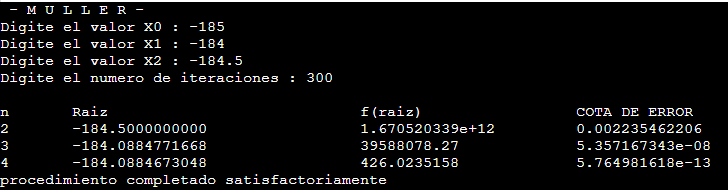
Para la cuarta parábola vamos a asignarle los siguientes valores a x:  
x0 = -185  
x1 = -184.5  
x2 = -184

Le daremos Un cupo de 300 interacciones.

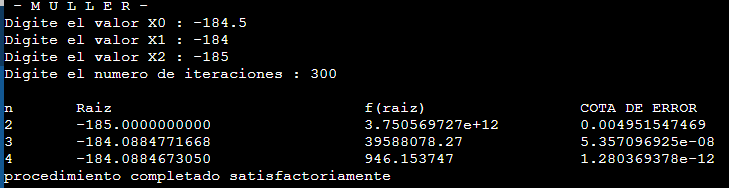


Aquí podemos ver que en solo 3 iteraciones el resultado aproximado de la raíz salió, con un error porcentual de 5.764981618e-13

Ahora intercambiemos los valores haber que sucede.



Tenemos la misma cantidad de iteraciones con un valor porcentual similar, cambiemos el orden de los valores una vez más.

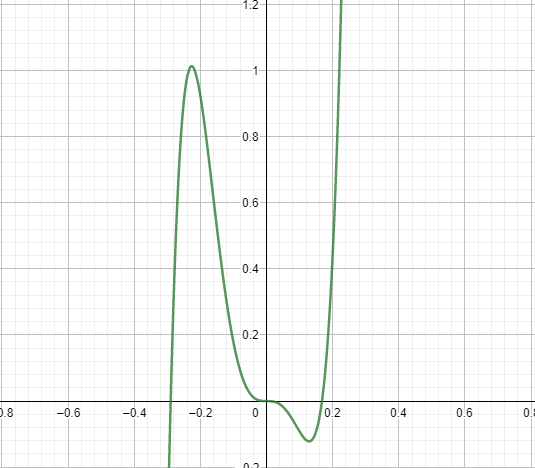


Tenemos después de 3 iteraciones un error porcentual más alto que los anteriores. 1.280369378e-12.

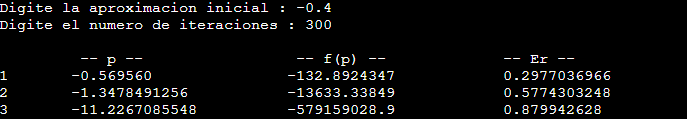
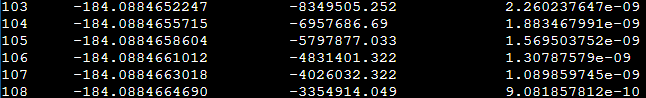
**Conclusión:** para el método de Müller podemos decir que funciono con una cantidad de iteraciones en su mayoría menores a 10, a excepción de la segunda parábola, la cual nos deja en duda la efectividad del método de Müller, vemos como al ejecutarse produce pocas iteraciones y menor error porcentual.

* **NEWTON**

**Secante 1**

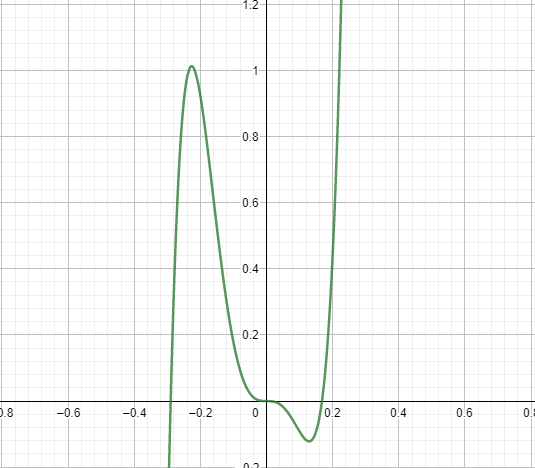


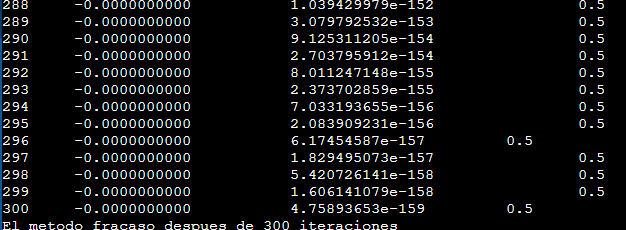
Empezaremos por intentar encontrar la primera raíz aproximada, usaremos las siguiente aproximación x = -0.4, con un cupo de 300 iteraciones.

Después de 108 iteraciones vemos que no fue posible encontrar la raíz 1 del polinomio, sin embargo se encontró la cuarta raíz con un error porcentual de 9.081857812e-10 pese a que no estábamos buscando dicha raíz.

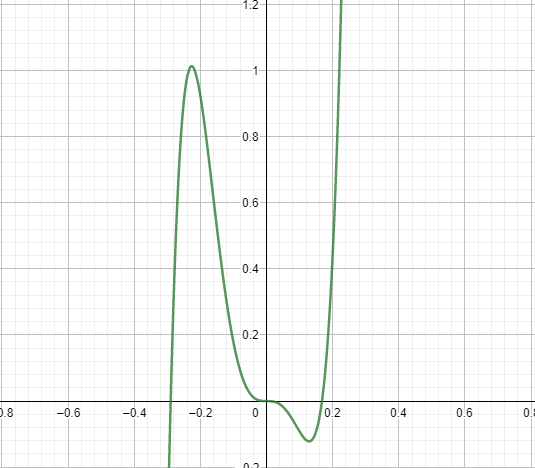
Intenten taremos con más valores para ver si podemos encontrar la raíz 1 usaremos x = -0.2





Notamos que el método fracaso después de 300 iteraciones. Lo mismo sucede con otros valores.

**Secante 2**



Usaremos los valores para x = -0.1 x = -0.01 x = 0.09 x = 0.15; respectivamente



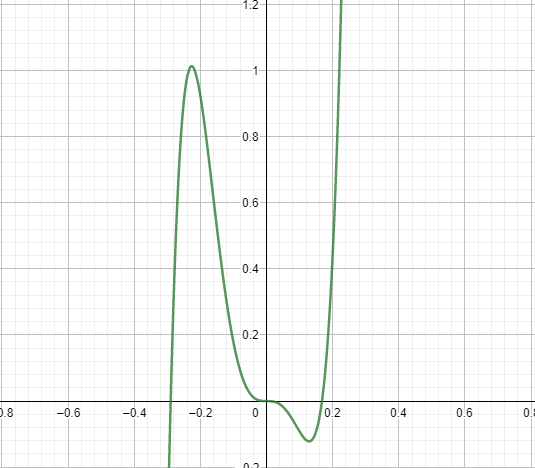




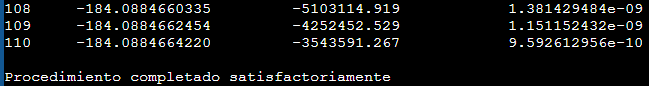


Notemos que en todos los 4 valores gastamos las 300 iteraciones y aun así no podemos encontrar la raíz.

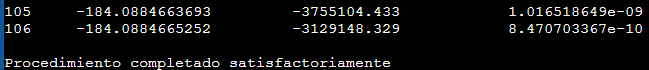
**Secante 3**



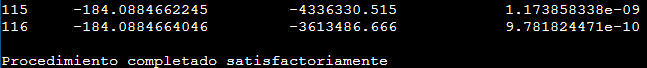
Usaremos los valores para x = 0.2 x = 0.25 x = 0.17 x = 0.4; respectivamente



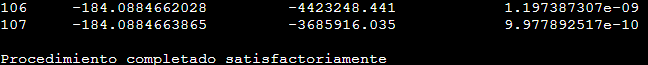
Para x = 0.2 no es posible encuentra la raíz 3 pero encuentra la cuarta raíz después de 110 iteraciones, con error porcentual de 9.592622956 e-10



Para x = 0.25 no es posible encuentra la raíz 3 pero también encuentra la cuarta raíz con erro porcentual de 8.470703367e-10 después de 106 iteraciones, un poco menos que el anterior aun así no es la raíz deseada

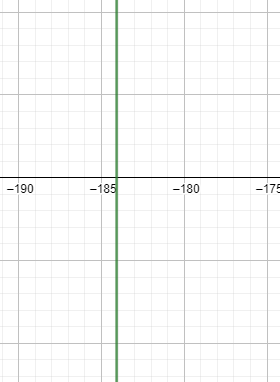


Con x = 0.17 sucede lo mismo, no se encuentra la raíz 3 pero si la raíz 4 con 116 iteraciones y un error porcentual de 9.781824471e-10



Finalmente comprobamos que no es posible encontrar la raíz tres con el método de newton.

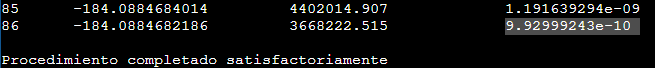
**Secante 4**



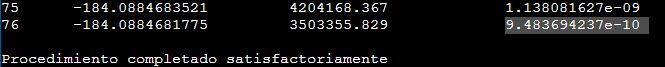
Para esta última raíz real usaremos los siguientes valores para x:

x = -190  
x = -185  
x = -184  
x = -183.5

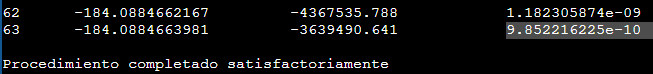
Respectivamente con un cupo de 300 iteraciones



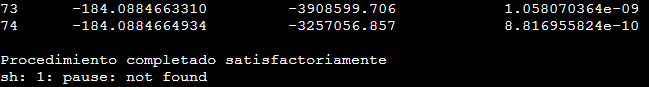
Con x = -190 tenemos un error porcentual de 9.92999243e-10 al encontrar la aproximación de la raíz tras 86 iteraciones



Con x = -185 tenemos un error porcentual 9.483694237e-10 y 76 iteraciones



Con x = -184 tenemos 63 iteraciones, un poco menos que la anterior, esto con un error porcentual de 9.852216225e-10

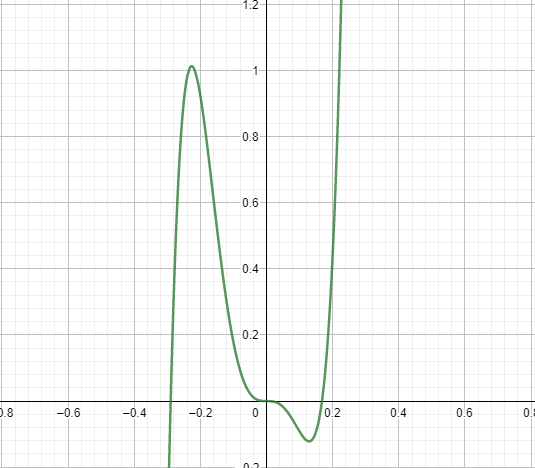


Por ultimo tenemos 74 iteraciones antes de encontrar la aproximación de la raíz 4, esto con un error porcentual de 8.816955824e-10

**Conclusión:** podemos observar que el método de newton resulto muy poco efectivo con los datos arrojados, no fue posible encontrar las tres primeras raíces, únicamente encontró la raíz 4, sin embargo tardo mucho en encontrar, cerca de 80 iteraciones por cada valor de x asignado, lo cual es poco eficaz.

Buscaremos si este polinomio tiene raíz de multiplicidad superior a uno

Evaluamos la segunda raíz, (0) de nuestro polinomio en nuestra ecuación, y derivamos hasta encontrar un numero diferente de 0



***f (x)*** *= 19x^6 + 3500x^5 + 426x^4 − 170x^3 = 0*

***f `(x)*** *= 114x^5 + 17500x^4 +1704x^3 −510x^2 = 0*

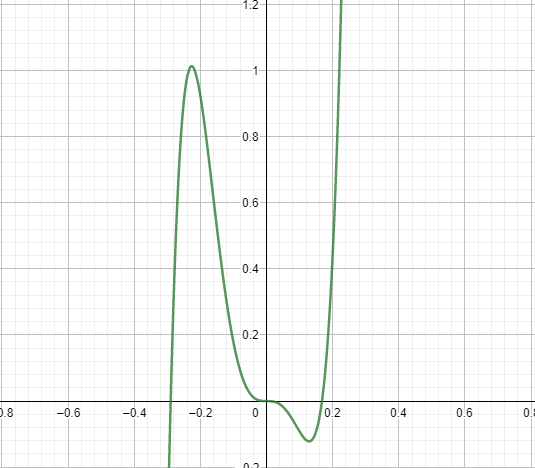
***f ``(x)*** *= 570x^4 + 70000x^3 + 5112x^2 – 1020x = 0*

***f ```(x)*** *= 2280x^3 + 210000x^2 + 1024 – 1020 = – 1020*

Podemos observar que el polinomio tiene raíz de multiplicidad superior a 1, asi que procederemos a usar newton generalizado

* **NEWTON GENERALIZADO**

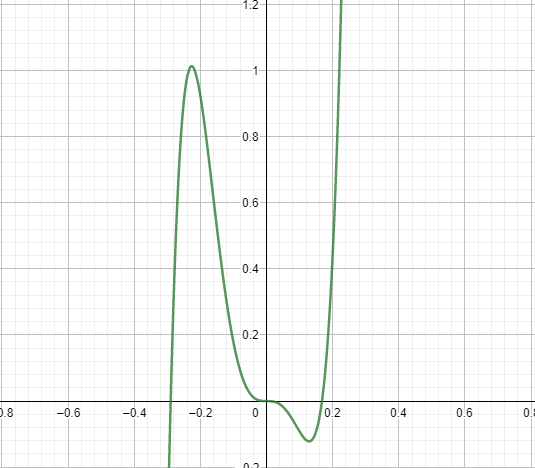
**Raíz 1**

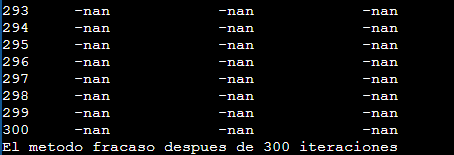


Usaremos las mismas raíces que para newton a ver cómo es su comportamiento, empezaremos x = -0.4

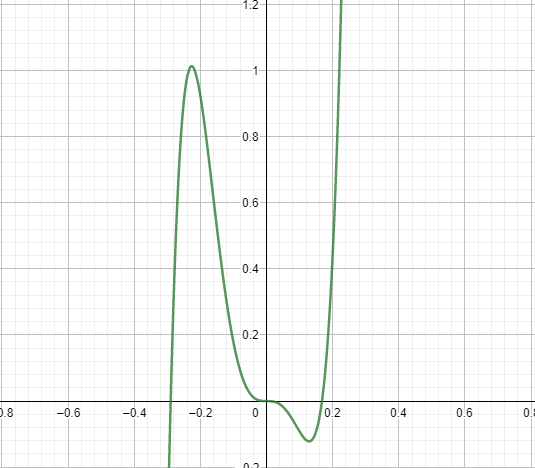


Este método no funciono con este valor, usaremos z = -0.2





Nuevamente ha fallado

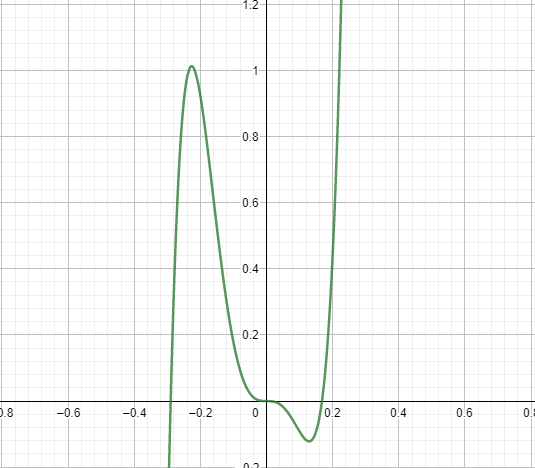
**Raíz 2**

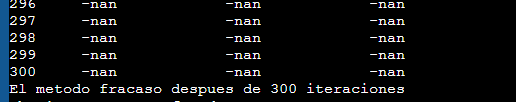
Usaremos x = -0.1



Este fallo nuevamente.

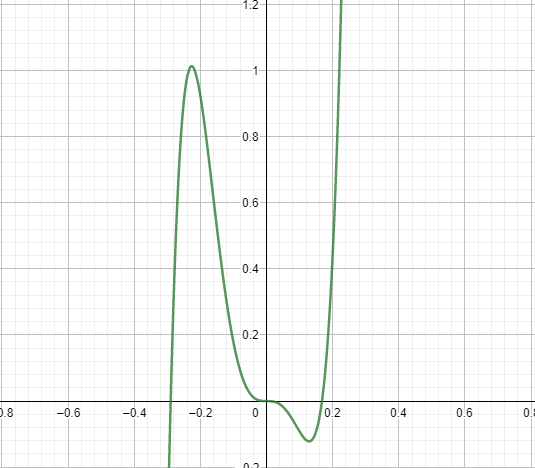
Usaremos ahora z = 0.05 para la misma raíz



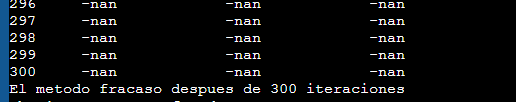


El método fracaso nuevamente

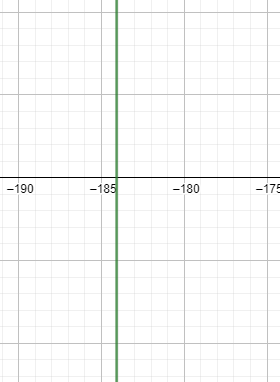
**Raíz 3**



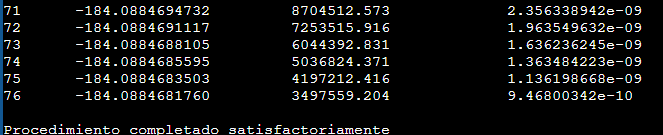
Usaremos x = 0.2



Vemos que el método ha vuelto a fracasar

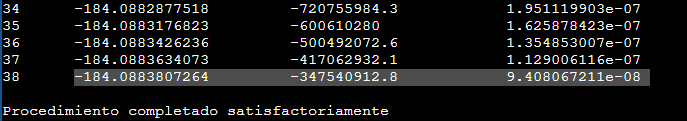


Para esta última raíz usaremos x = -185



En esta última raíz el método si funciono encontrándomela después de 76 iteraciones con error porcentual del 9.46800342e-10

Ahora probaremos con -184, un valor un poco más cercano a la raíz aproximada a ver qué sucede



Podemos ver que produce un error porcentual 9.853795651e-10 un poco más grande que el pasado pero no entrega la raíz con menos iteraciones

**Conclusión:** podemos observar que el método de newton generalizado también fue poco eficaz para este polinomio, no pudo encontrar las tres primeras raíz, aun así la cuarta la encontró en menos iteraciones que en newton normal.

**Tabla De Resultados**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Método | Valor(es)  Inicial(es) | Raíz | Iteraciones | Cota de Error |
| Regla Falsa | xi = -0.3  xs = -0.2 | 0.2897831494 | 10 | 3.798135461e-10 |
|  | xi = -0.02  xs = 0.02 | 0.0006356072 | 300 | 0.001001598284 |
|  | xi = 0.15  xs = 0.18 | 0.1677241483 | 26 | 8.605294822e-10 |
|  | xi = -185  xs = -184 | -184.0884673040 | 4 | 1.663600159e-10 |
| Newton | xi = -0.4 | -184.0884664690 | 108 | 9.081857812e-10 |
|  | xi = -0.1 | 0.000000000 | 300 | 0.5 |
|  | xi = 0.25 | -184.0884665252 | 106 | 8.470703367e-10 |
|  | xi = -185 | -184.0884681775 | 76 | 9.483694237e-10 |
| Müller | x0 = 0  x1 = -0.2  x2 = -0.4 | -0.2897831594 | 9 | 3.064975968e-15 |
|  | x0 = -0.1  x1 = 0  x2 = 0.1 | 0 | 2 | 0 |
|  | x0 = 0.1  x1 = 0.15  x2 = 0.2 | 0.1677241484 | 6 | 2.647735667e-15 |
|  | x0 = -184.5  x1 = -184  x2 = -185 | -184.0884673050 | 3 | 1.280369378e-12. |
| Newton | xi = -0.4 | nan | nan | nan |
|  | xi = -0.1 | nan | nan | nan |
|  | xi = 0.25 | nan | nan | nan |
|  | xi = -185 | -184.0884681760 | 76 | 9.46800342e-10 |

**Análisis:** para este segundo polinomio podemos ver una que no todos los métodos fueron efectivos.

Por ejemplo regla falsa vemos que de las 4 raíces reales del polinomio solo pudo encontrar 3, con una cantidad de iteraciones de 4 a 26, Newton y Newton generalizado no funcionaron en este polinomio, generaron solo una raíz. Y fracaso en las demás, ambos con bastantes iteraciones, cercanas a 100. Y por último está el método de Müller, este funciono para encontrar todas las raíces, con una cantidad de iteraciones menor a los anteriores, menor a 10

Ahora si observamos los valores de la raíces, son bastante similares, (de las raíces que se pudieron encontrar con cada método), comparando los errores vemos que el que genera las raíces con mayor error porcentual es newton, le sigue newton generalizado, después el método de la regla falsa, con diferencias no muy altas, ahora si comparamos con Müller vemos ya una diferencia un poco mayor, siendo este el método que genero las raíces con menor número porcentual de error.

**Conclusión:** para este polinomio vemos un gran desempeño en el método de Müller, a comparación de los otros tres métodos que no pudieron encontrara todas la raíces, también es el más eficaz por la mínima cantidad de iteraciones que genera cada ves que encuentra una raíz, y también por su error porcentual es el que tiene menor erro porcentual

Los métodos funcionan con menos iteraciones dependiendo de que tan cercano este el numero de que se le da de intervalo de la raíz.