# UNIVERSIDAD DEL CAUCA


Universidad
del Cauca

Presentado por:

Presentado a:

**CARLOS ALBERTO ARDILA**

Materia:

**ANALISIS NUMERICO**

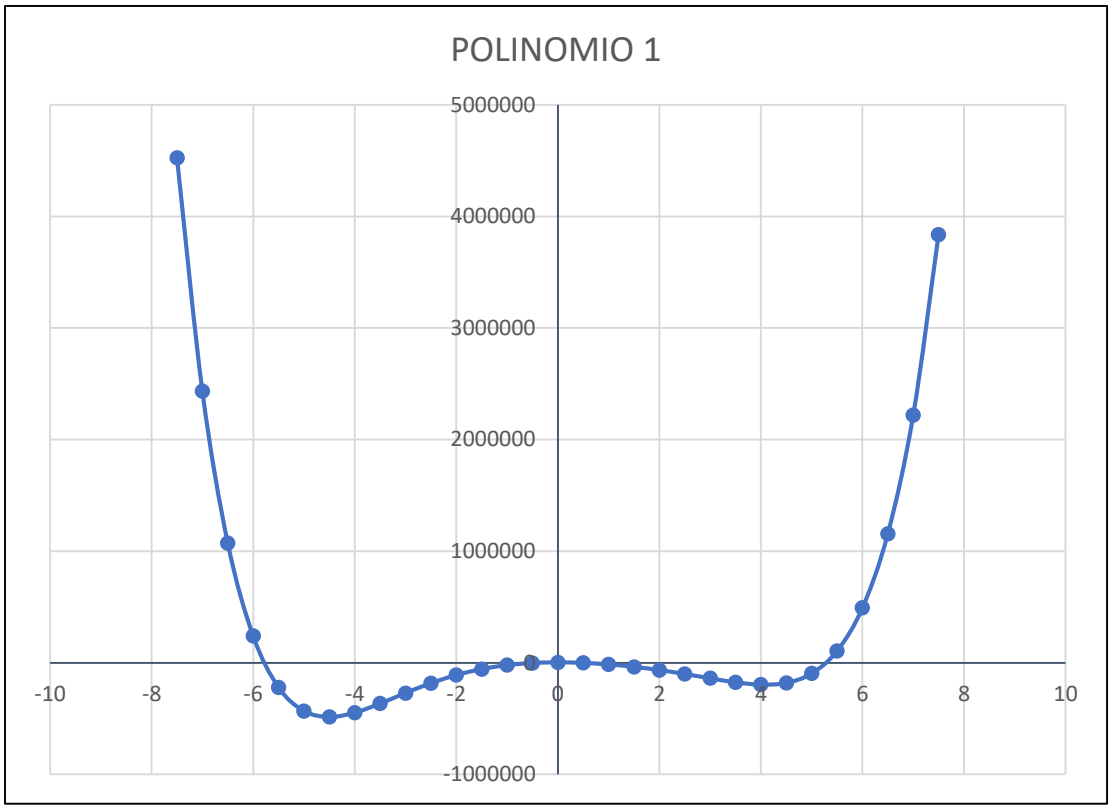
Trabajo:

**PRIMER PARCIAL**

**PRIMER POLINOMIO:**

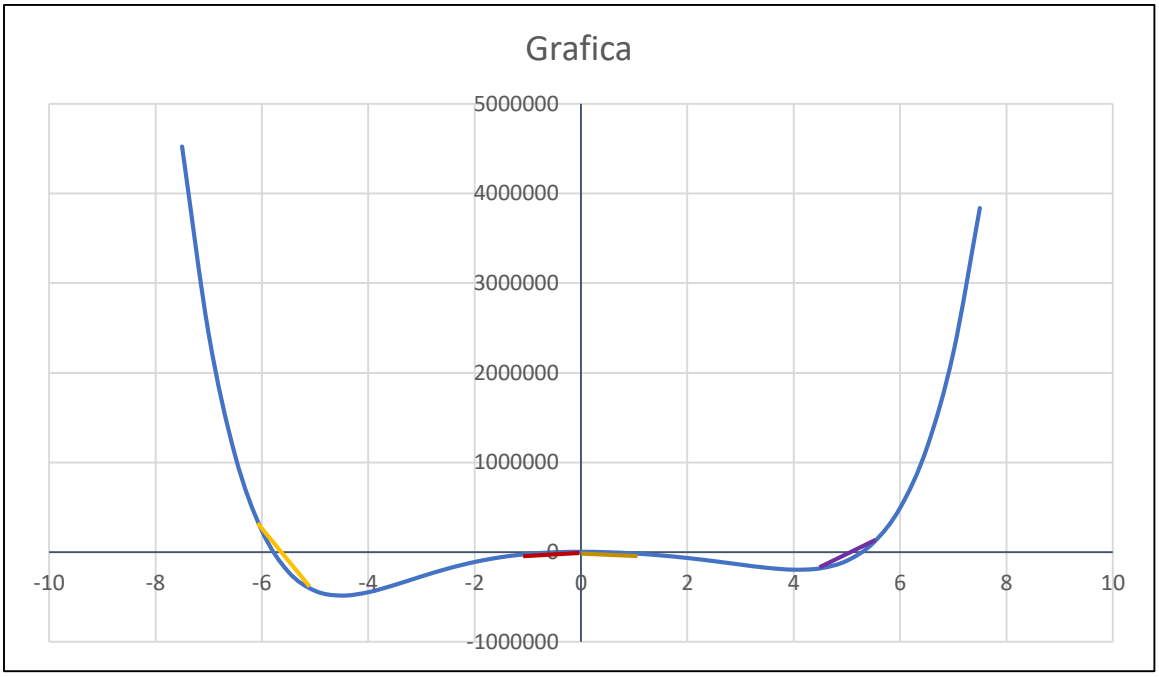
f(x) = _____

**GRÁFICA:**



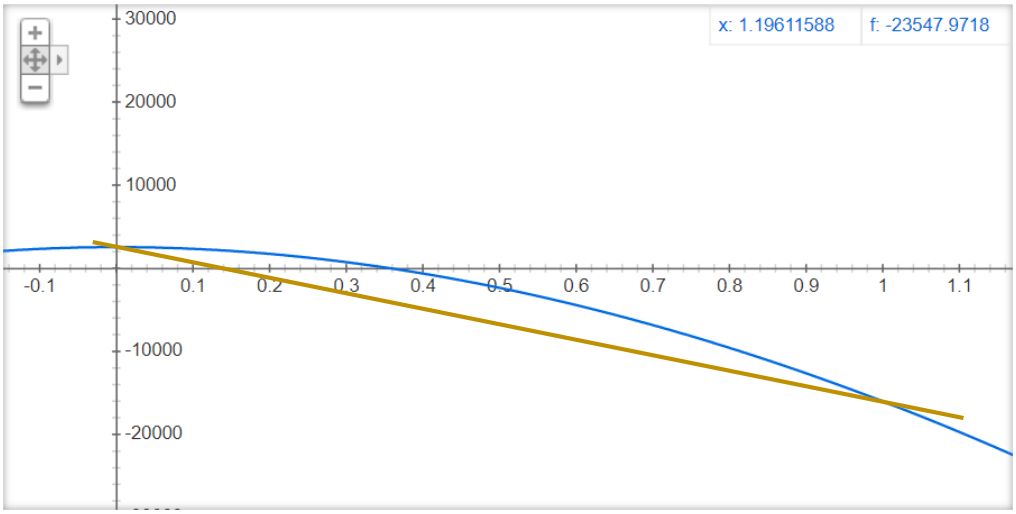

Según el grado del polinomio tenemos 6 raíces, de las cuales observamos que tenemos 4 raíces reales.

**REGLA FALSA:**




Procederemos a encontrar o aproximarnos a dicha raíz usando el método de regla falsa, para este caso usaremos el intervalo (0 , 1), teniendo en cuenta que tenemos 4 raíces reales.

RAIZ 1

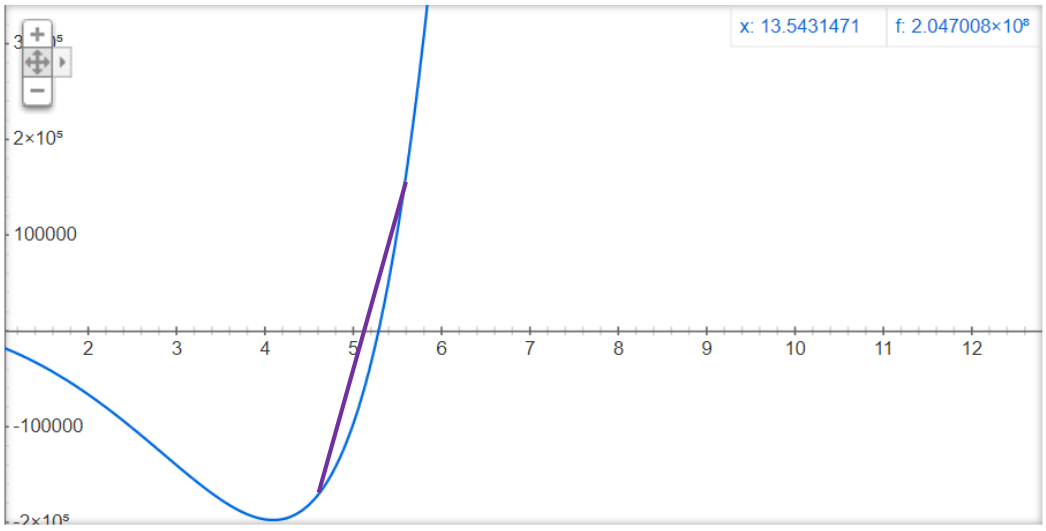


```
 - R E G L A   F A L S A -
Digite el extremo inferior (Xi) : 0
Digite el extremo superior (Xs) : 1
Digite el numero de iteraciones : 100

 XR : 0.1372960749
n       Raiz                    f(Raiz)                 Er
1       0.2398565983            1382.105209             0.4275910026
2       0.3002180924            735.1498126             0.2010588156
3       0.3309162628            353.9878838             0.09276718565
4       0.3453784788            162.3940984             0.04187352974
5       0.3519465407            72.85385974             0.01866210128
6       0.3548797928            32.35731834             0.008265480532
7       0.3561799420            14.30713811             0.003650259442
8       0.3567543041            6.313568819             0.001609965629
9       0.3570076634            2.78367362              0.0007096747349
10      0.3571193510            1.226859128             0.0003127457747
11      0.3571685717            0.5406266778            0.0001378080953
12      0.3571902606            0.2382142893            6.072065947e-005
13      0.3571998171            0.1049600074            2.675400108e-005
14      0.3572040278            0.0462459386            1.178791022e-005
15      0.3572058830            0.02037607681           5.193774096e-006
16      0.3572067004            0.0089777254            2.288381777e-006
17      0.3572070606            0.003955592341          1.008262345e-006
18      0.3572072193            0.001742835909          4.442408156e-007
19      0.3572072892            0.0007678941643         1.957326596e-007
20      0.3572073200            0.0003383344251         8.62398727e-008
21      0.3572073336            0.0001490702541         3.799731435e-008
22      0.3572073396            6.568039871e-005                1.674162837e-008
23      0.3572073422            2.893880376e-005                7.376366278e-009
Procedimiento completado satisfactoriamente
```

En este caso se encuentra la raíz en un total de 23 iteraciones, demora un poco al asignarle esos valores.

Ahora continuaremos con otra raíz, que diremos se “encuentra” en el intervalo de (4,5 y 5,5).

RAIZ 2

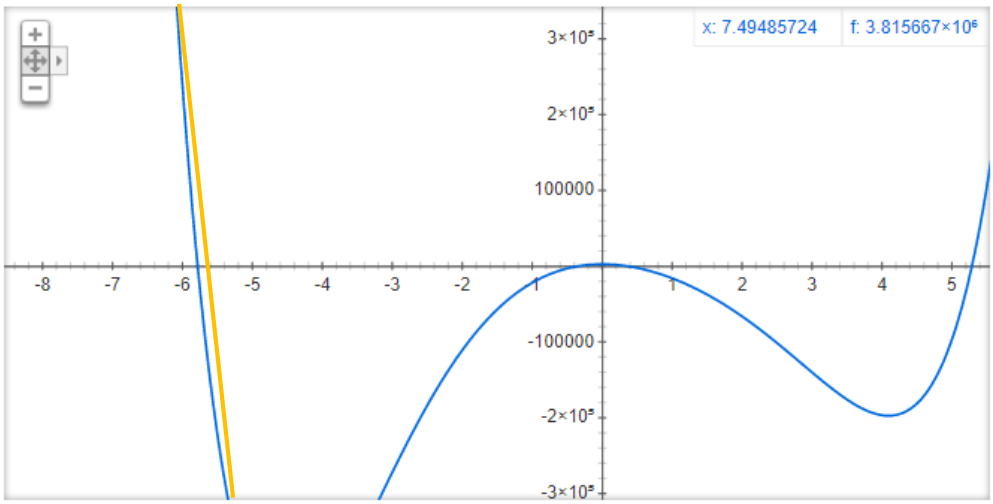


```
 - R E G L A   F A L S A -
Digite el extremo inferior (Xi) : 4.5
Digite el extremo superior (Xs) : 5.5
Digite el numero de iteraciones : 100

 XR : 5.134566738
n       Raiz                    f(Raiz)                 Er
1       5.2633032397            -9364.477427            0.0244592599
2       5.2827790949            -1358.148051            0.003686668487
3       5.2855674571            -193.2117629            0.0005275426324
4       5.2859634003            -27.41078671            7.490465798e-005
5       5.2860195577            -3.88722132             1.062375843e-005
6       5.2860275213            -0.5512300699           1.506534692e-006
7       5.2860286506            -0.0781669401           2.13634006e-007
8       5.2860288107            -0.01108441633          3.029425373e-008
9       5.2860288334            -0.001571818682         4.295858165e-009
Procedimiento completado satisfactoriamente
```

Dándole estos valores, observamos que el método cumple su función satisfactoriamente, encontrando esta raíz más rápido en comparación a los valores anteriores.

Probemos ahora el intervalo (-6,-5) donde tenemos otra posible raíz.

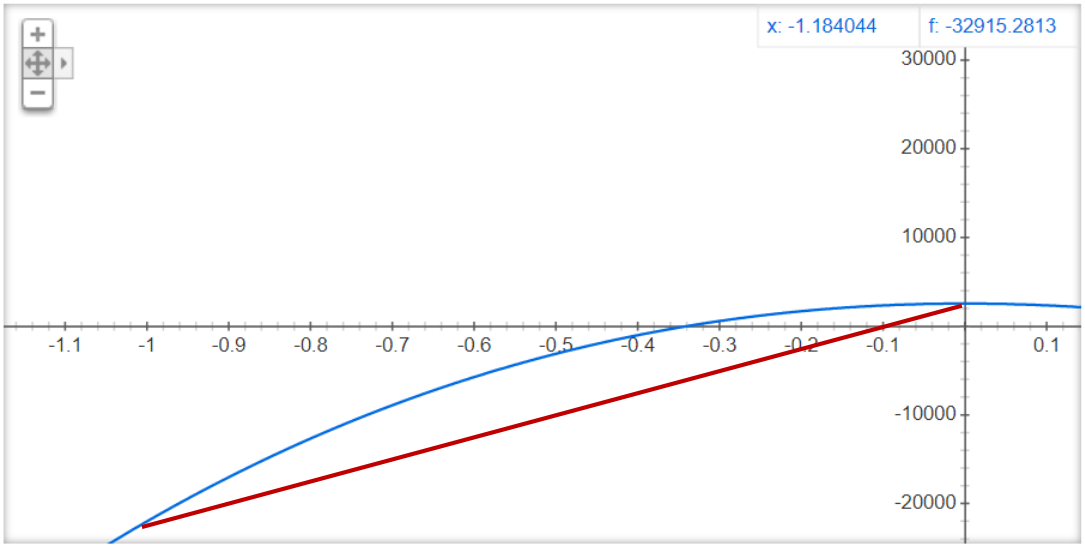
RAIZ 3




```
 - R E G L A   F A L S A -
Digite el extremo inferior (Xi) : -6
Digite el extremo superior (Xs) : -5
Digite el numero de iteraciones : 100

 XR : -5.644392634
n       Raiz                    f(Raiz)                 Er
1       -5.7628186451           -17819.42661            0.02055001526
2       -5.7792848442           -2386.856949            0.002849175901
3       -5.7814686181           -314.9182009            0.000377719592
4       -5.7817563625           -41.46671831            4.976764008e-005
5       -5.7817942446           -5.458672246            6.551952148e-006
6       -5.7817992312           -0.7185537993           8.624774908e-007
7       -5.7817998876           -0.09458659143          1.135321037e-007
8       -5.7817999741           -0.01245086723          1.494475471e-008
9       -5.7817999854           -0.001638964798         1.967246538e-009
Procedimiento completado satisfactoriamente
```

Efectivamente, se encuentra la iteración 9.

RAIZ 4

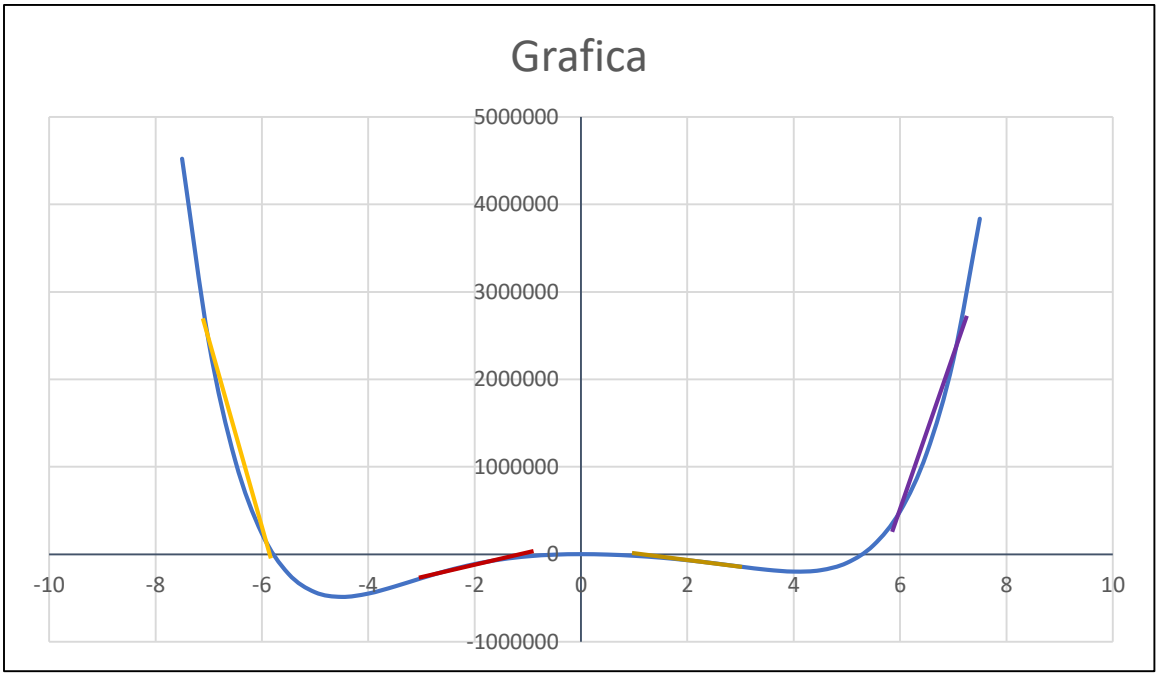



```
 - R E G L A   F A L S A -
Digite el extremo inferior (Xi) : -1
Digite el extremo superior (Xs) : 0
Digite el numero de iteraciones : 100

 XR : -0.1037808799
n       Raiz                    f(Raiz)                 Er
1       -0.1892123851           1776.622252             0.4515111692
2       -0.2497421012           1190.109562             0.2423688909
3       -0.2882102794           728.1140132             0.13347261
4       -0.3109920049           421.1882173             0.07325501976
5       -0.3239230991           235.8197323             0.03992025949
6       -0.3310863989           129.6331188             0.02163574187
7       -0.3350011113           70.54418224             0.01168566965
8       -0.3371246298           38.17807043             0.006298912372
9       -0.3382718757           20.60018009             0.00339149077
10      -0.3388903298           11.09758885             0.001824938836
11      -0.3392233314           5.973231853             0.0009816588335
12      -0.3394025195           3.213566125             0.0005279516925
13      -0.3394989077           1.728446564             0.0002839130549
14      -0.3395507469           0.9295354653            0.0001526699924
15      -0.3395786242           0.4998554102            8.209367632e-005
16      -0.3395936148           0.2687855088            4.414272182e-005
17      -0.3396016755           0.1445300571            2.373585893e-005
18      -0.3396060099           0.07771513186           1.276288578e-005
19      -0.3396083405           0.04178788226           6.862648945e-006
20      -0.3396095936           0.02246951597           3.690065974e-006
21      -0.3396102675           0.01208192982           1.984157673e-006
22      -0.3396106298           0.006496485743          1.066886134e-006
23      -0.3396108246           0.003493175859          5.736670085e-007
24      -0.3396109294           0.001878288467          3.084619729e-007
25      -0.3396109857           0.001009959758          1.658606507e-007
26      -0.3396110160           0.0005430574965         8.918361743e-008
27      -0.3396110323           0.0002920031511         4.795421611e-008
28      -0.3396110411           0.0001570107015         2.578508118e-008
29      -0.3396110458           8.442497926e-005                1.386469155e-008
30      -0.3396110483           4.539548578e-005                7.455073273e-009
Procedimiento completado satisfactoriamente
```

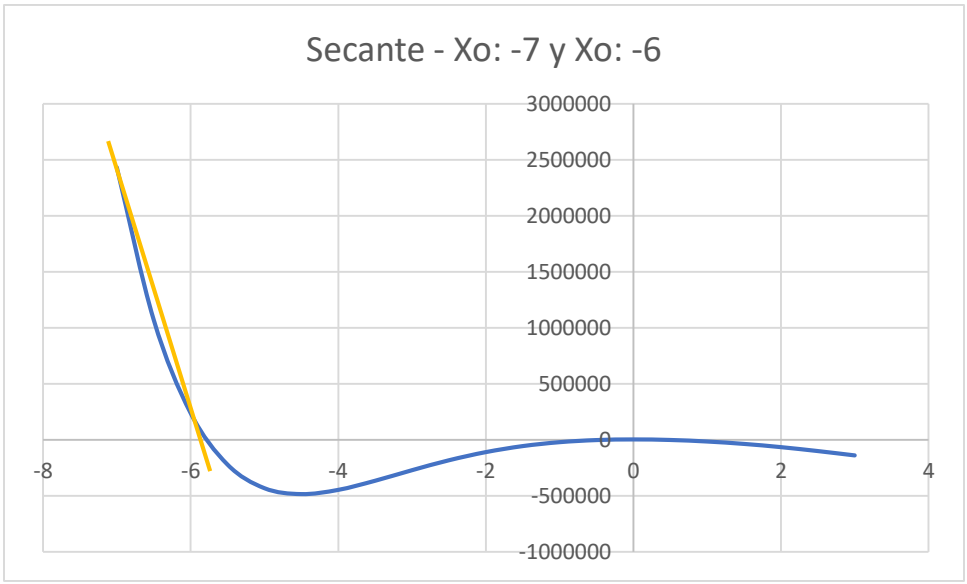
En esta ultima prueba, buscando la raíz que se encuentra entre (-1 , 0), vemos que el método es mas demorado, encontrando la raíz en la iteración 30.

***Conclusión:*** Se tomaron estos valores anteriormente, estando en el intervalo antes y después de la raíz para así saber cuál es la posible raíz. Al ser una raíz muy pequeña para determinar, este método demora un poco más por esa complejidad.

**SECANTE**



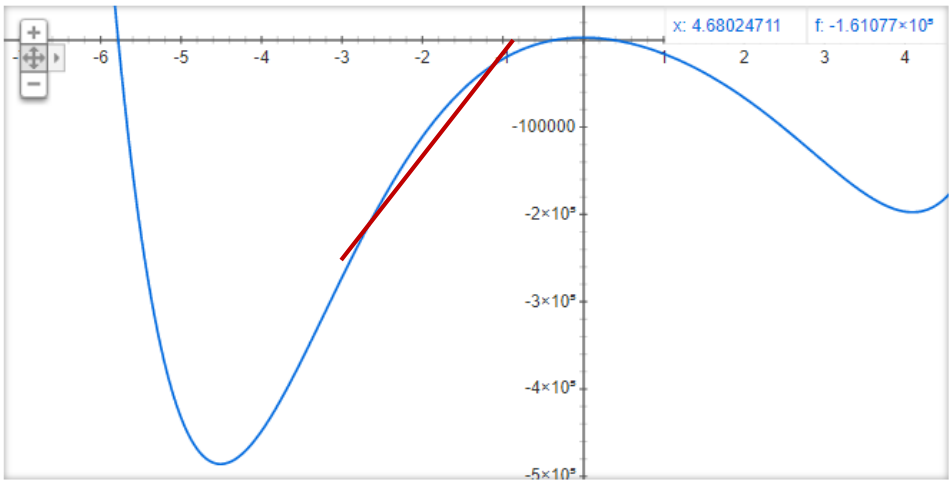

APROXIMACIÓN 1



```
 - S E C A N T E -
Digite Xo : -7
Digite X1 : -6
Digite el numero de iteraciones : 100

 XR : -5.891125696
n       Raiz                    f(raiz)                 Er
1       -5.7956598353                   13294.84263             0.01647195725
2       -5.7827477210                   901.4395615             0.002232868334
3       -5.7818085528                   8.142194855             0.0001624350301
4       -5.7817999925                   0.005059350191          1.480559189e-006
5       -5.7817999872                   2.845621339e-008                9.205533531e-010
Procedimiento completado satisfactoriamente
```

En esta ocasión tenemos un buen resultado, se encuentra la raíz en la iteración 5, con un porcentaje de error de 9,205533531e-010.

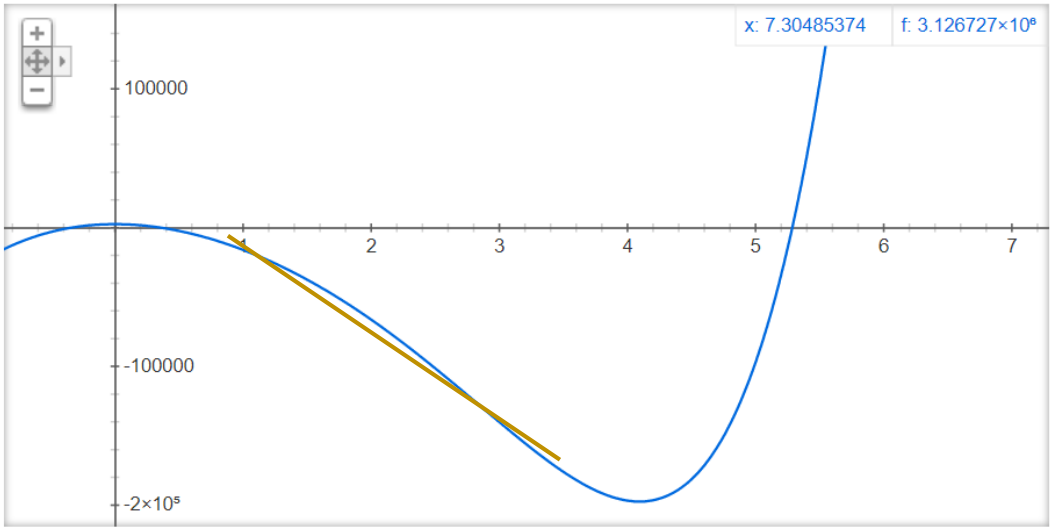
APROXIMACIÓN 2




```
 - S E C A N T E -
Digite Xo : -3
Digite X1 : -1
Digite el numero de iteraciones : 100

 XR : -0.8240164308
n       Raiz                    f(raiz)                 Er
1       -0.5359874670                   -4001.911357            0.5373800349
2       -0.4167539404                   -1336.842676            0.2861005381
3       -0.3569444249                   -274.5887255            0.1675597415
4       -0.3414838858                   -28.95679743            0.04527457857
5       -0.3396612901                   -0.7745224104           0.005365921157
6       -0.3396112005                   -0.002300224123         0.0001474912569
7       -0.3396110513                   -1.837586352e-007               4.393335263e-007
8       -0.3396110512                   2.435829316e-013                3.510001677e-011
Procedimiento completado satisfactoriamente
```

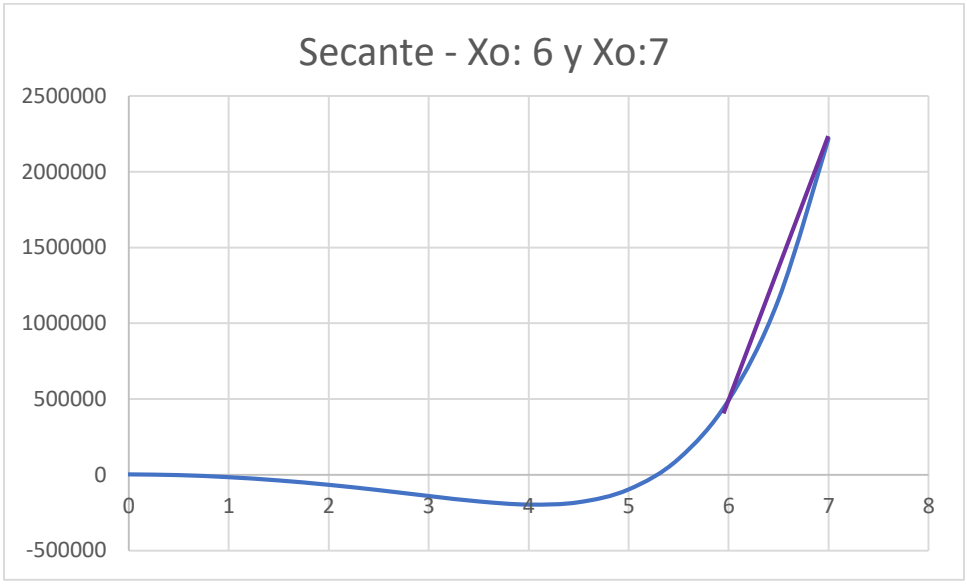
Para estas aproximaciones notamos un aumento en las iteraciones, pero esto va relacionado con los valores escogidos.

APROXIMACIÓN 3




```
 - S E C A N T E -
Digite Xo : 3
Digite X1 : 1
Digite el numero de iteraciones : 100

 XR : 0.7418185334
n       Raiz                    f(raiz)                 Er
1       0.4876929834                    -2124.889876            0.521076904
2       0.3949621222                    -552.2951304            0.2347841883
3       0.3623950522                    -72.8282822             0.0898662103
4       0.3574482998                    -3.361001396            0.01383907088
5       0.3572089635                    -0.02257950571          0.000670017606
6       0.3572073448                    -7.099837682e-006               4.531702529e-006
7       0.3572073443                    -1.523203785e-011               1.425384232e-009
Procedimiento completado satisfactoriamente
```

APROXIMACIÓN 4

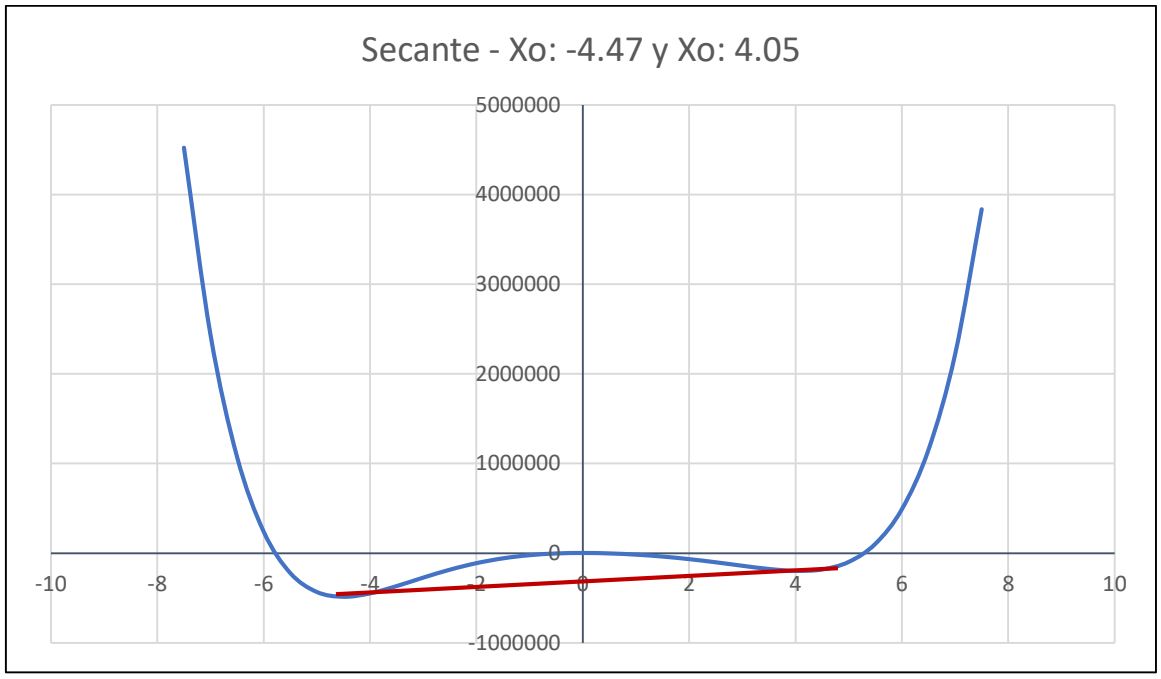



```
 - S E C A N T E -
Digite Xo : 7
Digite X1 : 6
Digite el numero de iteraciones : 100

 XR : 5.715584201
n       Raiz                    f(raiz)                 Er
1       5.4355010844                    69702.38531             0.05152848145
2       5.3232925779                    16035.83051             0.02107877876
3       5.2897641243                    1568.979581             0.006338364585
4       5.2861278486                    41.47973012             0.0006878902373
5       5.2860291044                    0.1119296687            1.868022075e-005
6       5.2860288372                    8.018976587e-006                5.054344213e-008
7       5.2860288372                    -1.601279109e-010               3.621416014e-012
Procedimiento completado satisfactoriamente
```

Haciendo una prueba en el método secante, ingresando Xo : -4.47 y X1 : 4.05 como aproximaciones iniciales.

APROXIMACIÓN 5

---




Tendremos los siguientes resultados:

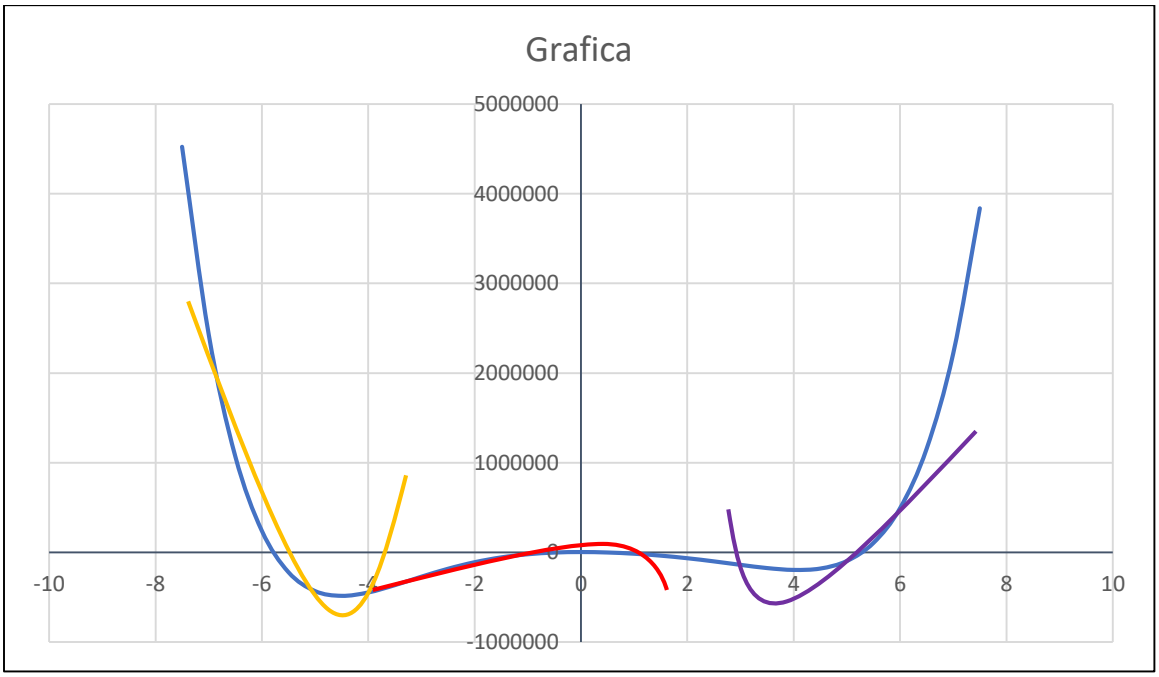
```
- S E C A N T E -
Digite Xo : -4.47
Digite X1 : 4.05
Digite el numero de iteraciones : 300

 XR : 9.879216302
n       Raiz                    f(raiz)                 Er
1       4.0927594057                    -197419.3347            1.413827768
2       4.1352370530                    -197276.8759            0.01027211905
3       62.9582573775                   2.476089265e+012                0.9343177968
4       4.1352417396                    -197276.8435            14.224807
5       4.1352464262                    -197276.8111            1.133328132e-006
6       32.7073327761                   4.699498534e+010                0.8735682162
7       4.1353663663                    -197275.9817            6.909174153
8       4.1354863055                    -197275.1499            2.90024319e-005
9       32.5793611618                   4.588806846e+010                0.8730642297
10      4.1356085866                    -197274.2993            6.877767076
180     8.4007833978                    8810620.695             0.4811124215
181     4.4448221179                    -185731.7443            0.8900156575
182     4.5264938392                    -178981.5698            0.01804304263
183     6.6920279159                    1508495.349             0.32359908
184     4.7561803519                    -149421.0586            0.4070172745
185     4.9306501810                    -114485.4898            0.03538475104
186     5.5023959568                    105793.5131             0.1039085119
187     5.2278033111                    -23381.48583            0.05252543552
188     5.2775063040                    -3548.204826            0.009417893611
189     5.2863982467                    154.7903447             0.001682041787
190     5.2860265512                    -0.957637346            7.031662856e-005
191     5.2860288366                    -0.0002559521203                4.323510299e-007
192     5.2860288372                    5.83952442e-010         1.155874063e-010
Procedimiento completado satisfactoriamente
```

Donde podemos observar la cantidad iteraciones que tuvo que usar el método para llegar a la raíz, cambiando las aproximaciones a Xo: -4.47 y X1: 4.05, tendremos reducción en las iteraciones, ya que teniendo en cuenta la sugerencia, X1 debe ser el valor más cercano a la raíz. Tenemos el siguiente resultado:

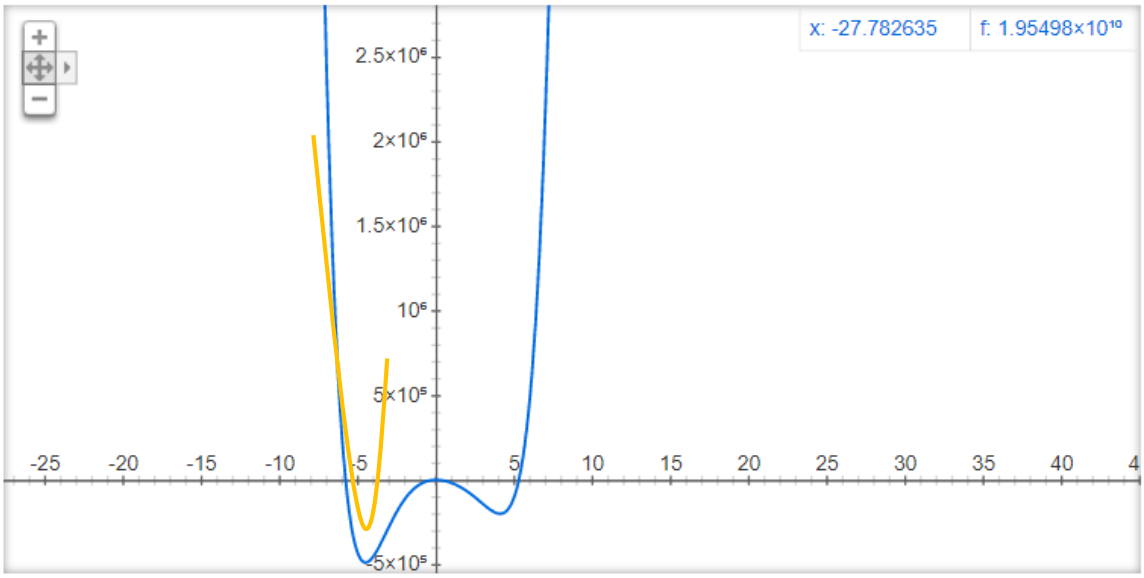
```
 - S E C A N T E -
Digite Xo : 4.05
Digite X1 : -4.47
Digite el numero de iteraciones : 500

 XR : 9.879216302
n       Raiz                    f(raiz)                 Er
1       -4.2136482844                   -471939.325             3.344575445
2       -3.9688362414                   -443944.4158            0.06168358381
3       -0.0865968539                   2390.494232             44.83118283
4       -0.1073894709                   2303.93743              0.193618768
5       -0.6608397777                   -7611.09963             0.8374954497
6       -0.2359936173                   1338.280201             1.800244283
7       -0.2995246346                   578.804057              0.2121061508
8       -0.3479422325                   -130.1335205            0.1391541279
9       -0.3390546341                   8.569890276             0.02621287985
10      -0.3396037608                   0.1123850426            0.001616962885
11      -0.3396110577                   -9.952551174e-005               2.148606866e-005
12      -0.3396110512                   1.153774853e-009                1.901071456e-008
13      -0.3396110512                   2.435829316e-013                2.203374466e-013
Procedimiento completado satisfactoriamente
```

**Conclusión:** Para este método utilizamos pruebas donde las aproximaciones iniciales estuvieran antes o después de la raíz. Como vemos la raíz en estos casos se encuentra más rápido teniendo en cuenta la sugerencia (Xo deber ser el valor más lejano a la raíz y X1 debe ser el valor más cercano a la raíz). Al no tener que buscar entre los intervalos como en el método anterior se le hace más fácil al programa lograr encontrar raíz ya que es un método iterativo.

**MÜLLER**



PARABOLA 1



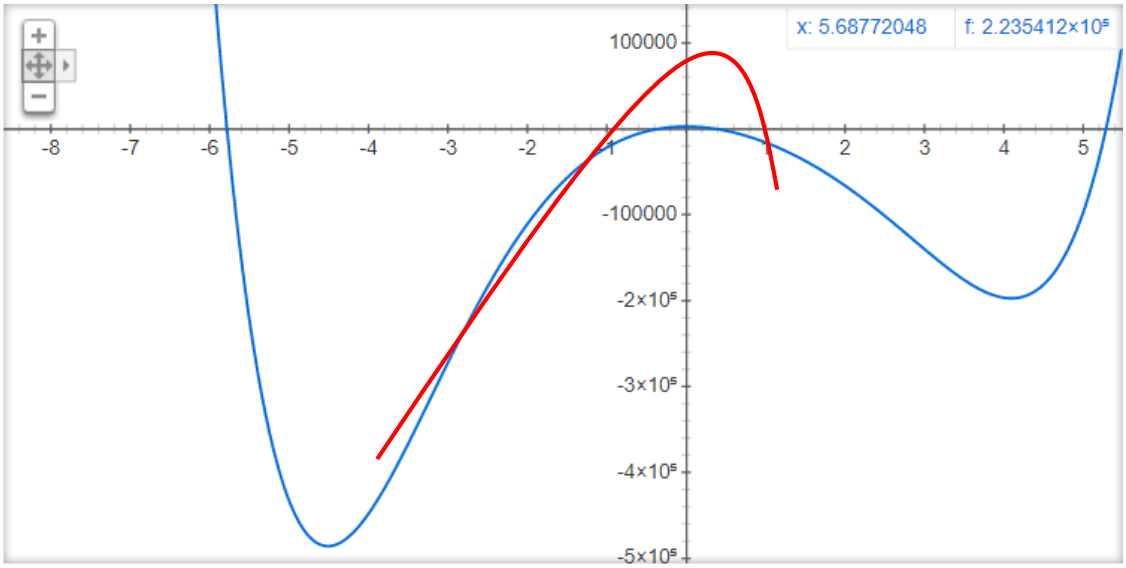

```
 - M U L L E R -
Digite el valor X0 : -7
Digite el valor X1 : -5
Digite el valor X2 : -4
Digite el numero de iteraciones : 100

n       Raiz                            f(raiz)                 COTA DE ERROR
2       -4.0000000000                   -448138         0.1777338115
3       -3.3963532004                   -347339.9608            0.4148283473
4       -2.4005408196                   -167915.2123            0.557208341
5       -1.5415668902                   -60905.29279            1.137696714
6       -0.7211345185                   -9667.836422            nan
7       nan                     nan             nan
8       nan                     nan             nan
9       nan                     nan             nan
10      nan                     nan             nan
```

Viendo esto diríamos que el método no funcionó para estos valores, pero al cambiarle el orden, vemos que funciona:

```
 - M U L L E R -
Digite el valor X0 : -4
Digite el valor X1 : -5
Digite el valor X2 : -7
Digite el numero de iteraciones : 100

n       Raiz                            f(raiz)                 COTA DE ERROR
2       -7.0000000000                   2432705         0.2564534057
3       -5.5712372365                   -174001.9188            0.03125531943
4       -5.7509861456                   -28703.26629            0.005075112706
5       -5.7803219308                   -1403.622075            0.0002559172726
6       -5.7818015937                   1.527139105             2.778677936e-007
7       -5.7817999871                   -1.197112931e-005               2.178124778e-012
procedimiento completado satisfactoriamente
```

Al cambiarle el orden vemos que el método funciona satisfactoriamente. Encontrando la raiz en 7 iteraciones.

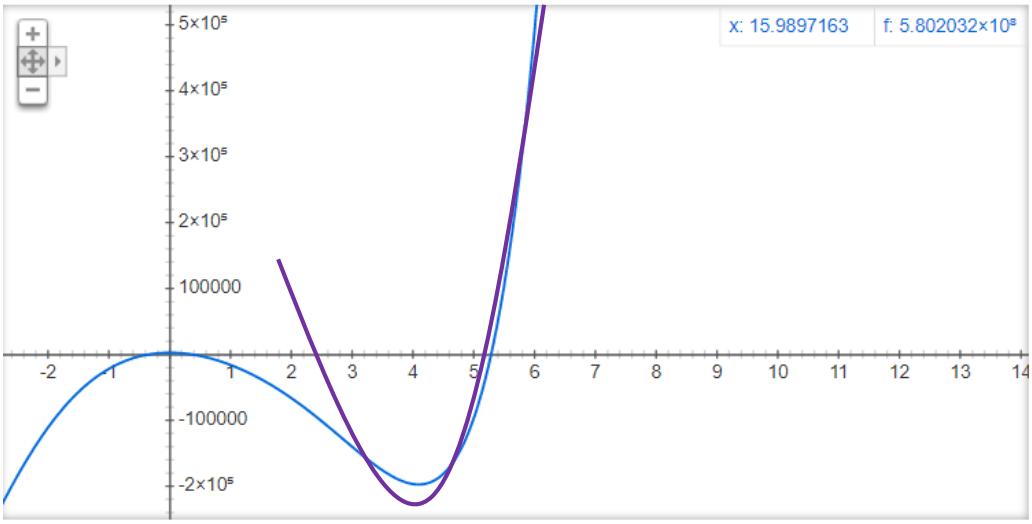
PARABOLA 2




Los valores que nos pueden servir para esta parábola son Xo = (-3,-1,1)

```
 - M U L L E R -
Digite el valor X0 : -3
Digite el valor X1 : -1
Digite el valor X2 : 1
Digite el numero de iteraciones : 100

n       Raiz                            f(raiz)                 COTA DE ERROR
2       1.0000000000                    -16023          0.503586325
3       0.6650765463                    -5961.680794            1.273529775
4       0.2925303876                    825.1356103             0.1777493139
5       0.3557678851                    20.03445541             0.004041166837
6       0.3572114361                    -0.05705716341          1.145475972e-005
7       0.3572073443                    -8.323572271e-007               1.671068863e-010
procedimiento completado satisfactoriamente
```

PARABOLA 3



```
 - M U L L E R -
Digite el valor X0 : 3
Digite el valor X1 : 5
Digite el valor X2 : 6
Digite el numero de iteraciones : 100

n       Raiz                            f(raiz)                 COTA DE ERROR
2       6.0000000000                    491142          0.1493881058
3       5.2201688617                    -26300.46719            0.0119018061
4       5.2830466586                    -1246.56795             0.0005577798662
5       5.2859950802                    -14.14074921            6.386344483e-006
6       5.2860288386                    0.0006029166659         2.722752716e-010
procedimiento completado satisfactoriamente
```

**Conclusión:** Como podemos ver en estas pruebas el programa nos da correctamente ya que son intervalos próximos a la raíz, pero para la PRUEBA 1 sucede un error, para el cual cambiamos las aproximaciones iniciales de orden y vemos un resultado exitoso, en general, es un metodo que asignandole unos valores aproximados y probando diferente orden de entrada, vemos que nos encuentra la raiz en pocas iteraciones.

---

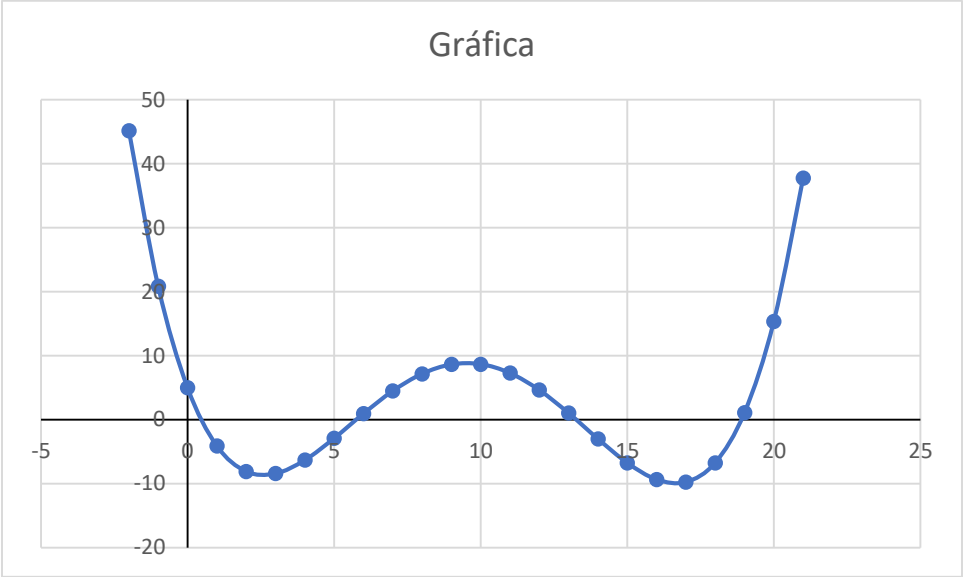
# TABLA RESUMEN

| METODO | VALOR(ES) INICIAL(ES) | RAIZ | ITERACIONES | COTA DE ERROR |
|---|---|---|---|---|
| REGLA FALSA | *[0 - 1]* | 0,3572073422 | 23 | 7,376366278*10^-9 |
| | *[4,5 - 5,5]* | 5,2860288334 | 9 | 4,295858165*10^-9 |
| | *[-6 – -5]* | -5,7817999854 | 9 | 1,967246538*10^-9 |
| | *[-1 - 0]* | -0,3396110483 | 30 | 7,455073273*10^-9 |
| SECANTE | *X0= -7 - X1= -6* | -5,7817999872 | 5 | 9,205533531*10^-10 |
| | *X0= -3 - X1= -1* | -0,3396110512 | 8 | 3,510001677*10^-11 |
| | *X0=3 - X1=1* | 0,3572073443 | 7 | 1,425384232*10^-9 |
| | *X0=7 - X1=6* | 5,2860288372 | 7 | 3,621416014*10^-12 |
| | *X0= 4,05- X1= -4,47* | -0,3396110512 | 13 | 2,203374466*10^-13 |

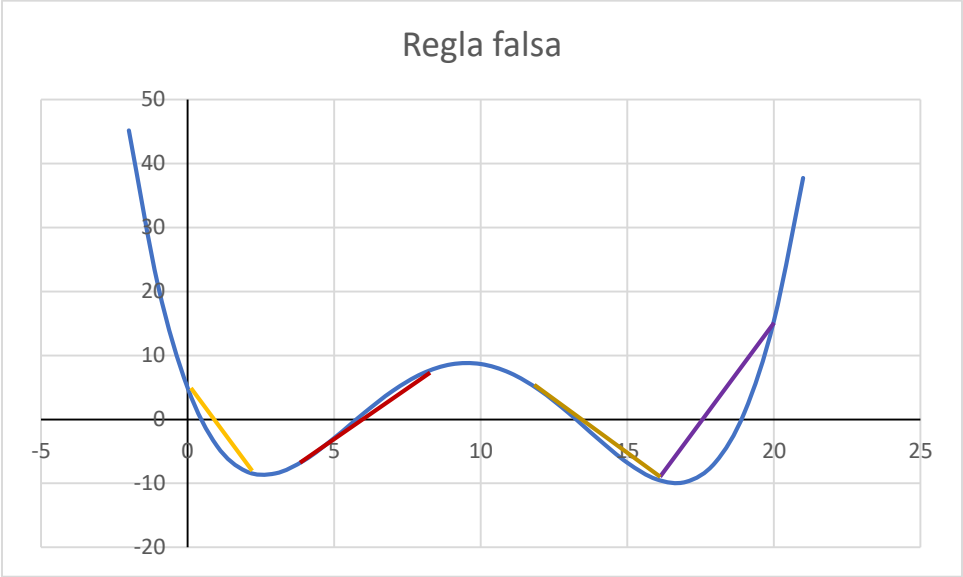
| | | | | |
|---|---|---|---|---|
| **MÜLLER** | **X0= -4 - X1= -5 - X2= -7** | -5,7817999871 | 6 | 2,178124778*10^-12 |
| | **X0= -3 - X1= -1 - X2= 1** | 0,3572073443 | 6 | 1,671068863*10^-10 |
| | **X0= 3 - X1= 5 - X2= 6** | 5,2860288386 | 5 | 2,722752716*10^-10 |

**SEGUNDO POLINOMIO**

f(x) = _____

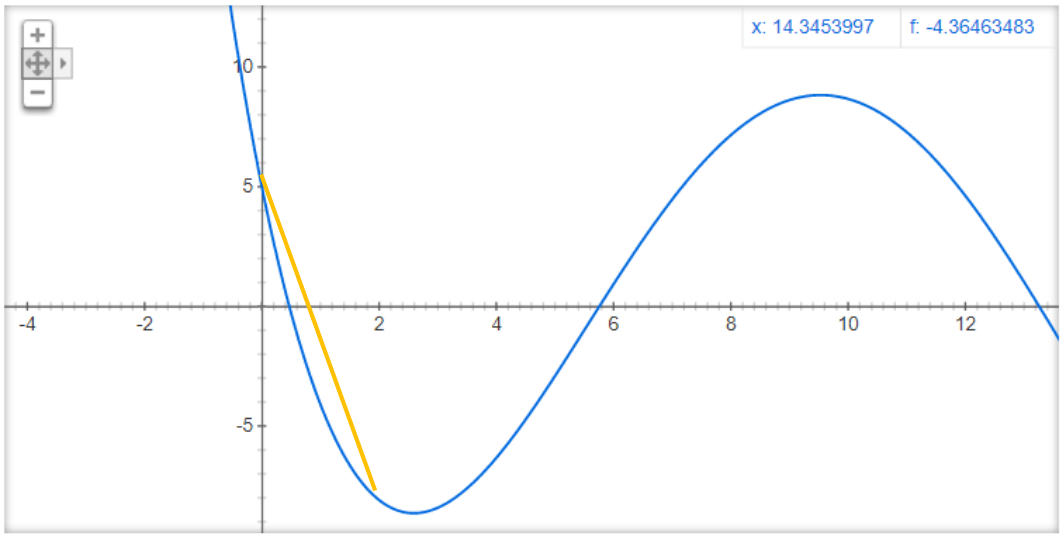



Para este polinomio vemos que el grado mas alto es 4 y efectivamente encontramos 4 raíces reales en la gráfica.



RAIZ 1

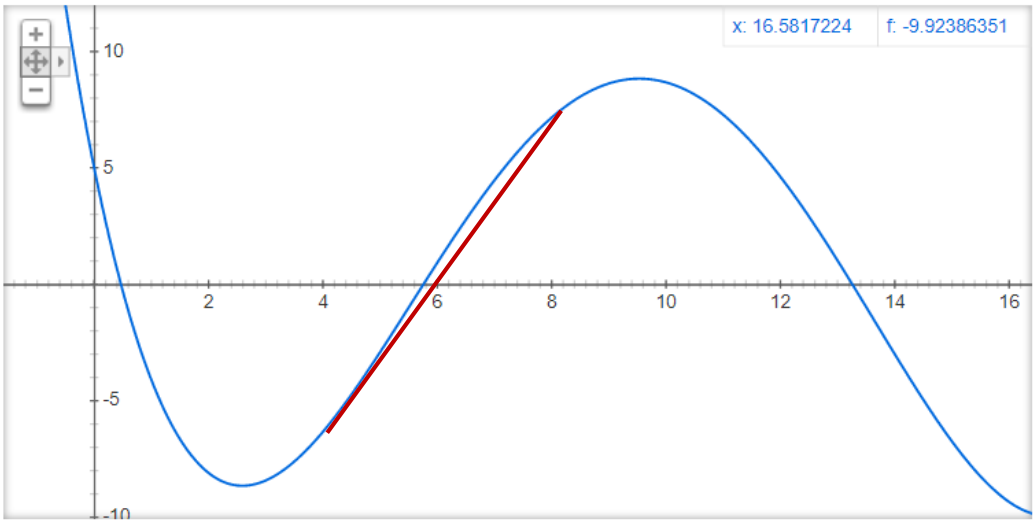
Tomaremos como intervalo de acotación a (0 y 2) garantizando así que nuestra primera raíz de nuestra función se encuentra en dicho intervalo.




```
 - R E G L A   F A L S A -
Digite el extremo inferior (Xi) : 0
Digite el extremo superior (Xs) : 2
Digite el numero de iteraciones : 100

 XR : 0.763382088
n       Raiz                    f(Raiz)                 Er
1       0.5110351531            -0.3871578211           0.4937956487
2       0.4743086893            -0.05366225634          0.07743156422
3       0.4692722478            -0.00730774128          0.01073245127
4       0.4685873847            -0.0009927699223                0.001461548256
5       0.4684943633            -0.0001348253296                0.0001985539845
6       0.4684817307            -1.830943771e-005               2.696506593e-005
7       0.4684800152            -2.486428036e-006               3.661887542e-006
8       0.4684797822            -3.376575182e-007               4.972856071e-007
9       0.4684797505            -4.585396626e-008               6.75315036e-008
10      0.4684797463            -6.226978659e-009               9.170793266e-009
Procedimiento completado satisfactoriamente
```

En esta ocasión Regla Falsa nos encuentra la raíz 0.4684797463 con un número de iteraciones de 10.

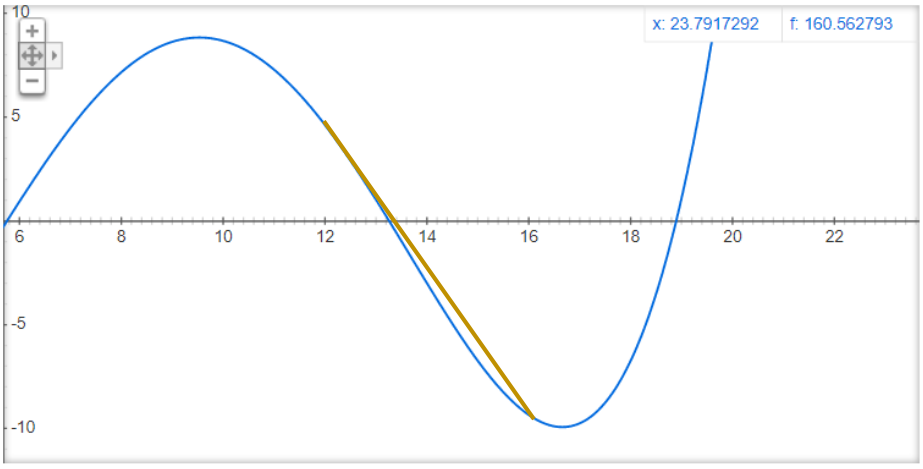
RAIZ 2




```
 - R E G L A   F A L S A -
Digite el extremo inferior (Xi) : 4
Digite el extremo superior (Xs) : 8
Digite el numero de iteraciones : 100

 XR : 5.877734954
n       Raiz                    f(Raiz)                 Er
1       5.7501311067            -0.0289653525           0.02219146746
2       5.7576625500            8.258200976e-005                0.001308073057
3       5.7576411384            1.25071706e-008         3.718808422e-006
4       5.7576411352            1.895116009e-012                5.632188886e-010
Procedimiento completado satisfactoriamente
```

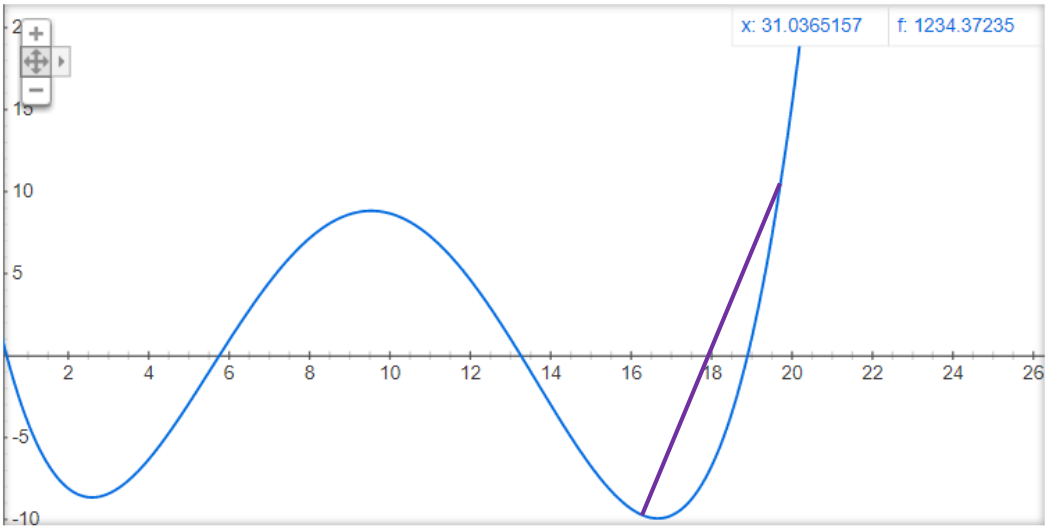
Para esta raíz, con los intervalos digitados tenemos un mejor resultado, encontrando la raíz en 4 iteraciones.

RAIZ 3



```
 - R E G L A   F A L S A -
Digite el extremo inferior (Xi) : 12
Digite el extremo superior (Xs) : 16
Digite el numero de iteraciones : 100

 XR : 13.32294471
n       Raiz                    f(Raiz)                 Er
1       13.2518089128           0.02273186079           0.005368006934
2       13.2574792094           4.828776564e-005                0.000427705486
3       13.2574912522           1.020491156e-007                9.083780893e-007
4       13.2574912777           2.156641532e-010                1.919723197e-009
Procedimiento completado satisfactoriamente
```

RAIZ 4

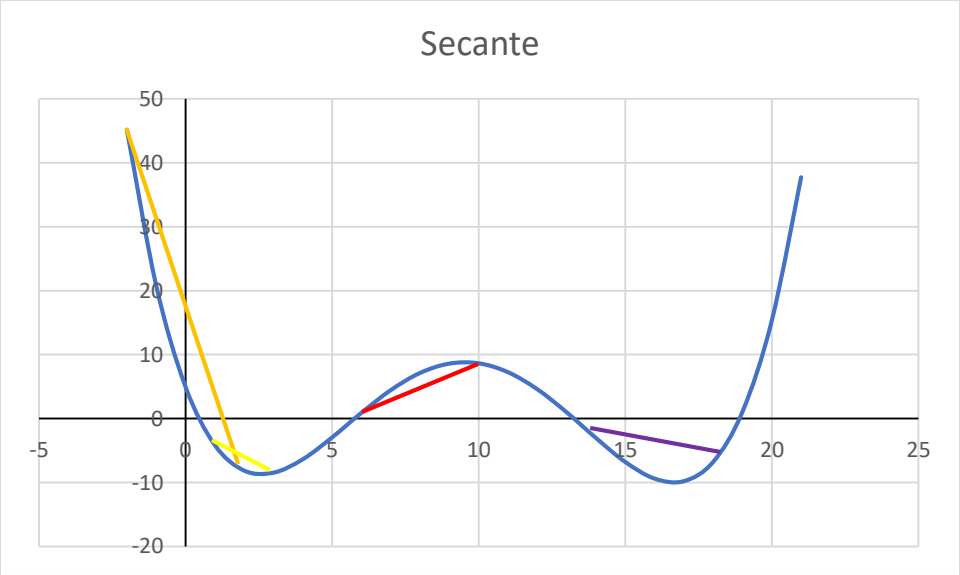



```
 - R E G L A   F A L S A -
Digite el extremo inferior (Xi) : 16
Digite el extremo superior (Xs) : 20
Digite el numero de iteraciones : 100

 XR : 17.51434034
n       Raiz                    f(Raiz)                 Er
1       18.4148421111           -4.160495639            0.0489008682
2       18.7530408218           -1.369508441            0.01803433982
3       18.8552413818           -0.3942868587           0.005420273224
4       18.8839279873           -0.109080673            0.001519101613
5       18.8918081900           -0.02984325471          0.0004171227316
6       18.8939599393           -0.00813987067          0.0001138855683
7       18.8945465266           -0.002218332223         3.104531999e-005
8       18.8947063641           -0.0006044173626                8.459384215e-006
9       18.8947499125           -0.0001646722314                2.304787988e-006
10      18.8947617771           -4.486384374e-005               6.27927475e-007
11      18.8947650095           -1.222279678e-005               1.710740947e-007
12      18.8947658901           -3.329999436e-006               4.660773473e-008
13      18.8947661300           -9.072303789e-007               1.269788715e-008
14      18.8947661954           -2.471672806e-007               3.45943246e-009
Procedimiento completado satisfactoriamente
```

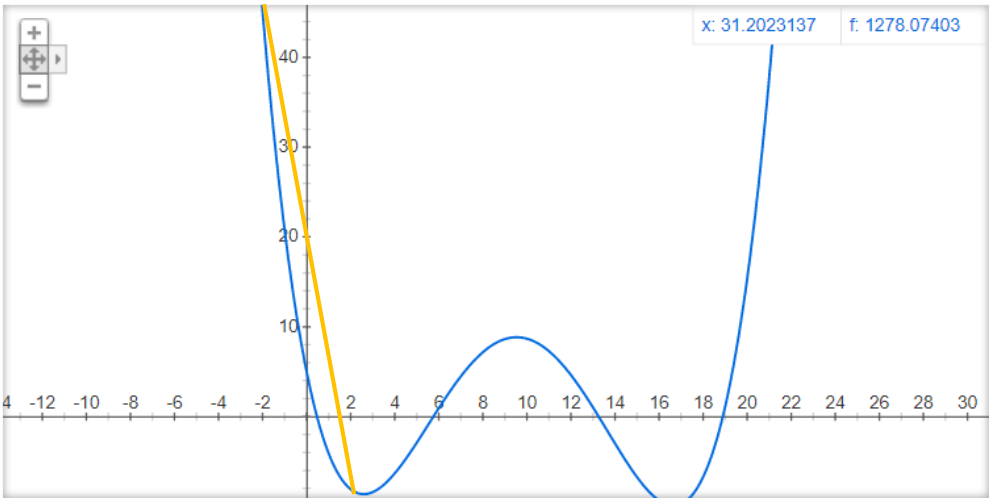
Con los valores suministrados, encontramos un notable incremento en las iteraciones en la búsqueda de la raíz.

**Conclusión:** De este método podemos concluir que, en algunos casos, no nos darán buenos resultados, tiende a demorar, para evitar esto, debemos usar intervalos mas cercanos a la raíz, reduciéndole trabajo al método.

**SECANTE**



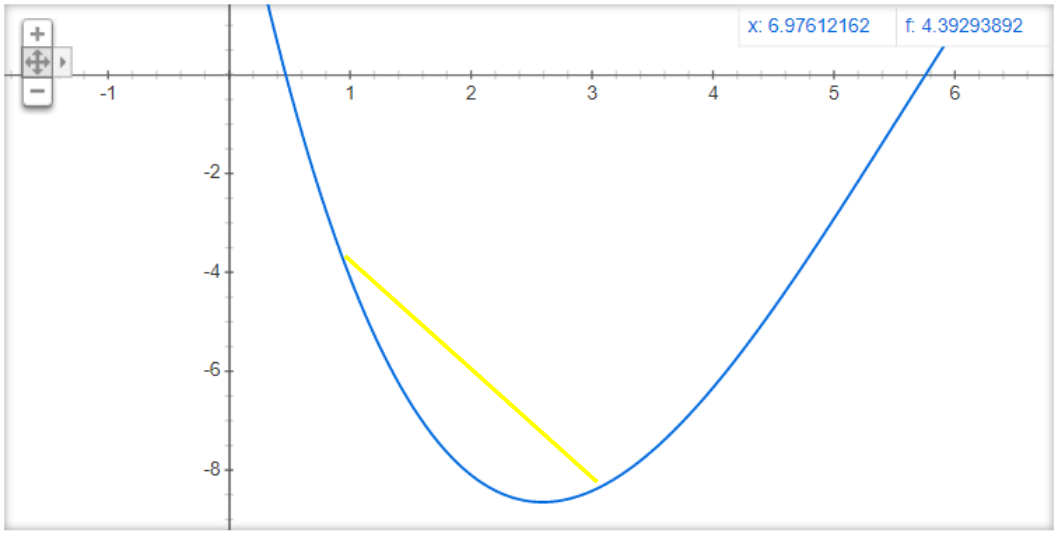

APROXIMACIÓN 1




Para este caso usaremos dos aproximaciones que contienen la raíz en el medio.

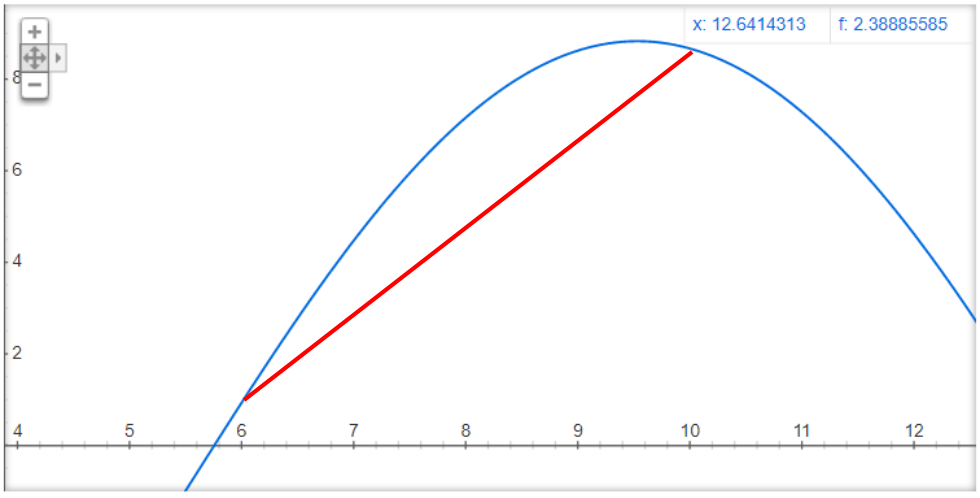
```
 - S E C A N T E -
Digite Xo : -2
Digite X1 : 2
Digite el numero de iteraciones : 100

 XR : 1.391876267
n       Raiz                    f(raiz)                 Er
1       -0.5868939386                   13.36402936             3.371597618
2       0.7650949800                    -2.481892365            1.76708638
3       0.5533375934                    -0.7614918338           0.382691126
4       0.4596085356                    0.0820567264            0.2039323695
5       0.4687260897                    -0.002271962773         0.01945177432
6       0.4684804468                    -6.468075308e-006               0.0005243397007
7       0.4684797455                    5.123439147e-010                1.497012441e-006
8       0.4684797456                    -3.113828639e-016               1.185707838e-010
Procedimiento completado satisfactoriamente
```

APROXIMACIÓN 2




```
 - S E C A N T E -
Digite Xo : 3
Digite X1 : 1
Digite el numero de iteraciones : 100

 XR : -0.9011579435
n       Raiz                    f(raiz)                 Er
1       0.6610223508                    -1.667882238            2.363279082
2       0.5344415744                    -0.5955697167           0.2368467994
3       0.4641377445                    0.04010415714           0.1514719085
4       0.4685731569                    -0.0008615490953                0.009465784437
5       0.4684798759                    -1.201647731e-006               0.0001991143957
6       0.4684797456                    3.609138075e-011                2.781032582e-007
7       0.4684797456                    2.003605615e-016                8.352508486e-012
Procedimiento completado satisfactoriamente
```

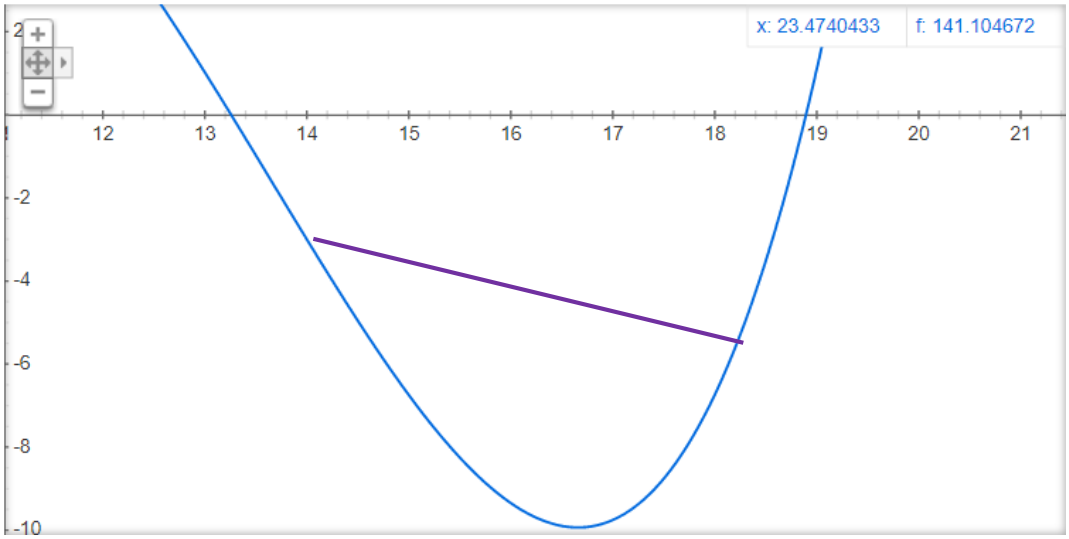
APROXIMACIÓN 3



```
 - S E C A N T E -
Digite Xo : 10
Digite X1 : 6
Digite el numero de iteraciones : 100

 XR : 5.52030588
n       Raiz                    f(raiz)                 Er
1       5.7588182621                    0.004539252275          0.04141689689
2       5.7576448469                    1.431346402e-005                0.0002038012548
3       5.7576411351                    -3.441449001e-010               6.446725423e-007
4       5.7576411352                    1.047772979e-015                1.549981827e-011
Procedimiento completado satisfactoriamente
```

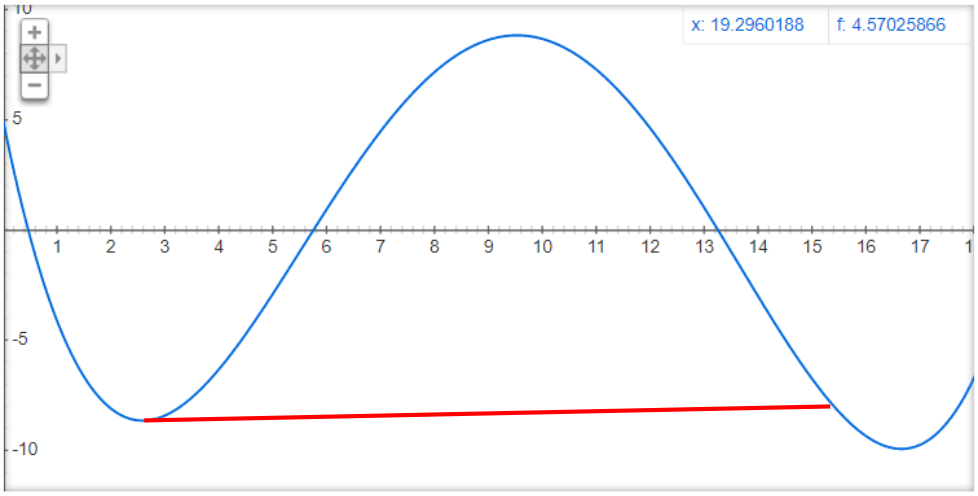
Como podemos apreciar el método ha resultado muy satisfactorio con solo 4 iteraciones para estas aproximaciones, y ha arrojado la raíz con un muy buen margen de error.

APROXIMACIÓN 4




```
 - S E C A N T E -
Digite Xo : 18
Digite X1 : 14
Digite el numero de iteraciones : 100

 XR : 10.79187166
n       Raiz                    f(raiz)                 Er
1       13.0992136197                   0.6294538045            0.1761435479
2       13.3051075883                   -0.1908204384           0.01547480674
3       13.2572104640                   0.001123581645          0.003612911208
4       13.2574908392                   1.754823429e-006                2.114843101e-005
5       13.2574912778                   -1.682422257e-011               3.308154187e-008
6       13.2574912778                   -6.383782392e-016               3.17151756e-013
Procedimiento completado satisfactoriamente
```

APROXIMACIÓN 5



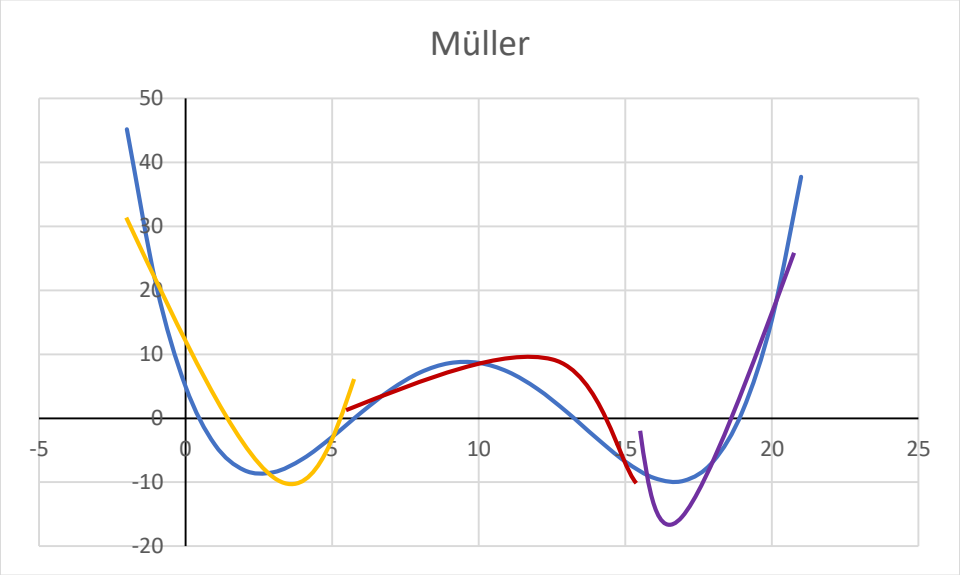

Ya sabemos que es un riesgo tomar aproximaciones como esta, pero en esta ocasión no nos presenta error, pero si presenta una notable incremento de iteraciones para encontrar la raiz.

```
 - S E C A N T E -
Digite Xo : 2.5
Digite X1 : 15.4
Digite el numero de iteraciones : 100

 XR : 171.3645926
n       Raiz                    f(raiz)                 Er
1       15.4002464378                   -7.978061108            10.12739288
2       15.4004928966                   -7.978751621            1.600330312e-005
3       12.5527090492                   2.715998816             0.2268660762
4       13.2759215826                   -0.07378909373          0.05447550506
5       13.2567928176                   0.002794613766          0.001442940634
6       13.2574908448                   1.732498432e-006                5.265153093e-005
7       13.2574912778                   -4.133109133e-011               3.266114201e-008
8       13.2574912778                   -6.383782392e-016               7.791455265e-013
Procedimiento completado satisfactoriamente
```

```
 - S E C A N T E -
Digite Xo : 15.4
Digite X1 : 2.5
Digite el numero de iteraciones : 100

 XR : 171.3645926
n       Raiz                    f(raiz)                 Er
1       2.5002888900                    -8.637265519            67.53791707
2       2.5005777820                    -8.637343288            0.0001155301305
3       -29.5848944613                  16325.10117             1.084522113
4       2.4836108618                    -8.632349736            12.91204907
5       2.4666627126                    -8.626492134            0.00687088233
6       -22.4928801632                  7102.431929             1.109664156
7       2.4363840554                    -8.613844601            10.23207493
8       2.4061864143                    -8.598413479            0.01255000066
9       -14.4203161267                  2049.939611             1.166860864
10      2.3359029293                    -8.55140684             7.173337062
11      2.2662940532                    -8.489203137            0.03071484745
12      -7.2335243849                   396.4222901             1.313304267
13      2.0671248772                    -8.221261565            4.499316594
14      1.8781608618                    -7.836385361            0.1006111985
15      -1.9692966629                   44.28332153             1.95372165
16      1.2996817869                    -5.769224113            2.51521448
17      0.9228883780                    -3.60388723             0.408276253
18      0.2957707385                    1.682829979             2.120282901
19      0.4953903366                    -0.246066088            0.4029541622
20      0.4699251939                    -0.01332582289          0.0541897796
21      0.4684671566                    0.0001161142887         0.003112357508
22      0.4684797514                    -5.398201638e-008               2.688447786e-005
23      0.4684797456                    -2.184251044e-013               1.249289789e-008
24      0.4684797456                    2.003605615e-016                5.059612886e-014
Procedimiento completado satisfactoriamente
```

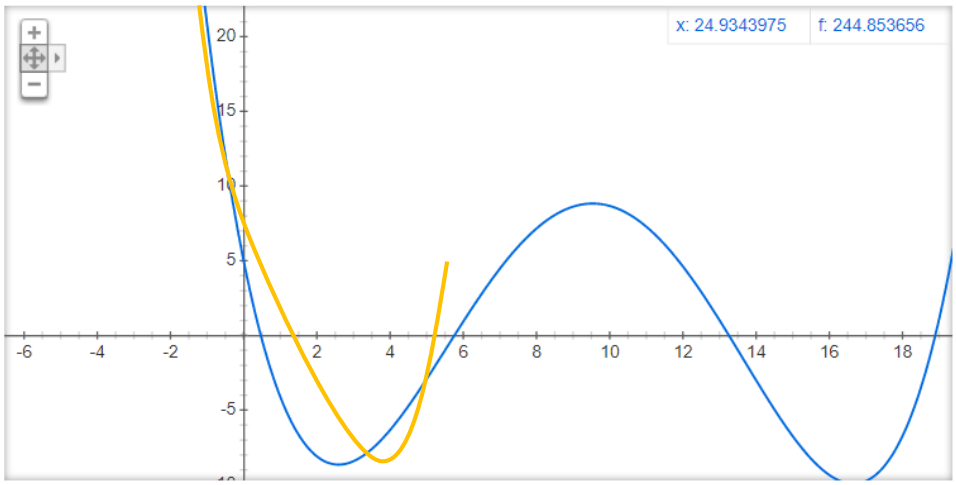
**Conclusión:** En este método comprobamos que, al digitar aproximaciones sin seguir la sugerencia, tendremos resultados poco favorables.

**MÜLLER**



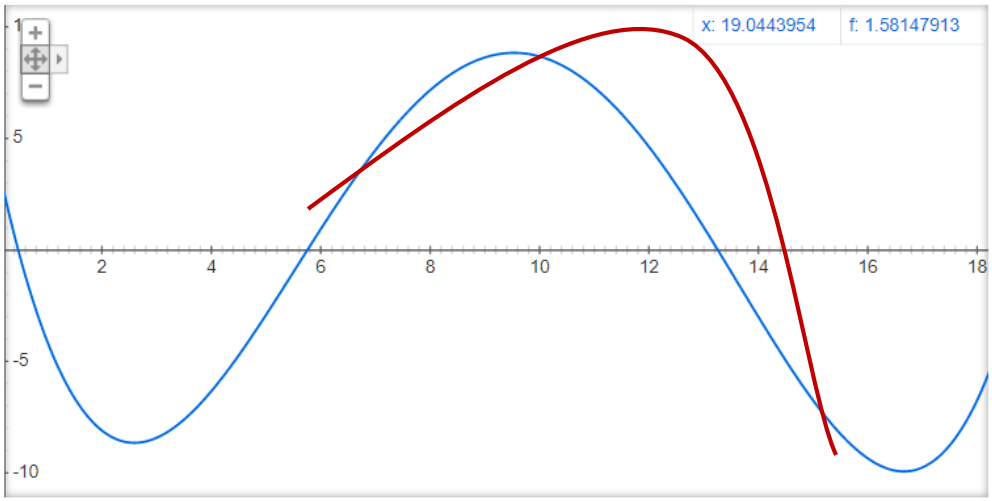

Ahora procedemos a utilizar el método de Müller como ya sabemos este método usa tres puntos con los cuales proyecta una parábola y va realizando sus aproximaciones.

PARABOLA 1




```
 - M U L L E R -
Digite el valor X0 : -1
Digite el valor X1 : 3
Digite el valor X2 : 5
Digite el numero de iteraciones : 100

n       Raiz                            f(raiz)                 COTA DE ERROR
2       5.0000000000                    -2.915          0.07868910239
3       5.4270496669                    -1.27926355             0.05367054987
4       5.7348417786                    -0.08796081512          0.003928909763
5       5.7574623285                    -0.0006895356005                3.10484873e-005
6       5.7576410946                    -1.565554109e-007               7.051020481e-009
procedimiento completado satisfactoriamente
```

PARABOLA 2

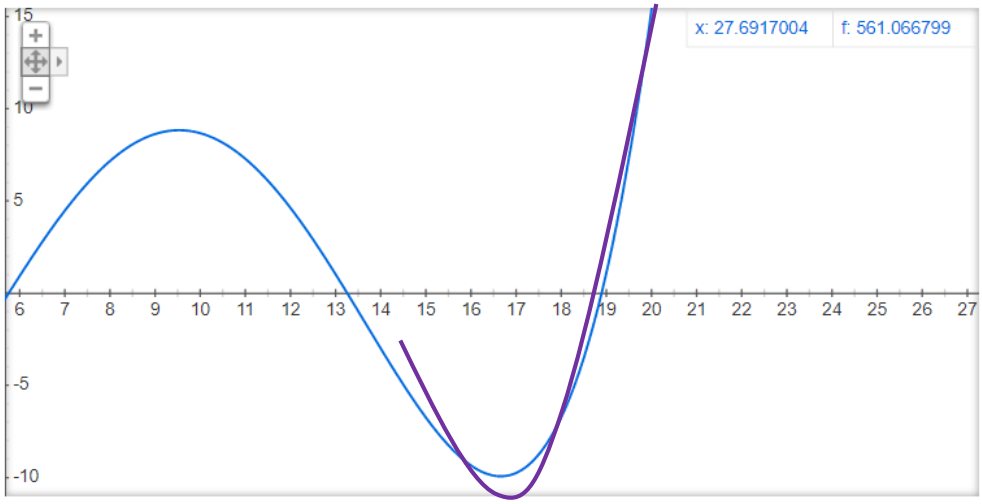



```
 - M U L L E R -
Digite el valor X0 : 7
Digite el valor X1 : 10
Digite el valor X2 : 15
Digite el numero de iteraciones : 100

n       Raiz                          f(raiz)                COTA DE ERROR
2       15.0000000000                 -6.745          0.09564231428
3       13.6905993904                 -1.749506258           0.0384083017
4       13.1842160429                 0.2924131973           0.005400878268
5       13.2558090539                 0.006730530605         0.0001270004206
6       13.2574927611                 -5.935157606e-006              1.118868223e-007
7       13.2574912778                 -1.972408348e-011              3.718192323e-013
procedimiento completado satisfactoriamente
```

Cambiando el orden de entrada de los valores, vemos una mejoría, la raíz es encontrado en una iteración menos.

```
 - M U L L E R -
Digite el valor X0 : 15
Digite el valor X1 : 10
Digite el valor X2 : 7
Digite el numero de iteraciones : 100

n       Raiz                          f(raiz)                COTA DE ERROR
2       7.0000000000                  4.4694          0.2052397506
3       5.8079730579                  0.1938816525           0.007949914065
4       5.7621643465                  0.01744126658          0.0007842783809
5       5.7576487471                  2.935377752e-005               1.322041573e-006
6       5.7576411352                  1.961112731e-010               8.832511541e-012
procedimiento completado satisfactoriamente
```

PARABOLA 3




```
 - M U L L E R -
Digite el valor X0 : 16
Digite el valor X1 : 18
Digite el valor X2 : 20
Digite el numero de iteraciones : 100

n       Raiz                            f(raiz)                 COTA DE ERROR
2       20.0000000000                   15.34           0.06246455329
3       18.8241574159                   -0.6976152835           0.003638837235
4       18.8929056244                   -0.01877769682          9.825666895e-005
5       18.8947621608                   -4.09890895e-005                2.148274399e-007
6       18.8947662199                   1.46511317e-010         7.678995599e-013
procedimiento completado satisfactoriamente
```

**Conclusión:** Para este polinomio en este método, tuvimos unas aproximaciones más acertadas, dándonos como resultado pocas iteraciones en la búsqueda y hallazgo de la raíz.

# TABLA RESUMEN

| METODO | VALOR(ES) INICIAL(ES) | RAIZ | ITERACIONES | COTA DE ERROR |
|---|---|---|---|---|
| **REGLA FALSA** | ***[0 - 2]*** | 0,4684797463 | 10 | 9,170793266*10^-9 |
| | ***[4 - 8]*** | 5,7576411352 | 4 | 5,632188886*10^-10 |
| | ***[12 – 16]*** | 13,2574912777 | 4 | 1,919723197*10^-9 |
| | ***[16 - 20]*** | 18,8947661954 | 14 | 3,45943246*10^-9 |
| **SECANTE** | ***X0= -2 - X1= 2*** | 0,4684797456 | 8 | 1,185707838*10^-10 |
| | ***X0= 3 - X1= 1*** | 0,4684797456 | 7 | 3,510001677*10^-12 |
| | ***X0=10 - X1=6*** | 5,7576411352 | 4 | 1,549981827*10^-11 |
| | ***X0=18 - X1=14*** | 13,2574912778 | 6 | 3,17151756*10^-13 |
| | ***X0=2,5 - X1=15,4*** | 13,2574912778 | 8 | 7,791455265*10^-13 |
| **MÜLLER** | ***X0= -1- X1= 3 – X2= 5*** | 5,7576410946 | 5 | 7,051020481*10^-9 |
| | **X0= 15 - X1= 10 - X2= 7** | 5,7576411352 | 5 | 8,832511541*10^-12 |
| | **X0= 16 - X1= 18 - X2= 20** | 18,8947662199 | 5 | 7,678995599*10^-10 |