Taller 1 - Análisis Numérico

Juan Camacho

j_camacho@javeriana.edu.co

Gabriela Camacho

gabriela.m.camacho@javeriana.edu.co

15 de febrero de 2020

1. **Problema:** Para calcular el error de redondeo del numero 536.78 se tomo como el valor de 536.7 puesto que es un redondeo inferior, por lo tanto el valor se deja en la cifra decimal anterior.

   Para hacer el calculo de el error relativo de el redondeo se hizo uso de la siguiente ecuacion:

$$ErrorRelativo = |\frac{ValorAntiguo - ValorRedondeado}{ValorAntiguo}| * 100\,\% \quad (1)$$

   Siendo asi, 536.78 el valor valor antiguo y 536.7 el valor redondeado, al hacer el calculo del error relativo este arrojo el resultado de $0.015\,\%$. El cual se dejo en dos cifras significativas.

   El codigo para comprobar, se encuentra en el archivo R.

2. **Problema:** Para realizar el algoritmo se hizo uso del lenguaje de programacion R, este codigo se encuentra adjuntado en el repositorio.

Iteracion	Valor Calculado	Error Absoluto
1	2.64583	$8{,}20 \cdot 10^{-5}$
2	2.64575	$1{,}27 \cdot 10^{-9}$
3	2.64575	0

Para llegar a los resultados mostrados en la tabla se usaron los valores de :

- Dato= 7
- Error permitido= $1 \cdot 10^{-5}$
- Valor inicial= 3

3. **Problema:** Al utilizar el teorema de Taylor para calcular la aproximacion de $e^{(0,5)}$ se obtuvo como resultado el siguiente polinomio:

$$P_3(X) = 1 + X + \frac{X^2}{2} + \frac{X^3}{6} \tag{2}$$

Reemplazando X=0.5 se hallo que $e^{(0,5)} \approx 1{,}6458$. El codigo de respuesta se encuentre adjunto para su verificacion.

4. **Problema:** Para determinar la propagacion de error al momento de usar instrumentos de medida se utilizo el siguiente algoritmo, con el cual se especifica en cada variable el proceso realizado teniendo en cuenta los metodos adecuados para obtener la respuesta requerida:

```
#Propagacion de Error

while(velocidadInferior <= velocidad + errorVelocidad)
{
  distancia = round(velocidadInferior*tiempoInferior,2)
  errorAbsoluto = round(abs(distancia-respuesta),2)
  errorRelativo = round(errorAbsoluto/respuesta*100,2)

  vectorVelocidad[iCounter] = velocidadInferior
  vectorTiempo[iCounter] = tiempoInferior
  vectorDistancia[iCounter] = distancia
  vectorEAbsoluto[iCounter] = errorAbsoluto
  vectorERelativo[iCounter] = errorRelativo

  iCounter=iCounter+1

  tiempoInferior = tiempoInferior + errorTiempo

  if( tiempoInferior > tiempo+errorTiempo)
  {
    tiempoInferior = tiempo-errorTiempo
    velocidadInferior = velocidadInferior + errorVelocidad
  }
}

resultados = matrix(c(vectorVelocidad,vectorTiempo,
    vectorDistancia,vectorEAbsoluto,vectorERelativo)
                    ,ncol=5,nrow = length(vectorEAbsoluto),
                        byrow = FALSE
                    , dimnames = list(seq(1,iCounter-1,1), c
                        ("Velocidad","Tiempo","Distancia","E
                        .Absoluto","E.Relativo")))

print(resultados)
```

Para el anterior algoritmo, se usaron las variables en los siguientes valores:

- velocidad = 4
- tiempo = 5
- d = velocidad * tiempo

El procedimiento arrojo la siguiente tabla que hace el analisis correspondiente a como se halla el error absoluto y relativo de las operaciones cambiando algunas aproximaciones:

	Velocidad	Tiempo	Distancia	E.Absoluto	E.Relativo
1	3.9	4.9	19.11	0.89	4.45
2	3.9	5.0	19.50	0.50	2.50
3	3.9	5.1	19.89	0.11	0.55
4	4.0	4.9	19.60	0.40	2.00
5	4.0	5.0	20.00	0.00	0.00
6	4.0	5.1	20.40	0.40	2.00
7	4.1	4.9	20.09	0.09	0.45
8	4.1	5.0	20.50	0.50	2.50
9	4.1	5.1	20.91	0.91	4.55

Resultado obtenido al realizar las operaciones aproximando al valor dado

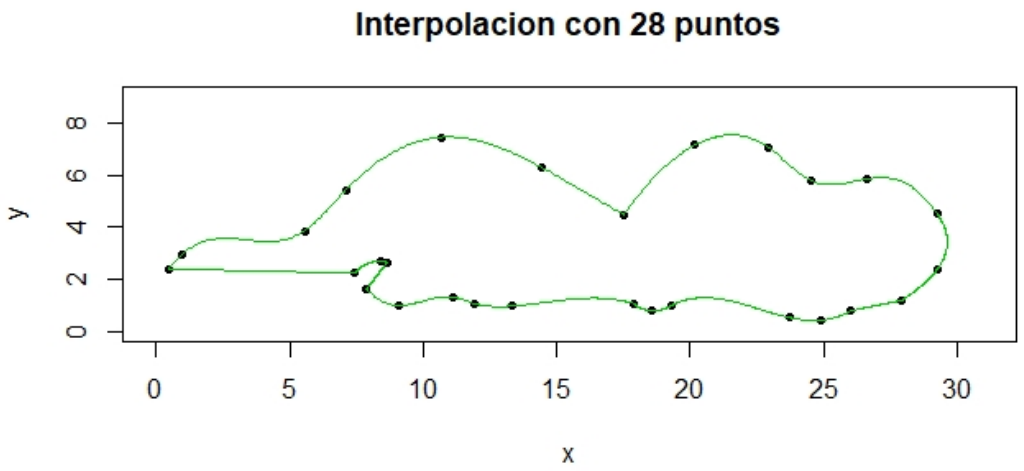
5. **Problema:** Para evaluar un polinomio realizando la menor cantidad de multicaciones posibles. Se utilizo el metodo de Horner:
   Pseudocodigo:

   - Se recibe un polinomio y el valor a evaluar.
   - for(rev), se recorre el arreglo del polinomio al reves, hasta finalizar el algoritmo. Durante este proceso se van realizando cambios a una variable con el resultado, la sumas y las multiplicaciones.
   - Hasta finalizar el ciclo y se entrega el resultado con el polinomio, el numero total de sumas y multiplicaciones que se ejecutaron.

   Para la comprobacion de este punto se adjunta el codigo en el repositorio

6. **Problema:** Con el objetivo de reconstruir la silueta de un perrito, utilizando la menor cantidad de puntos posibles se hizo uso de la herramienta Ilustrator, para asi poder calcar la imagen y lograr una aproximacion mas cercana a un plano cartesiano.

   El siguiente resultado se obtuvo al ejercutar la interpolacion por splines, dando 28 puntos de referencia en el cual se dividieron en 10 grupos para que asi la funcion generada por la interpolacion fuera mas precisa.



Resultado obtenido al realizar la interpolacion con splines

1. **Numero de Operaciones**

   a) **Problema: Teorema de Horner**
   El metodo de Horner busca solucionar de la manera mas eficiente el valor del polinomio utilizando el menor numeros de productos o adiciones

   Para este ejecicio es necesario:

   1) **Demostración por medio de inducción matemática que el número mínimo de multiplicaciones es n siendo n el grado del polinomio**
   Sea $P_0(x) = a_0x^0 = a_0$. El número de multiplicaciones para hallar $P_0(x_0)$ es igual a 0. Por lo que se cumple para el primer

caso $k = 0$.
Por lo tanto que $P_k(x) = a_0 + a_1x + ... + a_kx^k$ y que $P_k(x_0) = a_0 + a_1x_0 + ... + a_kx_0^k$ tiene k multiplicaciones y que el método de Horner se expresa:

$$\begin{gathered} b_k = a_k \\ b_{k-1} = a_{k-1} + b_k * x_0 \\ ... \\ b_0 = a_0 + b_1 * x_0 \end{gathered}$$

Y se debe llegar a la forma $k + 1$ del polinomio:

$$P_{k+1}(x_0) = a_0 + a_1x_0 + ... + a_{k+1}x_0^{k+1} \tag{3}$$

En otra forma:

$$P_{k+1}(x_0) = a_0 + x(a_1 + x(a_2 + ... + x(a_k + xa_{k+1}))) \tag{4}$$

Se reescribe $P_k(x_0)$ y se reemplaza a_k por b_k (Primera instrucción del método):

$$P_k(x_0) = a_0 + x_0(a_1 + ... + x_0(a_{k-1} + x_0 * b_k)) \tag{5}$$

Se añade una iteración al método de Horner para b_{k+1}, añadiendo una multiplicación más al método ($mult_{k+1} = k + 1$)

$$\begin{gathered} b_{k+1} = a_{k+1} \\ b_k = a_k + b_{k+1} * x_0 \\ ... \\ b_0 = a_0 + b_1 * x_0 \end{gathered}$$

Y se reemplaza b_k en $P_k(x_0)$:

$$P_k(x_0) = a_0 + x_0(a_1 + ... + x_0(a_{k-1} + x_0 * (a_k + b_{k+1} * x_0)) \tag{6}$$

Despejando la ecuación se llega a la forma:

$$P_k(x_0) = a_0 + x_0(a_1 + ... + x_0(a_{k-1} + x_0 * (a_k + b_{k+1} * x_0)) \tag{7}$$

La cual es equivalente a $P_{k+1}(x_0)$, quedando demostrado el número de múltiplicaciones iguales a k, el grado del polinomio.

1) Por medio de la programacion de R verificar los resultados de Horner

   Esta implementacion se encuentra en el archivo R en esta carpeta bajo el comentario #Metodo Horner

2) Evaluar con el siguiente polinomio y comparalo bajo la siguiente restriccion $x = 1{,}0001$ con $P(x) = 1+x+x^2+...+x^50$. Encuentre el error de calculo al comparalo con la expresion equivalente $Q(x) = (x^{51} - 1) = (x - 1)$:

   Al evaluar el polinomio con el metodo de Horner este arrojo **50.122696230512** por lo que al momento de compararlo con la expresion equivalente arrojo un error absoluto de **1.00501226962297** y un error relativo de **1.96569003208959 %**

b) **Problema: Numeros Binarios**

1) En este punto se realiza conversion de los 15 primero bits a binarios, para esto a π le quitamos el numero antes del punto (3), lo multiplicamos por dos, si este resultado es 1 o mayor igual a 1, se interpreta este como un 1 de binario, o si esta condicion no se cumple es 0 binario.

   Al ejecutar el codigo el resultado en binario del numero π es **11.0010010000111**

2) Para convertir los numeros binarios dados a base 10, se utilizo de 2 algoritmos el primero que nos permite encontrar la parte entera y el segundo que encuentra la parte decimal, de esta manera al final se unen los resultados por medio de la concatenacion.

Numero Binario	Numero en base 10
1010101	341
1011.101	11.625
10111.010101...	23.6665
111.1111...	7.996094

Resultados obtenidos para los numeros solicitados

3) Convierta los primeros numeros de base 10 a binaria, tambien se realizaron 2 algoritmos, primero se convirtio la parte decimal a binario y despues la parte la parte entera a binario, para que igualmente al final del ejercicio fuera posible unirlos por medio de la función cat.

Numeros en base 10	Numeros en binario
11.25	1011.0100000000
$\frac{2}{3}$	0.1010101010
30.6	11110.1001100110
99.9	1100011.1110011001

Resultados obtenidos para los numeros solicitados

Los algoritmos mencionados anteriormente se encuentran en el repositorio con los respectivos resultados obtenidos

c) **Problema: Epsilon de una Maquina**
Epsilon, es el numero de una maquina que contiene la distancia entre el 1 y el menor numero de puntos flotantes mayores 1, esto se genera gracias a que en los años 70s-80s, cada equipo podia manejar su propia interpretacion.
Es por esto que la IEEE decidio definir o estandarisar el punto flotante, esta trata el como se procesa un numero real, en el momento de realizar operaciones con este a nivel de procesador

1) ¿Como se ajusta un numero binario infinito en un numero finito de bits? La siguiente imagen es una representacion de como se representa un el infinito en bits. Se constituye en

exponente con el valor de 1, cero en la mantisa y no importa el signo puede ser 1/0 esto sin importar si es en 64 bits o 32 bits

$+\infty$	0	1111 1111	000 0000 0000 0000 0000 0000

Resultado obtenido al relizar el metodo de la secante

2) ¿Cual es la diferencia entre redondeo y recorte? Recorte: Es quitarle el decimal al numero, no se tiene encuenta la distancia eculidiana Redondeo: Es una aproximacion numerica que intenta hacer mas fiel al numero, funciones como piso y techo, estsa cumplen con este tipo de objetivos
   Ejemplo
   Redondeo 29.46->29.5 Recorte 29.46->29
3) El numero de punto flotante (IEEE) de precision doble asociado a x, el cual se denota como fl(x);para x(0:4) La represntacion de 0.4, es 00111110110011001100110011001101; para lograr este proceso primero se desarrollo el punto fijo, despues buscamos el valor del exponente y finamente cuadramos el signo
4) Error de redondeo En el modelo de la arimetica de computadora IEEE, el error de redondeo relativo no es mas de la mitad del epsilon de maquina:
   $$\frac{|fl(x)-x|}{|x|} \leq \frac{1}{2}\epsilon_{maq}$$
   En el repositorio se encuentra como por medio de python encontramos el epsilon de nuestra maquina es decir $2{,}220446e^{-16}$, al aplicar la formula comparamos $1{,}49*10-8$ <=$1{,}11*10-16$ Teniendo en cuenta lo anterior, encuentre el error de redondeo para x = 0.4
5) Verificar si el tipo de datos basico de R y Python es de precision doble IEEE y Revisar en R y Phython el format long Tanto python como R utilizan el la precision doble de IEEE en todos sus formatos
6) Encuentre la representacion en numero de maquina hexadecimal del nuemro real 9.4 Primero convertimos a binario y resultado es 01000001000101100110011001100110 Despues convertimos a hexadecimal 0x41166666

7) Encuentre las dos raıces de la ecuacion cuadratica $x^2 + 9^{12}x = 3$ Intente resolver el problema usando la arimetica de precision doble, tenga en cuenta la perdida de significancia y debe contrarestarla.

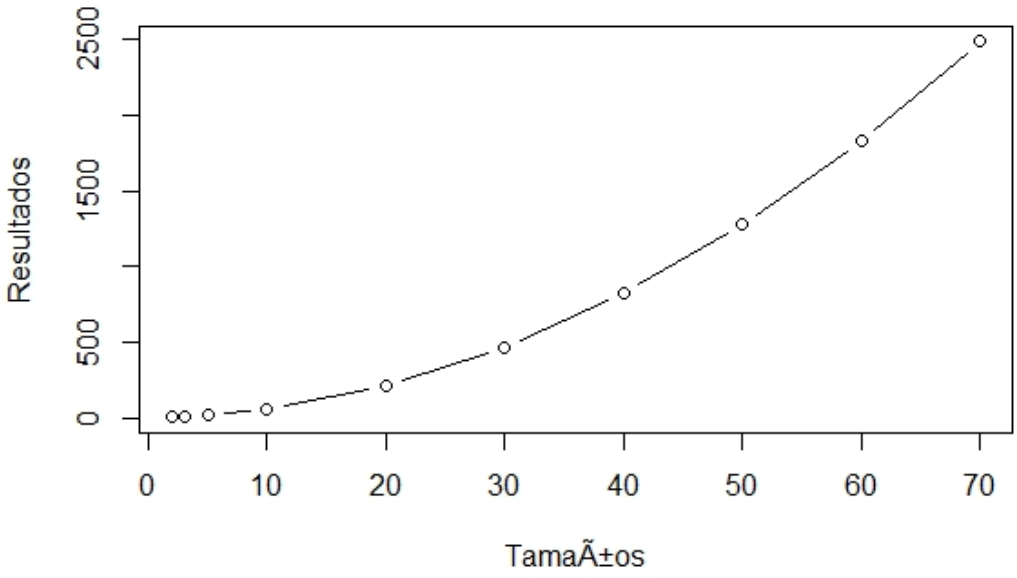
8) Explique como calcular con mayor exactitud las raıces de la ecuacion:

$$x^2 + bx - 10{-}12 = 0$$

Donde b es un numero mayor que 100 Para encontrar la mayor exactitude de esta raiz es necesario calcular para el caso critico b=100

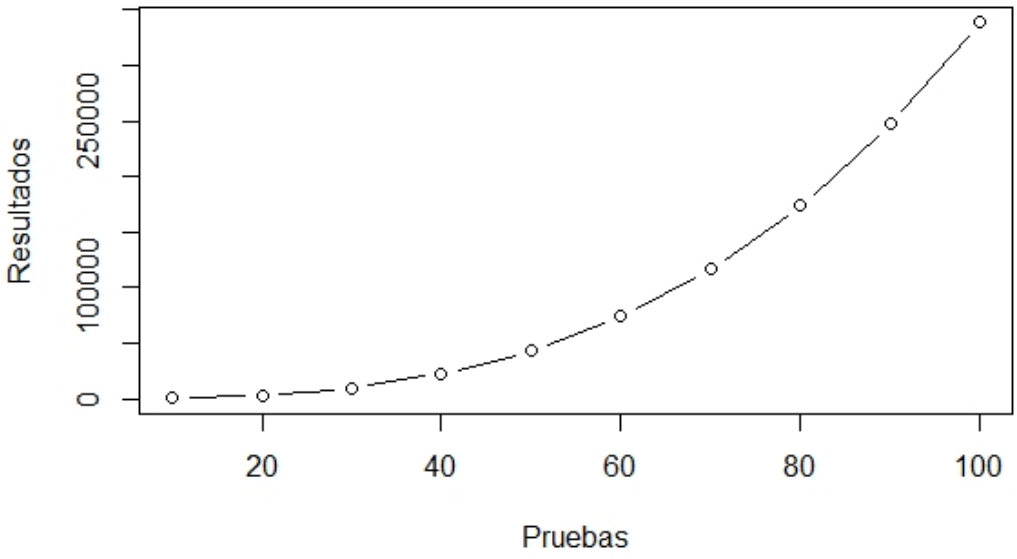
d) **Problema: Raices de una Ecuacion**

1) Algoritmo que permite sumar los elementos de una sub matriz cuadrada, para este algoritmo se tomaron varios tamaños para obtener la siguiente grafica :



Resultado obtenido al tomar varias matrices y obtener la suma de su triangular superior

2) Algoritmo que permite sumar los n^2 primeros numeros, para este caso, se tomo tambien varios datos para obtener la siguiente grafica:



Resultado obtenido al tomar varios tamaños y obtener la suma de sus primeros numeros al cuadrado

3) Para hayar la altura maxima del cohete de una funcion $y(t) = 6 + 2{,}13t^2 - 0{,}0013t^4$ Se utilizo el metodo de Newton Raphson para encontrar en que momento la primera derivada es cero, pues una altura maxima en una funcion es la derivada cuando se hace cero. Despues volvimos a derivar con el fin de encontra en la segunda derivada los mismos, si estos son negativos o opuestos a los de la primera funcion este es un punto maximo en que esta

e) **Problema: Covergencia de Metodos Iterativos / Newton**

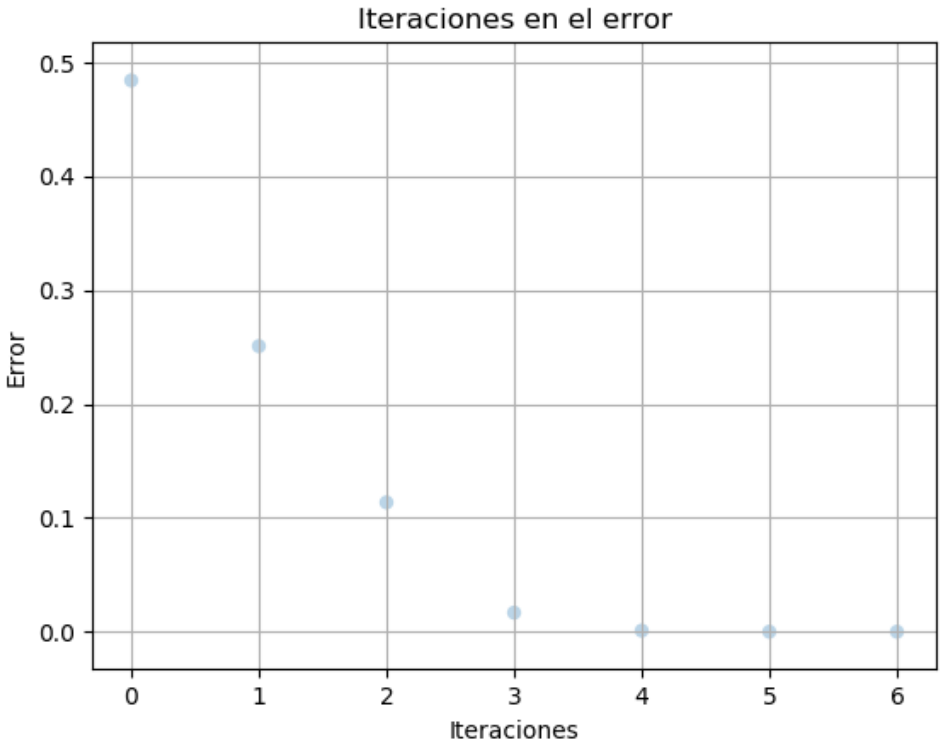
1) En este putno se quiere encontrar los puntos de convergencia de las 2 funciones dadas $f(x) = ln(x + 2)$ y $g(x) = sin(x)$ estas dos al restarlas nos dan otra funcion y es en esta en la cual vamos a encontrar los puntos de interseccion, para esto se nos otrogaron 2 formulas una que corresponde a el teorema de la secante

$$x_n = x_{n-1} \frac{f(x_{n-1}(x_1 - x_2)}{f(x_{n-1}) - f(x_{n-2})}$$

Este metodo se recorre poco a poco desde un punto a otro en nuetros caso (-1.9, -1.1) este punto lo elegimos por que en un comienzo seleccionamos (-1.9,0) pero este tenian pendientes contrarias, es decir -1.9 tenia una sentido y 0 el otro, cuando esto ocurre se tiene puntos maximo que no permiten la convergensia, mientras que con -1.9 y -1.1 esto no ocurre. Los resultados obtenidos son:

X	**Y**	**Error Absoluto**
-1.9	-1.1	0.484299
-1.393482	-1.9	0.250880
-1.631088	-1.625359	$9{,}019126^{-06}$
-1.631446	-1.631088	$4{,}822479^{-09}$

Su grafica es:



Resultado obtenido al relizar el metodo de la secante

Y es de tipo O(n)

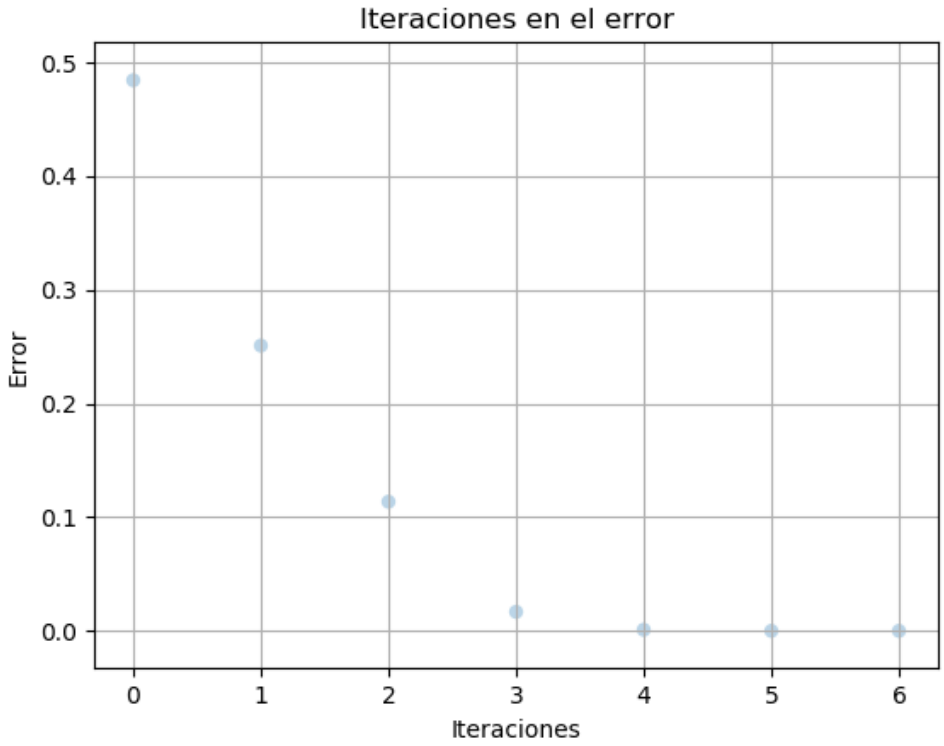
2) Para este punto el objetivo es el mismo que el punto anterio simplemente es necesario aplicarlo para la siguiente ecuación

$$x_{n+1} = x_n - f(x_n)\frac{x_n - x_{n+1}}{f(x_n) - f(x_{n-1})}$$

Utilizando el mismos rango de -1.9 a -1.1, los resultados obtenidos fueron:

X	**Y**	**Error Absoluto**
-1.9	-1.1	0.484299
-1.393482	-1.9	0.250880
-1.631088	-1.625359	$9{,}019126^{-06}$
-1.631446	-1.631088	$4{,}822479^{-09}$

La grafica obtenida es:



Resultado obtenido al relizar el metodo

y su notacion es de O(n)

f) **Problema: Newton**

1)

2) El metodo de Newton busca los ceros de una funcion por medio de las aproximaciones a partir de un valor incial para este caso en especifico $p_0 = 1$.
Pero a lo largo de la historia se ha modificando el algoritmo de Newton para hacerlo mejor, estas diferencias son radican con respecto a que funcion se evaluan es decir En el metodo de Newton Raphson se evalua en la primera derivada de la funcion y el metodo de Newton Raphson mejorado se evalua con respecto a la segunda derivada Los resultados encontrados fueron:

Iteracion	**Newton**	**Newton Raphson**	**Newton Raphson Mejorado**
1	$2{,}532439^{-08}$	$9{,}040218^{-09}$	$2{,}68144^{-08}$
2	$1{,}666869^{-08}$	$1{,}512395^{-08}$	$-6{,}022281^{-09}$
3	$2{,}833097^{-08}$	$1{,}418059^{-08}$	$3{,}323908^{-09}$
4	$8{,}855045^{-09}$	$5{,}853427^{-09}$	$9{,}684379^{-09}$
5	$-2{,}904790^{-11}$	$1{,}908672^{-08}$	$2{,}475979^{-08}$
6	$2{,}213079^{-08}$	$1{,}85908672^{-08}$	$2{,}475979^{-08}$
7	$1{,}98709^{-08}$	$-6{,}281730^{-10}$	$2{,}082099^{-08}$
8	$1{,}663693^{-08}$	$1{,}459126^{-8}$	$2{,}845568^{-08}$
9	$5{,}039701^{-08}$	$1{,}886822^{-8}$	$1{,}882883^{-08}$
10	$2{,}824407^{-08}$	$5{,}363914^{-9}$	$2{,}538493^{-09}$

g) **Problema: Convergencia Acelerada**

1) Para hacer el calculo de la convergencia acelerada usando el metodo de Aitken se uso un vector de 20 posiciones, el cual se lleno con la ecuacion requerida y luego se llamo al algoritmo y arrojo la siguiente tabla, el algoritmo se encuentra adjunto en el Script:

Valor Original	Valor Acelerado
0.5403023	0.9617751
0.8775826	0.9821294
0.9449569	0.9897855
0.9689124	0.9934156
0.9800666	0.9954099
0.9861432	0.99662
0.9898133	0.9974083
0.9921977	0.9979502
0.9938335	0.9983384
0.9950042	0.9986261
0.9958706	0.9988451
0.9965298	0.9990156
0.9970429	0.999151
0.9974501	0.9992603
0.9977786	0.9993497
0.9980475	0.9994239
0.9982704	No calculado
0.9984572	No calculado
0.9986153	No calculado
0.9987503	No calculado

Comparacion de datos usando el Metodo de Aitken

2) Para el segundo caso al usar el algoritmo de convergencia acelerada se tomaron las ecuaciones;

$$f(t) = 3\sin t^3 - 1 \tag{8}$$

y

$$g(t) = 4(\sin t \cos t) \tag{9}$$

Al momento de restarlas y enviarlas a al algoritmo de biseccion, luego con convergencia acelerada y al finalizar con el valor real se obtuvo:

- Valor sin aceleracion: **0.7661419**
- Valor con aceleracion: **0.7661438**
- Valor real de la solucion: **0.765642**

Todos los codigos expuestos en el repositorio fueron desarrollados por los autores de este archivo