Pontificia Universidad Javeriana
Departamento de Matemática
Análisis Numérico

Taller 1

Ejercicios realizados
por

Laura Mariana Jiménez Jiménez
la.jimenez@javeriana.edu.co

Paula Valentina Sanchez Peña
pa-sanchez@javeriana.edu.co

Sebastián Gutiérrez Zambrano
sebastian_gutierrez@javeriana.edu.co
Profesora: Eddy Herrera Daza

27 de septiembre de 2020

Índice

1. Número de operaciones

1.1. Horner

1.1.1. Definición

El método de Horner es un algoritmo iterativo que permite evaluar polinomios de forma monomial. El algoritmo evalúa el polinomio con más eficiencia. Si se evalúa normalmente el polinomio, se requerirían n sumas y $(n2 + n)/2$ multiplicaciones como minimo. Mientras que utilizando el algoritmo de Horner se requerirían simplemente n sumas y n multiplicaciones.

1.1.2. Solución

Para evaluar en $x = 1{,}00000000001$ con el polinomio $P(x) = 1 + x + x^2 + ... + x^{50}$ en la primera y la tercera derivada se uso el método de Horner, dando los siguientes resultados:

$$P'(1{,}00000000001) = 1275$$
$$P'''(1{,}00000000001) = 1499400$$

Para el error de cálculo se compara los resultados previamete obtenidos con la expresión equivalente definida como: $Q(x) = (x^{51} - 1)/(x - 1)$ tomando una precisión extendida a 256 bits, esto en razón de que el valor a evaluar $x = 1{,}00000000001$ es muy cercano a 1 y para este caso particular un valor como este puede llegar a dar una indeterminación debido a la división en cero que se generaría, es decir, resultados erróneos.

Se realizó el error absoluto y el error relativo tomando como valor real la versión simplificada $Q(x) = (x^{51} - 1)/(x - 1)$ y como experimental el polinomio extendido $P(x) = 1 + x + x^2 + ... + x^{50}$ lo que generó los siguientes errores:

—	Primera derivada	Tercera derivada
E. Relativo	$1{,}88856e^{-10}$	$9{,}340461e^{-17}$
E. Absoluto	$1{,}190909e^{-13}$	$1{,}259548e^{-16}$ height

[H]

Como se puede observar son errores pequeños que demuestran,una vez más, el buen funcionamiento del método.

1.2. Binario

1.2.1. Definición

El sistema de números binarios cuenta con solo dos dígitos, el uno (1) y el cero (0). Es la combinación de estos las que permite a este sistema numérico representar todos los números. Los números binarios son muy usados en el mundo de la computación y la electrónica debido a que su reducida cantidad de caracteres es fácil de representar con voltajes, como alto (1) o bajo (0) o también encendido (1) y apagado (0).

1.2.2. Solución

- **Primeros 15 bits en la representación binaria de π**

  Para hallar los primeros 15 bits de π se convirtió este número a doble mediante una función proveida por RStudio con error de redondedo de 8 y presición de 15 . Como indica el taller, la manera de convertir la parte entera es diferente a la manera de convertir la parte decimal, por esta razón cada una de estas partes fueron separadas y se les realizó un proceso de conversión individual. El resultado se muestra acontinuación:

  $$\pi = 3{,}141593, \text{ sistema base } 10$$
  $$\pi = 11{,}001001000011111, \text{ sistema base } 2$$

  A partir de los resultados anteriores se puede concluir que el calculo es correcto debido a que la parte entera del número binario representa el número 3 y en la parte decimal no se identifica ningún patrón, lo cual también es correcto debido a la naturaleza irracional de π.

- **Convertir números binarios a base 10**

  Para realizar la conversión binaria a decimal es necesario desarrollar el siguiente proceso:

  - Primero se toma solo la parte decimal del número binario, por ejemplo de 1011,101 → 101.
  - Se usa la siguiente formula:

$$\sum_1^n X * \frac{1}{2^n}$$

    Donde X corresponde al n-enésimo número de la secuencia de dígitos fracciónalabes binarios a convertir. Esto se puede observar en el siguiente ejemplo donde se toma 0,101 que corresponde a 0,625 en base 10.

$$1 * \frac{1}{2^1} + 0 * \frac{1}{2^2} + 1 * \frac{1}{2^3} = 0{,}625$$

  - Para la parte entera se toma un dígito a la vez y se multiplica por 2^x donde x corresponde a la posición del dígito.
  - Los números previamente obtenidos se van sumando hasta terminar todos los digítos.

  Realizando el proceso anterior se obtiene la siguiente conversión:

Sistema Binario	Sistema Decimal
1010101	85
1011.101	11.625
10111.010101...	23.33333
111.111111...	8

- **Convertir números base 10 a binario:**

  Igual que en el punto anterior se separa la parte entera de la decimal y se le aplican procesos independientes.

  - Para convertir la parte entera (10,3125 → 10) se utilizó a función llamada *as.binary*() de la libreria *binaryLogic* la cual recibe un valor entero y retorna su representación en binario.
  - Para la conversion de la parte decimal (10,3125 → 3125) tomamos el valor y lo multiplicamos por 2 hasta que el resultado sea 1 sin parte decimal. Este proceso puede conducirnos a dos casos; el primero es que el resultado de multiplicar por 2 sea menor a 1, en ese caso se escribe un 0. Por el contrario, si el resultado obtenido es un número mayor a 1 entonces se escribe un 1 y tomamos la parte fraccional para la siguiente iteración como se muestra en el siguiente ejemplo:

$$0,3125 * 2 = 0{,}625 \rightarrow 0$$
$$0,625 * 2 = 1,25 \rightarrow 1$$
$$0,25 * 2 = 0{,}5 \rightarrow 0$$
$$0,5 * 2 = 1 \rightarrow 1$$
Entonces el resultado es: 0,3125 $\rightarrow 0,0101$

Realizando el proceso anterior se obtiene la siguiente conversión, aqui es importante aclarar que la cantidad de decimales binarios varia dependiendo de la precisión que se exija en el programa, en este caso los resultados se realizaron con número de bits maximo igual 8:

Sistema Decimal	Sistema Binario
11.25	1011.01
0.6666667	0.10101010
30.6	11110.10011001
99.9	1100011.11100110

1.3. Representación del Punto Flotante de los Números Reales y epsilon de una Máquina

Es importante tener en cuenta que los computadores manejan un sistema aritmético binario lo cual representa cada número como una potencia de 2. En ese sentido ciertos números se pueden ver de manera exacta, pero por otro lado existen números como pi que llegan a tener una secuencia de decimales infinita, las cuales trascienden del sistema de representación binaria. Por eso mismo los computadores usan dos formatos diferentes de número: Punto fijo y punto flotante.

El estándar IEEE 754 establecido desde 1985, especifica todo lo relacionado con la aritmética computacional y especifica que un punto flotante está compuesto del signo, la mantisa y un exponente y de acuerdo a lo anterior se definen tres niveles de precisión:

- Sencilla (32 bits)
- Doble (64 bits)
- Completa (80 bits)

1.3.1. Solución

- **¿Cómo se ajusta un número binario infinito en un número finito de bits?**

  El ajuste de un número binario infinito en un numero finito de bits depende de la propiedad infinita del mismo número binario. Si el componente infinito del número se ubica en un fraccional periódico entonces se debe colocar, una vez, el periodo tras el punto decimal y se debe agregar alguna seña que indique la secuencia infinita.. Por otro lado, cuando el numero infinito que intentamos escribir de manera finita no presenta ningún patrón, ya sea porque es un numero infinitamente grande o porque es de naturaleza irracional se debería simplemente elegir una cantidad de bits a representar y redondear a esa cantidad.

Otra situación es si queremos representar el numero infinito en el estándar de IEEE, en este caso se colocan todos los bits del exponente mínimo en 1 y se usa el bit de signo para especificar si se trata de un infinito positivo o negativo.

- **¿Cuál 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 número, funciones como piso y techo,esa cumplen con este tipo de objetivos

  Redondeo 29,46 $\rightarrow$ 29,5

  Recorte 29,46 $\rightarrow$ 29

- **Número de punto flotante (IEEE) de precisión doble asociado a x, el cual se denota como fl(x); para x(0.4)**

  Primero se realiza la conversión del número decimal a binario:

$$0,4 * 2 = 0{,}8 \rightarrow 0$$
$$0,8 * 2 = 1{,}6 \rightarrow 1$$
$$0,6 * 2 = 1{,}2 \rightarrow 1$$
$$0,2 * 2 = 0{,}4 \rightarrow 0$$
$$0,4 * 2 = 0{,}8 \rightarrow 0$$

Entonces el resultado es: $0{,}4 \rightarrow 0,0110011001100110011001100110011001100$

  Como se puede ver 0.4 es un número infinito y además presenta periodicidad

  Representación de este número en punto flotante con precisión doble:

  Signo (1 bit): 0

  Exponente (11 bits): $(01111111101)_2$

  Mantisa(52 bits):$(1001100110011001100110011001100110011001100110011001)_2$

  De esta manera podemos concluir que fl(0, 4) en el estándar IEEE 754 es aproximadamente $(0,3999999999999999966693309261245)_{10}$

- **Error de redondeo**

  En el modelo de la arimética de computadora IEEE, el error de redondeo relativo no es más de la mitad del épsilon de máquina.

$$\frac{|fl(x) - x|}{|x|} \leq \frac{1}{2}\epsilon_{maq}$$

  Entonces el error de redondeo para $x = 0{,}4$ se denota como:

$$\frac{0{,}39999999999999966693309261245-0{,}4}{|0{,}4|} <= \frac{2^{-52}}{2}$$

$$\frac{0{,}39999999999999966693309261245-0{,}4}{|0{,}4|} <= 2^{-53}$$

$$8,3266726846887510^{17} <= 1,11022310^{16}$$

Esto quiere decir que el error es aproximadamente $8,32710^{17}$

- **Tipo de datos básico de R y Python**

  Para el caso de Python encontramos los siguientes niveles de presición dependiendo del tipo de dato:

  - Entero (int) - precisión sencilla.
  - Flotante (float) - precisión doble.
  - Enteros largos (long) - puede manejar números con cualquier nivel de precisión.

  Para el caso de R, los tipos de datos básicos como enteros, reales, enteros long e incluso números complejos se manejan siempre con precisión doble por defecto, sin embargo existe la posibilidad de alterarles esa precisión usando ciertas funciones.

- **Representación en número de máquina hexadecimal del número real 9.4**

  Conversión del número decimal a binario

  Utilizando el procedimiento previamente especificado se obtiene el siguiente resultado:

$$9 \rightarrow 1001$$

$$0{,}4 \rightarrow 0,0110011001100110011001100$$

$$9,4 \rightarrow 1001,0110011001100110011001100$$

  Representación en número de máquina hexadecimal: $0x9$

- **Raíces de la ecuacion cuadrática $x^2 + 9^{12}x = 3$ y como calcular raíces con mayor exactitud**

  Para el desarrollo de la ecuación $x^2 + 9^{12}x = 3$ se utilizó la formula cuadrática $x = \frac{-b \pm \sqrt{b^2-4ac}}{2a}$ teniendo en cuenta que, como se habia explicado anteriormente, los tipos de variables en R ya cuentan por defecto con una precisión doble. Sin embargo, para contrarestar la perdida de significancia y realizar un cálculo más preciso fue necesario acudir una ecuación concocida como la Fórmula de Citardauq.

  Citardauq mejora la precisión de la fórmula cuadrática racionalizando el numerador de la fórmula multiplicando por 1 en la forma:

$$\frac{-b \pm \sqrt{b^2-4ac}}{-b \pm \sqrt{b^2-4ac}}$$

Cuando multiplicamos por la ecuación obtenida previamente y simplificamos obtenemos la siguiente fórmula:

$$\frac{2c}{-b \pm \sqrt{b^2 - 4ac}}$$

Y esta es la utilizada para obtener las raíces con mayor exactitud.

2. Raíces de una ecuación

2.1. Definición

En los calculos numericos existen dos metodos de solución:

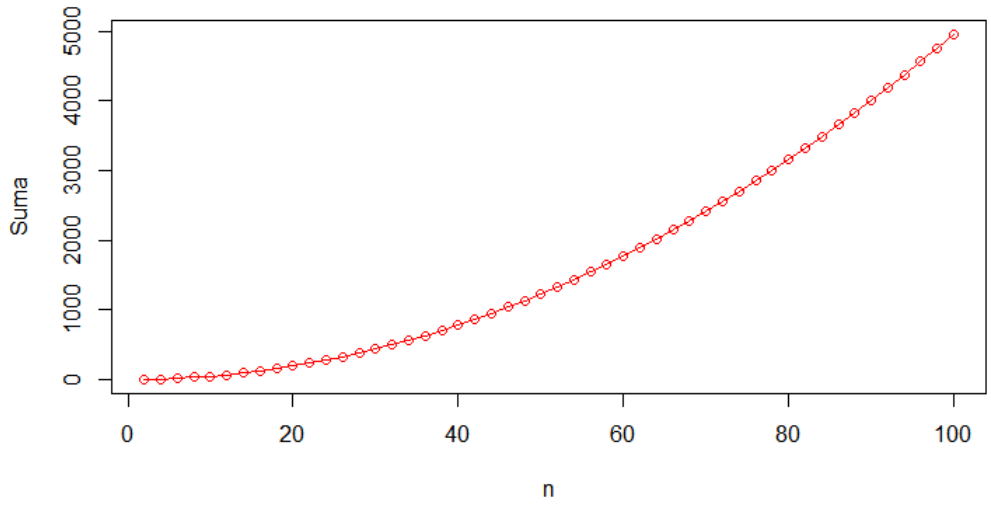
- **Metodos Iterativos** Consite en acercarse a la solución mediante aproximaciones sucesivas, a partir de un valor inicial estimado. En cada iteración se puede usar el resultado anterior para obtener una mejor aproximación
- **Métodos directos** La aproximación a la respuesta se produce a través de una serie finita de operaciones ariméticas y la eficiencia del método depende de la cantidad de operaciones el cual se puede asociar al tamaño del problema , en notación O() .

2.2. Solución

- **Implementar un algoritmo que permita sumar únicamente los elementos de la sub matriz triangular superior o triangular inferior, dada la matriz cuadrada An.**

  Se realizó un algoritmo que permite sumar los elementos de la sub matriz triangular superior haciendo uso de funciones proveídas por R como *ones*() y *upper*().

  Para este algoritmo se tomaron varios tamaños haciendo uso de secuencias con el fin de generar la siguiente gráfica que muestra que entre más grande sea la matriz mayor será el resultado de su suma:



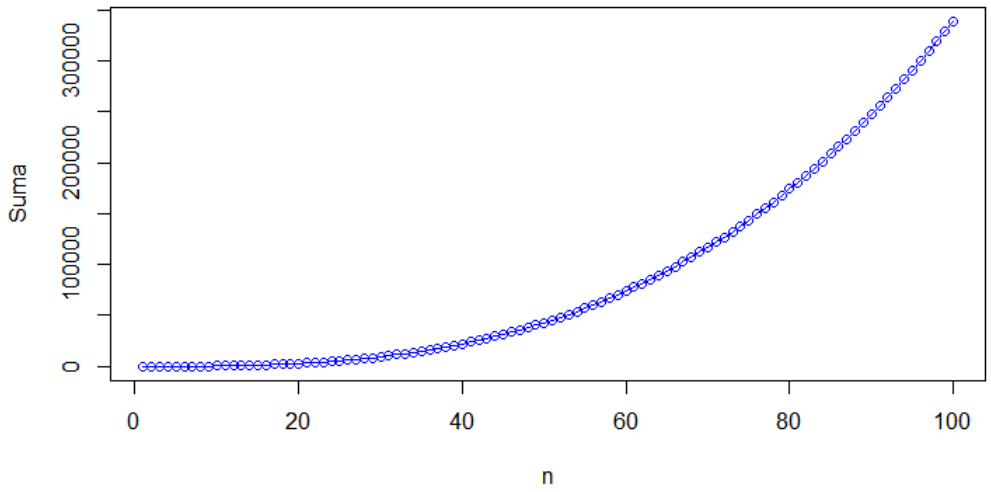
- **Sumar los n^2 primeros números naturales al cuadrado.**

  Se realizó un algoritmo que permite sumar los n^2 primeros números naturales al cuadrado en donde se evidencia que la tardanza de este método iterativo es lineal ya que depende del tamaño de n pero su crecimento no es exponencial porque a pesar del tamaño no es una operación compleja, en realidad puede ser fácilmente soportada por una máquina.

  Ejemplos

  - Para n = 20 el resultado de la sumatoria es: 2870
  - Para n = 50 el resultado de la sumatoria es: 42925
  - Para n = 150 el resultado de la sumatoria es: 1136275

  Para este algoritmo se generó una secuencia desde 1 hasta 100 con la cual se obtuvo la siguiente gráfica la cual confirma lo anteriormente dicho:



- **Utilizando dos métodos de solución de ecuación no lineal, encuentre la altura máxima que alcanza el cohete.**

  Dada la ecuación $y(t) = 6 + 2{,}13t^2 - 0{,}0013t^4$ calcular la altura máxima lo cual se hizo a partir de dos métodos, Newton-Raphson y el método de bisección. Se hallaron las raíces de la derivada de la función para de esa manera encontrar el máximo positivo obteniendo los siguientes resultados:

  - **Newton-Raphson**

    Se obtuvo una altura máxima de 878,480769230769183504339064609531622194 4 metros en 7 iteraciones en total.
  - **Bisección** Se obtuvo una altura máxima de 878,480769230769183504339064609531622188 4 metros en 128 iteraciones en total.

3. Convergencia de métodos iterativos

3.1. Primer ejercicio

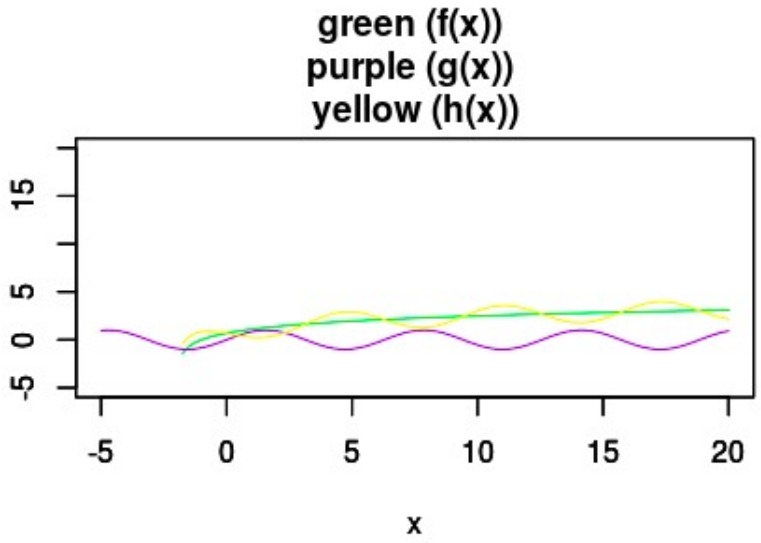
Teniendo las funciones $log(x+2)$ y $sin(x)$ se hizo uso de la siguiente ecuación para determinar el punto de intersección entre las dos funciones.

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

Primero se calcula el valor teórico haciendo uso de la función uniroot en Rstudio, con una tolerancia de $1e-16$.

teo	-1.63144359696888

Luego se decide graficar ambas funciones para observar mejor el problema. Y se calcula una nueva función que se va a llamar h con la cual se evaluará la ecuación anteriormente dada. Esta nueva funcion h es la resta de las dos funciones f y g.



Posteriormente se utiliza la ecuación recursivamente para encontrar el punto de intersección. Pero se puede notar que si bien la iteración va acercándose al valor teórico, no llega tan cerca, por lo cual el error el bastante grande (Pues es mayor que 1). Este fenómeno hace también que el error tenga un comportamiento bastante variable, no tiene una convergencia sino que más bien diverge, esto se verá en las gráficas del error.

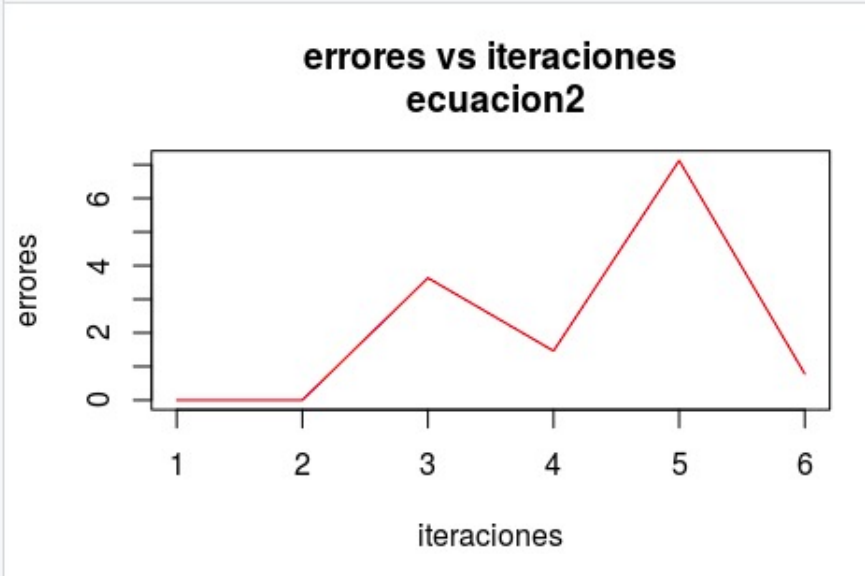
```
Iteracion 1 Error 0 Iteracion -1
Iteracion 2 Error 0 Iteracion 1
Iteracion 3 Error 3.511567651058261 Iteracion 1.880124054089376
Iteracion 4 Error 1.083319268167234 Iteracion -0.5481243288016509
Iteracion 5 Error 5.50780202876428 Iteracion 3.876358431795395
Iteracion 6 Error 1.472722895866294 Iteracion -3.104166492835179E
```

A continuación la gráfica de iteraciones versus error dónde se puede observar la divergencia del uso de este método.

Ahora, se da una nueva ecuación a utilizar con la cual se evaluará el punto de intersección para comparar ambas ecuaciones. De esta nueva ecuación se analizar que el error final es menor que la anterior, el número de iteraciones es el mismo, pero el valor final es más cercano al valor teórico que con la anterior ecuación.

```
Iteracion 1 Error 0 Iteracion -1
Iteracion 2 Error 0 Iteracion 1
Iteracion 3 Error 3.631443596968885 Iteracion 2
Iteracion 4 Error 1.461851972898963 Iteracion -0.1695916240699211
Iteracion 5 Error 7.123889379530706 Iteracion 5.492445782561822
Iteracion 6 Error 0.781895470321468 Iteracion -2.4133390672903535
```

Sin embargo, la gráfica de error versus número de iteraciones se comporta igual que con la ecuación anterior, con divergencia, sin un comportamiento tan predecible.



Es necesario recordar que para este último ejercicio con la ecuación 2 se utilizó una tolerancia mayor $(1e-8)$

3.2. Segundo ejercicio

Para resolver la problemática dada se utilizó Wolfram
$f(1) = 3,\ f(2) = 4,\ f(x) = a + (a * x + b) * e^(a * x + b)$

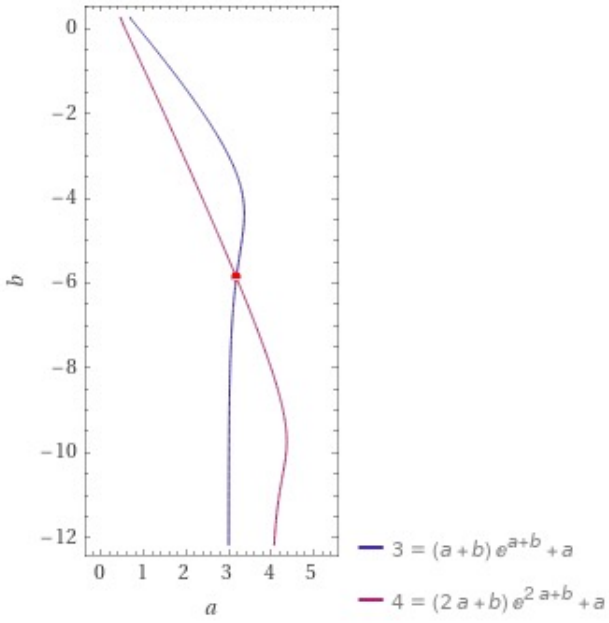
Input:

$$\left\{3 = a + (a \times 1 + b)\, e^{a \times 1 + b},\ 4 = a + (a \times 2 + b)\, e^{a \times 2 + b}\right\}$$

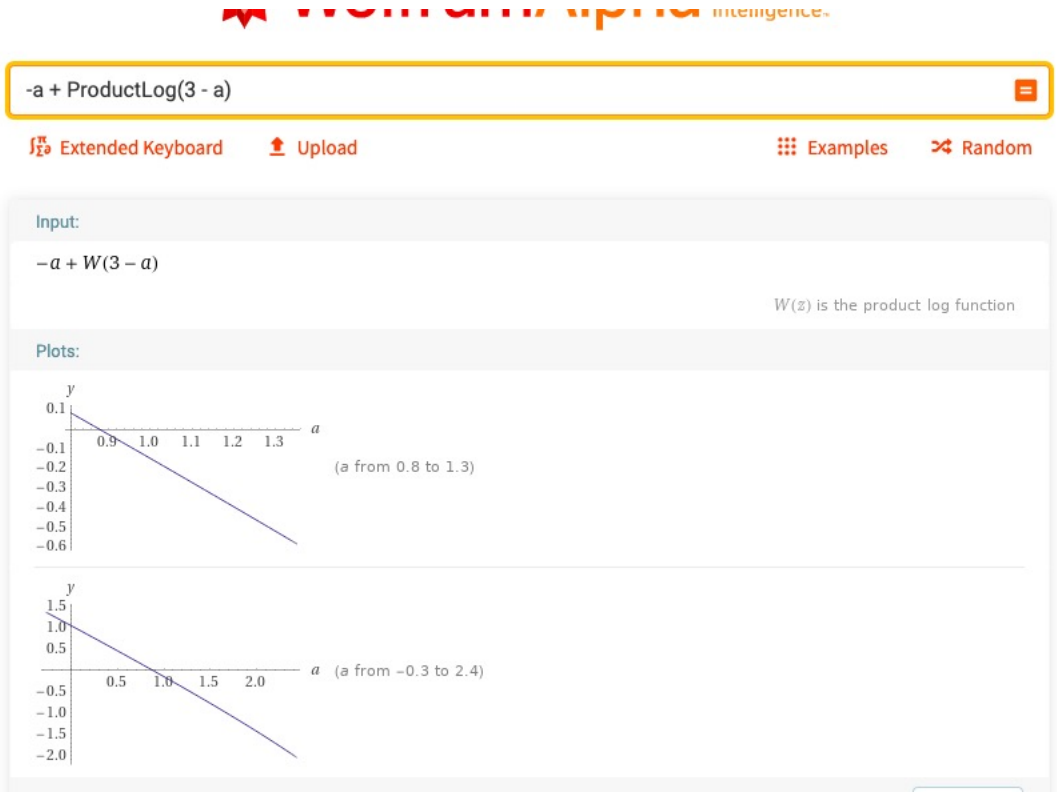
Exact result:

$$\left\{3 = a + (a + b)\, e^{a+b},\ 4 = a + (2a + b)\, e^{2a+b}\right\}$$

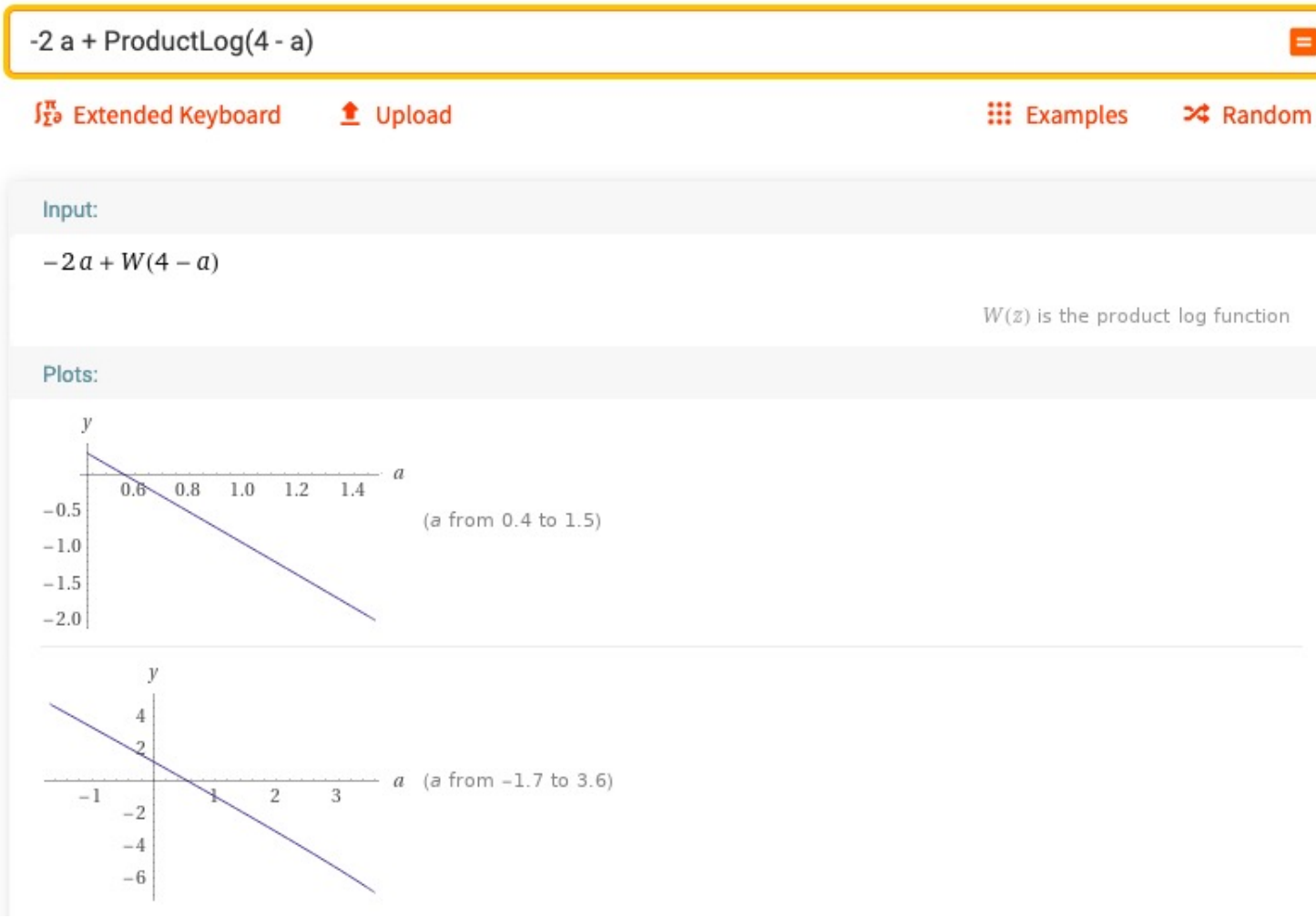
Plot of solution set:



Luego de ver la gráfica se entiende que finalmente el problema es de intersección entre dos funciones, entonces se maneja el método del ejercicio anterior. A la hroa de resolver el problema usando el metodo anterior, es necesario reformular las dos funciones que deben usarse. Haciendo uso de Wolfram se encuentra que la primera función a usar es $-a + W(3 - a)$ siendo W la función de Lambert.



Ahora para la segunda función, se realiza el mismo proceso usando Wolfram y da como resultado la función $-2a + W(4 - a)$



Una vez con las dos funciones listas, se realiza el procedimiento de usar la segunda ecuación del ejercicio anterior. Y los resultados son los siguientes. El error teórico sería el siguiente.

teo	0.157877179464472

Y el error que finalmente da es el siguiente, haciendo uso de tres iteraciones.

Resultado 0.15787718

Dicho ejercicio fue realizado con una tolerancia de $e^{-}8$

3.3. Tercer ejercicio

Teniendo la función $f(x) = e^x - x - 1$ se utilizará fórmula de Newton generalizada que es la sigueinte.

$$g(x) = x\frac{(x)}{\prime(x)} \tag{2}$$

Primero se pide demostrar que dicha funcion tiene un cero de multiplicidad 2 en x= 0. Para poder resolver esto, es necesario explicar que es la multiplicidad.

3.4. Multiplicidad

La multiplicidad indica cuántas raíces hay en un determinado punto. Lo que el enunciado realmente quiere expresar es que en el punto X=0 no existen dos raíces. Lo cual es cierto y se puede evidenciar en la gráfica y en el resultado de la utilización del método de Newton y Newton generalizado ya que la única raíz que encuentra es un valor cercano a 0 y eso se podrá observar a continuación en las tablas de resultados.

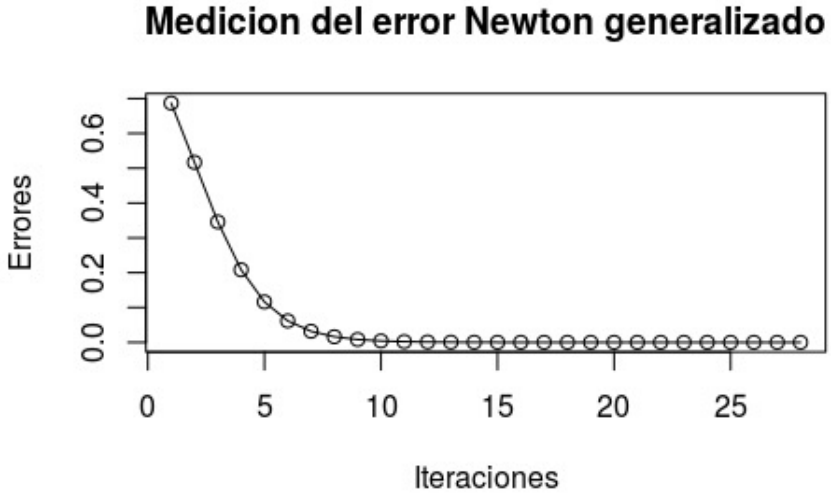
Primero, para evidenciar gráficamente que la afirmación es verdadera, se mostrará la gráfica de la función $f(x) = e^x - x - 1$ Utilizando como valor inicial 2 y máximo de iteraciones 29.

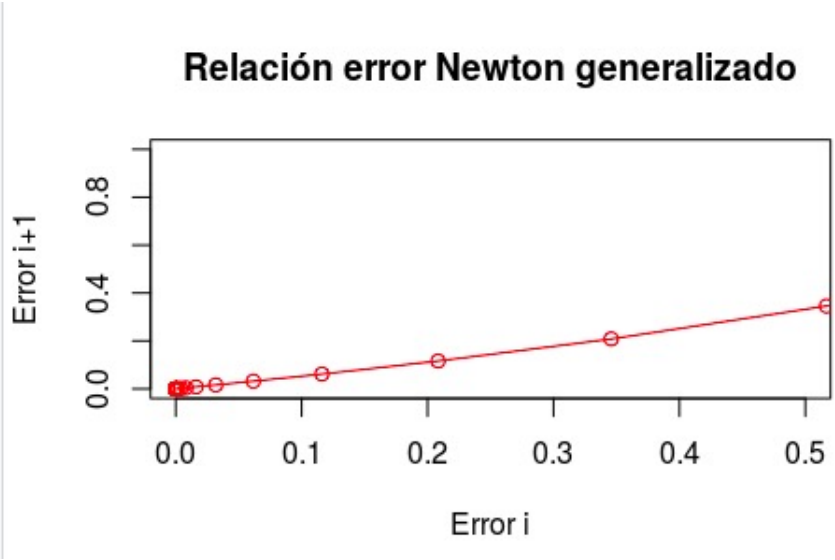
A continuación, los resultados luego de aplicar el método de Newton con la fórmula generalizada. El valor de la raíz calculada es de $1{,}848537046e-08$ con 29 iteraciones.

xn	f(xn)	Error estimado	Error anterior
1.313035285499331	1.404404811735363	0.686964714500669	0.000000000000000
0.796223594538966	0.420928638564356	0.516811690960366	0.686964714500669
0.450392870069002	0.118535579385603	0.345830724469964	0.516811690960366
0.242044035520060	0.031806250720908	0.208348834548941	0.345830724469964
0.125899366995155	0.008268660808180	0.116144668524905	0.208348834548941
0.064270222231248	0.002110297323536	0.061629144763907	0.116144668524905
0.032479309209032	0.000533209867719	0.031790913022216	0.061629144763907
0.016327561852855	0.000134023067785	0.016151747356177	0.031790913022216
0.008185996600725	0.000033596882257	0.008141565252130	0.016151747356177
0.004098582505826	0.000008410675969	0.004087414094899	0.008141565252130
0.002050691117366	0.000002104105074	0.002047891388460	0.004087414094899
0.001025696003140	0.000000526206039	0.001024995114226	0.002047891388460
0.000512935672600	0.000000131573997	0.000512760330540	0.001024995114226
0.000256489761567	0.000000032896311	0.000256445911033	0.000512760330540
0.000128250363524	0.000000008224429	0.000128239398043	0.000256445911033
0.000064126553251	0.000000002056151	0.000064123810273	0.000128239398043
0.000032063617683	0.000000000514043	0.000032062935568	0.000064123810273
0.000016031895484	0.000000000128511	0.000016031722199	0.000032062935568
0.000008015975363	0.000000000032128	0.000008015920121	0.000016031722199
0.000004008013049	0.000000000008032	0.000004007962314	0.000008015920121
0.000002004026709	0.000000000002008	0.000002003986339	0.000004007962314
0.000001002070328	0.000000000000502	0.000001001956382	0.000002003986339
0.000000501064973	0.000000000000126	0.000000501005355	0.000001001956382
0.000000250244777	0.000000000000031	0.000000250820196	0.000000501005355
0.000000125134132	0.000000000000008	0.000000125110645	0.000000250820196
0.000000061253836	0.000000000000002	0.000000063880295	0.000000125110645
0.000000032253909	0.000000000000000	0.000000028999927	0.000000063880295
0.000000018485370	0.000000000000000	0.000000013768539	0.000000028999927
0.000000018485370	0.000000000000000	0.000000000000000	0.000000013768539

Numero iteraciones = 29 Raíz = 1.848537046e-08 f(x) = 0 Error estimado = 0

Para poder observar la convergencia de este nuevo método a continuación ods gráficas del error.

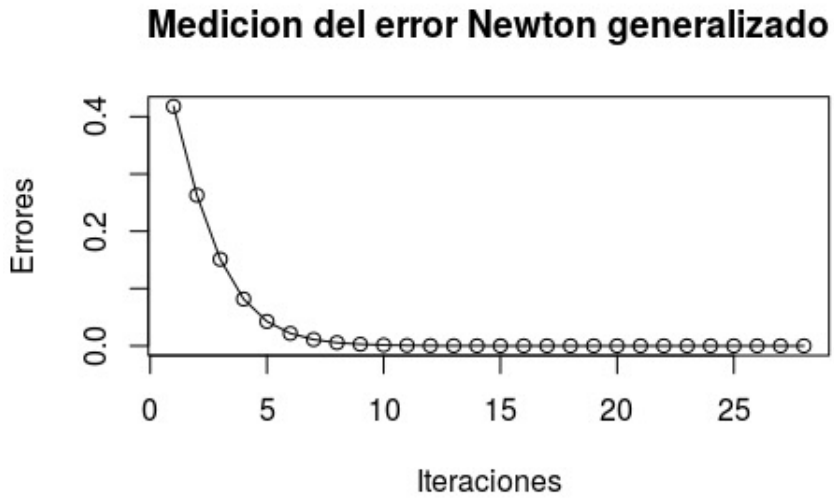


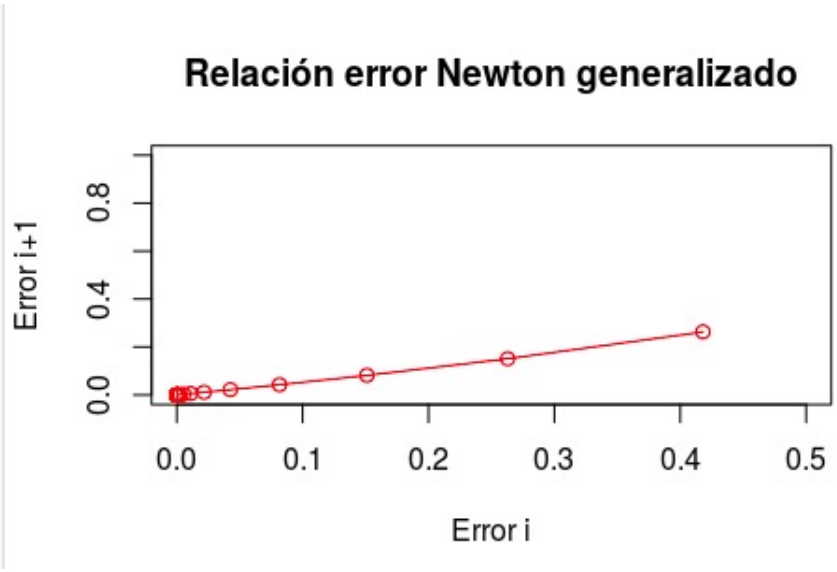


Para el siguiente ejercicio se utilizó como valor inicial 1 para reconocer la convergencia. Lo que se puede resaltar es que en este ejercicio al igual que el anterior, se acerca a la raíz en 0 pero esta vez por el lado negativo. sin embargo si se comparan las gráficas de error se comportan igual para ambos casos.

```
xn                  f(xn)              Error estimado       Error anterior
0.581976706869326  0.207595689972507  0.418023293130674  0.000000000000000
0.319055040910818  0.056772008685162  0.262921665958508  0.418023293130674
0.167996172885771  0.014935910537168  0.151058868025047  0.262921665958508
0.086348873747782  0.003837725704912  0.081647299137989  0.151058868025047
0.043795703673715  0.000973186964316  0.042553170074067  0.081647299137989
0.022057685365770  0.000245069312124  0.021738018307945  0.042553170074067
0.011069387477752  0.000061492354346  0.010988297888018  0.021738018307945
0.005544904662950  0.000015401437202  0.005524482814802  0.010988297888018
0.002775014494126  0.000003853916788  0.002769890168824  0.005524482814802
0.001388148972452  0.000000963924757  0.001386865521674  0.002769890168824
0.000694235066074  0.000000241036939  0.000693913906378  0.001386865521674
0.000347157696487  0.000000060266207  0.000347077369587  0.000693913906378
0.000173588891970  0.000000015067424  0.000173568804517  0.000347077369587
0.000086796957133  0.000000003766965  0.000086791934837  0.000173568804517
0.000043399105314  0.000000000941755  0.000043397851819  0.000086791934837
0.000021699712391  0.000000000235441  0.000021699392923  0.000043397851819
0.000010849891102  0.000000000058860  0.000010849821289  0.000021699392923
0.000005424958249  0.000000000014715  0.000005424932853  0.000010849821289
0.000002712480581  0.000000000003679  0.000002712477668  0.000005424932853
0.000001356302031  0.000000000000920  0.000001356178550  0.000002712477668
0.000000678038522  0.000000000000230  0.000000678263509  0.000001356178550
0.000000339095998  0.000000000000058  0.000000338942524  0.000000678263509
0.000000169499355  0.000000000000014  0.000000169596643  0.000000338942524
0.000000084349189  0.000000000000004  0.000000085150165  0.000000169596643
0.000000042230070  0.000000000000001  0.000000042119119  0.000000085150165
0.000000021198175  0.000000000000000  0.000000021031895  0.000000042119119
0.000000010723472  0.000000000000000  0.000000010474704  0.000000021031895
-0.000000009982939  0.000000000000000  0.000000020706410  0.000000010474704
-0.000000009982939  0.000000000000000  0.000000000000000  0.000000020706410
Numero iteraciones =  29  Raíz =  -9.982938826e-09  f(x) =  0  Error estimado =  0
```

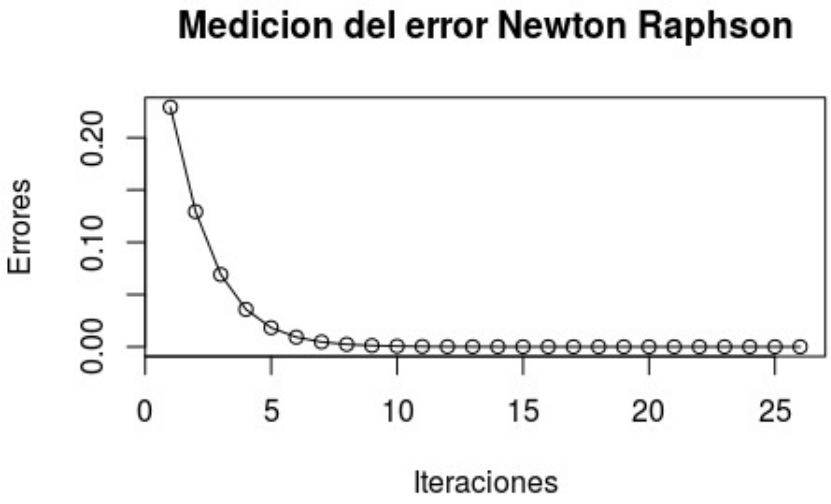
A continuación las gráficas de error

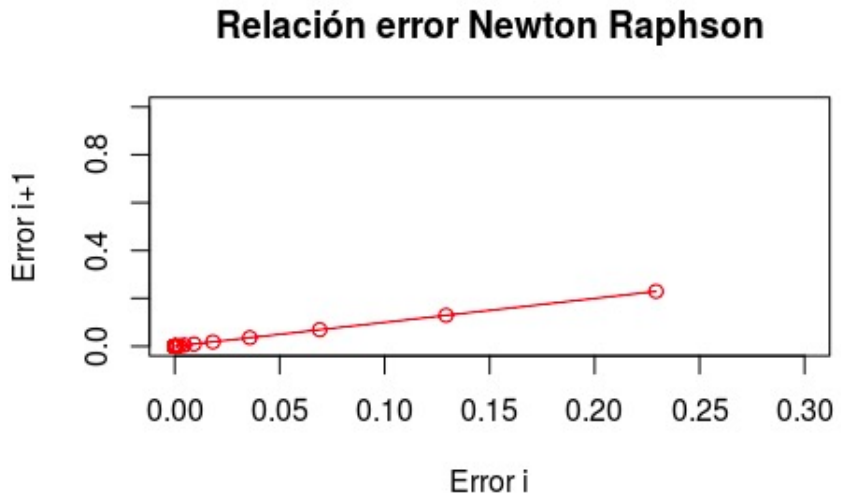




Ahora para saber si la convergencia mejora es necesario compararlo con el método de Newton Raphson. Por lo cual a continuación se mostrarán las gráficas para el método de Newton Raphson ya que en el primer ejercicio se mostraron las gráficas de Newton generalizado. Primero se evidenciarán los resultados con los mismos valores iniciales.

```
I= 1  X= 0.5  f(x)= 0.1487212707  E= 0.2292529587
I= 2  X= 0.2707470413  f(x)= 0.04019637259  E= 0.1292723074
I= 3  X= 0.1414747338  f(x)= 0.01049666416  E= 0.06906999803
I= 4  X= 0.07240473582  f(x)= 0.002685647788  E= 0.0357655356
I= 5  X= 0.03663920022  f(x)= 0.0006794887382  E= 0.01820773337
I= 6  X= 0.01843146686  f(x)= 0.0001709078983  E= 0.009187423675
I= 7  X= 0.009244043183  f(x)= 4.285812618e-05  E= 0.004614900574
I= 8  X= 0.004629142609  f(x)= 1.073103275e-05  E= 0.002312785559
I= 9  X= 0.002316357051  f(x)= 2.6848276e-06  E= 0.0011577314
I= 10  X= 0.001158625651  f(x)= 6.714660006e-07  E= 0.0005792009578
I= 11  X= 0.0005794246934  f(x)= 1.678989143e-07  E= 0.0002896843689
I= 12  X= 0.0002897403244  f(x)= 4.197878201e-08  E= 0.0001448631664
I= 13  X= 0.000144877158  f(x)= 1.049520226e-08  E= 7.243682989e-05
I= 14  X= 7.244032812e-05  f(x)= 2.623863793e-09  E= 3.621972676e-05
I= 15  X= 3.622060136e-05  f(x)= 6.559737198e-10  E= 1.811019135e-05
I= 16  X= 1.811041001e-05  f(x)= 1.639943736e-10  E= 9.055177671e-06
I= 17  X= 9.055232336e-06  f(x)= 4.09985379e-11  E= 4.527609335e-06
I= 18  X= 4.527623001e-06  f(x)= 1.024980101e-11  E= 2.263809792e-06
I= 19  X= 2.263813209e-06  f(x)= 2.562394741e-12  E= 1.131906177e-06
I= 20  X= 1.131907032e-06  f(x)= 6.405986852e-13  E= 5.65953409e-07
I= 21  X= 5.659536226e-07  f(x)= 1.600941602e-13  E= 2.829767846e-07
I= 22  X= 2.829768381e-07  f(x)= 4.019007349e-14  E= 1.414884123e-07
I= 23  X= 1.414884258e-07  f(x)= 1.021405183e-14  E= 7.074421115e-08
I= 24  X= 7.07442146e-08  f(x)= 2.442490654e-15  E= 3.537210683e-08
I= 25  X= 3.537210777e-08  f(x)= 6.661338148e-16  E= 1.768605373e-08
I= 26  X= 1.768605404e-08  f(x)= 2.220446049e-16  E= 8.843026942e-09
```





Con los resultados se observa que el método de Newton Rahpson requiere menos iteraciones para llegar al mismo resultado que requiere Newton generazilado. Además para Newton Raphson el error es más pequeño, se peude evidenciar en la gráfica de errori+1 y error1.

4. Convergencia acelerada

4.1. Teoría: Método de Δ^2 Aitken

Este método es un método de convergencia acelerada que permite que cualquier sucesión tenga una convergencia lineal, sin importar el origen de la sucesión. El nombre del método está en honor a su creador Alexander Aitken, el cual desarrolló este método en 1926.

Se supone una sucesión $\{x_n\}_{n=0}^{\infty}$ que converge linealmente a p se tiene en cuenta que la cantidad de n elementos es lo suficientemente grande para tener que $(x_n - p)(x_{n+1} - p > 0$. Entonces existe

una sucesión $\{\hat{X}_n\}_{n=0}^{\infty}$ definida de la siguiente manera:

$$\hat{X}_n = X_n - \frac{((\Delta X)_n)^2}{(\Delta^2 X)_n} \tag{3}$$

De tal manera que converge a p con mayor velocidad que $\{x_n\}_{n=0}^{\infty}$ al mostrarse de esta forma:

$$\lim_{n\to\infty} \frac{\hat{x}_n - p}{x_n - p} = 0 \tag{4}$$

Entonces de manera despejada la forma en la que se puede hallar una aproximación más acertada hacía la convergencia p se muestra finalmente de esta manera

$$\hat{P}_n = P_n - \frac{(p_{n+1} - p_n)^2}{p_{n+2} - 2p_{n+1} + p_n} \tag{5}$$

4.2. Solución

- **Verficar la convergencia de la sucesión $\{x_n\}_{n=0}^{\infty}$ con $x_n = cos(\frac{1}{n})$ y compararlo con la sucesión acelerada $\{A_n\}_{n=0}^{\infty}$**

  La sucesión $x_n = cos(\frac{1}{n})$ es una sucesión que converge linealmente a $p = 1$. Se usó los primeros $n = 15$ elementos de la sucesión y se comparó con la sucesión de convergencia acelerada. Se puede apreciar el comportamiento en la siguiente tabla.

n	$\{x_n\}_{n=0}^{\infty}$	$\{A_n\}_{n=0}^{\infty}$
1	0.5403023	0.9617751
2	0.8775826	0.9821294
3	0.9449569	0.9897855
4	0.9689124	0.9934156
5	0.9800666	0.9954099
6	0.9861432	0.99662
7	0.9898133	0.9974083
8	0.9921977	0.9979502
9	0.9938335	0.9983384
10	0.9950042	0.9986261
11	0.9958706	0.9988451
12	0.9965298	0.9990156
13	0.9970429	0.999151
14	0.9974501	
15	0.9977786	

Tabla de comparación de la convergencia de la sucesión $x_n = cos(\frac{1}{n})$

Se puede apreciar con claridad que la sucesión con convergencia presenta un comportamiento veloz a comparación de la sucesión inicial, aproximándose con aceleradamente a $p = 1$. Un

ejemplo claro de la aceleración es que en la aproximación $\hat{p} \approx 0{,}9974$ se llega en la séptima iteración de la sucesión acelerada mientras que en la sucesión original se tarda el doble.

- **Encontrar las coordenadas en dónde se intersectan las funciones paramétricas** $f(t) = 3sin^3(t) - 1$ **y** $g(t) = 4sin(t)cos(t)$

  Para hallar las coordenadas en dónde se intersectan las particulas al definir sus movimientos en en las funciones $f(t)$ y $g(t)$ se necesita igualar ambas, ya que así se puede determinar cuando ambas funciones pasan por un mismo punto

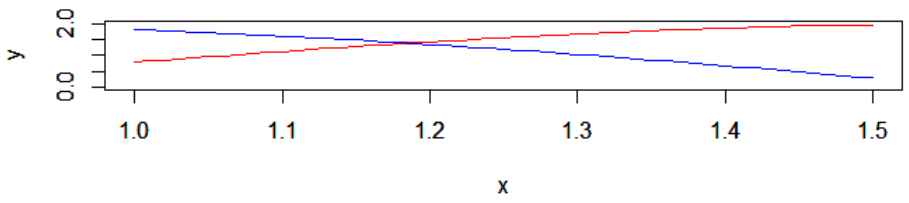
$$3sin^3(t) - 1 = 4sin(t)cos(t) \tag{6}$$

  Lo siguiente que se debe hacer es igualar a 0 la función $h(t)$ resultante y así obtener las raíces que permiten saber dónde intersectan las dos funciones. Habrán tantas raíces como puntos de intersección tengan las dos funciones.

$$h(t) = 3sin^3(t) - 1 - 4sin(t)cos(t) = 0 \tag{7}$$

  Se usa el método iterativo de Newton-Raphson para encontrar la raíces de la función h(t). Debido a la cantidad de intersecciones que tienen las dos funciones y la limitante del método de solo encontrar la raíz más cercana a un t inicial, se muestra una de las coordenadas dónde intersectan. El método encuentra la raíz $t_r = 1{,}18649502158199042122078800682107339616 5$ con una precisión extendida de 128 bits, con un $t_0 = 1$, una tolerancia $tol = 10^{-16}$, en 5 iteraciones.

  En ese sentido, al evaluar la raíz t_r encontrada previamente en cualquiera de las dos funciones, se puede encontrar la otra coordenada en dónde se intersectan las partículas. Entonces $f(t_r) = g(t_r) = 1{,}39026271679646745048396537917613613691 3$. En la siguiente gráfica se puede ver que ambas partículas coinciden en las coordenadas previamente halladas ($f(t)$ de color rojo y $h(t)$ de color azul.



Gráfica de las funciones $f(t)$ y $g(t)$ en dónde se intersectan

4.3. Teoría: Método de Steffensen

El método de Steffensen es un método iterativo para hallar las raíces de una función basándose en el método de punto fijo y usando el método de aceleración de convergencia de de Δ^2 Aitken. Fue

creado por Johan Frederik Steffensen y dado que usó la combinación de ambos métodos, este se le denomina como el método de punto fijo acelerado.

Este método converge cuadráticamente y al igual que el método de Newton, este necesita un x_0 lo suficientemente cerca para que no falle el método. El método consiste en realizar una doble evaluación de la función $f(x)$ en su forma $x = g(x)$ como se sugiere en el método del punto fijo, de tal forma que se vaya obteniendo los elementos de una sucesión que posteriormente se acelera con Aitken. Por tanto, el cálculo de cada aproximación de la raíz se obtiene de la siguiente manera:

$$x_{n+1} = x_n - \frac{[f(x_n)]^2}{f(x_n + f(x_n)) - f(x_n)} \tag{8}$$

4.4. Solución: Resolver $f(x) = x^2 - cos(x)$ con el Método de Aitken y método de Steffensen

Para la solución de la función se despejo de la forma $x = g(x)$ por tanto la función resultante es $g(x) = \sqrt{cos(x)}$. Adicionalmente, se utilizó una precisión extendida de 128 bits para encontrar las raíces en pocas iteraciones para las diferentes tolerancias. Se partió de un $x_0 = 5$ para encontrar únicamente la raíz positiva.

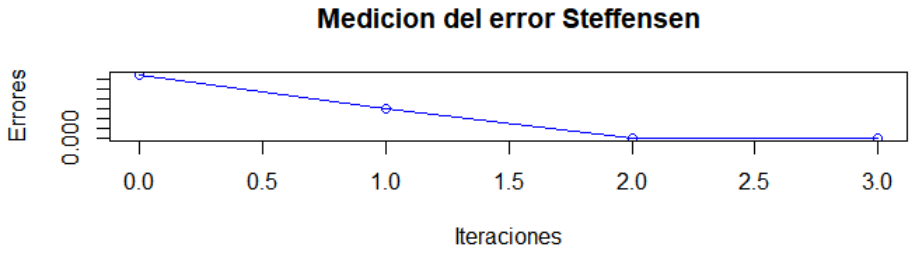
- **Para una tolerancia de** 10^{-8}

  Usando el método de Aitken se obtuvo la raíz $x_r = 0{,}82413231241263542681981328836173421366 99$ con unas 9 iteraciones. En la gráfica de relación de error con el número de iteraciones se puede apreciar su comportamiento.

  Aitken8.png

  Gráfica de Iteraciones vs. Error método de Aitken para una tolerancia de 10^{-8}

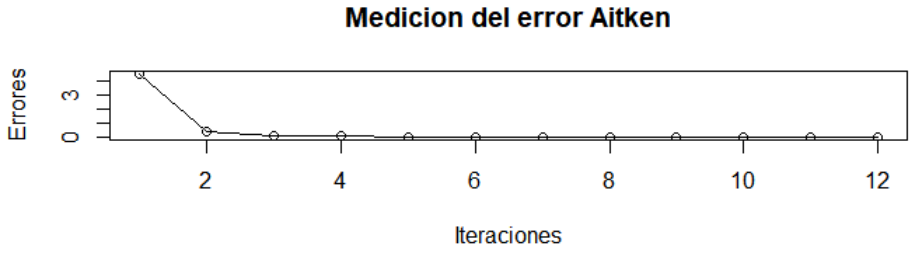
  Usando el método de Steffensen se puede apreciar una mejora bastante notable al encontrar la raíz $x_r = 0{,}82413231230252233038817981319735862959 5$ en sólo 3 iteraciones, tres veces menos iteraciones que el método de Aitken. En la gráfica de relación de error con el número de iteraciones se puede apreciar su comportamiento.



Gráfica de Iteraciones vs. Error método de Steffensen para una tolerancia de 10^{-8}

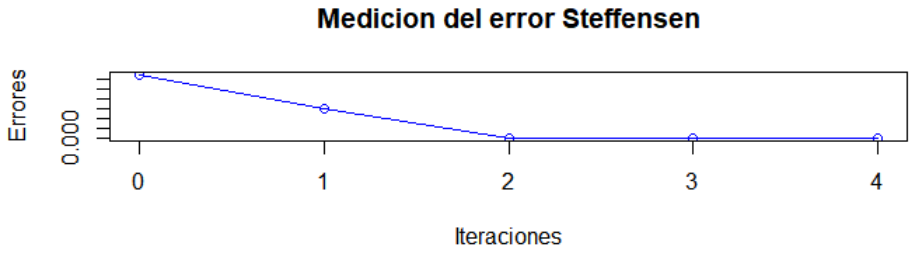
- **Para una tolerancia de** 10^{-16}

  Usando el método de Aitken se obtuvo la raíz $x_r = 0{,}824132312302522444751438706787960352193 7$ con unas 12 iteraciones. En la gráfica de relación de error con el número de iteraciones se puede apreciar su comportamiento.



Gráfica de Iteraciones vs. Error método de Aitken para una tolerancia de 10^{-16}

Usando el método de Steffensen se puede apreciar una mejora bastante notable al encontrar la raíz $x_r = 0{,}824132312302522422960956785771990246522 3$ en sólo 4 iteraciones, tres veces menos iteraciones que el método de Aitken. En la gráfica de relación de error con el número de iteraciones se puede apreciar su comportamiento.



Gráfica de Iteraciones vs. Error método de Steffensen para una tolerancia de 10^{-16}

Se puede concluir que el método de Steffensen es bastante eficiente a comparación del otro método ya que tarda tres iteraciones menos en llegar a al raíz aproximada, además, cuando la tolerancia se duplica el método solo le toma una iteración más llegar a un resultado aún más preciso, mientras que el método de Aitken triplica el número de iteraciones. Otra característica que cabe resaltar es que Steffensen presenta errores muy bajos desde sus primeras iteraciones y así se puede comprobar su convergencia.

5. Bibliografía

- Burden, R. L., & Faires, J. D. (1997). Numerical Analysis, Brooks. Cole, Belmont, CA
- Gautschi, W. (1997). Numerical analysis. Springer Science Business Media