Taller 1 Análisis Numérico

Daniel Hernández;
Juan Carlos Suárez;
Santiago Andrés Caroprese;

September 2020

1. 1. Número de Operaciones

1.1. Evaluar un polinomio

Ejercicio: Evaluar el polinomio en x=1.00000000001 con P(x)=1+x+x^2+...+x^{50} y la primera derivada. Encuentre el error de cálculo al compáralo con los resultados de la expresión equivalente Q(x)=(x^{51}-1)/(x-1).

Evaluación del Polinomio: 51.000000012750000002082500000249899998 84
Evaluación de Expresión Equivalente: 51.000000012750000002082500000379447829 27
Error: 1.29547830429433478487403391606226051995 6e-28

Para realizar la evaluación del polinomio se realizó un método en R el cual va acumulando las sumas y multiplicaciones de x. Ya que todos los coeficientes de P(x) son 1, se tiene el coeficiente 1 para x_0 que se usa como inicio y cada x_n se puede calcular de la siguiente manera:

$$x_n = x_{n-1} * x + 1 \tag{1}$$

1.2. Números Binarios

Ejercicio 1:Encuentre los primeros 15 bits en la representación binaria de pi. Se usó pi como 3.14159265358979.
En binario:11.00100100001111110110101010001000100001011010001000 10
Para este calculo el método separa la parte entera de la parte decimal y se calcula el equivalente en decimal por separado.

Ejercicio 2: Convertir los siguientes números binarios a base 10: 1010101; 1011.101; 10111.010101...; 111.1111 Para los binarios infinitos se realizo el calculo realizando un recorte en la cantidad de dígitos.

Ejercicio 3: Convierta los siguientes números de base 10 a binaria: 11.25; 2/3; 30.6; 99.9

Binario	Decimal
1010101	85
1011.101	11.625
10111.010101010101010...	23. 33331298828125...
111.111111111111111...	7. 999969482421875...

Tabla 1: Conversión Binario-Decimal

Decimal	Binario
11.25	1011.010
2/3	0.10101010101010101010101010101010...
30.6	11110.10011001100110011001100110011001...
99.9	1100011.11111111111111111111111111111111...

Tabla 2: Conversión Decimal-Binario

1.3. Punto Flotante

a. ¿Como se ajusta un número infinito en un número finito de bits?
Ya que la memoria de las máquinas no es infinita, para poder almacenar un número infinito en un número finito de bits es necesario utilizar redondeo o recorte para para obtener un número finito y luego se utiliza el método de punto flotante. El punto flotante descompone un número en tres partes: signo (s), mantisa (m) y exponente (E). El signo indica si el n+umero es positivo o negativo, el exponente representa el corrimiento que se realiza para normalizar el número, y la la mantisa corresponde al valor del número. Cuando el exponente es negativo representa un número menor a uno. En esta representación existen dos maneras de definir la precisión. Con precisión simple se tiene 32 bits: 1 para el signo, 23 para la mantisa y 8 para el exponente. Con precisión doble se tienen 64 bits para representar el número: 1 para el signo, 52 par la mantisa y 11 para el exponente. [2]
El calculo se realiza de la siguiente manera:
1. Definir signo: 0 si es positivo, 1 si es negativo.
2. Calcular el número en binario.
3. Normalizar número binario (mantisa).
4. Calcular exponente. Se cuentan cuantas posiciones se movió el punto para normalizar el número. Si se mueve a la izquierda es positivo, si se mueve a la derecha es negativo. A este valor es necesario sumarle el estándar de precisión (127 para simple y 1023 para doble).
5. Pasar el exponente a binario. Si hay bits disponibles se completan con 0.
6. Obtener mantisa. Es el número normalizado pero ignorando el punto. Si hay bits disponibles se completan con 0, si hay más dígitos de los que caben se hace un recorte. 7. Juntar resultado signo; exponente; mantisa.

b. ¿Cuál es la diferencia entre redondeo y recorte?
Recorte consiste en cortar el número a partir de ciertas cifras y simplemente ignorar lo que no se requiera. Redondeo igualmente corta el número a partir de ciertas cifras, pero el valor la última cifra que se va a almacenar depende de la parte que se va a eliminar. Al redondear se tiene en cuenta el valor que le sigue a la última cifra que se va a almacenar. Si es mayor a 5 se le suma 1 a la última cifra, si es menor a 5 se mantiene igual, pero si es 5 existen diferentes formas de hacer el redondeo. La más popular es aumentar en 1, pero hay otras como revisar el siguiente dígito después del 5.

c. Indique el número de punto flotante (IEEE) de precisión doble asociado a x, el cual se denota como fl(x) para x(0.4).
1. Pasar a binario: 0.4=0.0110011001100110011001100110011001100110011001100...
2. Normalizar: 1.1001100110011001100110011001100110011001100110011001100...
3. -2 posiciones+1023=1021
4. Pasar a binario: 1021=10000000010
5. Signo s=0
6. Exponente E=01111111101
7. Mantisa m=1001100110011001100110011001100110011001100110011001100110011001
8. Punto Flotante: 0;01111111101;1100110011001100110011001100110011001100110011001100110011001100
9. A decimal:0.40000000000000002

d. Error de redondeo: En el modelo de la aritmética de computadora IEEE, el error de redondeo relativo no es más de la mitad de la épsilon de máquina:

$$\frac{|fl(x) - x|}{|x|} <= \frac{e_{\text{maq}}}{2} \tag{2}$$

Teniendo en cuenta lo anterior, encuentre el error de redondeo para x = 0.4.

$$\frac{|0{,}0{,}40000000000000002 - 0{,}4|}{0{,}4} <= 1{,}11e^{-16} \tag{3}$$

$$2{,}000000000000000000000000e^{-17} <= 1{,}11e^{-16} \tag{4}$$

Nota: Debido a la precisión necesaria se utilizo la libreria mpfr de R.

e. Verificar si el tipo de datos básico de R y Python es de precisión doble IEEE y revisar en R y Phython el format long.
Python utiliza precisión doble en punto flotante [5]. Python tiene integers que están implementados usando el long de C.[3] R también utiliza precisión doble con punto flotante, y el format long esta relacionado con la manera de visualizar las tablas. [1]

	WolframAlpha Normal
x1	$1.062211848441644930860*10^{-11}$
x2	$-2.824295364810000000000*10^{11}$

Tabla 3: Resultados teóricos x1 y x2

f. Encuentre la representación en número de máquina hexadecimal del número real 9.4.
1. Calcular en binario: 9.4= 1001.0110011001100110011001100110011001100110011001100110011...
2. Normalizar: 1.0010110011001100110011001100110011001100110011001100110011...
3. 3 (posiciones) + 1023 =1026
4. Signo s=0
5. Exponente E=10000000010
6. Mantisa m=0010011001100110011001100110011001100110011001100110011001100110011
7. Punto Flotante: 0;10000000010;0010011001100110011001100110011001100110011001100110011001100110011
8. Agrupar por 4: (0100)(0000)(0010)(1001)(0000)(0000)(0000)(0000)(0000)(0000)(0000)
(0000)(0000)(0000)(0000)(0000)
9. Hexadecimal: 4029000000000000

g. Encuentre las dos raíces de la ecuación cuadrática $x^2+(9^{12})*x=3$. Intente resolver el problema usando la aritmética de precisión doble, tenga en cuenta la perdida de significancia y debe contrarrestarla.
Para este problema se realizaron 3 cálculos: Para el primero se utilizó la formula tradicional para encontrar las raíces de un polinomio de grado 2 de la manera $P(X)=ax^2+bx+c$.

$$x_1, x_2 = \frac{-b \pm \sqrt{b^2 - 4 * a * c}}{2 * a} \tag{5}$$

Ha este calculo se le llamó calculo .Expresión Normal". Para el segundo método se racionalizó la misma expresión, multiplicándolo por el conjugado. A este se le llamo .Expresión Racionalizada"

$$\frac{-b + \sqrt{b^2 - 4 * a * c}}{2 * a} * \frac{-b - \sqrt{b^2 - 4 * a * c}}{-b - \sqrt{b^2 - 4 * a * c}} \tag{6}$$

$$\frac{4 * a * c}{-2 * a(b + \sqrt{b^2 - 4 * a * c})} \tag{7}$$

Para el último cálculo se utilizó la excreción normal, pero a través de la herramienta WolframAlpha, y se tomo el resultado como valor teórico para realizar la comparación.

h. Explique cómo calcular con mayor exactitud las raíces de la ecuación: $x^2+b*x-10^{-12} = 0$ Donde b es un número mayor que 100.
Como se demostró en el ejercicio anterior, una de las maneras de contrarrestar la perdida de significancia consiste en encontrar una expresión equivalente para realizar el cálculo. En este caso se puede hacer racionalizando la ecuación para

	Expresión Normal	Expresión Racionalizada
x1	0	$1.062211848441645e^{-11}$
x2	-282429536481	-282429536481

Tabla 4: Resultados prácticos x1 y x2

	Expresión Normal	Expresión Racionalizada
Error x1	$1.0622118484416448e^{-11}$	$1.6155871338926322e^{-27}$
Error x2	0	0

Tabla 5: Error de x1 y x2

calcular las raíces de un polinomio de grado 2. Hay que tener en cuenta el resultado de este cálculo (7), ya que con ciertos valores muy cercanos entre sí, la expresión del denominador puede ser reducida a 0 por perdida de significancia. Para ello, antes de realizar la división es necesario verificar que a no sea 0 y que:

$$b \neq \sqrt{b^2 - 4 * a * c} \tag{8}$$

En caso de que la anterior condición no se cumpla, se calcula la raíz correspondiente con la ecuación original.

2. Raíces de Ecuación

2.1. Punto 1

Implemente en R o Python un algoritmo que le permita sumar únicamente los elementos de la submatriz triangular superior o triangular inferior, dada la matriz cuadrada An. Imprima varias pruebas,para diferentes valores de n, y exprese f(n) en notación O() con una gráfica que muestre su orden de convergencia.

Para realizar las pruebas, se utilizaron matrices con secuencia de números desde 1, hasta nxn. Como se puede ver en los resultados de la tabla 6, las sumas hechas en la submatriz inferior son casi el doble de las correspondientes a la superior, lo cual tiene sentido debido a la estructura de números secuenciales de la matriz.

En cuanto a la convergencia del método, se encontró que está dado por la ecuación 9. Y se muestra su comportamiento cuadrático en la siguiente gráfica.

$$O(n^2/2 - n/2) \tag{9}$$

N	Superior	Inferior
3	11	19
4	36	66
5	90	170
6	190	365
7	357	693
8	616	1204
9	996	1956
10	1530	3015
11	2255	4455
12	3212	6358

Tabla 6: Resultados para $N = 3$ hasta 12

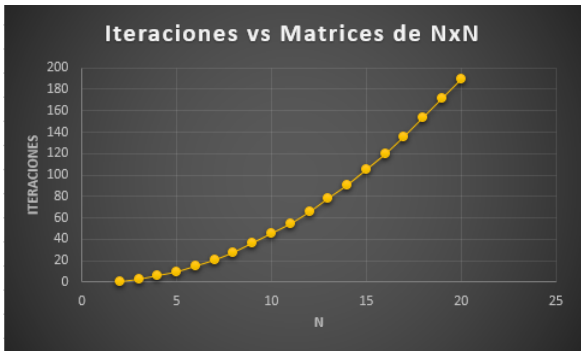


Figura 1: Iteraciones necesarias para cada matriz de NxN

2.2. Punto 2

Implemente en R o Python un algoritmo que le permita sumar los n^2 primeros números naturales al cuadrado. Imprima varias pruebas, para diferentes valores de n y exprese f(n) en notación O() con una gráfica que muestre su orden de convergencia.

El algoritmo implementado es bastante simple, se hace un ciclo que va desde i=1, hasta n^2 iteraciones. En cada iteración se calcula i^2 y este es añadido al valor total.

En la tabla 7, se muestra el considerable aumento de los resultados de las sumas a medida que se aumenta N.

N	Suma
1	1
2	30
3	285
4	1496
5	5525
6	16206
7	40425
8	89440
9	180441
10	338350

Tabla 7: Resultados de las sumas con diferentes N

El que se sumen los n^2 primeros números, implica que el orden de convergencia del método es cuadrático, y esta expresado por la ecuación 10. Comportamiento que se puede apreciar en la figura 2

$$O(n^2) \tag{10}$$

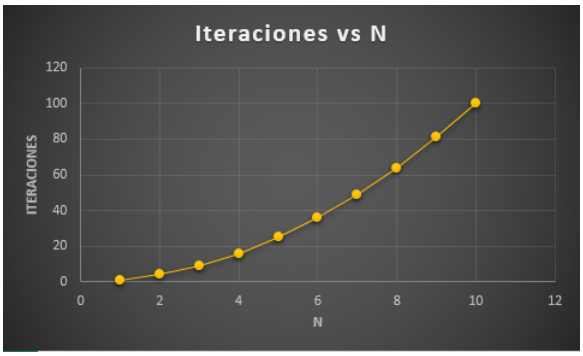


Figura 2: Iteraciones necesarias vs N

2.3. Punto 3

Hallar la altura máxima de un cohete de acuerdo a la ecuación 11, que expresa su trayectoria

$$y(t) = 6 + 2{,}13t^2 - 0{,}0013t^4 \tag{11}$$

Los puntos críticos de una función indican donde tiene esta sus valores máximos y mínimos. Así que si se quiere encontrar la altura máxima del cohete durante su trayectoria, se debe hallar un punto crítico p, de forma que y(p) sea mayor o igual que y(t) evaluada en los otros puntos críticos. Para este problema, se implemento un programa en R que primero encontraba los puntos críticos de la función. Los puntos críticos de una función, son las raíces de su derivada, por lo cual primero se derivo la expresión, y se hallaron todas sus raíces usando la función üniroot.all"de la librería rootsolve. Posteriormente, se evaluó y(t) en cada punto crítico, y estos fueron comparados entre sí, hasta encontrar el mayor valor que tomaba y(t). Se encontró que cuando t = 28.622207623290855, el cohete alcanza una altura máxima de 878.48076923.

3. Convergencia de Métodos Iterativos

3.1. Sección 1

1. Sean f(x)=ln(x+2) y g(x)=sin(x) dos funciones de valor real.

a) Utilice la siguiente formula recursiva con E=10^{-16} para determinar aproximadamente el punto de intersección.

$$x_n = x_{\text{n-1}} - \frac{f(x_{\text{n-1}}) * (x_{\text{n-1}} - x_{\text{n-2}})}{f(x_{\text{n-1}}) - f(x_{\text{n-2}})} \tag{12}$$

Al aplicar la anterior formula se obtiene que g(x) y f(x) tienen un punto de intersección en x=-1.631443596968884822896061383153441574973.

Método	Resultado	Error
1	-1.63144359696888482289	-7.68059498348977782098
	6061383153441574973	4766962663635416641e-17
2	-1.63144359696888486902	-1.22930274218867656433
	0385767123319798362	2361769411113010006e-16

Tabla 8: Resultados punto de intersección con (13)

b) Aplicar el método iterativo siguiente con E = 10^{-8} para encontrar el punto de intersección.

$$x_{n+1} = x_n - f(x_n) * \frac{(x_n - x_{n-1})}{f(x_n) - f(x_{n-1})} \tag{13}$$

Al analizar el problema el equipo de trabajo se dio cuenta que si la ecuación (12) se reescribe de tal manera que el índice de x no inicie en n-2 sino en n-1, es decir, x_n se convierte en x_{n+1}, x_{n-1} se convierte en x_n y x_{n-2} se reescribe como x_{n-1}, la ecuación (12) y (13) son equivalentes. A pesar de ello se realizo una modificación al código, de tal manera que la evaluación de la división ocurra previamente a la multiplicación por $f(x_n)$. Con ello se obtuvo que f(x) y g(x) tienen un punto de intersección en x=-1.631443596968884869020385767123319798362.

Luego se comparó los dos resultados con los resultados teóricos calculados en WolframAlpha (x=-1.631443596968884822896061387697825694063) y se obtuvo lo siguiente: Al observar los resultados se puede evidenciar que, aunque teóricamente el método es el mismo, al momento de realizar la evaluación de la división antes que la multiplicación se genera un cambio sutil en el resultado., haciendo que el error sea mayor. Esto seguramente se debe a la perdida de signifícanos que se en cada evaluación que se realiza.

2. Newton: Determine el valor de los coeficientes a y b tal que f(1)=3 y f(2)=4 con f(x)=a+(ax+b)e^{ax+b}. Obtenga la respuesta con E=10^{-8}. Lo primero que hay que notar es que se tiene un sistema de ecuaciones con dos variables y dos ecuaciones, y que es un sistema de ecuaciones no lineales. Teniendo eso en cuenta, lo primero que hay que hacer es definir esas dos ecuaciones, las cuales se consiguen reemplazando x y por los valores correspondientes:

$$3 = a + (a + b) * e^{a+b} \tag{14}$$

$$4 = a + (2a + b) * e^{2a+b} \tag{15}$$

Como es un sistema de ecuaciones no lineales y tenemos la multiplicación (a+b)*e^{a+b} es necesario buscar la manera de realizar una sustitución para simplificar las expresiones. El objetivo es relacionar ambas ecuaciones y tenerlas en función de una sola variable, para poder utilizar newton. para ello se utilizara:

$$c = a + b \tag{16}$$

$$a = c - b \tag{17}$$

Utilizando 16 y 17 en 14 y despejando b:

$$3 = (c - b) + c * e^{c} \tag{18}$$

$$b = c + c * e^{c} - 3 \tag{19}$$

Luego se utiliza (16) en (15), se simplifica y luego se utiliza (19):

$$4 = (c - b) + (2(c - b) + b) * e^{2(c-b)+b} \tag{20}$$

$$4 = (c - b) + (2c - b) * e^{2c-b} \tag{21}$$

$$4 = (c - (c + c * e^{c} - 3)) + (2c - (c + c * e^{c} - 3)) * e^{2c-(c+c*e^{c}-3)} \tag{22}$$

En este punto ya se tiene una ecuación en función de una sola variable:

$$4 = -c * e^{c} + 3 + (c - c * e^{c} + 3) * e^{c-c*e^{c}+3} \tag{23}$$

$$0 = -c * e^{c} + (c - c * e^{c} + 3) * e^{c-c*e^{c}+3} - 1 \tag{24}$$

La ecuación dada satisface las condiciones iniciales y es posible utilizarla con Newton para determinar c, pero antes se debe comprobar que su derivada con respecto a c es diferente de cero:

$$0 = (c * (c + 1) * e^{2c} - (c^2 + 6c + 4) * e^{c} + c + 4) * e^{-ce^{c}+c+3} + (-c - 1) * e^{c} \tag{25}$$

Después de aplicar newton se obtiene que c=1.022399211529483625450805118848162450114. Con este valores se puede calcular b en (18) y con b y c se calcula a en (15), lo que da b=0.864522032065014462995618647333491494089 4 y a=0.157877179464469162455186471514670956024 5. Una vez obtenido los valores de a y b se reemplazan en f(x), y para comprobar los resultados se calcula nuevamente f(1) y f(2):

$$f(1) = 3{,}000000000000000000000000000000000000000000 \tag{26}$$

$$f(2) = 4{,}000000000000000000002765162956017394437 \tag{27}$$

Dada la precisión de R, se asume que el error con respecto a f(1) es de 0, mientras que el error respecto a f(2) es de 2.765162956017394436652935071618337150486e-21, lo que es considerablemente menor a 10^{-8}, por lo que se considera que la respuesta obtenida es aceptable bajo los parámetros establecidos.

3.2. Múltiples soluciones

Sea f(x)=e^{x}-x-1:

a. Demuestre que tiene un cero de multiplicidad 2 en x=0. Se dice que una solución p de la ecuación f(x)=0 es un cero de multiplicidad m de f si f(x) puede

escribirse como f(x)=(x-p)^{m}*q(x). Si partimos de la base que la raíz de f(x) es 0 y que queremos comprobar su multiplicidad 2, se puede reescribir como:

$$f(x) = (x-0)^2 * q(x) \tag{28}$$

Además se sabe que f(x)=e^{x}-x-1, por lo tanto:

$$e^{x} - x - 1 = x^2 * q(x) \tag{29}$$

$$\frac{e^{x} - x - 1}{x^2} = q(x) \tag{30}$$

De ese modo, si se reescribe (28), reemplazando q(x) por el resultado obtenido en (30):

$$f(x) = x^2 * \frac{e^{x} - x - 1}{x^2} \tag{31}$$

Ya que se pudo reescribir f(x), se puede afirmar que para x=0, f(x) posee una raíz de multiplicidad 2 para todo x$\neq p$, donde limite$_{x\text{-}p}$ q(x)$\neq 0$.

b. Utilizando el método de Newton con p_0=1 verifique que converge a cero pero no de forma cuadrática. Para realizar el análisis, se guardaron los valores correspondientes al error en cada iteración, de la manera:

$$error = |valoractual - valoranterior| \tag{32}$$

Los resultados obtenidos se observan en la tabla 9.

Con estos datos, se realizo una gráfica que relacionara los errores en la iteración n, con los errores en la iteración n+1. De esa manera, en la siguiente gráfica se observa en el eje y el error en la iteración n+1, y en el eje x el error en la iteración n.

Con los puntos correspondientes se realizó una regresión lineal y una regresión exponencial. El resultado correspondiente a R indica que la regresión lineal es mas acertada ya que $R^2{}_{\text{lineal}}$ esta más cerca a 1 que $R^2{}_{\text{exponencial}}$. Por lo tanto no converge de forma cuadrática, con $R^2{}_{\text{lineal}}$.

c. Utilizando el método de Newton generalizado, mejora la tasa de rendimiento? Explique su respuesta.

Para comparar los dos métodos se tuvo en cuenta número de iteraciones, resultado y convergencia.

En la tabla 10 se puede observar que newton generalizado utiliza 2 iteraciones menos que newton para llegar al resultado, aunque el resultado de Newton es mas cercano a 0.

Al realizar las gráficas del error en la iteración n en relación con el error en la iteración n+1, se observa que Newton generalizado presenta también una convergencia lineal, aunque el R^2 de Newton es mas cercano a 1 que el R^2 de

Iteraciones	Error
1	0.41802329313067355
2	0.26292166595850802
3	0.15105886802504795
4	0.081647299137989113
5	0.042553170074067288
6	0.021738018307945847
7	0.010988297888028936
8	0.0055244828148080707
9	0.0027698901687939711
10	0.0013868655217479909
11	0.00069391390640960514
12	0.00034707736931453078
13	0.00017356880506783995
14	8.6791935023070596e-05
15	4.3397849540299617e-05
16	2.1699397179760574e-05
17	1.0849822556837258e-05
18	5.4249347556939693e-06
19	2.7124805213663479e-06
20	1.3561828300359917e-06
21	6.7810621549089107e-07
22	3.3919774053129604e-07
23	1.6965205894818476e-07
24	8.5233729703832846e-08
25	4.224419676155565e-08
26	2.1220239416608511e-08
27	0

Tabla 9: Resultados y error con p_1=1

	Iteraciones	Resultado
Newton	52	6.3154925280443565307911717686 2e-16
Newton Generalizado	50	8.2647083418937860969621264872 6e-16

Tabla 10: Comparación Newton y Newton Generalizado

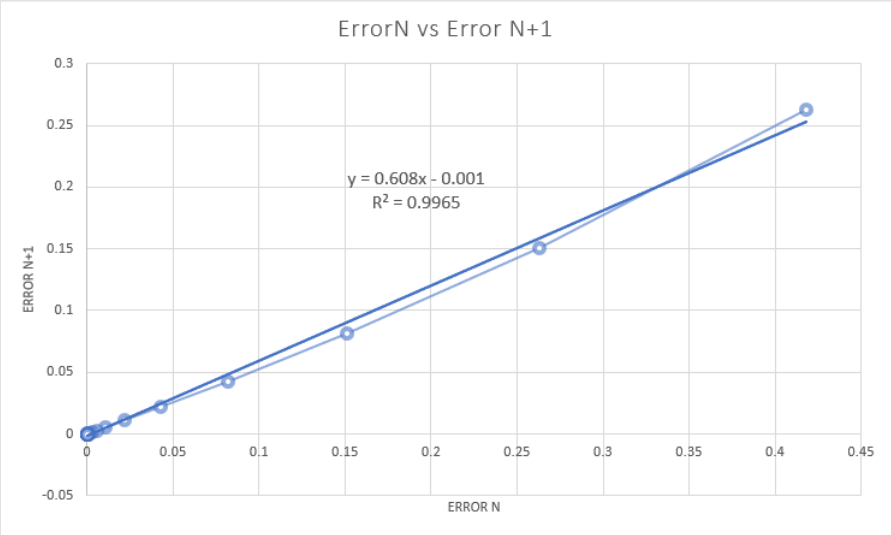


Figura 3: Newton: Error_n vs Error_{n+1}

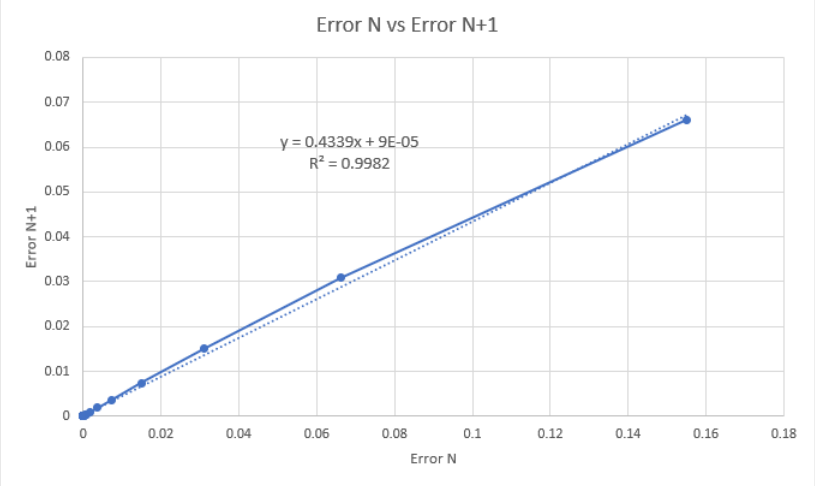


Figura 4: Newton generalizado: Error_n vs Error_{n+1}

Newton generalizado ambos presentan una convergencia lineal, pero, ya que el número de iteraciones es ligeramente mayor y el resultado posee un mayor error con respecto al valor teórico, se puede decir que Newton tiene un mejor tasa de rendimiento.

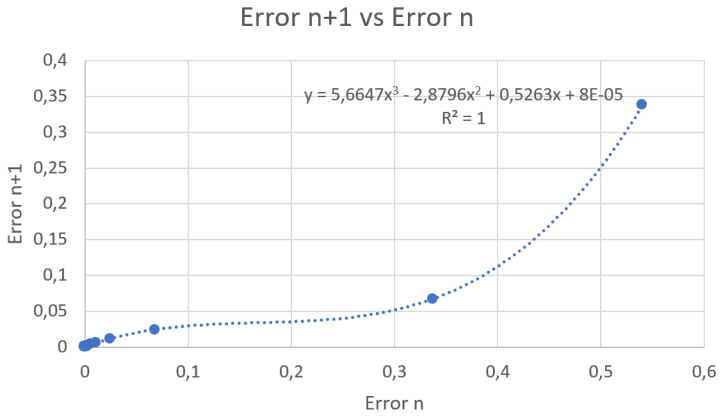
4. Convergencia Acelerada

4.1. Método de Aitken

Dada la sucesión $\{x_n\}_{n=0}^{\infty} = cos(1/n)$

1. Verifique el tipo de convergencia en x = 1 independiente del origen.

Se realizó en R una función que calcula los valores de la sucesión indicada y el error local para cada uno. A partir de los valores obtenidos, se construyó la gráfica que se muestra a continuación, la cual ilustra la relación entre el valor del error obtenido en dos iteraciones consecutivas. A partir de esta gráfica, se encontró que la mejor función de ajuste es cúbica, por lo que se puede concluir que la convergencia de la sucesión es de orden cúbico.



2. Compare los primeros términos con la sucesión $\{A_n\}_{n=0}^{\infty}$

A continuación, se muestra la tabla en la que se muestran los 20 primeros términos de la sucesión original y la acelerada. Como se puede observar, ambas sucesiones convergen hacia 1, pero la sucesión acelerada por Aitken lo hace más rápidamente. Se puede evidenciar que la sucesión acelerada obtiene en sus primeros términos valores cercanos a los que obtiene la sucesión original en términos posteriores.

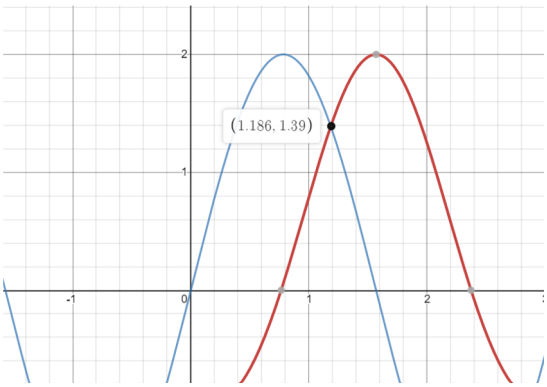
El ejecutar los algoritmos con una tolerancia de 10^{-8}, la sucesión original calculó 466 términos antes de lograr obtener un error local que cumpliera con esta tolerancia, mientras que la sucesión acelerada por el algoritmo de Aitken logró esto en 322 iteraciones.

n	Original	Aitken
1	0.54030230586813977	0.96177506016194292
2	0.87758256189037276	0.9821293544776053
3	0.9449569463147377	0.98978551355460054
4	0.96891242171064473	0.99341564966920237
5	0.98006657784124163	0.99540994165528651
6	0.98614323156292505	0.99661995874184672
7	0.98981326044661511	0.99740831513370942
8	0.99219766722932901	0.99795017270818498
9	0.99383350853889196	0.99833843863485561
10	0.99500416527802582	0.99862607500430112
11	0.99587061370056285	0.9988450527563334
12	0.99652978670055947	0.99901559238828397
13	0.99704287869644825	0.9991509871885178
14	0.99745006402491521	0.99926026631400522
15	0.99777860070112234	0.99934973615493794
16	0.99804751070009912	0.99942390899286948
17	0.99827039501276471	0.99948608219901669
18	0.99845718699874453	0.99953871052440746
19	0.99861527814258266	0.99958365181769859
20	0.99875026039496628	0.99962233305339177

Tabla 11: Resultados Original y Aitken

3. Sean $f(t) = 3sin^3t - 1$ y $g(t) = 4sin(t)cos(t)$ para t 0 las ecuaciones paramétricas que describe el movimiento en una partícula. Utilice un método de solución numérico con error de 10^{-16} para determinar donde las coordenadas coinciden y muestre gráficamente la solución.

Para obtener la intersección entre las funciones, se igualó a cero la resta de ambas expresiones para reescribir el problema como un problema de raíces. Una vez hecho esto, se utilizó el algoritmo de Newton acelerado por el método de Aitken para obtener la raíz de esta expresión. De esta manera, se obtuvo que la intersección se produce en t = 1.18649502158199042122078800682107339574 7.A continuación se muestran las gráficas de ambas funciones, en donde se puede apreciar gráficamente la solución que se obtuvo.



4.2. Método de Steffensen

Utilice el algoritmo de Steffensen para resolver la ecuación 33 y compararlo con el método de Aitken con Tolerancia de 10^{-8},10^{-16}, realice una gráfica que muestre la comparación entre los métodos.

$$f(x) = x^2 - cos(x) \tag{33}$$

Para implementar el método de Steffensen, se uso la ecuación 34 [4] (f(x) es la ecuación x). El método es bastante simple, y se siguen estos pasos:

- Se define un valor de aproximación p0 inicial, una tolerancia y un máximo de iteraciones.
- Se inicia un ciclo hasta el máximo de iteraciones, en el que se hace lo siguiente:
  - Se calcula una aproximación p con la ecuación x evaluada en p0.
  - Si $|p - p0| < tol$ entonces se retorna el valor actual de p como aproximación a la raíz.
  - Si no, entonces se hace p0 = p.

$$g(x) = f(x)^2/(f(p0 + f(p0)) - f(p0)) \tag{34}$$

A continuación se muestran las gráficas de los resultados para la ecuación 33, con el método de Steffensen y de Newton acelerado por Aitken. Se puede apreciar que Aitken logra acelerar newton y obtener una convergencia mas rápida que Steffensen. Sin embargo newton ya es un método que converge bastante rápido, Stefensen converge para las tolerancias $1x10^-8$ y $1x10^-16$ con 8 y 9 iteraciones respectivamente, qué son muy pocas iteraciones. Por lo cual Steffensen también es un método confiable para hallar raíces rápidamente.

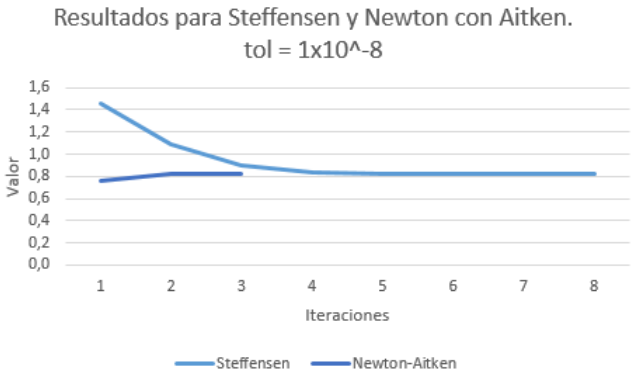


Figura 5: Resultados con tolerancia = $1x10^{-}8$

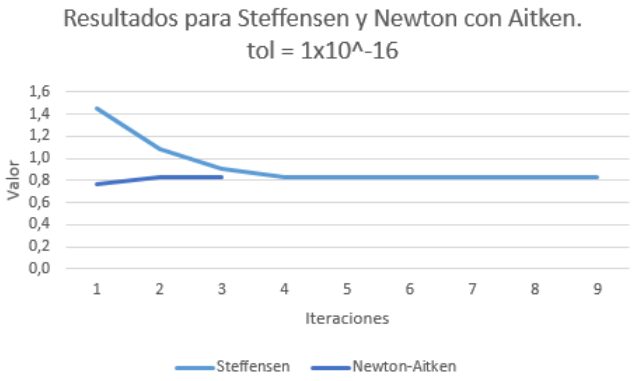


Figura 6: Resultados con tolerancia = $1x10^{-}16$

Referencias

[1] University of Cincinnati. "Dealing with Numbers". En: (22/11/2020). URL: https://uc-r.github.io/section3_numbers/.

[2] PUNTO FLOTANTE. "Matematicas Discretas". En: (22/11/2020). URL: https://medium.com/@matematicasdiscretaslibro/cap%C3%ADtulo-3-punto-flotante-c689043db98b.

[3] Python Software Foundation. "Built-in Types". En: (2020). URL: https://docs.python.org/3/library/stdtypes.htm.

[4] L. Ridgway Scott. "Numerical Analysis". En: (8/04/2011). URL: http://people.cs.uchicago.edu/~ridg/newna/nalrs.pdf.

[5] John Sturtz. "Basic Data Types in Python". En: (2020). URL: https://realpython.com/python-data-types/#floating-point-numbers.