Taller 1

Luisa Contreras Jairo Vanegas Gabriel Forero

26 de septiembre de 2020

1. Número de Operaciones

1.1. Números Binarios

1.1.1. Evualuar en $x = 1{,}00000000001$ con $P(x) = 1 + x + x^2 + ... + x^{50}$ y la primera derivada. Encuentre el error de cálculo al comparalo con los resultados de la expresión 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 %

1.1.2. Representación Binaria de pi (π)

A continuación denotaremos la representación binaria de pi en sus primeros 15 bits, teniendo en cuenta que este número es un número decimal no periódico e infinito, como lo podemos ver a continuación:

3.14159265358979323846...

Por lo tanto debemos usar el siguiente método por el cuál se transforman los numeros decimales en binario:
1. Empezamos transformando la parte entera la cuál corresponde a:

3→11

Ya tendríamos 2 bits, nos faltarían 13.

2. Ahora procedemos a multiplicar la parte decimal que seleccionamos en bits del siguiente modo

$0{,}1415926535897 * 2 = 0{,}28318530717 \rightarrow 0$

Y como se representa, cuando el numero resultante es mayor que 1, el bit resultante será 1, pero cuando es menor que 1, el bit resultante es 0

3. Así se realizo sucesivamente hasta que se obtuvo como resultado el siguiente número binario anteponiendo la parte entera por un punto de la decimal, como se hace normalmente en un número real.

11.0010010000111...

Sin embargo, este número no tiene una representación decimal finita por lo que tampoco tendrá una binaria, sólo aproximaciones. Además debemos tene en cuenta que las representaciones binarias cuando se trata de un decimal o un número que represente un fracción.

1.1.3. De Binarios a Base 10

A continuación convertiremos los siguientes números binarios a base 10, teniendo en cuenta el siguiente procedimiento.

- 1010101

1	0	1	0	1	0	1
$1 * 2^6+$	$0 * 2^5+$	$1 * 2^4+$	$0 * 2^3+$	$1 * 2^2+$	$0 * 2^1+$	$1 * 2^0$

  Que nos dá como resultado: $64 + 0 + 16 + 0 + 4 + 0 + 1 = 85$

- 1011.101

  Ahora como tenemos un decimal lo realizaremos de la siguiente manera

1	0	1	1	.	1	0	1
$1 * 2^4+$	$0 * 2^3+$	$1 * 2^2+$	$1 * 2^0+$	.	$1/2^1+$	$0/2^2+$	$1/2^3$

Que nos dá como resultado: 11,625

- 10111.010101...
  Para este item debemos truncar el número que podemos deducir por los puntos supensivos que es infito periodico pero lo cortaremos en 7 bits por como hemos manejado los binarios anteriores, dejando así sólo 2 bits significativos

1	0	1	1	1	.	0	1
$1 * 2^5+$	$0 * 2^4+$	$1 * 2^3+$	$1 * 2^2+$	$1 * 2^0+$	.	$0/2^1+$	$1/2^2$

  Que nos dá como resultado: 23,25

- 111.1111...
  Para este ejercicio haremos lo mismo que el anterior truncando el número en el bit 7, dejando 4 bits significativos

1	1	1	.	1	1	1	1
$1 * 2^2+$	$1 * 2^1+$	$1 * 2^0+$	.	$1/2^1+$	$1/2^2+$	$1/2^3+$	$1/2^4$

  Este ejercicio nos dió como resultado el siguiente decimal: 7,9375

1.1.4. De Base 10 a Binarios

- 11,25
  Estos procedimientos los realizaremos como ese explicó en el caso de π
  Obteniendo así como resultado: 1011,0100...
  Por lo que tuvimos que truncar el binario pues se realiza una división infinita, dando como resultado un binario infinito, dejando así 4 bits significativos

- 2/3
  Comenzamos por transformar la fracción en un número decimal,el cuál es: 0.6666666... periódico.
  Por lo que sólo tomaremos 7 bits significativos.
  Dando así como resultado: 0,1010101... binario periódico.

- 30,6
  En este caso sucede que lo mismo que en el caso de 11.25, sólo que

Figura 1: Representación de Número infinito en bits

el periodo de 30.6 es mucho más extenso como podemos observar : 11110,10011001100110011, con periodo 1001.

- 99,9
  Al igual que el ítem anterior, nos dió como resultado el siguiente binario infinito: 100011,111001100110011001100... Sin embargo, aunque aparentemente sea periódico igual que el punto anterior, llega un punto con la precision de bits significativos de 80 nos da como resultado

  1100011,11100110011001100110011001100110011001100110011001100110100000000000000000000

  siendo así un decimal no periódico. Aquí podemos observar la importancia de la precisión.

1.2. Representación del Punto Flotante de los Números Reales

1.2.1. Ajuste de un número binario infinito en un número finito de bits

Como lo comprobamos anterior existen números los cuales no es posible representar en totalidad en binario, por lo que se recurrió al recorte, por medio de cifras significativas de precision. Sin embargo es necesario tener un error con respecto a dicho recorte.
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

1.2.2. Diferencia entre redondeo y recorte

Recorte: Es quitarle decimales al numero, no se tiene en cuenta la distancia eculidiana. Es la técnica más sencilla consiste en truncar x a 10 dígitos, ignorando el resto.
Es decir, $r(x) = x_i$ si $x \geq 0$ y $r(x) = x_d$ si $x \leq 0$

$$r(x) = \begin{cases} x_i & \text{si } x_i \leq x < \frac{x_i + x_d}{2} \\ x_d & \text{si } \frac{x_i + x_d}{2} \leq x \leq x_d. \end{cases}$$

Figura 2: Fórmulas de Redondeo

Redondeo: Es una aproximacion númerica que intenta hacer mas fiel al número, funciones como piso y techo, esta cumplen con este tipo de objetivos. consiste en redondear x al más próximo entre x_i y x_d; en el caso en que está en a igual distancia se elige xd. Dado por las siguientes fórmulas: Ejemplo :

- Recorte 150,48 $\rightarrow$ 150
- Redondeo 150,48 $\rightarrow$ 150,5

La principal diferencia entre el redondeo y el recorte es que; Si queremos aproximar un número por el metodo de redondeo a n cifras para esta aproximación se tiene en cuenta la cifra n+1, en este caso si es igual o mayor a 5 se incrementa la cifra numero n. A diferencia del recorte que solo tiene en cuenta las n cifras que la persona quiere dejar y no importa cual sea la cifra n+1. Esto tiene una diferencia muy grande ya que con el redondeo estamos incluyendo mas precisión a nuestro numero y en el recorte podemos perder valor importante por ello debemos conocer muy bien sus diferencias y en que casos implementar cada técnica, según sea el caso.

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

a) Según la norma IEEE-754 un número infinito (+inf o −inf) se representa de forma que los bits correspondientes al exponente mínimo se colocan todos en 1 (8 para 32 bits, 11 para 64 bits) y el bit del signo indicará el signo del infinito.

b) En el proceso de redondeo se usa el valor de la siguiente para obtener un resultado más aproximado del valor. El corte (truncamiento) corta el número de cifras decimales sin tener en cuenta las siguientes para el valor de la última cifra. Se puede decir que el redondeo es más exacto en el valor decimal final, sin embargo, un redondeo a n decimales y corte a n decimales podrán ser los mismos dependiendo de la regla usada para el redondeo.

c) $x = 0,4$

Según el proceso para convertir un número decimal a su representación en binario el valor será:

$0{,}4 * 2 = 0{,}8 \rightarrow 0$
$0{,}8 * 2 = 1,6 \rightarrow 1$
$0,6 * 2 = 1,2 \rightarrow 1$
$0{,}2 * 2 = 0,4 \rightarrow 0$
$0{,}8 * 2 = 1,6 \rightarrow 1$
Repitiendose y volviendose cíclico infinitamente, dando como resultado el siguienete binario:
$= 0,01100110011001100110011... = 1,10011001100110011001100110011... * 2^{-2}$
Sin embargo, el carácter de este número es periódico infinito, por lo que para poder ser representado en un computador se utiliza la norma IEEE 754 para representaciones de números en coma flotante.
Usando un tamaño de número de 64 bits, dividos en 1 bit para el signo, 11 bits para el exponente mínimo (Biased Exponent) y 52 bits para la precisión de la fracción (Fraction) se encuentra que:

- Signo = 0
- BiasedExponent = $(1021_{10} = (01111111101)_2$
- Fraccin = $(1001100110011001100110011001100110011001100110011001)$

Por lo tanto, se tiene que el valor de $fl(0,4)$ es igual en el estandar IEEE 754
a: $(0,4)_{10} \cong (0011111111011001100110011001100...)_2 = (0,3999999999999999966693309261245)_{10}$

1.2.4. Error de redondeo

Continuando con el ejercicio anterior, encontremos el error de conversión estándar IEEE 754, se usará la siguiente fórmula

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

$$\frac{|0,39999999999999996693309261245 - 0,4|}{|0,4|} \leq \frac{2^{-52}}{2} \tag{2}$$

$$\frac{|0,39999999999999996693309261245 - 0,4|}{|0,4|} \leq 2^{-53} \tag{3}$$

$$8,32667268468875 * 10^{-17} \leq 1,110223 * 10^{-16} \tag{4}$$

El error de redondeo es por lo tanto $\cong 8,327 * 10^{-17}$ o $8,327 * 10^{-15}\,\%$

1.2.5. Verificación de doble precisión de tipo de datos de R y Phyton en format long

Tanto python como R utilizan el la precision doble de IEEE en todos sus formatos. En el caso de R, que es la herramienta que a la que más concurrimos en el grupo.Si el dato basico de r es de precision doble de IEEE se puede configurar para que de la cantidad de decimales que el usuario prefiera dejear en la mantisa asi modificando a su ves el exponente. El format long es un fromato de dato que permite almacenar de -2147483648 a 2147483647, esto le permite al usuario realizar operaciones con alta precision a la hora de no perder decimales.

Todas las plataformas R deben trabajar con valores que cumplan con el estándar IEC 60559 (también conocido como IEEE 754). Esto básicamente funciona con una precisión de 53 bits, y representa con esa precisión un rango de valores absolutos de aproximadamente $2e - 308a2e + 308$. También tiene valores especiales NaN(muchos de ellos), más y menos infinito y más y menos cero (aunque R actúa como si fueran lo mismo). También hay núméros desnormales (izados) (o subnormales) con valores absolutos por encima o por debajo del rango dado anteriormente pero representados con menor precisión. Sin embargo hay que tener en cuenta que, en última instancia, la forma en que se manejan los números de doble precisión depende de la CPU / FPU y el compilador. En IEEE 754-2008 / IEC60559: 2011, esto se denomina formato 'binary64'.

1.2.6. Representación hexadecimal de 9.4

Primero convertimos a binario, como lo hemos hecho ya anteiormente , lo que nos da como resultado: 1001,0110011001100110011001100110011001100..., lo truncamos en 8 bits significativos que nos daría como resultado 1001,01100110 Despues convertimos a hexadecimal 0x941166666

1.2.7. Encuentre las dos raíces de la ecuacion cuadrática $x^2+9^{12}x = 3$

Las raíces de la ecuación cuadrática $x^2 + 9^{12}x = 3$ son las siguientes:

$$x = -2{,}8242953648100000 \tag{5}$$

$$x = 1{,}0622118484416449 \tag{6}$$

Con una precisión de 10^{-16}

1.2.8. Calcular con mayor exactitud las raíces de la ecuación $x^2 + bx - 10^{-12} = 0$, donde b es un número mayor que 100

Las raíces de la ecuación estarán dadas por

$$x^2 + bx - \frac{1}{1000000000000} = 0, b \geq 100 \tag{7}$$

Donde hicimos las raices dadas por las primeros 5 raices, mayores o iguales a 100, y son las siguientes:

$\frac{1{,}0e14x+1000000000000x^2*1000000000000-1}{1000000000000}$	100
$\frac{1{,}01e1x4+1000000000000x^2*1000000000000-1}{1000000000000}$	101
$\frac{1{,}02e14+1000000000000x^2*1000000000000-1}{1000000000000}$	102
$\frac{1{,}03e14+1000000000000x^2*1000000000000-1}{1000000000000}$	103
$\frac{1{,}04e14+1000000000000x^2*1000000000000-1}{1000000000000}$	104

2. Raíces de una Ecuación

2.1. sumar únicamente los elementos de la sub atriz triangular superior o triangular inferior, dada la matriz cuadrada An

2.1.1. Pruebas

Para la solucion de este punto se hicieron varias pruebas de matrices cuadradas

N	Iteraciones
1	1
2	2
3	3
4	6
5	10
6	15
7	21
8	28
9	36
10	45

2.1.2. Gráfica

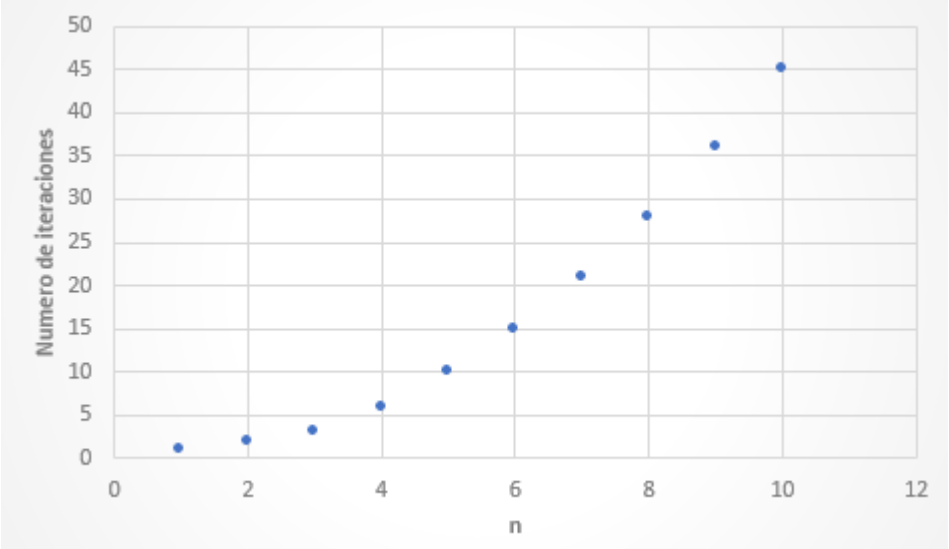

Figura 3: Iteraciones vs tamaño de la matriz

2.1.3. Convergencia

La convergencia es O(n*($\frac{n}{2}$-0.5))

2.2. Sumar los n primeros números naturales al cuadrado

2.2.1. Pruebas

N	Iteraciones	Resultado
1	1	1
2	2	5
3	3	14
4	4	30
5	5	55
6	6	91
7	7	140
8	8	204
9	9	285
10	105	385

2.2.2. Gráfica

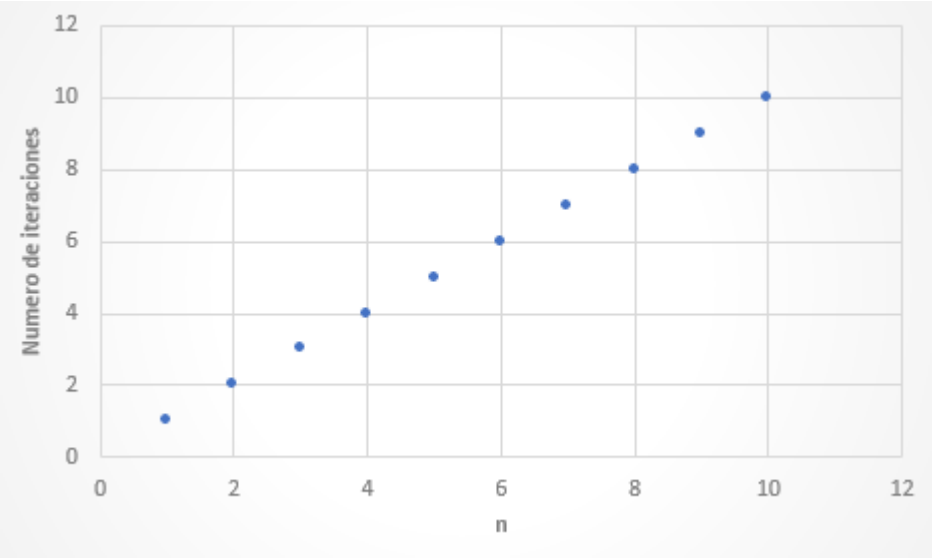

Figura 4: Iteraciones vs n

2.3. Problema del cohete

Para describir la trayectoria de un cohete se tiene el modelo:

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

Donde, y es la altura en [m] y t tiempo en [s]. El cohete esta colocado verticalmente sobre la tierra. Utilizando dos mótodos de solución de ecuación no lineal, encuentre la altura máxima que alcanza el cohete.

El mejor método para encontrar la máxima altura que alcanza el cohete es con la derivada.

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

Derivamos $y'(t) = 4{,}26t - 0{,}0052t^3$ Igualamos la derivada a 0

$$0 = 4{,}26t - 0{,}0052t^3$$

Despejamos t

$$0{,}0052t^3 = 4{,}26t$$

Cancelamos una t

$$0{,}0052t^2 = 4{,}26$$

Finalmente tenemos que t^2

$t^2 = \frac{4{,}261}{0{,}0052}$

3. Convergencia de Metodos Iterativos

3.1. Punto de intersección

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

Para este punto ambos métodos dieron el mismo resultado, para todas las iteraciones.

$$X_n = X_(n-1) - \frac{F(X_(n-1)) * (X_(n-1) - X_(n-2))}{F(X_(n-1)) - F(X_(n-2))} \tag{8}$$

$$X_(n+1) = X_n - F(X_n) * \frac{X_n - X_(n-1)}{F(X_n) - F(X_(n-1)} \tag{9}$$

Formula	Valores iniciales	Iteraciones	Resultado
8	$X_1 = -1{,}9$ $X_2 = -1{,}1$	7	$f(x) = -1{,}63144684837866$ $g(x) = -1{,}631088$
9	$X_n = -1{,}9$ $X_1 = -1{,}1$	7	$f(x) = -1{,}63144684837866$ $g(x) = -1{,}631088$

3.2. Newton

4. Punto cuatro: Convergencia Acelerada

4.1. Tipo de convergencia

Para analizar la convergencia de la sucesión $(x_n)_{n=0}^{\infty}$, con $x_n = cos\left(\frac{1}{n}\right)$ vamos a usar el limite de la sucesión descrito en la siguiente ecuación:

$$\lim_{n\to\infty} cos\left(\frac{1}{n}\right) \tag{10}$$

Para trabajar con este limite primero hacemos el limite dentro del coseno para ver cual es el resultado y luego aplicarlo a la función coseno:

$$\lim_{n\to\infty} \frac{1}{n} = 0 \tag{11}$$

Reemplazando en el limite original:

$$\lim_{n\to\infty} cos(0) = 1 \tag{12}$$

Con lo que verificamos que la sucesión converge a 1 independientemente de su origen.

4.2. Comparación con sucesión de Aitken

Para calcular A_n vamos a usar la ecuación que podemos encontrar en el enunciado del taller:

$$A_n = x_{n+2} - \frac{(x_{n+2} - x_{n+1})^2}{x_{n+2} - 2x_{n+1} + x_n} \tag{13}$$

En la siguiente tabla podemos ver los valores calculados para los primeros términos de las sucesiones:

n	x_n	A_n
1	0,540302306	0,96177506
2	0,877582562	0,982129354
3	0,944956946	0,989785514
4	0,968912422	0,99341565
5	0,980066578	0,995409942
6	0,986143232	0,996619959
7	0,98981326	0,997408315
8	0,992197667	0,997950173
9	0,993833509	0,998338439
10	0,995004165	0,998626075

Cuadro 1: Comparación A_n vs X_n

Como podemos ver en la tabla desde $n = 1$ la sucesión A_n ya llega a una convergencia que le toma a x_n 4 iteraciones mas para acercarse al valor convergente de la sucesión, por lo que podemos concluir que la sucesión de A_n converge de forma acelerada con respecto a la sucesión x_n.

4.3. Colisión de partículas

La trayectoria de dos partículas en el espacio es descrita por las ecuaciones paramétricas

$$f(x) = 3sin^3(x) - 1 \tag{14}$$

$$g(x) = 4sin(x)cos(x) \tag{15}$$

Se desea saber el valor de x en el que las partículas colisionarán, de hacerlo, y para eso tenemos como restricción que $x > 0$. Para encontrar este punto igualamos las dos ecuaciones y de forma algebraica igualamos a 0.

$$3sin^3(x) - 1 = 4sin(x)cos(x) \tag{16}$$

$$3sin^3(x) - 1 - 4sin(x)cos(x) = 0 \tag{17}$$

Con esta ultima ecuación vamos usar el método de la secante para hallar la primer raíz, ya que no tiene sentido una segunda raíz mayor que la primera ya que las partículas entraran en colisión en la primer raíz. Con esto en mente el rango en el que vamos a aplicar el método de la secante sera $(0, 2] \in R$. También usamos una tolerancia de 10^{-16} como criterio de parada para el método.

El método entrega como resultado el numero 1,18649502158199038 como la primer raíz en el intervalo elegido. Graficando la función (17) y una recta paralela al eje Y en el resultado, podemos ver que es una solución correcta.

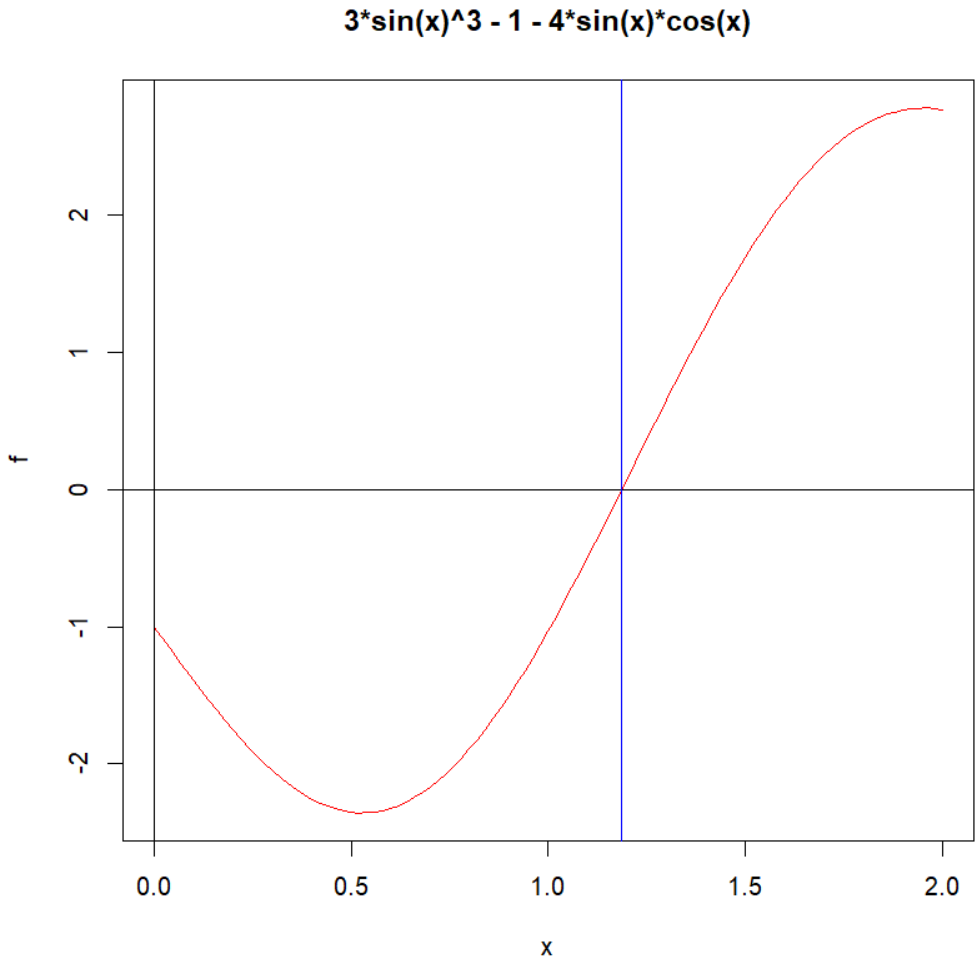

Figura 5: Colisión de partículas

4.4. Aitken vs Steffensen

Para este punto el se escoge la función $f(x) = x^2 - cos(x)$ y se desea comparar la ejecución de los algoritmos de Steffensen y de Aitken. Para el algoritmo que acelere Aitken usamos el algoritmo de Newton debido a que es el que hemos visto en clase. Debido a las características de la función tuvimos que derivarla directamente lo que nos dio como resultado $f'(x) = 2 * x + sin(x)$.

Como el algoritmo de Steffensen usa el método del punto fijo, para este experimento decidimos usar la siguiente función

$$f(x) = \sqrt{cos(x)} \tag{18}$$

En cada iteración del método.

Para los dos algoritmos usamos tolerancias de 10^{-8} y 10^{-16}, e iniciamos los algoritmos con $x = 1$ como valor inicial.

4.4.1. Resultados

Tanto para Aitken como para Steffensen los resultados fueron iguales y convergieron en los valores de la siguiente tabla

tol	Aitken	Steffensen
10^{-8}	0.8241323158	0.8241323158
10^{-16}	0.82413231230252243	0.82413231230252243

Cuadro 2: Resultados Aitken vs Steffensen

A continuación se presentan las gráficas de convergencia de los algoritmos para la tolerancia 10^{-16}

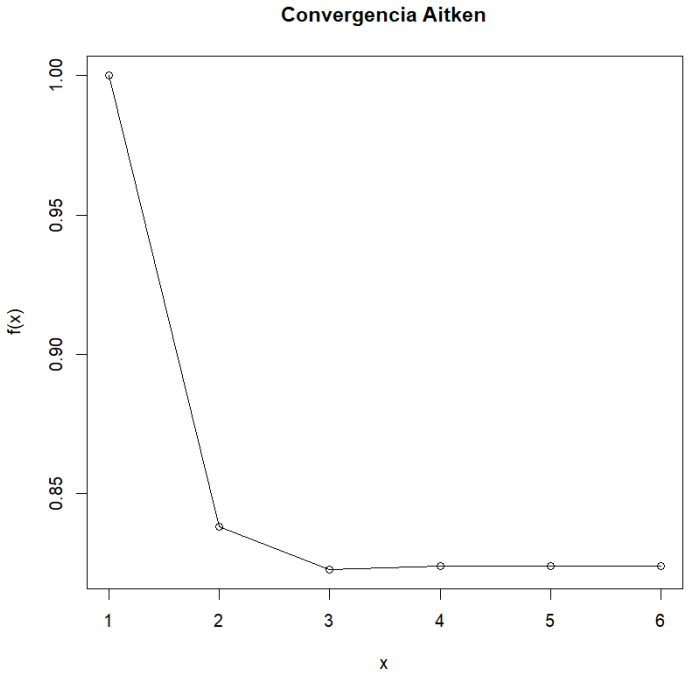

Figura 6: Convergencia Aitken

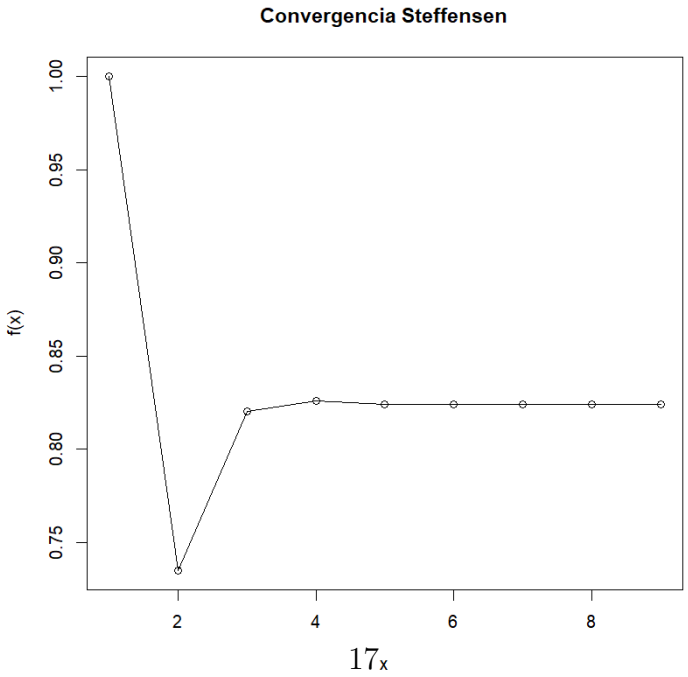

Figura 7: Convergencia Steffensen

Como se puede ver en las gráficas la convergencia de Aitken en el método de Newton, se alcanza mucho mas rápido y no solo eso sino que también converge de forma mas estable. El algoritmo de Steffensen converge rápidamente comparado con el método que usa, siendo este punto fijo, pero no lo suficiente para competir con Aitken con Newton, también se nota que la convergencia con este algoritmo se alcanza con fluctuaciones sobre y por debajo del valor deseado.