Número de Operaciones

Ejercicio.

```
Ejercicio.
 Evaluar el polinoio en x= 1.00000000001 conP(x) = 1 +x+x²+...+x³⁰ y la primera derivada.
Encuentreel error de c álculo al comparalo con los resultados de la expresi ón equivalenteQ(x) = (x³¹-1)/(x-1)

resultado de la original:  50.00000001225002
resultado de la derivada:  1225.0000003920004
Resultado de  Q(x):  51.0
```

1. Encuentre los primeros 15 bits en la representación binaria de π

```
a) Encuentre los primeros 15 bits en la representacion binaria de π
Primeros 15 digitos de π:  3.141592653589793
Primeros 15 digitos de π en binario:  11.100000001100011100011011110100011111011010010000 1
```

2. Convertir los siguientes números binarios a base 10: 1010101; 1011.101; 10111.010101...; 111.1111...

```
b) Convertir los siguientes numeros binarios a base 10: 1010101; 1011.101; 10111.010101...; 111.1111
85
11.5
23.21
7.15
```

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

```
c) Convierta los siguientes numeros de base 10 a binaria: 11.25;2/3; 30.6; 99.9
1011.11001
0.10111101011110100110001001010100000001010101010101010
11110.110
1100011.1001
0.100
```

4. ¿Cómo se ajusta un numero binario infinito en un numero finito de bits?

   Según la norma IEEE-754, se puede representar al comparar el número con el valor absoluto numérico que puede soportar la máquina. El máximo valor absoluto de un número es *MAX_VALUE* = 2^{1023} * (2 - ε) = 1.7976931348623157 × 10^{308} por lo que se realiza una comprobación booleana:

   Si (n > *MAX_VALUE), entonces n = +INFINITO*

   Si (n < *-MAX_VALUE), entonces n = -INFINITO*

   El registro del formato es un bit cero o uno para el signo, todos los bits del exponente puestos a uno y todos los bits de la fracción puestos a cero. Esto resultará en los hexadecimales 7FF0000000000000 para +Infinity con esta presentación:

   $$+\infty \Rightarrow (-1)s \times b.f \times 2e =$$

   (-1)0 × 0.0000000000000000000000000000000000000000000000000000 × 21024

   Y el hexadecimal FFF0000000000000 para -Infinity con la siguiente presentación:

   $$-\infty \Rightarrow (-1)s \times b.f \times 2e =$$

(-1)1 × 0.00000000000000000000000000000000000000000000000000000 × 21024

5. ¿Cuál es la diferencia entre redondeo y recorte?

   En el recorte o truncamiento, se eliminan las cifras o dígitos hasta el nivel de precisión establecido mientras que en el redondeo se tienen en cuentan las cifras que están después del nivel de precisión establecido. En este caso, si el digito es mayor 5 entonces la cifra anterior se incrementa el digito y con cada uno de ellos hasta retroceder al nivel de precisión.

6. ¿Cómo se ajusta un numero binario infinito en un numero finito de bits?

   Hecho arriba.

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

   Para encontrar el numero punto flotante fl(x) de x = 0,4, debe convertir este número base 10 en uno base 2, es decir, en un numero binario.

   $$0,4 * 2 = 0,8 \rightarrow 0$$

   $$0,8 * 2 = 1,6 \rightarrow 1$$

   $$0,6 * 2 = 1,2 \rightarrow 1$$

   $$0,2 * 2 = 0,4 \rightarrow 0$$

   $$0,4 * 2 = 0,8 \rightarrow 0$$

   $$....$$

   $$(0,4)_2 = 0,01100110011001100110011....$$

   $$= 1,10011001100110011001100110011... * 2^{-2}$$

   Sin embargo, como se puede observar que su cola es infinita por lo que para representarse se usara la norma para expresar esta cola (descrita en el punto 6). Como se trata de un numero de doble precisión, usamos el tamaño de 64 bits los cuales están divididos 52 bits para la fracción, 1 bit para el signo y 11 para el exponente (biased exponent).

   Signo = 0

   Exponente = $(01111111101)_2$

   Fracción = $(1100110011001100110011001100110011001100110011001100)_2$ (52 Bits)

   Teniendo esto, se puede decir que el valor de numero de punto flotante fl(0,4) es igual a (según IEEE-754):

   $(0,4)_{10} \approx (0, 01100110011001100110011....)_2 \approx 0,4+0,1*2^{(-52)}$

   $fl(0,4) \approx (0,4)10 \approx 0.4000000000000000222044604925031308084726333618164062 5$

8. 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 maquina:

$$\frac{|fl(x)-x|}{|x|} \leq \frac{1}{2}\ni_{maquina}$$

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

$$\frac{|fl(0{,}4)-0{,}4|}{|0{,}4|} \leq \frac{2^{-52}}{2}$$

$$\frac{|(0{,}4+0{,}1*2^{-52})-0{,}4|}{|0{,}4|} \leq \frac{2^{-52}}{2}$$

$$5.55111512312578270211815834045410156 25 \times 10\text{^} - 17$$
$$\leq 1.11022302462515654042363166809082031 25 \times 10\text{^} - 16$$

El error de redondeo es ≈ $5{,}551 \times 10\text{^} - 17$

9. Verificar si el tipo de datos básico de R y Python es de precisión doble IEEE y Revisar en R y Phytho el format long.

   Casi todas las máquinas actuales (noviembre de 2000) utilizan aritmética de punto flotante IEEE-754, y casi todas las plataformas asignan flotantes de Python a IEEE-754 de "doble precisión". 754 dobles contienen 53 bits de precisión. Según la norma IEEE-754, se tienen el siguiente estándar:

Format	Total bits	Significand bits	Exponent bits	Smallest number	Largest number
Single precision	32	23 + 1 sign	8	ca. $1.2 \cdot 10^{-38}$	ca. $3.4 \cdot 10^{38}$
Double precision	64	52 + 1 sign	11	ca. $2.2 \cdot 10^{-308}$	ca. $1.8 \cdot 10^{308}$

   Al imprimir la información que usa Python cuando trata valores de punto flotante, podemos observar lo siguiente:

```
PS C:\Users\user> & "C:/Program Files (x86)/Python38-32/python.exe" "c:/Users/user/Desktop/verificar float.py"
sys.float_info(max=1.7976931348623157e+308, max_exp=1024, max_10_exp=308, min=2.2250738585072014e-308, min_exp=-1021, min_10_exp=-307, dig=15, mant_dig=53, epsilon=2.220446049
9250313e-16, radix=2, rounds=1)
```

   Dentro de la información se puede evidenciar que utiliza las mismas especificaciones como la cantidad de bits significantes (53 que corresponde a 52 bits + 1 bit para el signo) así como el menor y mayor numero que se puede expresar que corresponde a $2{,}2 \times 10^{-308}$ y $1{,}8 \times 10^{308}$

10. Encuentre la representación en número de maquina hexadecimal del número real 9.4

    Se usó las funciones 'pack' y 'unpack' de la librería 'struct'. Nosotros primero empaquetamos (pack) en un binario y luego desempaquetamos esto (unpack) como un entero.

```
PS C:\Users\user> & "C:/Program Files (x86)/Python38-32/python.exe"
Numero a convertir:  9.4
Float a Hex (32-Bit):  0x41166666
Double a Hex (64-Bit):  0x4022cccccccccccd
```

11. 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.

```
with(RootFinding)
```
[Analytic, AnalyticZerosFound, BivariatePolynomial, EnclosingBox, HasRealRoots, Homotopy, Isolate, NextZero, Parametric, WitnessPoints] (1)

```
fx := x^2 + 9^12 x - 13
```
$$fx := x^2 + 282429536481\,x - 13 \quad (2)$$

```
solve(fx)
```
$$-\frac{282429536481}{2} + \frac{\sqrt{7976644307687250986 3413}}{2}, -\frac{282429536481}{2} - \frac{\sqrt{7976644307687250986 3413}}{2} \quad (3)$$

```
Isolate(fx, digits = 15)
```
$$[x = -2.82429536481000\,10^{11}, x = 4.60291800991379\,10^{-11}] \quad (4)$$

12. Explique cómo calcular con mayor exactitud las raíces de la ecuación

$x^2 + bx - 10^{-12} = 0$ donde b es un número mayor a 100

```
gx := x^2 + 101 x - 10^-12
```
$$gx := x^2 + 101\,x - \frac{1}{1000000000000} \quad (5)$$

```
solve(gx)
```
$$-\frac{101}{2} + \frac{\sqrt{2550250000000001}}{1000000}, -\frac{101}{2} - \frac{\sqrt{2550250000000001}}{1000000} \quad (6)$$

```
Isolate(gx, digits = 15)
```
$$[x = -101.000000000000, x = 9.90099009900990\,10^{-15}] \quad (7)$$

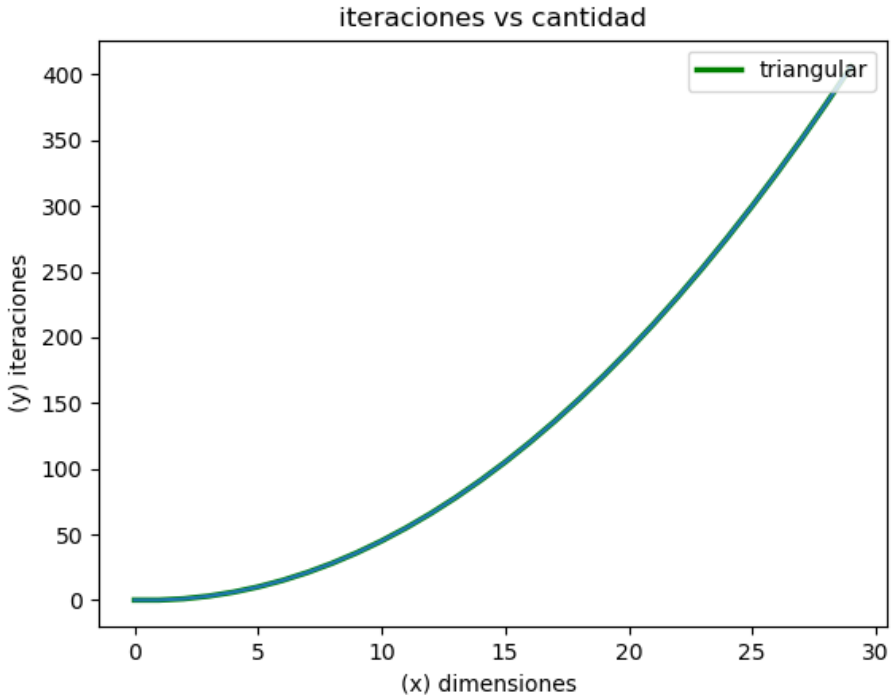
```
Isolate(gx, digits = 30)
```
$$[x = -101.000000000000009900990099010, x = 9.90099009900990001950886197335\,10^{-15}] \quad (8)$$

Raíces de una Ecuación

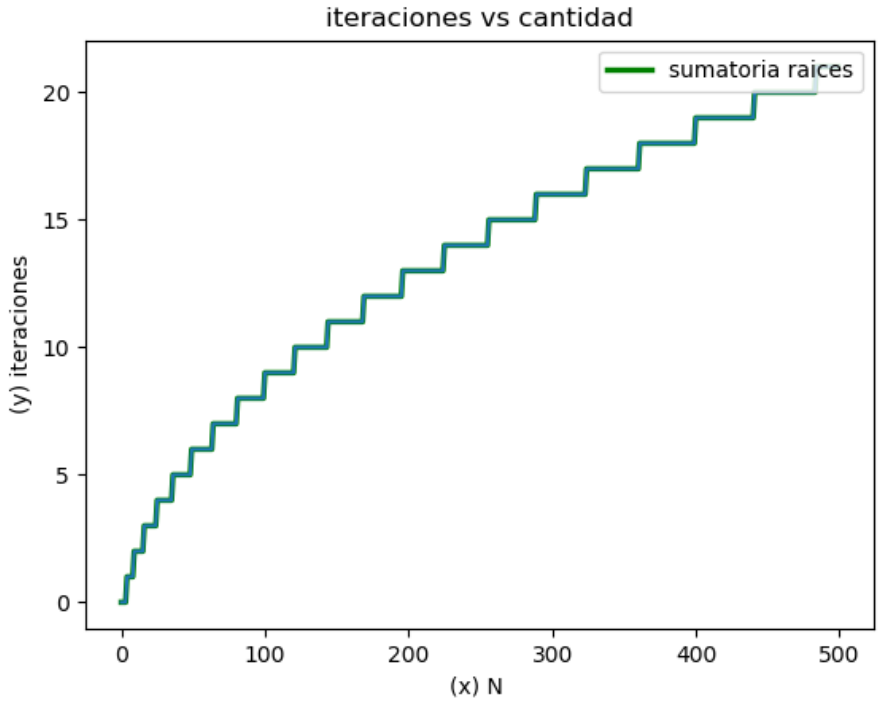
1. Implemente en R o Python un algoritmo que le permita sumar únicamente los elementos de la submatriz triangular superior o triangular inferior, dada la matriz cuadrada A_n. Imprima varias pruebas, para diferentes valores de n y exprese *f(n)* en notación *O()* con una gráfica que muestre su orden de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

```
Ejercicio #1
realice un algoritmo que le permita sumar únicamente los elementos de la submatriz triangular superior o triangular infe
rior, dada la matriz cuadrada A_n.
Imprima varias pruebas,para diferentes valores deny exprese f(n) en notacion O() con una grafica que muestre su orden de
convergencia.

matriz:
 [[1 1 1 1 1]
 [1 1 1 1 1]
 [1 1 1 1 1]
 [1 1 1 1 1]
 [1 1 1 1 1]]
suma de la triangular superior:
la cantidad de iteraciones fueron:  10
suma de la triangular inferior:  10
```



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.



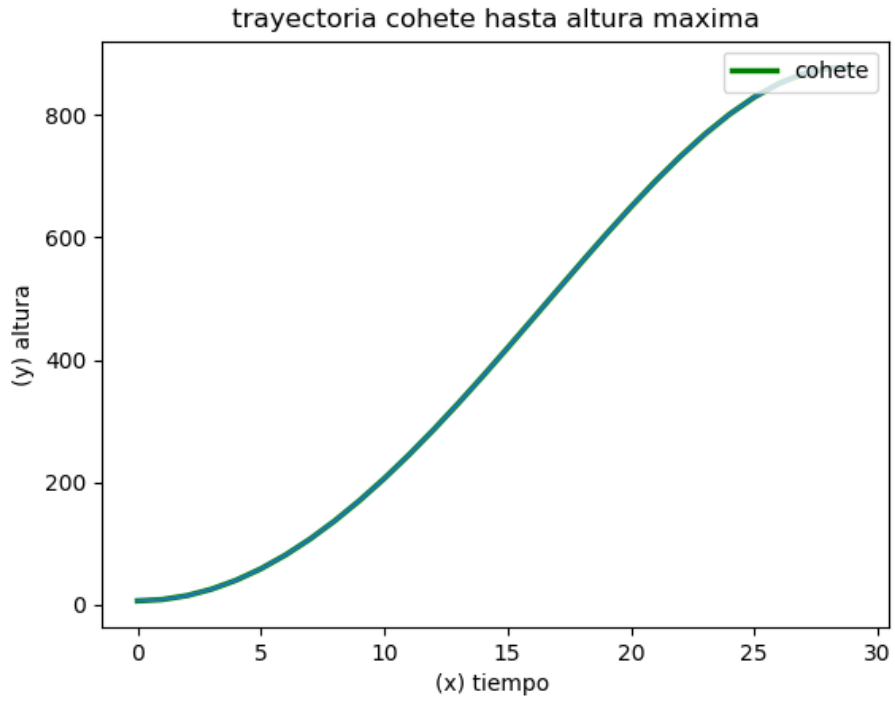
```
Ejercicio #2
Realice un algoritmo que le permita sumar los n² primeros números naturales al cuadrado. Imprima varias pruebas para dif
erentes valores de n y exprese f(n) en notación O() con una gráfica que muestre su orden de convergencia.

la cantidad de iteraciones fueron:  10
la sumatoria de los cuadrados hasta 121 es:  506
```

3. 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 está colocado verticalmente sobre la tierra. Utilizando dos métodos de solución de ecuación no lineal, encuentre la altura máxima que alcanza el cohete




```
Ejercicio #3
y(t) = 6 + 2,13t²−0.0013t⁴
Donde y es la altura en [m] yttiempo en [s]. El cohete esta colocado verticalmente sobre la tierra. Utilizando dos metod
os de solución de ecuación no lineal, encuentre la altura m áxima que alcanza elcohete.

la cantidad de iteraciones fueron:  29
La altura maxima es:  877.8647  metros
```

Convergencia de métodos Iterativos

Para cada uno de los siguientes ejercicios implemente en R o Python, debe determinar el número de iteraciones realizadas, una gráfica que evidencie el tipo de convergencia del método y debe expresarla en notación O():

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.

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

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

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

2. Determine el valor de los coeficientes a y b tal que

$$f(1) = 3 \text{ y } f(2) = 4 \text{ con } f(x) = a + (ax + b)e^{ax+b}. \text{ Obtenga la respuesta con } E = 10^{-8}$$

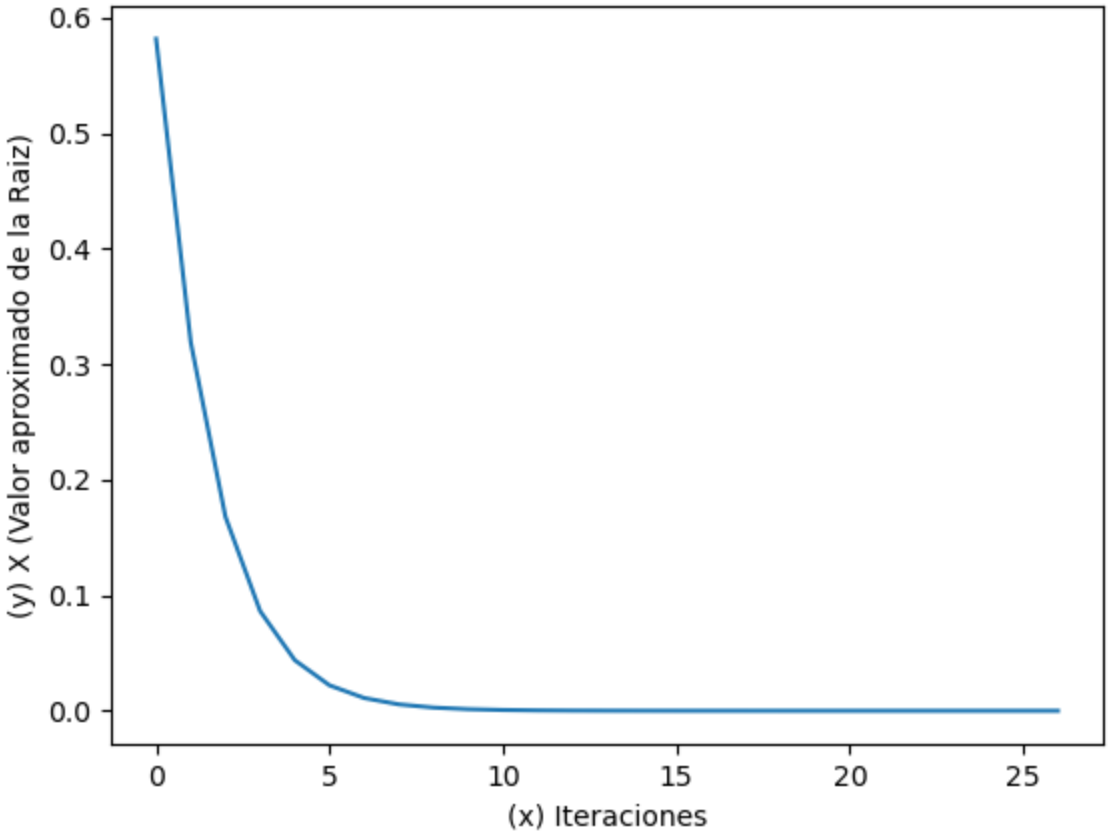
3. Sea *f(x) = e^x –x –1*

a) Demuestre que Tiene un cero de multiplicidad 2 en x = 0.

En la siguiente imagen se puede comprobar o verificar la multiplicidad 2 al observar las ultimas dos iteraciones que realiza Newton-Raphson en su aproximacion a la raiz para f(x).

```
PS C:\Users\user> & "C:/Program Files (x86)/Python38-32/
x1: 0.581976706869326
x2: 0.319055040910818
x3: 0.167996172885770
x4: 0.086348873747781
x5: 0.043795703673714
x6: 0.022057685365768
x7: 0.011069387477739
x8: 0.005544904662931
x9: 0.002775014494137
x10: 0.001388148972389
x11: 0.000694235065980
x12: 0.000347157696665
x13: 0.000173588891597
x14: 0.000086796956574
x15: 0.000043399107034
x16: 0.000021699709854
x17: 0.000010849887297
x18: 0.000005424952542
x19: 0.000002712472020
x20: 0.000001356289190
x21: 0.000000678182975
x22: 0.000000338985234
x23: 0.000000169333175
x24: 0.000000084099446
x25: 0.000000041855249
x26: 0.000000020635009
x27: 0.000000020635009
Estimacion de Newton = 0.000000020635009
Iteraciones: 27
```

b) Utilizando el método de Newton con p0 = 1 verifique que converge a cero pero no de forma cuadrática.



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

El rendimiento mejora respecto a su par debido a que en este metodo se requiere un menor numero de itereaciones para determinar la raiz aproximada al hacer uso de la segunda derivada de la funcion f(x) para calcular el valor de la raiz en cada iteracion. Sin embargo el error relativo entre los valores aproximados de la raiz es 2.0592823×10^{-8}

```
x1: -0.23421061355351425
x2: -0.00845827991076109
x3: -1.1890183808588653e-05
x4: -4.218590698935789e-11
x5: -4.218590698935789e-11
Estimacion de Newton Modificado = 0.000000000042186
Iteraciones: 5
PS C:\Users\user>
```

Convergencia Acelerada

1. Dada la sucesión $\{x_n\}_{n=0}^{\infty}$ con $x_n = \cos(1/n)$
    a. Verifique el tipo de convergencia en x = 1 independiente del origen

```
punto 1
Verifique el tipo de convergencia enx= 1 independiente del origen
f(a)*f(b) >= 0 entonces no es posible de realizar
raiz:  None
```

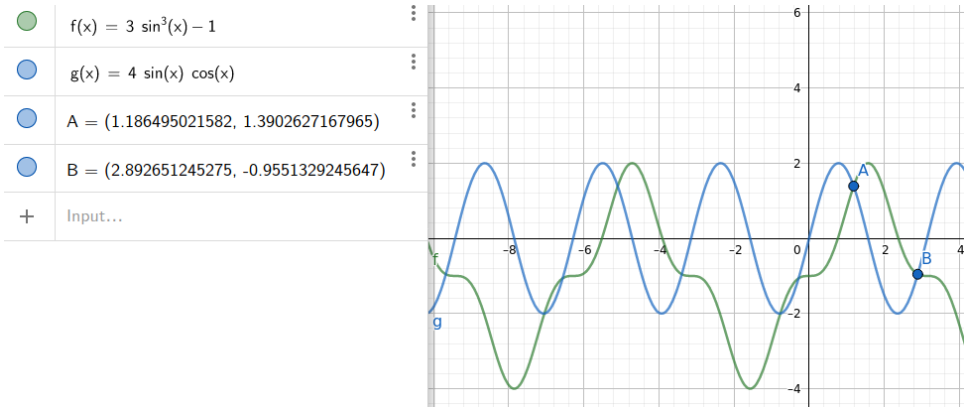
b. Compare los primeros términos con la sucesión $\{A_n\}^{\infty}{}_{n=0}$

```
punto 2
Compare los primeros t érminos con la sucesión { An }∞ n=0
cos(1/ 1 )= 0.5403023058681398
cos(1/ 2 )= 0.8775825618903728
cos(1/ 3 )= 0.9449569463147377
cos(1/ 4 )= 0.9689124217106447
cos(1/ 5 )= 0.9800665778412416
cos(1/ 6 )= 0.986143231562925
cos(1/ 7 )= 0.9898132604466151
cos(1/ 8 )= 0.992197667229329
cos(1/ 9 )= 0.993833508538892
cos(1/ 10 )= 0.9950041652780258
cos(1/ 11 )= 0.9958706137005628
cos(1/ 12 )= 0.9965297867005595
cos(1/ 13 )= 0.9970428786964483
cos(1/ 14 )= 0.9974500640249152
cos(1/ 15 )= 0.9977786007011223
cos(1/ 16 )= 0.9980475107000991
cos(1/ 17 )= 0.9982703950127647
cos(1/ 18 )= 0.9984571869987445
cos(1/ 19 )= 0.9986152781425827
cos(1/ 20 )= 0.9987502603949663
cos(1/ 21 )= 0.9988664273811704
cos(1/ 22 )= 0.9989671200045982
cos(1/ 23 )= 0.9990549693005716
cos(1/ 24 )= 0.9991320700239181
raiz: 0.6366197723675813
```

c. Sean $f(t) = 3\sin^3 t - 1$ y $g(t) = 4\sin(t)\cos(t)$ para $t \geq 0$ 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.

```
punto 3
Sean f(t) = 3sin³t −1 y g(t) = 4sin(t)cos(t) parat≥0 las ecuaciones paramétricas que describe el movimiento en una parti
cula. Utilice un método de solución numérico con error de 10⁻¹⁰ para determinardonde las coordenadas coiciden y muestre
gráficamente la solución
uno de los puntos:  1.1864950215819903
otro de los puntos:  2.892651245274954
```

Comprobación de puntos:



2. Utilice el algoritmo de Steffensen para resolver $x^2 - cosx$ y compararlo con el método de Aitken con Tolerancia de 10^{-8}, 10^{-16}, realice una gráfica que muestre la comparación entre los métodos.

```
[1] 0.8241329
i=  3  x= 0.9
i=  4  x= 0.8333333
i=  5  x= 0.8333333
i=  6  x= 0.8333333
i=  7  x= 0.8125
i=  8  x= 0.8229167
i=  9  x= 0.8229167
i=  10  x= 0.8229167
i=  11  x= 0.8242188
i=  12  x= 0.8242188
i=  13  x= 0.8242188
i=  14  x= 0.8242188
i=  15  x= 0.8241374
i=  16  x= 0.8241374
i=  17  x= 0.8241374
Resultado sin aceleracion:  0.8241329
Resultado con aceleracion:  0.8241374
Valor real de la solucion: 0.8241323123
```