Taller 1 Análisis Númerico

Gabriel Andrés Niño Carvajal - Juliana Garcia Mogollon

14 de febrero de 2020

1 Problema De Redondeo

Problema: Suponga que un dispositivo solo puede almacenar únicamente los cuatro primeros dígitos decimales de cada número real, y trunca los restantes (esto es redondeo inferior). Calcule el error de redondeo si se quiere almacenar el número 536.78.
Solución: "Si n es la cantidad de enteros del número normalizado con potencias de 10 y m es la cantidad de cifras decimales que se pueden almacenar en un dispositivo, entonces el error de redondeo absoluto esta acotado por" [1]: $|E| < 1 * 10^{n-m}$
Se procede a resolver el problema por este método. El número debe ser expresado en forma normalizada: $x = 536.78, x = 0.5367810^3$

Lo siguiente es descomponer el número en dos partes:

$$x = 0.536710^3 + 0.0000810^3$$

El dispositivo almacena:

$$= 0.536710^3$$

El error de redondeo sería [1]:

$$E = 0.0000810^3 = 0.810^(3 - 4) = 0.810^{(-1)}$$

El algoritmo fue implementado en el lenguaje R:

```
#Error de Redondeo
###############
#Autores:
#Gabriel Niño
#Juliana Garcia
#Taller 1 - Analisis Numerico
```

```
##############
rm(list=ls())
valreal<-536.78
valaprox<-536.7
E<-abs(valreal-valaprox)
print(E)
#######################
```

El resultado del error de redondeo fue:

$$0.08 = 0.810^{(-1)}$$

2 Problema Raíz Cuadrada de 7

Implemente en cualquier lenguaje el siguiente algoritmo que sirve para calcular la raíz cuadrada. Aplíquelo para evaluar la raíz cuadrada de 7, analice su precisión, como podría evaluar la convergencia y validez del algoritmo.

Algoritmo: Raíz cuadrada
Entra: n Dato
E Error permitido
x Valor inicial
Sale: y Respuesta calculada con error E

$y \leftarrow \frac{1}{2}(x+\frac{n}{x})$

Repetir mientras |x - y| > E
    x ← y
    $y \leftarrow \frac{1}{2}(x+\frac{n}{x})$
Fin

Ilustración 1 - Algoritmo Raíz

Para evaluar la convergencia y validez del algoritmo se puede utilizar la potenciación ya que siendo la operación inversa de la radicación debería dar como resultado el dato inicial al cual se le iba a hallar la raíz, en este caso 7
El algoritmo fue implementado en el lenguaje R:

```
###############
#Autores:
#Gabriel Niño
#Juliana Garcia
#Taller 1 - Analisis Numerico
##############
rm(list=ls()) #Borrar los datos guardados en memoria
x<-3 #Valor Inicial
n<-7 #Dato
E<-1e-5 #Error Permitido
cont<-1 #Contador de iteración
```

```
valit<- c() #Valor de cada iteracion
valx<-c() #Valores que tomara x en cada iteracion
valy<-c() #Valores que tomara y en cada iteracion
valE<-c() #Valores del Error Absoluto en cada iteracion
vale<-c() #Valores del error relativo en cada iteracion
valval<-c() # Validacion de la raiz en en cada iteracion
valaprox<-c() #Valor aproximado que es la media de las diferentes medidas
f<-function(x,n,E)
{
    1/2*(x+n/x) #Función para calcular la raiz cuadrada
}
  y<-f(x,n,E) #Resultado de la funcion
  valit<-c(valit,cont)
  valx<-c(valx,x)
  valy<-c(valy,y)
  vale<-c(vale,(abs(y-x)/abs(y))*100)
  valval<-c(valval,y^2)
  valaprox<-c(valaprox,mean(valy))
  precision<-c() #Precision del calculo

   repeat
    {
      cont<-cont+1
      x<-y
      y<-f(x,n,E)
      valit<-c(valit,cont)
      valx<-c(valx,x)
      valy<-c(valy,y)
      valval<-c(valval,y^2)
      vale<-c(vale,(abs(y-x)/abs(y))*100)
      valaprox<-c(valaprox,mean(valy))
      if(abs(x-y)<E) break
    }
  valE<-abs(y-valaprox)
  precision<-100-vale
  #Guardar resultados en una tabla de datos
  resultados<-data.frame("Iteracion"=valit,"X"=valx,"Y"=valy,"Error_Absoluto"=valE,"Error_Relativo"=va
  #Exportar los resultados
  write.table(resultados,file="ResultadosRaiz7.txt")
  #Imprimir resultados
  print(resultados)
##########################################################
```

Estos fueron los resultados de la prueba:

Iteración	X	Y	Error Absoluto	Error Relativo	Precisión	Validación (Y^2)
1	3	2.6666666666 66667	0.0209153556020 762	12.5%	87.5%	7.11111111111111
2	2.6666666666 66667	2.6458333333 33333	0.0104986889354 093	0.78740157480 3172%	99.2125984251 968%	7.00043402777778
3	2.6458333333 33333	2.6457513123 33596	0.0069991263807 2853	0.00310010230 337531%	99.9968998976 966%	7.00000000672744
4	2.6457513123 33596	2.6457513110 6459	0.0052493447855 4639	4.80531562566 483e-08%	99.9999999519 468%	7

Tabla 1 - Raíz de 7

El algoritmo convergió al siguiente resultado:

$$y = 2.64575131106459$$

Este resultado se eleva al cuadrado para evaluar la validez de la respuesta, pues puede que el algoritmo llegue a un punto en que converja, pero no necesariamente es correcto:

$$y^2 = 2.64575131106459^2 = 7$$

Se comprueba que el algoritmo converge y es da como resultado una respuesta válida. El error disminuye con cada iteración mientras la precisión, lo que significa que después de cada iteración el algoritmo se acerca a una respuesta más precisa. El último cálculo de la precisión dio el siguiente resultado:

$$precision = 99.9999999519468\%100\%$$

Siendo un valor tan cercano al 100%, esto soporta la conclusión de que la respuesta que da el algoritmo es válida.

3 Problema Aproximación Utilizando el Teorema de Taylor

Utilizando el teorema de Taylor hallar la aproximación de $e^0.5 concincocifrassignificativas.$
$Primerohayquedefinirel Teoremade Taylor[2] : f(x) = P_n(x) + R_n(x)$
$DondeP_n(x)$ es el polinomio de Taylor:

$$P_n(x) = f(0) + f'(0)x + (f''(0)x^2)/2 + +(f^{((n)}(0)x^n)/n!$$

Y $R_n(x)$ es el error de truncamiento.
Hay que tener en cuenta que la derivada

$$f(x) = e^x = f'(x)$$

y que

$$e^0 = 1.$$

Primero hay que hallar el criterio de error con 5 cifras significativas:

$$E_s = (0.510^{(2-5)})\% = 0.0005\%$$

Ahora aplicando el polinomio de Taylor la primera aproximación es:

$$e^{0.5} = e^0 = 1 = e^{0.5} = 1$$

Segunda aproximación:

$$e^{0.5} = e^0 + e^0(0.5) = e^{0.5} = 1 + 0.5 = 1.5$$

Tercera aproximación:

$$e^{0.5} = e^0 + e^0(0.5) + (e^0(0.5)^2)/2 = e^{0.5} = 1 + 0.5 + 0.125 = 1.625$$

Cuarta aproximación:

$$e^{0.5} \approx e^0 + e^0(0.5) + \frac{e^0(0.5)^2}{2} + \frac{e^0(0.5)^3}{3!} \rightarrow$$

$$e^{0.5} \approx 1 + 0.5 + 0.125 + 0.02083333333 = 1.645833333$$

Quinta aproximación:

$$e^{0.5} \approx e^0 + e^0(0.5) + \frac{e^0(0.5)^2}{2} + \frac{e^0(0.5)^3}{3!} + \frac{e^0(0.5)^4}{4!} \rightarrow$$

$$e^{0.5} \approx 1 + 0.5 + 0.125 + 0.02083333333 + 2.604166667 \times 10^{-3} = 1.666666666$$

Sexta aproximación:

$$e^{0.5} \approx e^0 + e^0(0.5) + \frac{e^0(0.5)^2}{2} + \frac{e^0(0.5)^3}{3!} + \frac{e^0(0.5)^4}{4!} + \frac{e^0(0.5)^5}{5!}$$

$$e^{0.5} \approx 1 + 0.5 + 0.125 + 0.02083333333 + 2.604166667 \times 10^{-3} + 2.604166667 \times 10^{-4} = 1.666927083$$

El algoritmo fue implementado en el lenguaje R:

```
###############
#Autores:
#Gabriel Niño
```

```
#Juliana Garcia
#Taller 1 - Analisis Numerico
###############
#Teorema de Taylor
rm(list=ls()) #Borrar los datos guardados en memoria
n<-0 #Valor Inicial
x<-0.5 #Constante
y<-function(x,n){
  x^n/factorial(n)
}
valx<-c(x)
valy<-c(y(x,n)) #Valores que tomara y en cada iteracion
valtay<-c(sum(valy))#Valores de aproximacion del Teorema de Taylor
temp<-sum(valy) #Auxiliar
valn<-c(n) #Valores de n en cada iteracion
valE<-c(abs(exp(x)-mean(valtay))) #Valores del Error Absoluto en cada iteracion
vale<-c(abs(exp(x)-mean(valtay))/mean(valtay)) #Valores del error relativo en cada iteracion
n<-n+1
repeat{
  valx<-c(valx,x)
  valy<-c(valy,y(x,n))
  valtay<-c(valtay,sum(valy))
  valn<-c(valn,n)
  valE<-c(valE,abs(exp(x)-mean(valtay)))
  vale<-c(vale,abs(exp(x)-mean(valtay))/mean(valtay))
  temp<-sum(valy)
  n<-n+1
  if(n>5) break
}
precision<-100-vale
#Guardar resultados en una tabla de datos
resultados<-data.frame("X"=valx,"N"=valn,"Y"=valy,"Taylor"=valtay,"Error_Absoluto"=valE,"Error_Relativ
#Exportar los resultados
write.table(resultados,file="ResultadosTeoremaTaylor.txt")
#Imprimir resultados
print(resultados)
#########################################################
```

Estos fueron los resultados de la prueba:

X	N	Y	Taylor	Error Absoluto	Error Relativo	Precisión
0.5	0	1.0000000000	1.000000	0.6487213	0.64872127	99.35128
0.5	1	0.5000000000	1.500000	0.3987213	0.31897702	99.68102
0.5	2	0.1250000000	1.625000	0.2737213	0.19907002	99.80093
0.5	3	0.0208333333	1.645833	0.2060129	0.14279597	99.85720
0.5	4	0.0026041667	1.648438	0.1648671	0.11110735	99.88889
0.5	5	0.0002604167	1.648698	0.1373931	0.09090888	99.90909

Tabla 2 - Teorema de Taylor

La precisión de la aproximación se va aumentando con cada iteración, se puede deducir que si el algoritmo continua, posiblemente se llegue a una aproximación que tenga una precisión casi del 100%.
Se hizo una gráfica combinando el teorema de Taylor y la función exponencial. Rojo para exponencial, azul para Taylor y y un punto en $e^{0.5}$:

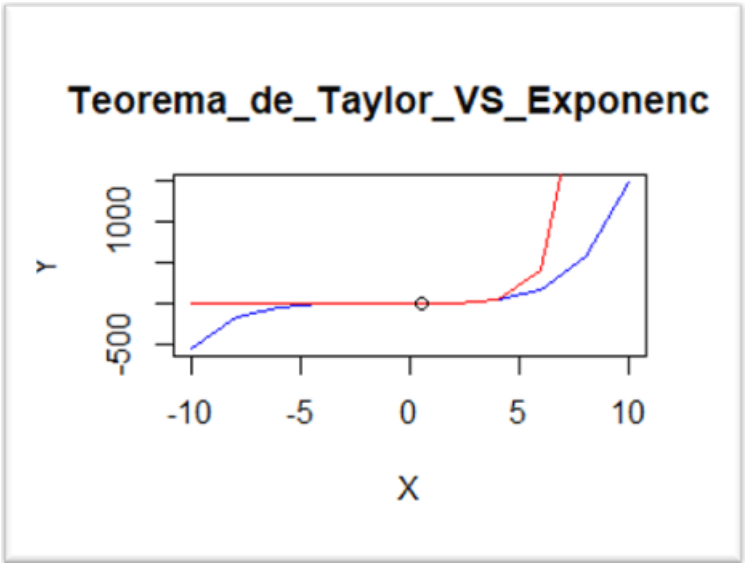

Ilustración 2 - Taylor vs Exponencial

El código fue implementado en R:

```
rm(list=ls())
x<-c(-10,-8,-6,-4,-2,0,2,4,6,8,10)
f<-function(x){
  1+x+x^2/2+x^3/factorial(3)+x^4/factorial(4)+x^5/factorial(5)
}
plot(x,f(x),main="Teorema_de_Taylor_VS_Exponencial",xlab="X",ylab="Y",type="l",col="blue")
points(x,exp(x),type="l",col="red")
points(0.5,exp(0.5))
#########################################################
```

4 Error de la Aritmética Computacional

El error de redondeo en la aritmética computacional se debe a que los datos pueden llegar a ser inexactos ya que los dispositivos no pueden almacenar todas las cifras [1]. Las operaciones aritméticas pueden producir resultados que no se pueden guardar ni representar en dispositivos de almacenamiento, y a medida que se ejecutan las operaciones aritméticas el error puede llegar a crecer de manera muy significativa [1]. Hay 3 fuentes de error en la aritmética computacional:

- Análisis: Se producen errores inherentes que son provocados por suposiciones, omisiones o simplificaciones mal hechas.
- Diseño: Se producen errores de truncamiento que son provocados utilizar procedimientos simplificados dentro de los métodos numéricos establecidos o al utilizar algoritmos iterativos.
- Instrumentación: Se producen errores de redondeo en dispositivos que no tengan la capacidad de almacenamiento para procesar datos reales con muchas cifras decimales.

5 Épsilon de la máquina

El épsilon de la máquina es la distancia entre 1 y el número de punto flotante mayor que 1. Para el punto flotante de precisión doble se tiene que:

$$E_{maq} = 2^{(-52)}$$

Donde E_{maq} representa épsilon[3].

6 Problema Calcular Tamaño del Error

Calcule el tamaño del error dado por las operaciones aritméticas, para la solución del siguiente problema:
La velocidad de una partícula es constante e igual a 4m/s, medida con un error de 0.1m/s durante un tiempo recorrido de 5seg medido con un error de 0.1seg. Determine el error absoluto y el error relativo en el valor de la distancia recorrida.

Velocidad: $v = 4, E_v = 0.1$
Tiempo: $t = 5, E_t = 0.1$
Distancia: $d = vt = 4x5 = 20$
Error absoluto: $E_d = vE_t + tE_v = 4(0.1) + 5(0.1) = 0.9$
Rango de variación de la distancia: $20 - 0.9d20 + 0.919.1d20.9$
Error relativo: $e_d = E_v/v + E_t/t = 0.1/4 + 0.1/5 = 0.045 = 4.5\%$

7 Problema Evaluar Polinomio

Evaluar el valor de un polinomio es una tarea que involucra para la maquina realizar un número de operaciones la cual debe ser mínimas. Como se puede evaluar el siguiente polinomio con el número mínimo de multiplicaciones. $P(x) = 2x^4 - 3x^2 + 3x - 4$ en $x_0 = -2$
Para encontrar la solución a este problema deberemos de usar el Teorema de Horner. Este es un método que consiste en determinar de manera eficaz el valor del polinomio de grado n utilizando el menor número de productos.
Código implementado en R:

```
###############
#Teorema Horner
#Autores:
#Gabriel Niño
#Juliana Garcia
#Taller 1 - Analisis Numerico
##############

coef=c(2,0,-3,3,-4)
valor=0
x_0=-2
i=1

numDeSumas = 0
numDeMultiplicaciones = 0

while(i<=5){
  valor=valor*x_0+coef[i]
  i=i+1;
  numDeSumas = numDeSumas + 1
  numDeMultiplicaciones = numDeMultiplicaciones +1
}
print(valor)
cat("El numero total de sumas es :",numDeSumas,"\n")
cat("El numero total de Multiplicaciones es :",numDeMultiplicaciones,"\n")
#########################################################
```

Resultado por consola:

```
 > print(valor)
[1] 10
> cat("El numero total de sumas es :",numDeSumas,"\n")
El numero total de sumas es : 5
> cat("El numero total de Multiplicaciones es :",numDeMultiplicaciones,"\n")
El numero total de Multiplicaciones es : 5
```

8 Problema Silueta Perrito

Reconstruir la silueta del perrito utilizando la menor cantidad de puntos para reproducir el dibujo del contorno completo del perrito sin bigotes, con la información dada:
Coordenadas:
$y = c(3, 3.7, 3.9, 4.5, 5.7, 6.69, 7.12, 6.7, 4.45, 7, 6.1, 5.6, 5.87, 5.15, 4.1, 4.3, 4.1, 3)$
$x = c(1, 2, 5, 6, 7.5, 8.1, 10, 13, 17.6, 20, 23.5, 24.5, 25, 26.5, 27.5, 28, 29, 30)$

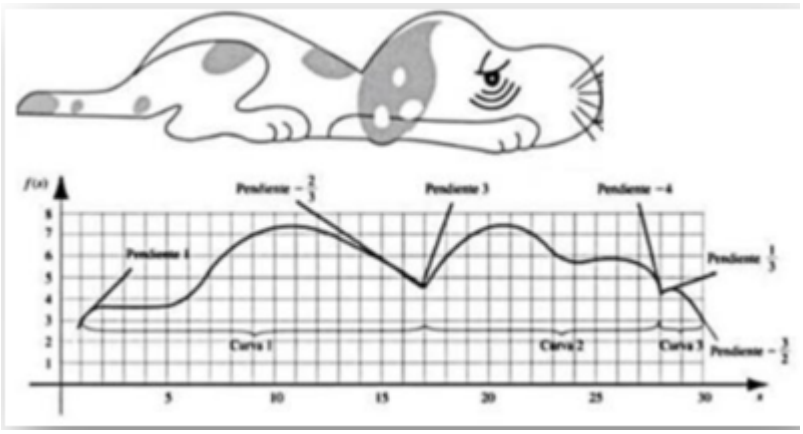

Ilustración 3 - Silueta Perro

Solución: Para la correcta realización de este punto del taller fue necesario aplicar ciertos conocimientos adquiridos en la clase de Análisis Numérico, específicamente el de interpolación con splines[13]. Es importante destacar que para realizar este ejercicio en primera instancia se siguieron las instrucciones del problema (utilizar las coordenadas dadas por la profesora). Sin embargo, el grupo llegó a la conclusión de crear las coordenadas para de esta manera aproximarnos lo máximo posible a la figura original [Ilustración 2]. Dicho resultado se ve reflejado en la ilustración 3.
Código implementado en R:

```
###############
#Silueta Perrito
#Autores:
#Gabriel Niño
#Juliana Garcia
#Taller 1 - Analisis Numerico
##############

rm(list=ls())

x = c(0.5, 1.1, 5.4, 7.5, 10.6, 16, 17.8, 21, 24, 27, 30, 32, 31.2, 30, 27.5, 26.5, 24.7, 22, 20.5, 18
y = c(2.1, 3, 3.8, 05.5, 7.5, 6, 5.2, 7.2, 6.9, 5.4, 5.2, 4.3, 2.5, 1.50, 0.95, 00.55, 00.52, 1, 00.90
y_1= y[1:7]; x_1 = x[1:7]
y_2= y[7:12]; x_2 = x[7:12]
y_3= y[12:14]; x_3 = x[12:14]
y_4= y[14:15]; x_4 = x[14:15]
y_5= y[15:18]; x_5 = x[15:18]
y_6= y[18:20]; x_6 = x[18:20]
y_7= y[20:25]; x_7 = x[20:25]
y_8= y[25:27];  x_8 = x[25:27]
y_9= y[27:28];  x_9 = x[27:28]
y_10= y[28:30]; x_10 = x[28:30]

num_puntos = length(y)
```

```
color = 1
linea = spline(y_3,x_3)
lin = linea$x
linea$x = linea$y
linea$y = lin
plot(x, y, main = paste("Silueta Perrito: ", num_puntos, "puntos"),xlim=c(0,35),ylim=c(0,10))
lines(spline(x_1, y_1), col = color)
lines(spline(x_2, y_2), col = color)
lines(spline(x_4, y_4), col = color)
lines(spline(x_5,y_5), col = color)
lines(spline(x_6, y_6), col = color)
lines(spline(x_7, y_7), col = color)
lines(spline(x_8, y_8), col = color)
lines(spline(x_9, y_9), col = color)
lines(spline(x_10, y_10), col = color)
lines(linea, col = color)
#########################################################
```

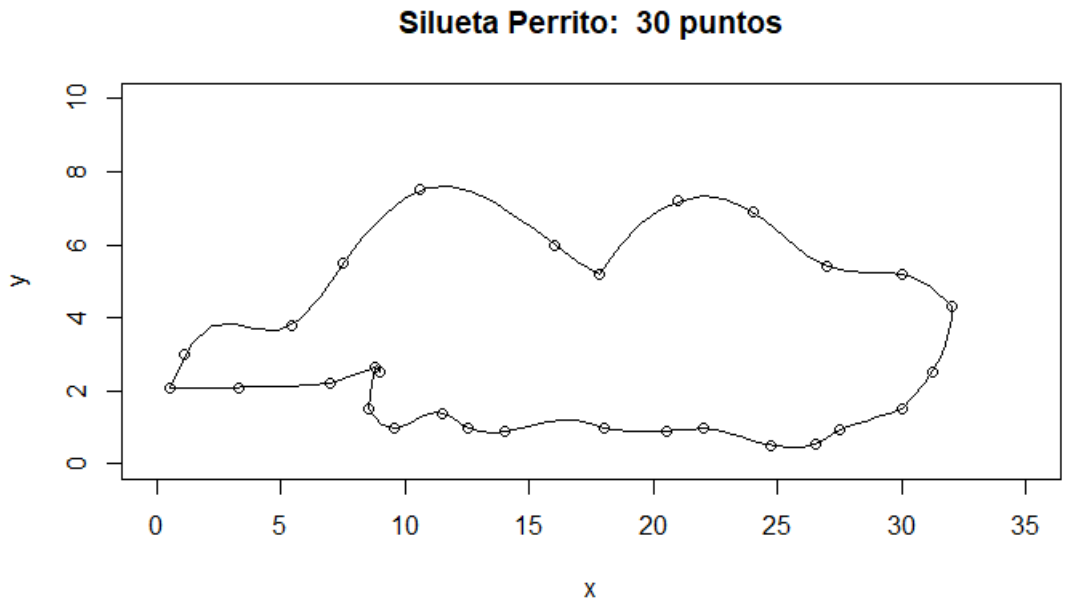

Ilustración 4 - Silueta Perro Graficada en R

Parte 2 del Taller

9 Teorema De Horner

a. Utilice el método de inducción matemática para demostrar el resultado del método 2

b. Implemente en R o Python para verificar los resultados del método de Horner
El código fue implementado en R:

```
###############
#Teorema Horner
#Autores:
#Gabriel Niño
#Juliana Garcia
#Taller 1 - Analisis Numerico
###############

coef=c(2,0,-3,3,-4)
valor=0
x_0=-2
i=1

numDeSumas = 0
numDeMultiplicaciones = 0

while(i<=5){
  valor=valor*x_0+coef[i]
  i=i+1;
  numDeSumas = numDeSumas + 1
  numDeMultiplicaciones = numDeMultiplicaciones +1
}
print(valor)
cat("El numero total de sumas es :",numDeSumas,"\n")
cat("El numero total de Multiplicaciones es :",numDeMultiplicaciones,"\n")
#########################################################
```

c. Evaluar en $x = 1.0001$ con $P(x) = 1 + x + x^2 + ... + x^{50}$. Encuentre el error de cálculo al compararlo con la expresión equivalente $Q(x) = (x^{51} - 1)/(x - 1)$
El código fue implementado en R:

```
###############
#Autores:
#Gabriel Niño
#Juliana Garcia
#Taller 1 - Analisis Numerico
###############


P_x = function(x)
{
  return (1 + x + x^2 + x^3 + x^4 + x^5 + x^6 + x^7 + x^8 + x^9 + x^10
          + x^11 + x^12 + x^13 + x^14 + x^15 + x^16 + x^17 + x^18 + x^19 +
            x^20 + x^21 +x^22 + x^23 + x^24 + x^25 + x^26 + x^27 + x^28 + x^29 +
            x^30 + x^31 + x^32 + x^33 + x^34 + x^35 + x^36 + x^37 + x^38 + x^39 +
            x^40 + x^41 + x^42 + x^43 + x^44 + x^45 + x^46 + x^47 + x^48 + x^49 +
```

```
            x^50)
}


cat("Este es el resultado por P(x):",
    round(P_x(1.0001), 4),
    "\n")

Q_x = function(x)
{
  return((x^51-1)/(x-1))

}

cat("Este es el resultado por Q(x):",
    round(Q_x(1.0001), 4),
  "\n")

valor_real = round(P_x(1.0001), 4)
Valor_exp = round(Q_x(1.0001), 4)

cat("El error absoluto del calculo es de: ", abs(Valor_exp - valor_real), ".\n")
##########################################################
```

El resultado por consola:

```
> Este es el resultado por P(x): 51.1277
> Este es el resultado por Q(x): 51.1277

> valor_real = round(P_x(1.0001), 4)
> Valor_exp = round(Q_x(1.0001), 4)
> El error absoluto del calculo es de:  0 .
```

10 Números Binarios

a) Encuentre los primeros 15 bits en la representación binaria de
Código implementado en R:

```
###############
#Pi en Binario
#Autores:
#Gabriel Niño
#Juliana Garcia
#Taller 1 - Analisis Numerico
###############
rm(list=ls()) #Borrar los datos guardados en memoria
#Primeros 15 bits decimales de Pi
bitsEntero<-c()
bitsDecimal<-c()
```

```
entero<-floor(pi)
decimal<-pi-entero
aux<-0
#Hayando los bits enteros
repeat{
  aux<-entero%%2
  entero<-entero%/%2
  bitsEntero<-c(aux,bitsEntero)
  if(entero == 1){
    bitsEntero<-c(entero,bitsEntero)
    break
  }
}
#Hayando bits decimales
repeat{
  if((length(bitsEntero)+length(bitsDecimal))>=15) break
  decimal<-decimal*2
  aux<-floor(decimal)
  bitsDecimal<-c(bitsDecimal,aux)
  if(aux == 1){
    decimal<-decimal-aux
  }
}
print(bitsEntero)
print(bitsDecimal)
#########################################################
```

El resultado por consola:

```
 > print(bitsEntero)
[1] 1 1
> print(bitsDecimal)
 [1] 0 0 1 0 0 1 0 0 0 0 1 1 1
```

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

```
###############
 #De Binario a base 10
#Autores:
#Gabriel Niño
#Juliana Garcia
#Taller 1 - Analisis Numerico
###############
calcular_la_parte_entera = function(X){
  cont_potencias = 0
  digito = 0
  num=0
```

```
  while(X > 0){
    digito = X %% 10

    num = num + (digito*(2^cont_potencias))

    cont_potencias = cont_potencias + 1

    X = X %/% 10
  }

  return(num)}
calcular_la_parte_decimal = function(X){
  X_2 = X
  X_2
  cont_potencias_2 = 0
  nums = 0
  digito = 0
  total=0

  while(X > 0){
    nums = nums+1

    X = X %/% 10
  }
  cont_potencias_2 =-nums;

  while(X_2 > 0){
    digito = X_2 %% 10

    total = total + (digito*(2^cont_potencias_2))

    cont_potencias_2 = cont_potencias_2 + 1

    X_2 = X_2 %/% 10
  }
  return(total)}
cat("Numero 1: ",calcular_la_parte_entera(10111) + calcular_la_parte_decimal(010101))
cat("Numero 2: ",calcular_la_parte_entera(1011) + calcular_la_parte_decimal(101),"\n")
cat("Numero 3: ",calcular_la_parte_entera(111) + calcular_la_parte_decimal(111111))
cat("Numero 4: ",calcular_la_parte_entera(101010101),"\n")
#########################################################
```

El resultado por consola:

```
Numero 1:  23.65625
Numero 2:  11.625
Numero 3:  7.984375
Numero 4:  341
#########################################################
```

c) Convierta los siguientes números de base 10 a binaria: 11.25;2/3; 30.6; 99.9
El código implementado en R:

```
#De Base 10 a binaria

calcular_numero_entero_a_binario = function(x)
{
  binario=0
  expo=1
  while(x>0)
  {

    digito = x%%2
   binario =  binario+expo*digito
    expo=expo*10
    x = x%/%2

  }
  return (binario)
}


calcular_numero_decimal_a_binario = function(x,numero_de_bits)
{
  num=0
  for (i in 1:numero_de_bits)
  {
    x = 2*x;


    if(x < 1)
    {
      num = paste(num,"0",sep="")

    }
    else
    {
      num = paste(num,"1",sep="")
      x = x - 1;
    }
  }
  return(num)
}

cat("1er numero en binario: ", calcular_numero_entero_a_binario(11),".",calcular_numero_decimal_a_bina
cat("2do numero en binario: ",calcular_numero_decimal_a_binario(0.6666666666,08), "\n")
cat("3er numero en binario: ", calcular_numero_entero_a_binario(30),".",calcular_numero_decimal_a_bina
cat("4to numero en binario: ", calcular_numero_entero_a_binario(99),".",calcular_numero_decimal_a_bina
########################################################
```

El resultado por consola:

```
1er numero en binario:  1011 . 001000000
2do numero en binario:  010101010
3er numero en binario:  11110 . 010011001
4to numero en binario:  1100011 . 011100110
```

11 Épsilon De La Máquina

1. ¿Cómo se ajusta un número binario infinito en un número finito de bits?
Con el método de punto flotante donde los números se expresan en la siguiente notación: $n = fx10^e$
Donde f es la fracción o mantisa y e es un entero positivo o negativo conocido como exponente. "El intervalo está determinado por el número de dígitos del exponente y la precisión esta determinada por el número de dígitos de la fracción" [5].
Cuando el número binario infinito se quiere representar en un número finito de bits es necesario truncarlo o redondearlo, pero siempre teniendo en cuenta el error que estas operaciones pueden generar. Este error se conoce como error de redondeo tanto para aproximación por truncamiento como por redondeo [6].
"En el modelo de la aritmética de computadora IEEE, el error de redondeo relativo no es más de la mitad del épsilon de la máquina" [3]:
$(|fl(x) - x|)/(|x|) < 1/2E_{maq}$

2. ¿Cuál es la diferencia entre redondeo y recorte?
El recorte consiste en eliminar los bits que están más allá de cierto umbral [3] mientras que el redondeo aproxima el número que no puede ser representado al número más cercano que puede ser representado [5].

3. ¿Cómo se ajusta un número binario infinito en un número finito de bits?
Con el método de punto flotante donde los números se expresan en la siguiente notación: $n = fx10^e$
Donde f es la fracción o mantisa y e es un entero positivo o negativo conocido como exponente. "El intervalo está determinado por el número de dígitos del exponente y la precisión está determinada por el número de dígitos de la fracción" [5].

Cuando el número binario infinito se quiere representar en un número finito de bits es necesario truncarlo o redondearlo, pero siempre teniendo en cuenta el error que estas operaciones pueden generar. Este error se conoce como error de redondeo tanto para aproximación por truncamiento como por redondeo [6].
"En el modelo de la aritmética de computadora IEEE, el error de redondeo relativo no es más de la mitad del épsilon de la máquina" [3]:
$(|fl(x) - x|)/(|x|) < 1/2E_{maq}$

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

Parte Entera	Mantisa	Exponente
4	0	-1

Tabla 3 - Número Punto Flotante

La notación científica sería [7]:
$410^{(-1)}$

5. Error de redondeo En el modelo de la aritmética de computadora IEEE, el error de redondeo relativo no es más de la mitad del épsilon de máquina:
$(|fl(x) - x|)/(|x|) < 1/2E_{maq}$
Teniendo en cuenta lo anterior, encuentre el error de redondeo para x = 0.4
$(|410^{(} - 1) - 0.4|)/(|0.4|) = 0$

6. Verificar si el tipo de datos básico de R y Python es de precisión doble IEEE y revisar en R y Phython el formato long.
R maneja un tipo de atributo conocido como numérico que son números de punto flotante de precisión doble y se escriben en la siguiente notación [8]: $4e - 1$
En este caso el 10 es reemplazado por e y el número que viene después de este es el exponente.
R no maneja un tipo de dato long, sino que representa los números muy grandes con notación de punto flotante. Ejemplo: $4e + 1$ donde e es 10
Python maneja un tipo de dato conocido como float que son números de punto flotante de precisión doble. También se puede utilizar notación científica, y añadir una e (de exponente) para indicar un exponente en base 10 [9]. Ejemplo: $0.1e - 3$
Python maneja un tipo de dato conocido como long que es un número entero en caso de desbordamiento que permite almacenar números muy grandes. Un número puede almacenarse como long poniendo una L al final del número [9]. Ejemplo: 23L

7. Encuentre la representación en número de máquina hexadecimal del número real 9.4.
La representación hexadecimal de 9.4 es 9.6666666666668

8. Encuentre las dos raíces de la ecuación cuadrática $x^2+9^12x = 3.Intenteresolverelproblemausandolaaritméticadeprec$
$Pararesolverelproblemasevaautilizarlaecuacióncuadrática$:
$x = (-b(b^2 - 4ac))/2a$
Donde: $a = 1 = 110^0, b = 9^{12} = 2.8210^{11}, c = -3 = -310^0$
El procedimiento es:

$$x = (-(2.8210^{11})((2.8210^{11})^2 - 4(1)(-3)))/(2(1))$$

Valor Real	Valor aproximado	Error Absoluto	Error Relativo
9^12	2.82*10^11	429536481	0.1523179%

Tabla 4 - Aritmética Precisión Doble 1

Valor Real	Valor aproximado	Error Absoluto	Error Relativo
(9^12)^2	(2.82*10^11)^2	2.424431*10^20	0.3048678%

Tabla 5 - Aritmética Precisión Doble 2

$$x = (-(2.8210^{11})((7.9510^{22}) + 12))/2$$

Para la suma:

$1210^0 + 7.9510^{22} = (1210^{(0-22)} + 7.95) * 10^{22}$

$$(12 * 10^{(-22)} + 7.95) * 10^{22} = (12 * 10^{(-22)}) + 7.95) * 10^{22} = 7.95 * 10^{22}$$

Volviendo a la ecuación:

Valor Real	Valor aproximado	Error Absoluto	Error Relativo
(9^12)^2 + 12	(2.82*10^11)^2 + 12	2.424431*10^20	0.3048678%

Tabla 6 - Aritmética Precisión Doble 3

$$x = (-(2.82 * 10^{11})(7.95 * 10^{22}))/2 = (-(2.8210^{11})2.82 * 10^{11})/2$$

Valor Real	Valor aproximado	Error Absoluto	Error Relativo
Sqrt((9^12)^2 + 12)	Sqrt((2.82*10^11)^2 + 12)	429536481	0.1523179%

Tabla 7 - Aritmética Precisión Doble 4

$$x_1 = 0; x_2 = -9^{12} = -2.82 * 10^{11}$$

Las raíces de la ecuación son 0 y -b, donde b maneja los siguientes valores:

Valor Real	Valor aproximado	Error Absoluto	Error Relativo
-9^12	-2.82*10^11	429536481	0.1523179%

Tabla 8 - Aritmética Precisión Doble 5

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

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

Donde b es un número mayor que 100.
Tomando como referencia el punto 8, las raíces serían 0 y -b ya que $-10^{(-12)}$ es un número bastante pequeño.

$$x = \frac{-(b) \pm \sqrt{(b)^2 - 4(1)(-10^{-12})}}{2(1)}$$

$$x = \frac{-b \pm \sqrt{(b)^2}}{2}$$

$$x = \frac{-b \pm b}{2} \rightarrow x_1 = 0\ ;\ x_2 = -b$$

12 Raíces De Una Ecuación

1. Implemente en R o Python un algoritmo que le permita sumar únicamente los elementos de la sub matriz 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.
Código implementado en R:

```
###############
#Matriz Triangular Superior
#Autores:
#Gabriel Niño
#Juliana Garcia
#Taller 1 - Analisis Numerico
##############

matTriangular = function(tamanyo)
{
  suma = c()

  for (z in tamanyo)
  {
    mat= matrix(1,nrow = z, ncol = z)
    cont = 0

    for (i in 1:z)
    {
      for (j in 1:i)
      {
        cont = cont + mat[i,j]
      }
    }

    suma = c(suma,cont)
    cont = 0
  }

  return(suma)
}

val = c(5,10,15,20,25,40,50,60,65,75)

graph = matTriangular(val)
plot(val,graph,xlab = "Tamanyo",ylab = "sumasXmatriz", type = 'b',col = 4)
#########################################################
```

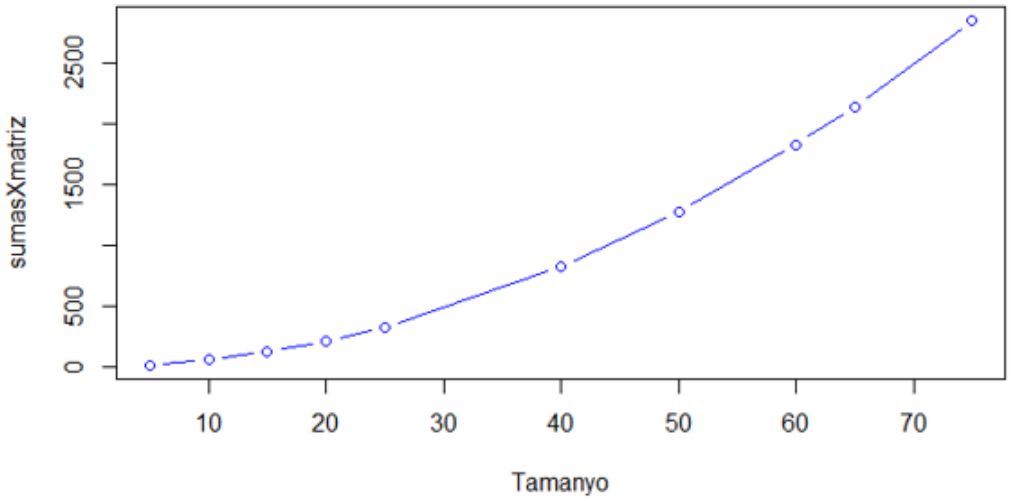

Ilustración 5 - sumasXMatriz

2. Implemente en R o Python un algoritmo que le permita sumar los $n^2 primerosnúmerosnaturalesalcuadrado.Imprima$
$CódigoimplementadoenR$:

```
###############
#Suma Números Naturales
#Autores:
#Gabriel Niño
#Juliana Garcia
#Taller 1 - Analisis Numerico
###############
rm(list=ls())
n<-100000
valn<-c()
valn2<-c()
cont<-1000
tiempos<-c()

repeat
{
  t<-proc.time()
  cont2<-1
  acum<-0
  repeat
  {
    acum<-acum+cont2^2
    cont2<-cont2+1
    if(cont2>cont) break
  }
  valn<-c(valn,cont)
  valn2<-c(valn2,acum)
```

```
  temp<-proc.time()-t
  tiempos<-c(tiempos,temp[["elapsed"]])
  cont<-cont+1000
  if(cont>n) break

datos<-data.frame("n"=valn,"Tiempos"=tiempos)
print(datos)
plot(valn,tiempos,main="Orden_Convergencia",xlab="n",ylab="Tiempo",ylim=c(0,0.05),type="l",col="red")
##########################################################
```

Con los datos obtenidos se hizo una tabla del valor n contra el tiempo de ejecución para ver el orden de convergencia. La variable n tuvo distintos valores avanzando de 1000 en 1000 hasta llegar a 1000000:

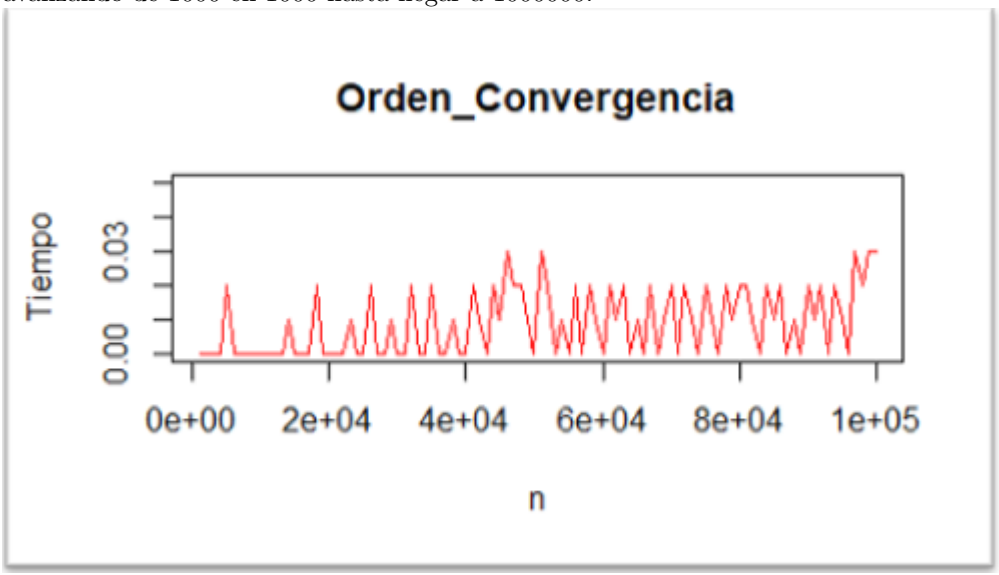

Ilustración 6 - Orden de convergencia

Se puede observar que en las primeras pruebas el tiempo de ejecución no fue muy largo pero a medida que n incrementa el tiempo de ejecución comienza a elevarse, se puede deducir que si n tomara valores más grandes de 1000000 el tiempo también aumentaría.

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.
Código implementado en R:

```
###############
#Altura Cohete
```

```
#Autores:
#Gabriel Niño
#Juliana Garcia
#Taller 1 - Analisis Numerico
###############
rm(list=ls())
E<-1e-6
t<-1
f<-function(t)
{
  6+2.13*t^2-0.0013*t^4
}
#Metodo de Biseccion
a<-1
b<-100
n<-0
nmax<-floor(1/log(2)*log(abs(a-b)/E))
x0<-a
x1<-b
repeat
{
  x2<-(x0+x1)/2
  if(f(x0)*f(x2) < 0)
  {
    x0<-x0
    x1<-x2
  }
  else
  {
    x0<-x2
    x1<-x1
  }
  n<-n+1
  if((abs(a-b)/(2^n))<=E) break
}
print(x2)
#########################################################
```

Altura máxima de:

$$y = 40.51263m$$

Luego se utilizó el método de Newton, el algoritmo fue implementado en R:

```
###############
#Altura Cohete - Newton
#Autores:
#Gabriel Niño
#Juliana Garcia
#Taller 1 - Analisis Numerico
```

```
###############
rm(list=ls())
E<-1e-6
t<-1
f<-function(t){
  6+2.13*t^2-0.0013*t^4
}
#Metodo de Newton
df<-function(t){
  4.26*t-(0.0052)*t^3
}
xk<-1
repeat{
  dx<-f(xk)/df(xk)
  xk1<-xk-dx
  xk<-xk1
  if(dx<=E) break
}
print(xk1)
#########################################################
```

El resultado fue:

$$y = 1.093m$$

El motivo por el cual los dos algoritmos no dan el mismo resultado puede deberse al hecho de que la ecuación no da una única respuesta, al ser una ecuación de grado 4 debe tener 4 respuestas posibles.

13 Convergencia Acelerada

Dada la sucesión $x_{n(n=0)}$ con $x_n = cos(1/n)$
1. Verifique el tipo de convergencia en $x = 1$ independiente del origen.
$cos(1/n)$ converge linealmente a 1. Primero hay que calcular el limite:

$$\lim_{n\to\infty} \frac{\cos\left(\frac{1}{n+1}\right) - 1}{\cos\left(\frac{1}{n}\right) - 1} = \frac{0}{0}$$

Da una indeterminación, es necesario utilizar l'Hôpital:

$$\lim_{n\to\infty} \frac{[-\frac{1}{(n+1)^2}]\sin\left(\frac{1}{n+1}\right)}{[-\frac{1}{n^2}]\sin\left(\frac{1}{n}\right)} =$$

$$\lim_{n\to\infty} [\frac{(n+1)^2}{n^2}] \frac{\sin\left(\frac{1}{n+1}\right)}{\sin\left(\frac{1}{n}\right)} =$$

$$\lim_{n\to\infty} [\frac{n}{n+1}]^2 \lim_{n\to\infty} \frac{\sin\left(\frac{1}{n+1}\right)}{\sin\left(\frac{1}{n}\right)} =$$

$$\lim_{n\to\infty} [\frac{n}{n+1}]^2 \lim_{n\to\infty} \frac{[-\frac{1}{(n+1)^2}]\cos\left(\frac{1}{n+1}\right)}{[-\frac{1}{n^2}]\cos\left(\frac{1}{n}\right)} =$$

$$\lim_{n\to\infty} [\frac{n}{n+1}]^2 \lim_{n\to\infty} [\frac{n}{n+1}]^2 \lim_{n\to\infty} \frac{\cos\left(\frac{1}{n+1}\right)}{\cos\left(\frac{1}{n}\right)} =$$

$$= 1 \cdot 1 \cdot 1 = 1$$

2. Compare los primeros términos con la sucesión $A_{n\,(n=0)}$

Valores de n	Cos(1/n)	Aitken
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	0.9966200
7	0.9898133	0.9974083
8	0.9921977	0.9979502
9	0.9938335	0.9983384
10	0.9950042	0.9986261

Tabla 9 - Coseno Aitken

En los primeros valores de n (1, 2, 3) se puede ver como el método de Aitken converge más rápido a 1.

Código implementado en R:

```
###############
#Aitken Coseno
#Autores:
#Gabriel Niño
#Juliana Garcia
#Taller 1 - Analisis Numerico
###############
rm(list=ls())
x<-1
vec<-1:22
valcoseno<-cos(1/vec)
valcos<-c()
calculos<-c()
cont<-1
repeat{
  a<-valcoseno[cont+2]-((valcoseno[cont+2]-valcoseno[cont+1])^2)/(valcoseno[cont+2]-2*valcoseno[cont+1
  calculos<-c(calculos,a)
  valcos<-c(valcos,valcoseno[cont])
  cont<-cont+1
  if(cont>20) break
}
datos<-data.frame("Coseno"=valcos,"Aitken"=calculos)
print(datos)
#########################################################
```

3. Sean $f(t) = 3sin^3t - 1$ y $g(t) = 4sin(t)cos(t)$ para $t > 0$ las ecuaciones paramétricas que describen el movimiento en una partícula. Utilice un método de solución numérico con error de $10^{(-6)}$ para determinar donde las coordenadas coinciden.

Se hizo una gráfica de las dos funciones. Para f(t) rojo y para g(t) azul:

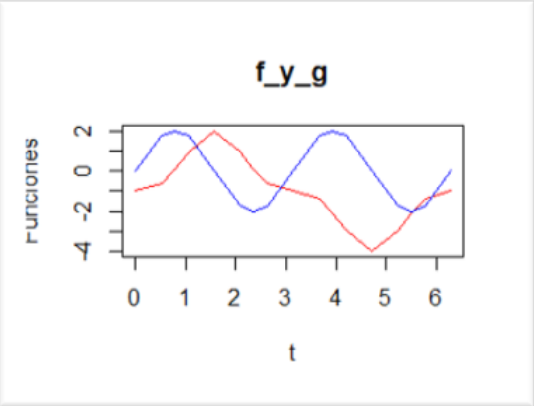

Ilustración 7 - f y g

Código implementado en R:

```
###############
#Polares
#Autores:
#Gabriel Niño
#Juliana Garcia
#Taller 1 - Analisis Numerico
##############

rm(list=ls())
t<-1
E<-1e-6
f<-function(t){
  3*(sin(t))^3-1
}
g<-function(t){
  4*cos(t)*sin(t)
}
h<-function(t){
  f(t)-g(t)
}
t<-seq(0,10,len=100)
x<-f(t)
y<-g(t)
plot(x,y,main="f_y_g",xlab="t",ylab="Funciones",type="l",col="red")
#########################################################
```

Hay que pasar de ecuaciones paramétricas a ecuaciones cartesianas:

$$y = 3sin^3t - 1 \rightarrow \left(\frac{y+1}{3}\right)^{\frac{\tilde{}}{3}} = sin^2t$$

$$x = 4\sin(t)\cos(t) \rightarrow \frac{x^2}{16} = sin^2(t)cos^2(t)$$

4. Utilice el algoritmo de Steffensen para resolver $x^2 - cosx$ y compararlo con el método de Aitken. Nota: Debe implementar los códigos en R, utilice Rcpp para mejorar el rendimiento.
Se implementó el algoritmo de Steffensen en R:

```
###############
#Steffensen vs Aitken
#Autores:
#Gabriel Niño
#Juliana Garcia
#Taller 1 - Analisis Numerico
##############
rm(list=ls())
x<-1
E<-1e-6
f<-function(x)
{
```

```
  x^2-cos(x)
}
valx<-c(x)
steff<-c()
esteff<-c()
Esteff<-c()
cont<-1
maxi<-300
repeat
{
  y<-x-(f(x))^2/(f(x+f(x))-f(x))
  steff<-c(steff,y)
  Esteff<-c(Esteff,abs(x-y))
  esteff<-c(esteff,abs(x-y)/abs(y)*100)
  x<-y
  valx<-c(valx,x)
  cont<-cont+1
  if(abs(x-y)<E) break
}
datosSteff<-data.frame("x"=valx,"y"=steff,"E"=Esteff,"e"=esteff)
print(datosSteff)
############################################################
```

Resultados:

x	Y	Error Absoluto	Error relativo
1.0000000	0.8645502	0.1354498	15.66709%
0.8645502	0.8645502	0.1354498	15.66709%

Tabla 10 - Resultados

Se implementó el algoritmo de Aitken en R:

```
###############
#Steffensen vs Aitken
#Autores:
#Gabriel Niño
#Juliana Garcia
#Taller 1 - Analisis Numerico
###############
rm(list=ls())
x<-1
E<-1e-6
f<-function(x){
  x^2-cos(x)
}
#Aitken-------------------------------------------------------------------------
x<-1
val2x<-c(x)
ait<-c()
eait<-c()
```

```
Eait<-c()
cont<-1
maxi<-300
a<-c()
repeat
{
  a<-c(a,f(cont))
  cont<-cont+1
  if(cont>maxi) break
}
cont<-1
repeat
{
  y<-a[cont+2]-((a[cont+2]-a[cont+1])^2)/(a[cont+2]-2*a[cont+1]+a[cont])
  ait<-c(ait,y)
  Eait<-c(Eait,abs(x-y))
  eait<-c(eait,abs(x-y)/abs(y)*100)
  x<-y
  val2x<-c(val2x,x)
  cont<-cont+1
  if(abs(x-y)<E) break
}
datosait<-data.frame("x"=val2x,"y"=ait,"E"=Eait,"e"=eait)
print(datosait)
```

Resultados:

X	Y	Error Absoluto	Error Relativo
1.000000	-9.218504	10.2185	110.8477%
-9.218504	-9.218504	10.2185	110.8477%

Tabla 11 - Resultados

14 Referencias

[1] Rodríguez Ojeda, Luis;Escuela Superior Politécnica del Litoral, Análisis Numérico un enfoque algorítmico con el soporte de Python, 2014.
[2] Herrera, Eddy;Pontificia Universidad Javeriana, Clase Análisis Numérico 1930, Bogotá, 2019.
[3] E. Herrera Daza, Análisis Numérico Primera Parte, Bogotá, 2020.
[4]Herrera, Eddy;Pontificia Universidad Javeriana, Problemas Análisis Numérico, Bogotá, 2020.
[5] B, Números de Punto Flotante, [En línea]. Available: http://materias.fi.uba.ar/6601/comaflotante.pdf. [Último acceso: 13 febrero 2020].
[6] ERRORES DE REDONDEO Y ARITMÉTICA DE PRECISIÓN FINITA, [En línea]. Available: https://rua.ua.es/dspace/bitstream/10045/16373/1/Microsoft%20Word%20-%201.REDONDEO%20Y%20PRECISION.pdf. [Último acceso: 13 febrero 2020].
[7] Matemáticas Discretas, Capítulo 3: PUNTO FLOTANTE, 22 noviembre 2017. [En línea]. Available: https://medium.com/@matematicasdiscretaslibro/cap%C3%ADtulo-3-punto-flotante-c689043db98b. [Último acceso: 13 febrero 2020].
[8] J. Ortega Sánchez, Introducción a R Sesión 2 Nociones Básicas, Julio 2009. [En línea]. Available: https://www.cimat.mx/ jortega/MaterialDidactico/TallerR09/VeranoClase2.pdf. [Último acceso: 13 febrero 2020].
[9] COVANTEC, 3.6. Tipo números, [En línea]. Available: https://entrenamiento-python-basico.readthedocs.io/es/latest/leccion3/tipo$_n umericos.html$.[$Últimoacceso$: $13 febrero 2020$].
[10]$Aritméticaencomaflotante.$, 2010.[$Enlínea$].$Available$: $https$: $//www.uv.es/varnau/AEC_5 20.pdf$.[$Últimoacceso$: