Informe Laboratorio: Análisis Numérico
Práctica No. 0

Estudiante: Jorge Sandoval
Código: 2182028
Grupo: B2
Escuela de Ingeniería de Sistemas e Informática
Universidad Industrial de Santander

20 de abril de 2021

1. Introducción

En este informe se encuentran la solución de los problemas propuestos para el laboratorio 0,para ello se realizo el código para calcular el redondeo a 3 dígitos de las sumatorias propuestas,también se encuentra la demostración de el calculo de las raíces cuadráticas con las formulas equivalentes, así mismo se encuentra los códigos que permiten desarrollar los ejercicios enunciados en la sección 3,3.

2. Desarrollo

2.1. Sumatorias

2.1.1. a.

Para la solución de este ejercicio se implemento un código de matlab,el cual consta de un ciclo for que va desde 1 hasta a 6 iterando el siguiente algoritmo $suma = suma + 1/(3^i)$ que representa el termino interior de la sumatoria.así mismo en el script de matlab se utiliza

```
function sumatoria
suma=0;
for i=1:6
suma=suma+1/(3^(7-i));
end
suma=round(suma,3);
disp(suma)
end
```

Figura 1: Ciclo for

la función round la cual permite redondear un numero al valor mas cercano.El resultado arrojado por esta linea de codigo es 0.4990

2.1.2. b.

Siguiendo la metodología aplicada para desarrollar el item anterior,para este caso solo se cambio en el código matlab la parte interna del ciclo,por la linea de código que representa la sumatoria de este ejercicio.la respuesta obtenida es 0.4990.

```
%sumatoria b
suma=0;
for i=1:6
    suma=round(suma+1/(3^(7-i)),3);
end

disp(suma)
```

Figura 2: Ciclo for

2.2. Demostración

Tenemos que

$$x_1 = \frac{-b + \sqrt{b^2 - 4ac}}{2a} \tag{1}$$

Multiplicando el denominador y numerador por $-b - \sqrt{b^2 - 4ac}$ tenemos

$$x_1 = \frac{-b + \sqrt{b^2 - 4ac}}{2a} * \frac{-b - \sqrt{b^2 - 4ac}}{-b - \sqrt{b^2 - 4ac}} \tag{2}$$

Despues de hacer las operaciones correspondientes nos queda

$$\frac{4ac}{2a(-b - \sqrt{b^2 - 4ac})} \tag{3}$$

Sacando factor común (-1) y factorizando tenemos

$$\frac{\cancel{(-1)}}{(-1)} \frac{-\overset{2}{\cancel{4}}\cancel{a}c}{\cancel{2a}(b + \sqrt{b^2 - 4ac})} \rightarrow \frac{-2c}{b + \sqrt{b^2 - 4ac}} = x_1 \tag{4}$$

Para X_2 tenemos

$$x_2 = \frac{-b - \sqrt{b^2 - 4ac}}{2a} \tag{5}$$

Multiplicando el denominador y numerador por $-b + \sqrt{b^2 - 4ac}$ tenemos

$$x_1 = \frac{-b + \sqrt{b^2 - 4ac}}{2a} * \frac{-b + \sqrt{b^2 - 4ac}}{-b + \sqrt{b^2 - 4ac}} \tag{6}$$

Después de hacer las operaciones correspondientes nos queda

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

Sacando factor común (-1) y factorizando tenemos

$$\frac{\cancel{(-1)}}{(-1)} \frac{-\overset{2}{\cancel{4}}\cancel{a}c}{\cancel{2a}(b - \sqrt{b^2 - 4ac})} \rightarrow \frac{-2c}{b - \sqrt{b^2 - 4ac}} = x_2 \tag{8}$$

3. Implementación en matlab

3.1. función floating point

Para realizar esta función fue necesario crear un algoritmo que permita convertir los números con parte decimal a binarios.

```
function [binario]=decimal_binario(number)
if number<0
number=number*-1;
end
parteE=floor(number);
parteD=rem(number,1);
j=1;
entero=floor(parteE/2);
sobra=rem(parteE,2);
binario(j)=num2str(sobra);

while entero>=1
    j=j+1;
    parteE=entero;
    entero=floor(parteE/2);
    sobra=rem(parteE,2);
    binario(j)=num2str(sobra);

end
j=j+1;
binario=fliplr(binario);
binario(j)=num2str('.');

if parteD==0
    binario(j+1)=num2str(0);
    decimal=1;
else
    decimal=0;
```

Figura 3: algoritmo para convertir a binario

Luego de hacer el script anterior se procedió a programar una función que permitiera pasar un numero a punto flotante en una computadora de 16 bits haciendo uso de el algoritmo visto en clase.

```
function P=floating_point_function_sandoval_jorge(n)
if n<0
    signo=num2str(1);
else

   signo=num2str(0);
end
suma=-1;
j=0;
binario=decimal_binario(n);
for i=2:length(binario)
    j=j+1;
    suma=suma+1;
    if binario(i)=='.'
    exponente=suma;
    j=j-1;
    else
        matissa(j)=binario(i);
    end
end
bias=2^(7-1)-1;
exp=exponente+bias;
expB=dec2bin(exp);


k=length(matissa);
valor=1;
manti(1)=num2str(0);
```

Figura 4: función punto flotante

A continuación se testeara la función con los números propuestos en el laboratorio para probar dicho script.

3.1.1. a.

```
>> floating_point_function_sandoval_jorge(1612.078125)

ans =

    '0100100110010011'
```

Figura 5: $1612{,}078125_{10}$ a punto flotante 16 bits

3.1.2. b.

```
>> floating_point_function_sandoval_jorge(6317.9136)

ans =

    '0100101110001010'
```

Figura 6: $6317{,}9136_{10}$ a punto flotante 16 bits

```
>> floating_point_function_sandoval_jorge(-962.0153)

ans =

    '1100100011100001'
```

Figura 7: $-962{,}0153_{10}$ a punto flotante 16 bits

3.2. relative and absolute error

Para desarrollar esta parte se creo un script en matlab que permite calcular el error relativo y el absoluto, para ello se tuvo que convertir de punto flotante a binario,el procedimiento de cada item es desarrollado mas adelante.

```
function error_relativo_absoluto(valorR,valorA)
Ep=abs(valorR-valorA);
Rp=abs(valorR-valorA)/abs(valorR);
P='el error absoluto es:';
Q='el error relativo es:';
disp(P)
disp(Ep)
disp(Q)
disp(Rp)
end
```

Figura 8: Script para calcular el error absoluto y relativo

3.2.1. a.

El procedimiento de pasar un numero punto flotante a decimal es el siguiente:
Numero punto flotante:0100100110010011
S=0
$1 + 2^{-1} + 2^{-4} + 2^{-7} + 2^{-8}$=1.57421875
2^{73-63}=2^{10}
=1.57421875*2^{10}=1612
luego se coloco el resultado obtenido en la función para obtener el error relativo y el absoluto.

```
>> error_relativo_absoluto(1612.078125,1612)
el error absoluto es:
    0.0781

el error relativo es:
   4.8462e-05
```

Figura 9: error absoluto y relativo

3.2.2. b.

El procedimiento de pasar un numero punto flotante a decimal es el siguiente:
Numero punto flotante:0100101110001010
S=0
$1 + 2^{-1} + 2^{-5} + 2^{-7}$=1.5390625
2^{75-63}=2^{10}
=1.5390625*2^{10}=6304
luego se coloco el resultado obtenido en la función para obtener el error relativo y el absoluto.

```
>> error_relativo_absoluto(6317.9136,6304)
el error absoluto es:
   13.9136

el error relativo es:
    0.0022
```

Figura 10: error absoluto y relativo

3.2.3. b.

El procedimiento de pasar un numero punto flotante a decimal es el siguiente:
Numero punto flotante:0100101110001010

S=0
$1 + 2^{-1} + 2^{-5} + 2^{-7}$=1.5390625
2^{75-63}=2^{10}
=1.5390625*2^{10}=6304
luego se coloco el resultado obtenido en la función para obtener el error relativo y el absoluto.

```
>> error_relativo_absoluto(6317.9136,6304)
el error absoluto es:
   13.9136

el error relativo es:
    0.0022
```

Figura 11: error absoluto y relativo

3.2.4. c.

El procedimiento de pasar un numero punto flotante a decimal es el siguiente:
Numero punto flotante:0100101110001010
S=0
$1 + 2^{-1} + 2^{-5} + 2^{-7}$=1.5390625
2^{75-63}=2^{10}
=1.5390625*2^{10}=6304
luego se coloco el resultado obtenido en la función para obtener el error relativo y el absoluto.

```
>> error_relativo_absoluto(6317.9136,6304)
el error absoluto es:
   13.9136

el error relativo es:
    0.0022
```

Figura 12: error absoluto y relativo

4. Improving the Quadratic Formula

En este caso se creo una función para hallar las raíces de la formula cuadrática con las recomendaciones dadas en el pdf del laboratorio,así mismo se dieron ciertos números donde se tenia que hallar sus respectivas raíces,los resultados se muestran a continuación.

```
function[x1,x2]=cuadratica(a,b,c)
if b>0
   x1=(-2*c)/(b+sqrt(b^2-4*a*c));
   x2=(-b-sqrt(b^2-4*a*c))/(2*a);
else
   x1=(-b+sqrt(b^2-4*a*c))/(2*a);
   x2=(-2*c)/(b-sqrt(b^2-4*a*c));
end
end
```

Figura 13: Función para calcular las raíces de un numero

4.1. a.

```
>> [x1,x2]=cuadratica(1,-1000.001,1)

x1 =

        1000


x2 =

   1.0000e-03

>>
```

Figura 14: Raíces de $x^2 - 1,000,001x + 1 = 0$

4.2. b.

```
>> [x1,x2]=cuadratica(1,-10000.0001,1)

x1 =

       10000


x2 =

   1.0000e-04

>> |
```

Figura 15: Raíces de $x^2 - 10,000,0001x + 1 = 0$

4.3. c.

```
>> [x1,x2]=cuadratica(1,-100000.00001,1)

x1 =

      100000


x2 =

   1.0000e-05

>>
```

Figura 16: Raíces de $x^2 - 100,000{,}00001x + 1 = 0$

4.4. d.

```
>> [x1,x2]=cuadratica(1,-1000000.000001,1)

x1 =

     1000000


x2 =

   1.0000e-06

>>
```

Figura 17: Raíces de $x^2 - 1000,000{,}000001x + 1 = 0$