

# Tarea1

Benjamin Rivera

September 6, 2020

## Contents

<b>1 Ejercicio 1</b>	<b>3</b>
1.1 Calcular el valor del $\epsilon$ de la máquina . . . . .	3
1.2 Dar la representación en notación científica (mantisa base 2, multiplicada por 2 elevada al exponente correspondiente) del número 5. . . . .	3
1.3 Dar la representación científica del número consecutivo a 5 en la computadora. Escribir la distancia $d_c$ entre 5 y su consecutivo. Expresar $d_c$ en términos del $\epsilon$ de la máquina. . . . .	3
1.4 Tenemos que el consecutivo de 5 es expresable como $5 + d_c$ . Si tenemos un $x$ real tal que $x \in (5, 5 + d_c)$ entonces la computadora representará a $x$ como $fl(x) = 5$ o $fl(x) = 5 + d_c$ . Escribir una cota para el error relativo para las dos posibles representaciones de $x$ . . . . .	4
1.5 Explique si los siguientes números tienen representación exacta en la computadora, es decir, si $fl(a_i) = a_i$ . . . . .	5
1.6 De una cota para el error relativo de las restas 4 y 5 respecto al verdadero valor. Suponga que $fl(x)$ se obtiene por redondeo hacia abajo ( <i>truncamiento</i> ) . . . . .	6
<b>2 Ejercicio 2</b>	<b>9</b>

## Tarea 1

Tarea 1 de Benjamín Rivera para el curso de **Métodos Numéricos** impartido por Joaquín Peña Acevedo. Fecha límite de entrega **6 de Septiembre de 2020**.

## Como ejecutar

**Requerimientos** Este programa se ejecuto en mi computadora con la version de **Python 3.8.2** y con estos [requerimientos](#)

### Jupyter

En caso de tener acceso a un *servidor jupyter*, con los requerimientos antes mencionados, unicamente basta con ejecutar todas las celdas de este *notebook*. Probablemente no todas las celdas de *markdown* produzcan el mismo resultado por las [Nbextensions](#).

### Consola

Habr  archivos e instrucciones para poder ejecutar cada uno de los ejercicios desde la consola.

### Si todo sale mal

En caso de que todo salga mal, tratare de dejar una copia disponible en **GoogleColab** que se pueda ejecutar con la versi n de **Python** de *GoogleColab*

# 1 Ejercicio 1

Supongamos que una computadora tiene 8 dígitos para representar la parte fraccionaria de un número de punto flotante.

## 1.1 Calcular el valor del épsilon de la máquina

En clase vimos que si conocemos la cantidad de bits  $p$  para representar la parte fraccionaria de la mantisa, entonces

$$\epsilon_m = (1.0)_2 \times 2^{-p} \quad (1)$$

Dado que sabemos que la máquina de este ejercicio usara *8bits* para representar la parte fraccionaria entonces, por 1, el  $\epsilon_m$  para este ejercicio es

$$\epsilon_m = (1.0)_2 \times 2^{-8} = 3.90625 \times 10^{-3}$$

```
[1]: print(2**(-8))
```

0.00390625

## 1.2 Dar la representación en notación científica (mantisa base 2, multiplicada por 2 elevada al exponente correspondiente) del número 5.

Sabemos que el número 5 se representa en binario como  $(101)_2$ . Como los números se prefieren normalizados entonces debemos representar  $(1.01)_2$  en la notación solicitada. Por lo que

$$5 = (101)_2 \times 2^0 = (1.01)_2 \times 2^2 = 1.25 \times 2^2$$

## 1.3 Dar la representación científica del número consecutivo a 5 en la computadora. Escribir la distancia $d_c$ entre 5 y su consecutivo. Expresar $d_c$ en términos del épsilon de la máquina.

Sabemos que el número consecutivo  $(5_c)$  a 5 en esta computadora es  $5 + \epsilon_m$ . Si extendemos toda la mantisa de 5 este se ve  $(1.01000000)_2$ . Y sumar  $\epsilon_m$  implica sumar 1 a la mantisa, específicamente en esta computadora, al *bit8* de la mantisa. De manera que

$$\begin{aligned} 5_c &= 5 + \epsilon_m = (1.01)_2 \times 2^2 + (1.0)_2 \times 2^{2-8} \\ &= ((1.01000000)_2 + (0.00000001)_2) \times 2^2 \\ &= (1.01000001)_2 \times 2^2 \\ &= 5.015625 \end{aligned}$$

por lo que la distancia entre 5 y su consecutivo  $5 + \epsilon_m$  es

$$d_c = (5 + \epsilon_m) - 5 = \epsilon_m \times 2^2 = (0.00000001)_2 \times 2^2 = 0.015625$$

**1.4 Tenemos que el consecutivo de 5 es expresable como  $5 + d_c$ . Si tenemos un  $x$  real tal que  $x \in (5, 5 + d_c)$  entonces la computadora representara a  $x$  como  $fl(x) = 5$  o  $fl(x) = 5 + d_c$ . Escribir una cota para el error relativo para las dos posibles representaciones de  $x$**

Error relativo

$$\left| \frac{fl(x) - x}{x} \right|$$

Sabemos que las dos posibles representaciones son el *truncamiento* y el *redondeo*, para dar la cota se deben calcular los valores minimos y maximos del error relativo. En general trabajaremos con

$$\left| \frac{fl(x) - x}{x} \right| = \left| \frac{fl(x)}{x} - 1 \right| \quad (2)$$

Antes de continuar es importante notar lo siguiente. Sabemos que, para ambos casos, el dominio de la función sera  $[5, 5 + d_c]$ . Además, notemos que el rango de  $\frac{fl(x)}{x}$ , para nuestro dominio y los dos valores que puede tomar  $fl(x)$ , es  $\left[ \frac{5}{5+d_c}, \frac{5+d_c}{5} \right]$ . Por ultimo, sabemos que  $\frac{fl(x)}{x}$  es descendente para  $x > 0$ .

**Truncamiento** Para el truncamiento  $fl(x) = 5$  por lo que debemos calcular la cota para

$$Er_- = \left| \frac{5}{x} - 1 \right|$$

En este caso, el rango de  $\frac{fl(x)}{x} = \frac{5}{x}$  es  $\left[ \frac{5}{5+d_c}, \frac{5}{5} = 1 \right]$  por lo que el rango de  $Er_-$  es  $\left[ 0, \left| \frac{5}{5+d_c} - 1 \right| \right]$ . Por iluminación divina, suponemos que esta función es ascendente, y por lo tanto  $Er_-(5) < Er_-(5 + d_c)$ . Primero evaluamos la función en estos puntos

$$Er_-(5) = \left| \frac{5}{5} - 1 \right| = 0 \quad Er_-(5 + d_c) = \left| \frac{5}{5 + d_c} - 1 \right| \sim 0.00311$$

Por lo que, usando el truncamiento, esta **función esta acotada** por  $[Er_-(5), Er_-(5 + d_c)]$

**Redondeo** Procedemos de manera similar al anterior. Para el redondeo tenemos que  $fl(x) = 5 + d_c$ , esto nos da la expresion

$$Er_+ = \left| \frac{5 + d_c}{x} - 1 \right|$$

Y de manera similar al anterior podemos ver que esta función es descendente, por lo que  $Er_+(5 + d_c) < Er_+(5)$

$$Er_+(5 + d_c) = \left| \frac{5 + d_c}{5 + d_c} - 1 \right| = 0 \quad Er_+(5) = \left| \frac{5 + d_c}{5} - 1 \right| \sim 0.00312$$

De manera que esta función queda acotada por  $[Er_+(5 + d_c), Er_+(5)]$

**1.5 Explique si los siguientes números tienen representación exacta en la computadora, es decir, si  $fl(a_i) = a_i$**

- $a_1 = \epsilon/2$

Sabemos que en el sistema de este ejercicio (8bits parte flotante)  $\epsilon_m = (1.0)_2 \times 2^{-8}$ , de manera que,

$$\epsilon_m/2 = \frac{(1.0)_2 \times 2^{-8}}{2} = (1.0)_2 \times 2^{-9}$$

por lo que, mientras que el rango del exponente sea suficiente (dado que el ejercicio solo da información del tamaño de la *matissa*), este número **es representable** en el sistema.

- $a_2 = 1 + \epsilon/2$

De manera que  $a_2$  se expresa como

$$a_2 = (1.0)_2 + \epsilon/2 = (1.0)_2 + (1.0)_2 \times 2^{-9} = (1.000000001)_2$$

pero como este sistema solo usa 8bits para la mantisa entonces, para este sistema

$$a_2 = (1.0)_2 + \epsilon/2 = (1.000000001)_2 = (1.0)_2$$

por lo que este número **no tiene representación** en este sistema.

- $a_3 = 1 - \epsilon$

Este número se escribe como

$$\begin{aligned} a_3 = 1 - \epsilon &= (1.0)_2 - (1.0)_2 \times 2^{-8} \\ &= (1.00000000)_2 - (0.00000001)_2 \\ &= (0.11111111)_2 \\ &= (1.1111111)_2 \times 2^{-1} \end{aligned}$$

por lo que este número **si tiene una representación** en este sistema.

- $a_4 = 1 - \epsilon/2$

Como con los anteriores, esta operación se expresa como

$$\begin{aligned} a_4 = 1 - \epsilon/2 &= (1.0)_2 - (1.0)_2 \times 2^{-9} \\ &= (1.00000000)_2 - (0.000000001)_2 \\ &= (0.11111111)_2 \\ &= (1.1111111)_2 \times 2^{-1} \end{aligned}$$

el cual **si tiene representación** en el sistema de este ejercicio.

- $a_5 = 1 - \epsilon/4$

En este inciso primero calcularemos  $\epsilon/4$ , para el cual expandimos lo siguiente

$$\epsilon/4 = \frac{(1.0)_2 \times 2^{-8}}{2^2} = (1.0)_2 \times 2^{-10}$$

el cual, mientras el exponente alcance, *si es representable* en el sistema. Por otro lado, el numero de este inciso nos da

$$\begin{aligned} a_5 = 1 - \epsilon/4 &= (1.0)_2 - (1.0)_2 \times 2^{-10} \\ &= (1.00000000)_2 - (0.0000000001)_2 \\ &= (0.11111111)_2 \\ &= (1.11111111)_2 \times 2^{-1} \end{aligned}$$

pero este número tiene una mantisa de 9bits, por lo cual, **no tiene representación** en el sistema.

- $a_6 = \epsilon^2$   
Para este inciso

$$a_6 = \epsilon^2 = ((1.0)_2 \times 2^{-8})_2^2 = (1.0)_2 \times 2^{-16} \quad (3)$$

el cual **si tiene representación** en el sistema.

- $a_7 = 0.125$   
Primero pasamos de el número de decimal a binario, de manera que  $(0.125)_{10} = (0.001)_2$ . Luego hay que normalizarlo, por lo que  $(0.001)_2 = (1.0)_2 \times 2^{-3}$ . Por lo que este numero **si tiene representacion** en el sistema.
- $a_8 = 2^{-10}$   
Y por 'ultimo, tenemos que

$$a_8 = 2^{-10} = ((10.0)_2)^{-10} = ((1.0)_2 \times 2^1)^{-10} = (1.0)_2 \times 2^{-10}$$

**1.6 De una cota para el error relativo de las restas 4 y 5 respecto al verdadero valor. Suponga que  $fl(x)$  se obtiene por redondeo hacia abajo (*truncamiento*)**

$$fl(0.9) - fl(0.5) \quad (4)$$

$$fl(0.9) - fl(0.895) \quad (5)$$

Para este ejercicio, dado que nos pide usar el *truncamiento*, definimos a la **unidad de redondeo** como  $u = \epsilon_m/2$ . Esto tambien nos define a  $fl(x) = x(1 + \delta)$  y  $|\delta| \leq u$ . Adem'as, siguiendo las notas, podemos usar la cota que ah'i se proporciona y sustituir  $u$ .

$$|\delta_{x-y}| \leq \frac{\epsilon}{2} \frac{|x| + |y|}{|x - y|} \quad (6)$$

Y antes de continuar calculamos  $\epsilon_m/2$  para poder usarlo mas adelante. De manera que

$$\epsilon_m/2 = \frac{(1.0)_2 \times 2^{-8}}{2} = (1.0)_2 \times 2^{-9}$$

Y, como tambien lo usaremos, es bueno tener en cuenta.

$$\delta_{x-y} = \frac{fl(x) - fl(y) - (x - y)}{x - y}$$

Empezamos por la operaci3n 4, de donde obtenemos  $x = 0.9$  y  $y = 0.5$ . Primero calculamos los valores de maquina de estos.

$$\begin{aligned} fl(x) = fl(0.9) &= (1.1100(1100))_2 \times 2^{-1} \\ &= (1.11001100)_2 \times 2^{-1} \quad (\text{Truncamiento (no cabe)}) \\ &\sim 0.8984375 \end{aligned}$$

$$\begin{aligned} fl(y) = fl(0.5) &= (1.0)_2 \times 2^{-1} \\ &= (1.00000000)_2 \times 2^{-1} \quad (\text{Truncamiento (si cabe)}) \\ &= 0.5 \end{aligned}$$

De manera que, como ya conocemos  $\epsilon_m$ , los valores de la operaci3n y sus representaciones en el sistema, procedemos a encontrar la cota. Dado que usamos la cota de las notas, unicamente queda sustituir, de esto obtenemos que:

$$\begin{aligned} \delta_{x-y} &\sim -0.00390625 \\ \frac{\epsilon}{2} \frac{|x| + |y|}{|x - y|} &= 0.00683593 \end{aligned}$$

Por lo que, siguiendo la cota de la ecuaci3n 6, podemos decir que esta operaci3n esta acotada por

$$|\delta_{x-y}| \sim 0.00390625 \leq 0.00683593 \sim \frac{\epsilon}{2} \frac{|x| + |y|}{|x - y|}$$

De manera similar, para la operaci3n 5, tenemos que  $x = 0.9$  y  $y = 0.895$ . Calculamos los redondeos de la computadora.

$$\begin{aligned} fl(x) = fl(0.9) &= (1.1100(1100))_2 \times 2^{-1} \\ &= (1.11001100)_2 \times 2^{-1} \quad (\text{Truncamiento (no cabe)}) \\ &\sim 0.8984375 \end{aligned}$$

$$\begin{aligned} fl(y) = fl(0.895) &= (1.1100101000111101)_2 \times 2^{-1} \\ &= (1.11001010)_2 \times 2^{-1} \quad (\text{Truncamiento (no cabe)}) \\ &= 0.89453125 \end{aligned}$$

Ahora procedemos a calcular los límites de la cota, por lo que

$$\delta_{x-y} \sim -0.21875$$
$$\frac{\epsilon}{2} \frac{|x| + |y|}{|x - y|} = 0.70117187$$

Y por último, según la cota 6, acotamos esta operación por

$$|\delta_{x-y}| \sim -0.21875 \leq 0.70117187 \sim \frac{\epsilon}{2} \frac{|x| + |y|}{|x - y|}$$



## 2 Ejercicio 2

Programa la función `epsilonFloat` que devuelve el  $\epsilon_m$  de la máquina para números de simple precisión y la función `epsilonDouble` para números de doble precisión.

```
epsilon = 1.0
unidad = 1.0
valor = unidad + epsilon
while valor > unidad
    epsilon = epsilon/2
    valor = end + epsilon
end while
epsilon = epsilon*2
```

Usar el algoritmo visto en clase.

```
[3]: # Primera parte del ejercicio 1
import numpy as np

def epsilonMaquina(tipoDato):
    """ Funcion que trata de calcular el epsilon
    de la maquina mediante el algoritmo antes pre_
    sentado.
    Input:
        tipoDato := esta pensado para ser uno de
        los tipos proporcionados por la li_
        breria numpy.
    Output:
        Regresa el epsilon de la maquina calcula_
        do con el tipo de dato especificado.
    """
    return epsilon*2

def epsilonFloat():
    """ Calculamos el epsilon de la maquina con
    precision de 32bits
    """
    return epsilonMaquina(np.float32)

def epsilonDouble():
    """ Calculamos el epsilon de la maquina con
    precision de 64 bits
    A pesar de que el flotante de python ya
    tiene esta precision, creo que es convenien_
    te especificarlo.
    """
    return epsilonMaquina(np.float64)

# Epsilons calculados

eF = np.float32(epsilonFloat())
```

```
eD = np.float64(epsilonDouble())

# Imprimir en pantalla
print(...)
```

Se calcularon los epsilons  
 eF=1.1920929e-07 y  
 eD=2.220446049250313e-16  
 para 32 y 64 bits correspondientemente.

```
[4]: # Segunda parte del ejercicio 1
def respuesta(res):
    """ Funcion para formatear la respuesta. """
    return "iguales" if res else "diferentes"
def comparacion(epsilon):
    """ Esta funcion recibira el epsilon a evaluar y el tipo de dato al que este correspondiente para hacer las comparaciones solicitadas en el ejercicio.
    Input:
        epsilon a considerar en la comparacion
    Output:
        Las respuestas son procesadas por la funcion respuesta para obtener el formato solicitado
    """
    # Comprobaciones
    print(f'{respuesta( tD(1 + epsilon ) == 1 ) =}')
    print(f'{respuesta( tD( epsilon/2 ) == 0 ) =}')
    print(f'{respuesta( tD(1 + epsilon/2 ) == 1 ) =}')
    print(f'{respuesta( tD(1 - epsilon/2 ) == 1 ) =}')
    print(f'{respuesta( tD(1 - epsilon/4 ) == 1 ) =}')
    print(f'{respuesta( tD( epsilon**2 ) == 0 ) =}')
    print(f'{respuesta(epsilon + tD(epsilon**2) == epsilon) =}')
    print(f'{respuesta(epsilon - tD(epsilon**2) == epsilon) =}')

# Hacemos la comparacion para 32bits
comparacion(eF)
# y para 64
comparacion(eD)
```

Con epsilon=1.1920929e-07 y tipo de dato = <class 'numpy.float32'> se da que

```
respuesta( tD(1 + epsilon ) == 1 ) = 'diferentes'
respuesta( tD( epsilon/2 ) == 0 ) = 'diferentes'
respuesta( tD(1 + epsilon/2 ) == 1 ) = 'iguales'
respuesta( tD(1 - epsilon/2 ) == 1 ) = 'diferentes'
respuesta( tD(1 - epsilon/4 ) == 1 ) = 'iguales'
respuesta( tD( epsilon**2 ) == 0 ) = 'diferentes'
respuesta(epsilon + tD(epsilon**2) == epsilon) = 'diferentes'
```

```

respuesta(epsilon - tD(epsilon**2) == epsilon)
↳='diferentes' [width=50mm,scale=0.5]
...

```

Con  $\epsilon=2.220446049250313e-16$  y tipo de dato = <class 'numpy.  
 ↳float64'> se da

```

que
respuesta( tD(1 + epsilon )    == 1 )  ='diferentes'
respuesta( tD( epsilon/2 )     == 0 )  ='diferentes'
respuesta( tD(1 + epsilon/2 )  == 1 )  ='iguales'
respuesta( tD(1 - epsilon/2 )  == 1 )  ='diferentes'
respuesta( tD(1 - epsilon/4 )  == 1 )  ='iguales'
respuesta( tD( epsilon**2 )    == 0 )  ='diferentes'
respuesta(epsilon + tD(epsilon**2) == epsilon) ='diferentes'
respuesta(epsilon - tD(epsilon**2) == epsilon) ='diferentes'
...

```

De manera que el programa calculo que el *epsilon de la maquina* es  $\{\{eF\}\}$  y  $\{\{eD\}\}$  para las precisiones de 32 y 64 bits correspondientemente, y respecto a las comparaciones unicamente se encontraron **dos igualdades** cuando se uso precision de 64bits.

## Como ejecutar [GoogleColab](#)

Para ejecutar este ejercicio en **consola** es importante ubicarse en la misma carpeta del [archivo T1.py](#) y ejecutar el siguiente comando en consola

```
python3 T1.py
```

Este programa no espera recibir argumento alguno. La salida debe ser similar a la siguiente imagen

```

bench@bench-dell: ~/Documents/Academico/UG/LicenciaturaDEMAT/GitHubRepo/M...
bench@bench-dell:~/Documents/Academico/UG/LicenciaturaDEMAT/GitHubRepo/MN/Tareas/T1$ python3 T1.py
Se calcularon los epsilons
eF=1.1920928955078125e-07 y
eD=2.220446049250313e-16
para 32 y 64 bits correspondientemente.

Con epsilon=1.1920928955078125e-07 y tipo de dato = <class 'numpy.float32'> se da que
respuesta(tD(1) + epsilon == tD(1))      ='diferentes'
respuesta(epsilon/tD(2) == tD(0))        ='diferentes'
respuesta(tD(1) + epsilon/tD(2) == tD(1)) ='diferentes'
respuesta(tD(1) - epsilon/tD(2) == tD(1)) ='diferentes'
respuesta(tD(1) - epsilon/tD(4) == tD(1)) ='diferentes'
respuesta(epsilon**2 == tD(0))            ='diferentes'
respuesta(epsilon + epsilon**2 == epsilon) ='diferentes'
respuesta(epsilon - epsilon**2 == epsilon) ='diferentes'
...

Con epsilon=2.220446049250313e-16 y tipo de dato = <class 'numpy.float64'> se da que
respuesta(tD(1) + epsilon == tD(1))      ='diferentes'
respuesta(epsilon/tD(2) == tD(0))        ='diferentes'
respuesta(tD(1) + epsilon/tD(2) == tD(1)) ='iguales'
respuesta(tD(1) - epsilon/tD(2) == tD(1)) ='diferentes'
respuesta(tD(1) - epsilon/tD(4) == tD(1)) ='iguales'
respuesta(epsilon**2 == tD(0))            ='diferentes'
respuesta(epsilon + epsilon**2 == epsilon) ='diferentes'
respuesta(epsilon - epsilon**2 == epsilon) ='diferentes'
...
bench@bench-dell:~/Documents/Academico/UG/LicenciaturaDEMAT/GitHubRepo/MN/Tareas/T1$

```