UNIVERSIDAD MAYOR DE SAN ANDRÉS

FACULTAD DE INGENIERÍA

“LABORATORIO 1”

1.SERIES DE MACLAURIN Y TAYLOR

ESTUDIANTE: SALINAS MAMANI CARLA JAEL

DOCENTE: ING. DUCHEN

CARRERA: INGENIERIA ELECTRÓNICA

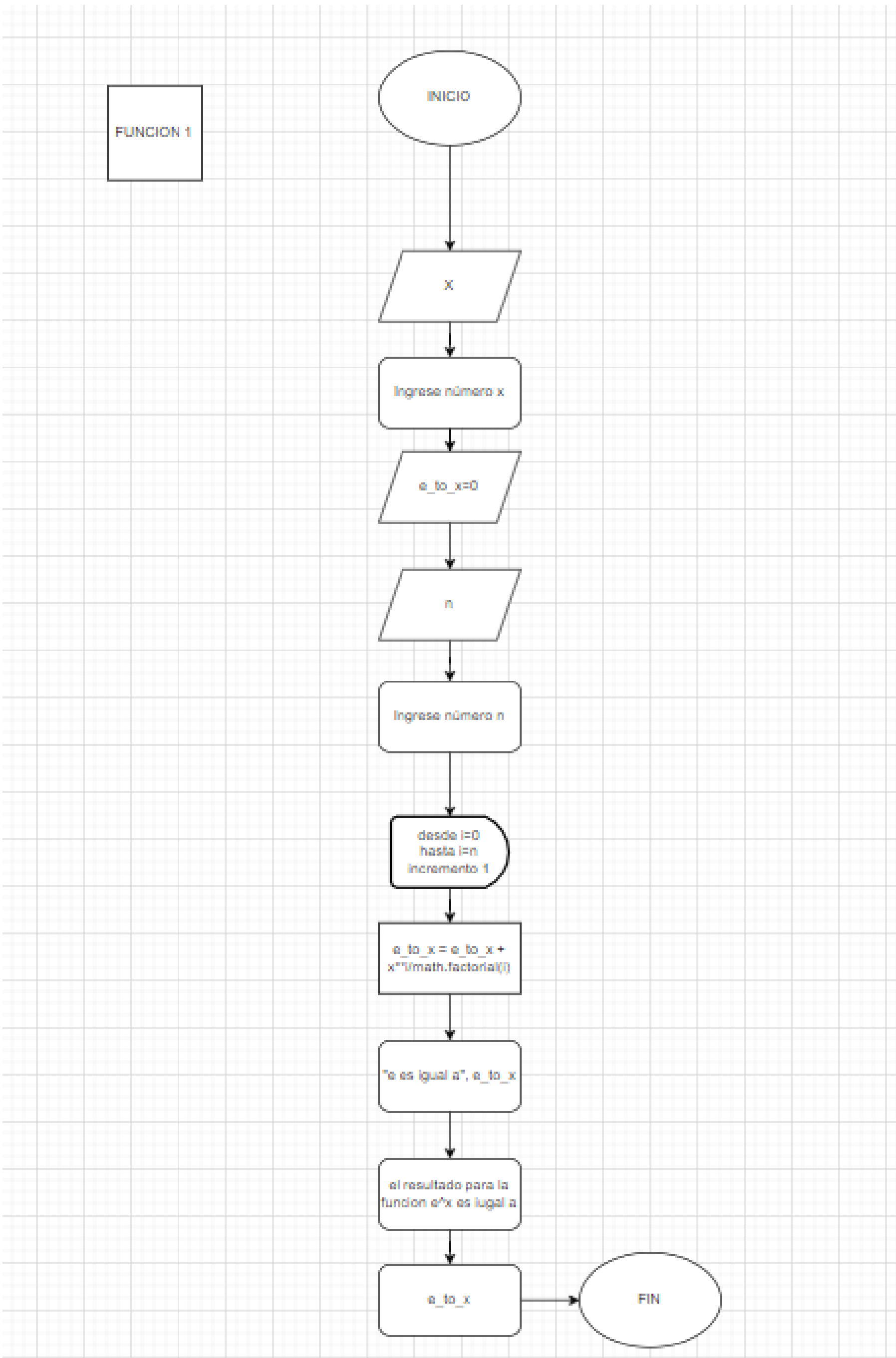
Objetivos: Realizar el código basándose en las series matemáticas de Taylor.

Marco teórico: A partir de los diagramas de flujo, armamos las siguientes funciones:

1. $\boldsymbol{Funcion\ 1: e^{x}, donde\ x\ es\ dato\ digitado, c = 0\ por\ que\ f(c) = f(0)(maclaurin)}$

$$P_n(x) = f(0) + f'(0)x + \frac{f''(0)}{2!}x^2 + \frac{f'''(0)}{3!}x^3 + \ldots + \frac{f^{(n)}(0)}{n!}x^n$$

Grafico1.



Código fuente.

i. Ingresar las librerías:

```python
import math
```

ii. Ingresar x, n siendo x el exponente y n el número de repeticiones de la serie.

```python
x = int(input('Ingrese número x : '))
e_to_x = 0
n = int(input('Ingrese número n : '))
```

iii. Seguimos con un for para n, llamando a la variable e_to_x (la función) y operando mediante la formula. (grafico1.)

```python
for i in range(n):
    print(i)
    e_to_x = e_to_x + x**i/math.factorial(i)
    print('e es igual a : ', e_to_x)
    print('======================')
print("el resultado para la funcion e^x es iugal a " ,e_to_x)
```

iv. Comprobamos el código y lo ejecutamos.

Ejemplo. x=2 n=9

```
Ingrese número x :
2
Ingrese número n :
9
```

Tenemos la serie:

Inicio i=0

```
0
e es igual a :  1.0
======================
1
e es igual a :  3.0
======================
2
e es igual a :  5.0
======================
3
e es igual a :  6.333333333333333
======================
4
e es igual a :  7.0
======================
5
e es igual a :  7.266666666666667
======================
6
e es igual a :  7.355555555555555
======================
7
e es igual a :  7.3809523809523805
======================
8
e es igual a :  7.387301587301587
```

El resultado es:

```
======================
el resultado para la funcion e^x es iugal a  6.38871252204585
45
```

Aclarando los valores reales y aproximados podríamos agregar:

```python
print("Tomando como referencia el valor real de la funcion e^x = ",math.exp(x))
print(" ")
print("Para la serie con n = ", n,"      El resultado de la aproximacion de taylor es: ",e_to_x)
print(" ")
print("Observamos que mientras mas se incrementa la serie, la aproximación es mas cercana al valor real")
```

Obteniendo:

```
Tomando como referencia el valor real de la funcion e^x =  7.38905609893065

Para la serie con n =  9       El resultado de la aproximacion de taylor es:  6.3887125220458545

Observamos que mientras mas se incrementa la serie, la aproximación es mas cercana al valor real
```

v. Ahora realizamos un método de series para crear un gráfico de la función con diferentes valores de "x"}

a. Definimos al método:

```python
def func_exponente(x, n):

    e_to_x = 0
    for i in range(n):
        e_to_x = e_to_x + x**i/math.factorial(i)

    return e_to_x
```

```python
n_series = int(input('¿Cuántas series desea generar?: '))
n = int(input('Ingrese n de la serie : '))

resultados_aproximados = []
resultados_reales = []

valores_x = np.linspace(1,20,n_series) #PARA NO DIGITAR X TANTAS VECES, LE DAREMOS UN RANGO

for i in range(n_series):

        #x = int(input('Ingrese número x : '))
    #valores_x.append(x)
    print('valor de x es : ', valores_x[i])
    serie_resultado = func_exponente(valores_x[i], n)
    print('valor aprox ', serie_resultado)

    resultados_aproximados.append(serie_resultado)
    #print('mi lista de resultados aproximados es ', resultados_aproximados)

    valor_real = math.exp(valores_x[i])
    print('valor real ', valor_real)

    resultados_reales.append(valor_real)
    #print('mi lista de resultados reales es ', resultados_reales)

    print('=================')
```

b. Ejemplo:

Num de serie: 5   n=12

```
¿Cuántas series desea generar?: 5

Ingrese n de la serie :
 12
```

```
valor de x es :  1.0
valor aprox  2.718281826198493
valor real  2.718281828459045
==================
valor de x es :  5.75
valor aprox  309.470468844243
valor real  314.1906602856942
==================
valor de x es :  10.5
valor aprox  23195.627592530684
valor real  36315.502674246636
==================
valor de x es :  15.25
valor aprox  708236.990383311
valor real  4197501.3938479675
==================
valor de x es :  20.0
valor aprox  10376141.474587142
valor real  485165195.4097903
==================
```

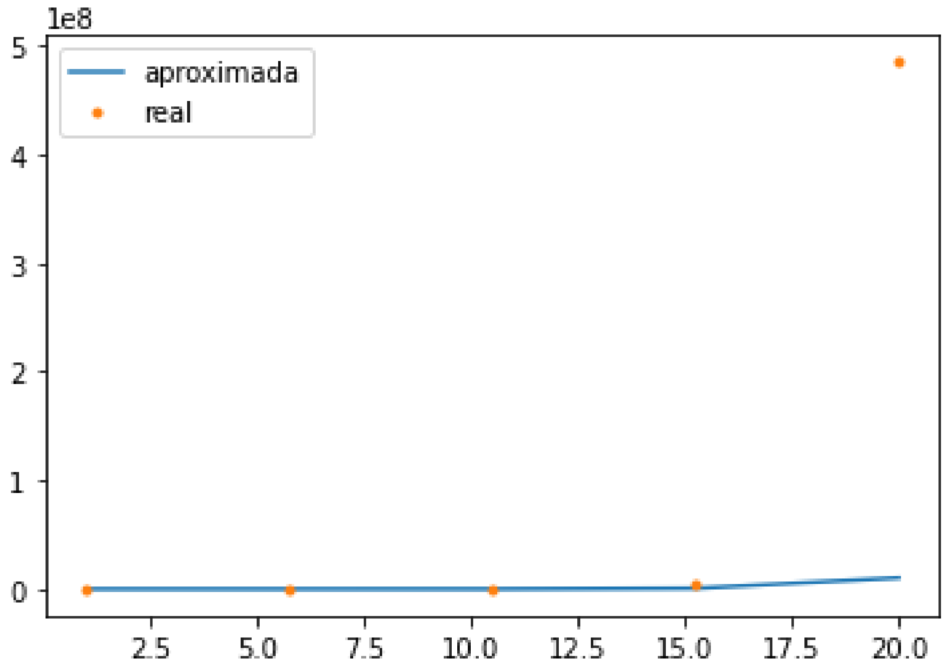
vi. Ahora graficaremos:

a. Llamando a las librerías:

```
import numpy as np
import matplotlib.pyplot as plt

%matplotlib inline
```

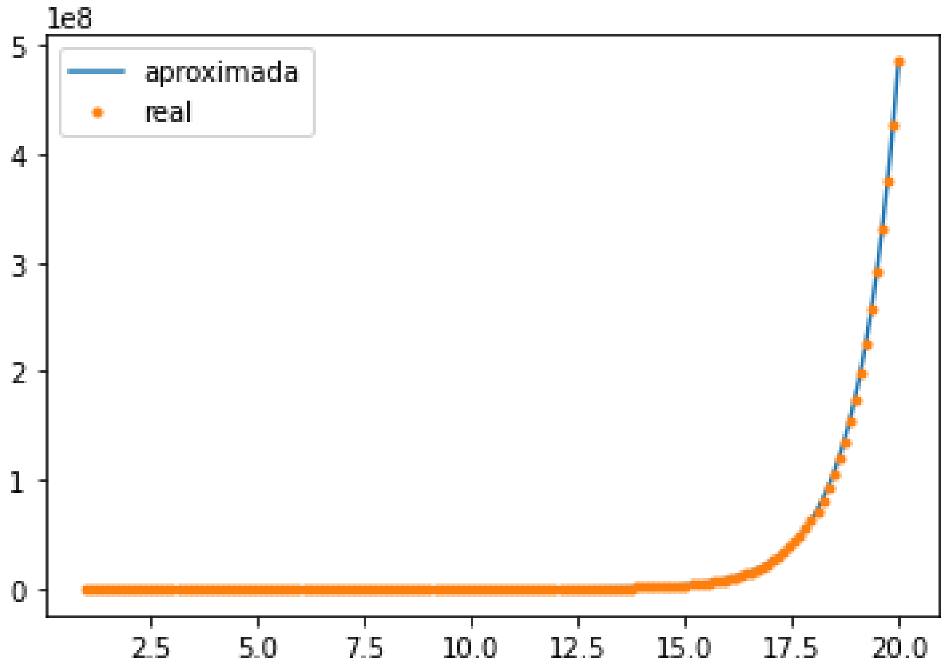
b. Agregando el código para mi lista de valores de x:

```
plt.plot(valores_x, resultados_aproximados,label='aproximada')
plt.legend()

plt.plot(valores_x, resultados_reales, '.',label = 'real')
plt.legend()

```



Pero si agrandamos el numero se series que queremos, la gráfica se hace mas exponencial:

Ejemplo. num de serie=150   n =92



2. considerando la expansion de cos(x) (maclaurin)

$$= 1 + \frac{\frac{d}{dx}(\cos(x))(0)}{1!}x + \frac{\frac{d^2}{dx^2}(\cos(x))(0)}{2!}x^2 + \frac{\frac{d^3}{dx^3}(\cos(x))(0)}{3!}x^3 + \ldots$$

$$= 1 + \frac{0}{1!}x + \frac{-1}{2!}x^2 + \frac{0}{3!}x^3 + \frac{1}{4!}x^4 + \frac{0}{5!}x^5 + \frac{-1}{6!}x^6 + \frac{0}{7!}x^7 + \frac{1}{8!}x^8 + \ldots$$

$$= 1 - \frac{1}{2!}x^2 + \frac{1}{4!}x^4 - \frac{1}{6!}x^6 + \frac{1}{8!}x^8 + \ldots$$

$$= \sum_{n=0}^{\infty} (-1)^n \frac{x^{2n}}{(2n)!}$$

i. Llamamos al método func_taylor:

```python
def func_taylor(x, n):



    cos_aproximacion = 0

    for i in range(n):

        coef = (-1)**i
        print('coef : ', coef)
        print('i ' , i)
        num = x**(2*i)
        denom = math.factorial(2*i)
        print('el factorial denominador =  ', denom)
        cos_aproximacion = cos_aproximacion + (( coef ) * ( (num)/(denom) ))
        print("la aproximacion con i = ",i,"    La aproximación es : ", cos_aproximacion)
        print('====================')

    return cos_aproximacion
```

ii. Ingresamos Angulo y n:_

```
angulo = int(input('Ingrese ángulo : '))
x_angulo = (math.radians(angulo))

print("El ángulo",angulo,"tranformado en radianes es: ",x_angulo)

n = int(input('Ingrese el valor n: '))
```

iii. Agregamos el valor real y el aproximado

```
n = int(input('Ingrese el valor n: '))
resultado = func_taylor(x_angulo, n)
valor_real_coseno = math.cos(x_angulo)
```

iv. Imprimimos el resultado:

```
print("Tomando como referencia el valor real de la funcion cos(x) = ",valor_real_coseno)
print(" ")
print("Para la serie con n = ", n,"      El resultado de la aproximacion de taylor es: ",resultado)
print(" ")
print("Observamos que mientras mas se incrementa la serie, la aproximación es mas cercana al valor real")
```

Ejemplo. Con ángulo=45 y n=5

```
Ingrese ángulo : 45
El ángulo 45 tranformado en radianes es:  0.7853981633974483
Ingrese el valor n: 5
coef :  1
i  0
el factorial denominador =   1
la aproximacion con i =  0     La aproximación es :  1.0
=====================
coef :  -1
i  1
el factorial denominador =   2
la aproximacion con i =  1     La aproximación es :  0.6915748624659576
=====================
coef :  1
i  2
el factorial denominador =   24
la aproximacion con i =  2     La aproximación es :  0.707429206709773
=====================
coef :  -1
i  3
el factorial denominador =   720
la aproximacion con i =  3     La aproximación es :  0.7071032148228457
=====================
```

```
coef :  -1
i  3
el factorial denominador =   720
la aproximacion con i =  3     La aproximación es :  0.7071032148228457
====================
coef :  1
i  4
el factorial denominador =   40320
la aproximacion con i =  4     La aproximación es :  0.7071068056832942
====================
Tomando como referencia el valor real de la funcion cos(x) =  0.7071067811865476

Para la serie con n =  5       El resultado de la aproximacion de taylor es:  0.7071068056832942

Observamos que mientras mas se incrementa la serie, la aproximación es mas cercana al valor real
```

3. $\boldsymbol{Funcion\ 1: e^x, donde\ x\ es\ dato\ digitado, c = 1\ (TAYLOR)}$

i. Ingresamos x y n:

```
x = int(input('Ingrese número x : '))

n = int(input('Ingrese número n : '))
```

ii. Declaramos suma_serie = 0

iii. Iniciamos el ciclo:

```
for i in range(n):

    numerador = (x+1)**i
    denominador = math.exp(1)*math.factorial(i)

    valor = numerador / denominador

    suma_serie =  suma_serie + valor

    print('e es igual a : ', suma_serie)
    print('======================')
```

iv. Imprimimos el resultado:

print("el resultado para la función e^x es igual a " ,suma_serie)

ejemplo. X=5   n=5

```
Ingrese número x : 5
Ingrese número n : 5
e es igual a :  0.36787944117144233
======================
e es igual a :  2.5751560882000963
======================
e es igual a :  9.196986029286059
======================
e es igual a :  22.440645911457985
======================
e es igual a :  42.306135734715866
======================
el resultado para la funcion e^x es igual a  42.3061357347158
66
```