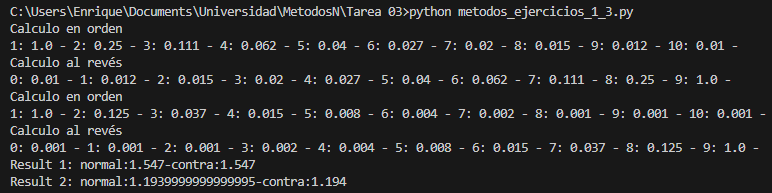
**Tarea 3 Ejercicios Unidad 1B**

**Nombre:** Luis Enrique Pérez Señalin.

1. Utilice aritmética de corte de tres dígitos para calcular las siguientes sumas. Para cada parte, ¿qué método es más preciso y por qué?
2. primero por y luego por

**R=** Ambos dan 1.547, sin hacer el corte de tres dígitos, es más preciso el primero, porque el segundo empieza a perder exactitud

1. primero por y luego por

**R=** La forma en orden da 1.19399 mientras que en orden contrario da 1.194.

1. La serie de Maclaurin para la función arcotangente converge para −1 < 𝑥 ≤ 1 y está dada por:
2. Utilice el hecho de que tan para determinar el número n de términos de la serie que neesita sumar para garantizar que
3. El lenguaje de programación C++ requiere que el valor de se encuentro dentro de . ¿Cuántos términos de la serie se necesitarán sumar para obtener este grado de precisión?
4. Otra fórmula para caluclar se puede deducir a partir de la identidad . Determine el número de términos que se deben sumar para garantizar una aproximación dentro de .

Basándonos en estos valores

1. Compare los siguientes tres algoritmos. ¿Cuándo es correcto el algoritmo de la parte 1a?
2. ENTRADA 𝑛, 𝑥1, 𝑥2, ⋯ , 𝑥𝑛.

SALIDA PRODUCT.

Paso 1 Determine PRODUCT = 0.

Paso 2 Para 𝑖 = 1, 2, ⋯ , 𝑛 haga

Determine PRODUCT = PRODUCT ∗ 𝑥𝑖.

Paso 3 SALIDA PRODUCT; PARE.

1. ENTRADA 𝑛, 𝑥1, 𝑥2, ⋯ , 𝑥𝑛.

SALIDA PRODUCT.

Paso 1 Determine PRODUCT = 1.

Paso 2 Para 𝑖 = 1, 2, ⋯ , 𝑛 haga

Set PRODUCT = PRODUCT ∗ 𝑥𝑖.

Paso 3 SALIDA PRODUCT; PARE.

1. ENTRADA 𝑛, 𝑥1, 𝑥2, ⋯ , 𝑥𝑛.

SALIDA PRODUCT.

Paso 1 Determine PRODUCT = 1.

Paso 2 Para 𝑖 = 1, 2, ⋯ , 𝑛 haga

si 𝑥𝑖 = 0 entonces determine PRODUCT = 0;

SALIDA PRODUCT;

PARE

Determine PRODUCT = PRODUCT ∗ 𝑥𝑖.

Paso 3 SALIDA PRODUCT; PARE.

**R:** El algoritmo 1a será correcto siempre que x , ósea todo número real con o sin fracción, ya sea negativo o positivo y sean igual a 0, porque al inicio inicializa la variable producto en 0, por lo que al final terminará siendo 0 el resultado.

1. a. ¿Cuántas multiplicaciones y sumas se requieren para determinar una suma de la forma

**R:** La multiplicación es de cantidad, y las sumas es

b. Modifique la suma en la parte a) a un formato equivalente que reduzca el número de cálculos.

**R:** La multiplicación es de cantidad, y las sumas es lo sacamos de

**Discusiones**

1. Escriba un algoritmo para sumar la serie finita en orden inverso

def sumar\_serie\_inversa(x):

suma = 0

n = len(x)

for i in range(n-1, -1, -1):

suma += x[i] return suma

# Ejemplo de uso   
x = [1, 2, 3, 4, 5] # Arreglo de ejemplo  
resultado = sumar\_serie\_inversa(x)  
print("La suma de la serie en orden inverso es:", resultado)

O

def sumar\_serie\_inversa(x):

x = x[-1::]

suma = 0

n = len(x)

for i in range(x):

suma += x[i] return suma

1. Las ecuaciones (1.2) y (1.3) en la sección 1.2 proporcionan formas alternativas para las raíces 𝑥1 y 𝑥2 de 𝑎𝑥 2 + 𝑏𝑥 + 𝑐 = 0. Construya un algoritmo con entrada 𝑎, 𝑏, 𝑐 c y salida 𝑥1, 𝑥2 que calcule las raíces 𝑥1 y 𝑥2 (que pueden ser iguales con conjugados complejos) mediante la mejor fórmula para cada raíz.

Forma alternativa:

Si a = 0, no es una ecuación cuadrática valida.

El discriminante es D =

Si D < 0, entonces es un número complejo, caso contrario no.

Si D>= 0, entonces tenemos las formulas:

Si b>= 0:

Sino:

1. Suponga que

para 𝑥 < 1 y si 𝑥 = 0.25. Escriba y ejecute un algoritmo que determine el número de términos necesarios en el lado izquierdo de la ecuación de tal forma que el lado izquierdo difiera del lado derecho en menos de 10−6

def lado\_derecho(x):

    return (1 + 2\*x) / (1 + x + x\*\*2)

def termino\_izquierdo(x, n):

    numerador = (1 - 2\*x) \* (2\*\*n \* x\*\*(2\*\*n - 1))

    denominador = 1 - x\*\*(2\*\*n) + x\*\*(2\*\*(n+1))

    return numerador / denominador

def calcular\_terminos\_necesarios(x, tolerancia=1e-6):

    suma\_parcial = 0

    valor\_lado\_derecho = lado\_derecho(x)

  n = 0

    while True:

        termino = termino\_izquierdo(x, n)

        suma\_parcial += termino

        if abs(suma\_parcial - valor\_lado\_derecho) < tolerancia:

            return n + 1  # Devuelve el número de términos necesarios

        n += 1

# Valor de x dado

x = 0.25

tolerancia = 1e-6

# Calcular el número de términos necesarios

terminos\_necesarios = calcular\_terminos\_necesarios(x, tolerancia)

print("Número de términos necesarios:", terminos\_necesarios)