

# Métodos Numéricos - DEBER 03 - Algoritmos y Convergencia

Alicia Pereira

## Tabla de Contenidos

<b>Escuela Politécnica Nacional</b>	<b>1</b>
Discusiones . . . . .	4

## Escuela Politécnica Nacional

**2. La serie de Maclaurin para la función arcotangente converge para  $-1 < x \leq 1$  y está dada por**

$$\arctan x = \lim_{n \rightarrow \infty} P_n(x) = \lim_{n \rightarrow \infty} \sum_{i=1}^n (-1)^{i+1} \frac{x^{2i-1}}{2i-1}. \quad (1)$$

**a. Utilice el hecho de que  $\tan \frac{\pi}{4} = 1$  para determinar el número  $n$  de términos de la serie que se necesita sumar para garantizar que  $|4P_n(1) - \pi| < 10^{-3}$**

$$\tan \frac{\pi}{4} = 1$$

$$\arctan(1) = \frac{\pi}{4}$$

$$\arctan(1) = \sum_{i=1}^n (-1)^{i+1} \frac{1^{2i-1}}{2i-1}$$

$$P_n(1) = \sum_{i=1}^n (-1)^{i+1} \frac{1}{2i-1}$$

Asignar valores a  $n$  para garantizar que:  $|4P_n(1) - \pi| < 10^{-3}$

```
import math

contador = 0
diferencia = 100
```

```
def obtener_arcotangente(terminos: int) -> float:
    resultado = 0
    for j in range(1, terminos + 1):
        resultado += (-1) ** (j + 1) * (1 / (2 * j - 1))
    return resultado

while diferencia >= 1e-3:
    contador += 1
    valor_arcotangente = obtener_arcotangente(contador)
    diferencia = abs(4 * valor_arcotangente - math.pi)

print(f'Se requieren {contador} términos')
```

Se requieren 1000 términos

b. El lenguaje de programación C++ requiere que el valor  $\pi$  se encuentre dentro de  $10^{-10}$ . ¿Cuántos términos de la serie se necesitarían sumar para obtener este grado de precisión?

$$4 * \sum_{i=1}^n (-1)^{i+1} \frac{1}{2i-1} = \pi$$

```
import math

def aproximar_pi(terminos):
    acumulador = 0
    for k in range(terminos):
        acumulador += (-1) ** k * (1 / (2 * k + 1))
    return 4 * acumulador

def verificar_pi(aprox, pi_real, decimales):
    str_aproximacion = str(aprox)[:decimales + 2]
    str_pi_real = str(pi_real)[:decimales + 2]
    coincidencias = 0
    for idx in range(decimales + 2):
        if str_aproximacion[idx] == str_pi_real[idx]:
            coincidencias += 1
        else:
            break
    return coincidencias

decimales_requeridos = 6
valor_pi = math.pi
```

```

contador = 1

while True:
    valor_aprox = aproximar_pi(contador)
    coincidencias = verificar_pi(valor_aprox, valor_pi, decimales_requeridos)
    if coincidencias >= decimales_requeridos:
        break
    contador += 1

print(f"Se necesitaron {contador} términos para obtener {decimales_requeridos} decimales correctos.")

```

Se necesitaron 10794 términos para obtener 6 decimales correctos.

**3. Otra fórmula para calcular  $\pi$  se puede deducir a partir de la identidad  $\pi/4 = 4 \arctan \frac{1}{5} - \arctan \frac{1}{239}$ . Determine el número de términos que se deben sumar para garantizar una aproximación dentro de  $10^{-3}$ .**

Despejando  $\pi \rightarrow \pi = 4 * (4 * \arctan \frac{1}{5} - \arctan \frac{1}{239})$

El valor de la arcotangente se obtiene a partir de Maclaurin

```

import math

def aproximar_pi(terminos):
    suma_arctan1 = 0
    suma_arctan2 = 0
    for j in range(terminos):
        suma_arctan1 += (-1)**j * (1 / (5**(2*j+1) * (2*j+1)))
        suma_arctan2 += (-1)**j * (1 / (239**(2*j+1) * (2*j+1)))
    return 4 * (4 * suma_arctan1 - suma_arctan2)

def verificar_precision(aprox, valor_pi, decimales):
    str_aprox = str(aprox)[:decimales + 2]
    str_pi = str(valor_pi)[:decimales + 2]
    coincidencias = 0
    for k in range(decimales + 2):
        if str_aprox[k] == str_pi[k]:
            coincidencias += 1
        else:
            break
    return coincidencias

```

```

valor_pi = math.pi
decimales_objetivo = 3
contador = 1

while True:
    valor_aprox = aproximar_pi(contador)
    coincidencias = verificar_precision(valor_aprox, valor_pi, decimales_objetivo)
    if coincidencias >= decimales_objetivo:
        break
    contador += 1

print(f"Se necesitaron {contador} términos para obtener {decimales_objetivo} decimales correctos")

```

Se necesitaron 1 términos para obtener 3 decimales correctos.

**5. ¿Cuántas multiplicaciones y sumas se requieren para determinar una suma de la forma  $\sum_{i=1}^n \sum_{j=1}^i a_i b_j$ ?**

Para un valor dado de i, j toma valores de 1 hasta i.

Se realizan i multiplicaciones, dando un total de:

$$\sum_{i=1}^n i = \frac{n(n+1)}{2}$$

```

n = 3
conteo_multiplicaciones = 0

for a in range(1, n + 1):
    for b in range(1, a + 1):
        resultado = a * b
        conteo_multiplicaciones += 1

print(f'La operación de multiplicación se realizó {conteo_multiplicaciones} veces')

```

La operación de multiplicación se realizó 6 veces

## Discusiones

**2.** Las ecuaciones(1.2) y (1.3) en la sección 1.2 proporcionan formas alternativas para las raíces.  $x_1$  y  $x_2$  de  $ax^2 + bx + c = 0$ . Construya un algoritmo con entrada a,b,c y salida x\_1,x\_2 que calcule las raíces  $x_1$  y  $x_2$  (que pueden ser iguales con conjugados complejos) mediante la mejor fórmula para cada raíz

**GitHub:** git@github.com: dayapt04

[<https://github.com/dayapt04/MetodosNumericos>]