Conjunto de ejercicios 1

Boris Garcés

Tabla de Contenidos

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

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

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 $|4P_n(1) - \pi| < 10^{-3}$

Partiendo de $\tan \frac{\pi}{4} = 1$, podemos obtener lo siguiente: $\arctan(1) = \frac{pi}{4}$ De acuerdo con la serie de Maclaurin $\arctan(1) = \sum i = 1^n (-1)^{i+1} \frac{1^{2i-1}}{2i-1}$ $P_n(1) = \sum i = 1^n (-1)^{i+1} \frac{1}{2i-1}$ A continuación daremos valores a n para garantizar que $|4P_n(1) - pi| < 10^{-3}$

```
import math
n = 0
valor_encontrado = 100
def calcular_arcotangente(n: int) -> float:
    arcotangente = 0
    for i in range(1, n + 1):
        arcotangente += (-1)**(i + 1) * (1 / (2 * i - 1))
    return arcotangente

while valor_encontrado >= 10**-3:
    n += 1
    arcotangente = calcular_arcotangente(n)
    valor_encontrado = abs(4 * arcotangente - math.pi)

print(f'El número de términos necesarios es {n}')
```

```
El número de términos necesarios es 1000
```

b) El lenguaje de programación c++ requiere que el valor π se encuentre dentro de 10^{-10}. ¿Cuántos términos de la sere se necesitarían sumar para obtener este grado de presición?

A partir del razonamiento anterior se obtiene: $4 * \sum_{i=1}^{n} (-1)^{i+1} \frac{1}{2i-1} = \pi$

```python
import math

def calcular_pi(n):
    suma = 0
    for i in range(n):
        suma += (-1) ** i * (1 / (2 * i + 1))
    return 4 * suma

def comparar_pi(aproximacion, pi_exacto, precision):

    str_aprox = str(aproximacion)[:precision+2]
    str_pi = str(pi_exacto)[:precision+2]
    contador_acierto = 0
    for i in range(precision+2):
        if str_aprox[i] == str_pi[i]:
            contador_acierto += 1
        else:
            break
    return contador_acierto

precision_deseada = 6
pi_exacto = math.pi
n = 1
while True:
    aproximacion = calcular_pi(n)
    aciertos = comparar_pi(aproximacion, pi_exacto, precision_deseada)
    if aciertos >= precision_deseada:
        break
    n += 1
print(f"Se necesitaron {n} términos para obtener {precision_deseada} decimales correctos.")
```

```
Se necesitaron 10794 términos para obtener 6 decimales correctos.
```

3. Otra fórmula para calcular π 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}

Para resolver este problema se debe despejar el valor de pi, obteniendo de este modo: $\pi = 4 * (4 * \arctan\frac{1}{5} - \arctan\frac{1}{239})$ y el valor de la arcotangente lo podemos determinar como resultado de la serie de Maclaurin

```
import math

def calcular_pi(n):
    arctan1 = 0
    arctan2 = 0
    for i in range(n):
        arctan1 += (-1)**i * (1 / (5**(2*i+1) * (2*i+1)))
        arctan2 += (-1)**i * (1 / (239**(2*i+1) * (2*i+1)))
    return 4 * (4*arctan1 - arctan2)
def comparar_pi(aproximacion, pi_exacto, precision):
    str_aprox = str(aproximacion)[:precision+2]
    str_pi = str(pi_exacto)[:precision+2]
    contador_acierto = 0
    for i in range(precision+2):
        if str_aprox[i] == str_pi[i]:
            contador_acierto += 1
        else:
            break
    return contador_acierto
pi_exacto = math.pi
precision_deseada = 3
n = 1
while True:
    aproximacion = calcular_pi(n)
    aciertos = comparar_pi(aproximacion, pi_exacto, precision_deseada)
    if aciertos >= precision_deseada:
        break
    n += 1
print(f"Se necesitaron {n} términos para obtener {precision_deseada} decimales correctos.")
```

```
Se necesitaron 1 términos para obtener 3 decimales correctos.
```

5.

a) ¿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, por lo que se realizan i multiplicaciones, dando que en total se realizarán $\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$, siendo esta la cantidad de veces que se realiza la multiplicación.

```
n = 3
multiplicaciones = 0
for i in range(1, n+1):
    for j in range(1, i+1):
        multiplicacion = i * j
        multiplicaciones += 1

print(f'La multiplicación se ha realizado {multiplicaciones} veces')
```

```
La multiplicación se ha realizado 6 veces
```

Discuciones

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

```
import cmath
def obtener_raices(a, b, c):
    discriminante = b ** 2 - 4 * a * c
    if discriminante > 0:
        if b > 0:
            raiz1 = (-b - (discriminante) ** 0.5) / (2 * a)
        else:
            raiz1 = (-b + (discriminante) ** 0.5) / (2 * a)
        raiz2 = c / (a * raiz1)
    elif discriminante == 0:
        raiz1 = raiz2 = -b / (2 * a)
    else:
        raiz_discriminante = cmath.sqrt(discriminante)
        raiz1 = (-b + raiz_discriminante) / (2 * a)
        raiz2 = (-b - raiz_discriminante) / (2 * a)
    return raiz1, raiz2
a, b, c = 1, -3, 2
resultado1, resultado2 = obtener_raices(a, b, c)
print(f"Las raíces son: x1 = {resultado1}, x2 = {resultado2}")
```

```
Las raíces son: x1 = 2.0, x2 = 1.0
```