Algoritmos y Convergencia

Mateo Cumbal

2024-10-30

Tabla de Contenidos

1 EJERCICIOS

1.1 Ejercicio 2

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

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

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

```python
import math

def n_terminos_para_pi(precision):
    n = 0
    suma = 0.0
    error = float('inf')
    pi_real = math.pi
    while error >= precision:
        n += 1
        termino = (-1)**(n+1) / (2*n - 1)
```

```
        suma += termino
        pi_aprox = 4 * suma
        error = abs(pi_real - pi_aprox)
    return pi_aprox, n

pi_aprox, n_terminos = n_terminos_para_pi(10**-3)
print(f"Valor aproximado de  : {pi_aprox}")
print(f"Mínimo de términos necesarios: {n_terminos}")
```

```
Valor aproximado de  : 3.140592653839794
Mínimo de términos necesarios: 1000
```

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

```
pi_aprox, n_terminos = n_terminos_para_pi(10**-10)
print(f"Valor aproximado de  : {pi_aprox}")
print(f"Mínimo de términos necesarios: {n_terminos}")
```

> **Se murió el código.** En esta serie, para alcanzar una precisión de 10^{-10}, es probable que necesitemos sumar varios miles de millones de términos, debido a la naturaleza lenta de convergencia de esta serie.

1.2 Ejercicio 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}.

Despejando π de la ecuación resulta en:

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

```
def calcular_pi_machin(precision=1e-3):
    def arctan_leibniz(x, precision):
        suma = 0.0
        n = 1
        termino = x
        while abs(termino) >= precision:
            suma += termino
```

```
            n += 1
            termino = (-1) ** (n + 1) * (x ** (2 * n - 1)) / (2 * n - 1)
        return suma, n

    arctan1, terms1 = arctan_leibniz(1/5, precision)
    arctan2, terms2 = arctan_leibniz(1/239, precision)

    pi_aproximado = 4 * (4 * arctan1 - arctan2)
    total_terms = terms1 + terms2

    return pi_aproximado, total_terms

# Ejemplo de uso
pi_aproximado, total_terms = calcular_pi_machin(precision=1e-3)
print(f"Valor aproximado de  : {pi_aproximado}")
print(f"Número total de términos sumados: {total_terms}")
```

```
Valor aproximado de  : 3.1405969316596933
Número total de términos sumados: 5
```

1.3 Ejercicio 5

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

```
def calcular_operaciones(n):
    multiplicaciones = n * (n + 1) // 2
    sumas = multiplicaciones - 1
    return multiplicaciones, sumas

n = int(input("Ingrese el valor de n: "))
multiplicaciones, sumas = calcular_operaciones(n)
print(f"Para n = {n}:")
print(f"Número de multiplicaciones requeridas: {multiplicaciones}")
print(f"Número de sumas requeridas: {sumas}")
```

```
Ingrese el valor de n:  10

Para n = 10:
Número de multiplicaciones requeridas: 55
Número de sumas requeridas: 54
```

Para cada i de 1 a n, el índice j recorre desde 1 hasta i. Por lo que para cada valor de i, hay i multiplicaciones. Lo que resulta que el número total de multiplicaciones es la sumatoria de Gauss.

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

Después de cada multiplicación, el resultado es sumado al acumulador. Por lo que se tiene solo una suma menos que el número de multiplicaciones porque la primera multiplicación no necesita ser sumada.

$\frac{n(n+1)}{2} - 1$

GitHub: Métodos Númericos - @mateobtw18