Ejercicios: Algoritmos y convergencia

Luis Tipan

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

Partiendo de $\tan\frac{\pi}{4} = 1$, obtenemos: $\arctan(1) = \frac{pi}{4}$

Por Maclaurin:

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

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

Se da valores a n para garantizar

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

```python
import math

def calcular_nuevo_n():
    n = 1
    suma = 0.0
    error = float('inf')
    pi_real = math.pi
    while error >= 1e-3:
        suma += (-1)**(n + 1) / (2 * n - 1)
        pi_aproximado = 4 * suma
        error = abs(pi_real - pi_aproximado)
        n += 1
    return n, pi_aproximado, error

n, pi_aproximado, error = calcular_nuevo_n()
print(f"Cantidad mínima de términos: {n}")
print(f"Valor aproximado de  : {pi_aproximado}")
print(f"Error absoluto: {error}")
```

```
Cantidad mínima de términos: 1001
Valor aproximado de  : 3.140592653839794
Error absoluto: 0.000999999749998981
```

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?

A partir del enunciado anterior | $4 * \sum_{i=1}^{n} (-1)^{i+1} \frac{1}{2i-1} = \pi$

```python
import math

def calcular_pi_precision(n):
    suma = 0.0
    for i in range(n):
        termino = (-1)**i * (1 / (2 * i + 1))
        suma += termino
    return 4 * suma

precision_deseada = 10
pi_real = math.pi
n = 1
decimales_correctos = 0

while decimales_correctos < precision_deseada:
    aproximacion = calcular_pi_precision(n)
    str_aprox, str_pi = f"{aproximacion:.15f}", f"{pi_real:.15f}"
    decimales_correctos = sum(1 for a, b in zip(str_aprox[2:], str_pi[2:]) if a == b)
    n += 1

print(f"Se requirieron {n} términos para obtener {precision_deseada} decimales.")
print(f"Valor aproximado de  : {aproximacion}")
print(f"Valor exacto de  : {pi_real}")
```

```
Se requirieron 501 términos para obtener 10 decimales.
Valor aproximado de  : 3.139592655589785
Valor exacto de  : 3.141592653589793
```

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

El valor de la arcotangente se obtiene a partir de Maclaurin

```python
import math

def calcular_pi_modificado(n):
    arctan1, arctan2 = 0, 0
    for i in range(n):
        arctan1 += (-1)**i / (5**(2 * i + 1) * (2 * i + 1))
        arctan2 += (-1)**i / (239**(2 * i + 1) * (2 * i + 1))
    return 4 * (4 * arctan1 - arctan2)

precision_deseada = 3
n = 1
pi_real = math.pi

while abs(calcular_pi_modificado(n) - pi_real) >= 10**-precision_deseada:
    n += 1

print(f"Términos necesarios para precisión de 10^-{precision_deseada}: {n}")
```

```
Términos necesarios para precisión de 10^-3: 2
```

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, 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.

```python
n = 5
multiplicaciones_totales = sum(i for i in range(1, n + 1))
print(f"Multiplicaciones totales para n={n}: {multiplicaciones_totales}")
```

```
Multiplicaciones totales para n=5: 15
```

Discuciones

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: https://github.com/LuisTipan005/MetodosNumericos_2024B