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?

a. $\sum_{i=1}^{10}\left(\frac{1}{i^2}\right)$ primero por $\frac{1}{1}+\frac{1}{4}\ldots.+\frac{1}{100}$ y luego por $\frac{1}{100}+\frac{1}{81}\ldots+\frac{1}{1}$

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

b. $\sum_{i=1}^{10}\left(\frac{1}{i^3}\right)$ primero por $\frac{1}{1}+\frac{1}{9}+\frac{1}{27}\ldots+\frac{1}{1000}$ y luego por $\frac{1}{1000}+\frac{1}{729}\ldots+\frac{1}{1}$

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

```
C:\Users\Enrique\Documents\Universidad\MetodosN\Tarea 03>python metodos_ejercicios_1_3.py
Calculo en orden
1: 1.0 - 2: 0.25 - 3: 0.111 - 4: 0.062 - 5: 0.04 - 6: 0.027 - 7: 0.02 - 8: 0.015 - 9: 0.012 - 10: 0.01 -
Calculo al revés
0: 0.01 - 1: 0.012 - 2: 0.015 - 3: 0.02 - 4: 0.027 - 5: 0.04 - 6: 0.062 - 7: 0.111 - 8: 0.25 - 9: 1.0 -
Calculo en orden
1: 1.0 - 2: 0.125 - 3: 0.037 - 4: 0.015 - 5: 0.008 - 6: 0.004 - 7: 0.002 - 8: 0.001 - 9: 0.001 - 10: 0.001 -
Calculo al revés
0: 0.001 - 1: 0.001 - 2: 0.001 - 3: 0.002 - 4: 0.004 - 5: 0.008 - 6: 0.015 - 7: 0.037 - 8: 0.125 - 9: 1.0 -
Result 1: normal:1.547-contra:1.547
Result 2: normal:1.1939999999999995-contra:1.194
```

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

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

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

b. El lenguaje de programación C++ requiere que el valor de π se encuentro dentro de 10^{-10}. ¿Cuántos términos de la serie se necesitarán sumar para obtener este grado de precisión?

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

Basándonos en estos valores

4. Compare los siguientes tres algoritmos. ¿Cuándo es correcto el algoritmo de la parte 1a?
   a. 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.
   b. 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.
   c. 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 ∈ R, 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.

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

$$\sum_{i=1}^{n}\sum_{j=1}^{i} a_i b_j$$

   R: La multiplicación es de $\frac{n(n+1)}{2}$ cantidad, y las sumas es $\frac{n(n+1)}{2} - 1$

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

   R: La multiplicación es de n cantidad, y las sumas es $\frac{(n-1)(n+2)}{2}$ lo sacamos de $\frac{n(n+1)}{2} + (n-1)$

Discusiones

1. Escriba un algoritmo para sumar la serie finita $\sum_{i=1}^{n} x_i$ 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
```

2. 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 = $b^2 - 4ac$
   Si D < 0, entonces es un número complejo, caso contrario no.
   Si D>= 0, entonces tenemos las formulas:
   Si b>= 0:
   $x1 = \frac{-b-\sqrt{D}}{2a}$
   $x2 = \frac{2c}{-b-\sqrt{D}}$
   Sino:
   $x1 = \frac{-b+\sqrt{D}}{2a}$
   $x2 = \frac{2c}{-b+\sqrt{D}}$

3. Suponga que

$$\frac{1-2x}{1-x+x^2}+\frac{2x-4x^3}{1-x^2+x^4}+\frac{4x^3-8x^7}{1-x^4+x^8}\pm\cdots=\frac{1-2x}{1+x+x^2}$$

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)
```