Ciencias de la Computación
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS UNIVERSIDAD DE CHILE

CC1001 – Tarea #1

Dos Métodos Numéricos para Aproximar el Valor de π

Profesor: Nelson Baloian T.
Profesores Auxiliares: Mauricio Giadach P. – Francisco Gutiérrez F.

INDICACIONES GENERALES

- El desarrollo de la tarea es individual. Sin embargo, está permitido (y recomendado) discutir sobre posibles caminos de solución con sus compañeros y profesores auxiliares, con la condición de que al momento de sentarse a escribir el código de sus programas no reciba ayuda externa de ningún tipo (esto incluye mostrar todo o parte de su código a otra persona).

- En caso de detectarse algún indicio de copia, se calificará con la nota mínima tanto a quien copia, como a quien se deja copiar.

- Su tarea debe estar compuesta de un archivo llamado **tarea1.py** (cuyo esqueleto encontrará adjunto a este enunciado) y un informe **en formato PDF** que documente su razonamiento e ilustre el correcto funcionamiento de su programa.

- La entrega se debe hacer vía U-Cursos **hasta las 23:59 del día MIÉRCOLES 6 DE ABRIL**. No se recibirán tareas pasado este plazo.

INTRODUCCIÓN

La constante *Pi* (π) se define en la Geometría Euclidiana como la razón entre el perímetro de una circunferencia y su diámetro. Otra definición que se le suele dar es la superficie cubierta por un círculo de radio unitario. *Pi* es considerada una de las constantes más importantes, empleándose frecuentemente en las distintas ramas de la Matemática, la Física y las Ciencias de la Ingeniería.

Uno de los problemas que involucra el cálculo de funciones que contienen esta constante deriva del hecho que *Pi* es un número irracional, es decir, no puede expresarse como la fracción de dos números enteros. En otras palabras, se dice que tiene una representación decimal infinita no periódica.

Por esta razón, es imposible calcular su valor exacto y se hace entonces necesario aproximar su valor mediante métodos numéricos con un cierto margen de error tolerable, que depende de la circunstancia. En esta tarea, le proponemos estudiar dos de los algoritmos más clásicos para el cálculo de π: la aproximación mediante el uso de la serie de Leibniz y la aplicación de un método de Montecarlo.

1. APROXIMACIÓN POR LA SERIE DE LEIBNIZ

Gottfried Leibniz fue un matemático alemán al que, junto a Newton, se le atribuye haber inventado el cálculo infinitesimal. Uno de sus trabajos fue el haber desarrollado una serie infinita de términos, cuya suma en el límite infinito tiende exactamente al valor $\frac{\pi}{4}$. Dado que no podemos efectuar la suma de "infinitos" términos (puesto que el computador no lo permite), nos contentaremos con una aproximación finita de un número arbitrario de ellos.

Esta serie se define como: $\frac{1}{1}-\frac{1}{3}+\frac{1}{5}-\frac{1}{7}+\frac{1}{9}-...+...=\frac{\pi}{4}$.

a) Implemente en Python una función de nombre **`termino(i)`** tal que, dado el parámetro *i*, devuelva el término i-ésimo de la serie de Leibniz.

```
termino(1) devuelve  1
termino(2) devuelve -0.3333333
```

b) Defina la función **`suma(n)`** tal que, dado el parámetro *n*, devuelva la suma de los *n* primeros términos de la serie de Leibniz.

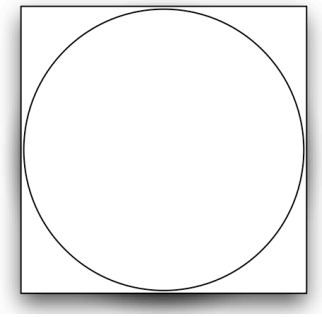
`suma(4)` corresponde al resultado de calcular: $1-\frac{1}{3}+\frac{1}{5}-\frac{1}{7}$

c) Usando lo anterior y el hecho que la suma corresponde teóricamente al valor $\frac{\pi}{4}$, defina la función **`leibniz(n)`** tal que, dado el parámetro *n*, devuelva la aproximación de π con los primeros *n* términos de la serie de Leibniz.

2. APROXIMACIÓN POR EL MÉTODO DE MONTECARLO

Otro método para aproximar el valor de π consiste en el uso del método de Montecarlo. Este consiste en repetir un número "grande" de veces el lanzamiento de una moneda en una baldosa que tiene un círculo inscrito. Se cuenta cuántas veces cae la moneda dentro del círculo, cuántas veces cae fuera de él y con esa información se puede determinar la relación que existe entre el número de repeticiones y la razón entre las áreas de ambas zonas.

Para simplificar el problema, consideraremos una baldosa cuadrada de largo 2 y un círculo centrado en el punto (1,1), tal como lo muestra la figura:



De esta forma, si *nTotal* representa el número total de monedas lanzadas y *nBlanco*, el número de monedas que caen dentro del círculo, tendremos la relación:

$$\frac{nBlanco}{nTotal} = \frac{Área\ círculo}{Área\ cuadrado} = \frac{\pi R^2}{(2R)^2} \xrightarrow{radio\ círculo = 1} \frac{\pi}{4}$$

Note que para obtener las coordenadas de los puntos donde caen las monedas, usando la función **`random()`** es posible generar 2 números aleatorios (uno para eje, X e Y) reales en el intervalo [0,1). Como buscamos números aleatorios entre 0 y 2 (los vértices de cada uno de los lados de la baldosa), podemos proceder simplemente a evaluar **`2 * random()`** para obtener un número que representa el valor del punto donde cae la moneda en cada eje.

Por último, recuerde que un punto *(x, y)* está contenido dentro de un círculo si la distancia entre su centro y dicho punto es menor o igual que el radio. Para este caso, tenemos entonces que un punto *(x, y)* se encontrará dentro del círculo cuyo centro es el punto *(1, 1)* si se verifica la relación: $\sqrt{(x-1)^2 + (y-1)^2} \leq 1$.

a) Implemente en Python la función **`estaDentro(x, y)`** tal que, dadas las coordenadas *x* e *y* de un punto cualquiera, devuelva `True` si está dentro del círculo y `False` en caso contrario.

b) Utilizando la función anterior, implemente **`montecarlo(n)`** tal que, dado el parámetro *n* que representa el número total de lanzamientos, devuelva una aproximación del valor de π.

3. ANÁLISIS

En esta parte vamos a implementar el programa principal que nos permitirá comparar los valores dados por las aproximaciones en ambos algoritmos y el valor teórico de π (que asumiremos correspondiente al devuelto por `math.pi`).

a) Defina la función **`errorPorcentual(valor)`** tal que, dado el parámetro *valor* que corresponde a una aproximación del valor de π, devuelva el error porcentual entre el valor aproximado y el valor teórico. Note que este valor debe ser siempre positivo, ya sea si la aproximación fue por exceso o por defecto.

b) Implemente el programa principal, tal que le pida al usuario ingresar el número de iteraciones a realizar (el mismo para ambos algoritmos) y devuelva en pantalla una tabla con los resultados y el error porcentual en cada caso. Si el parámetro ingresado es negativo o cero, debe repetir esta operación hasta que el usuario ingrese un número válido.

```
Ingrese el numero de iteraciones: -5
Por favor, ingrese un numero positivo!!

Ingrese el numero de iteraciones: 100

    N         Leibniz          Error         Montecarlo         Error
    1
    2
   ...
   100
```