INF285 - Computación Científica

Tarea 0 - Parte 2

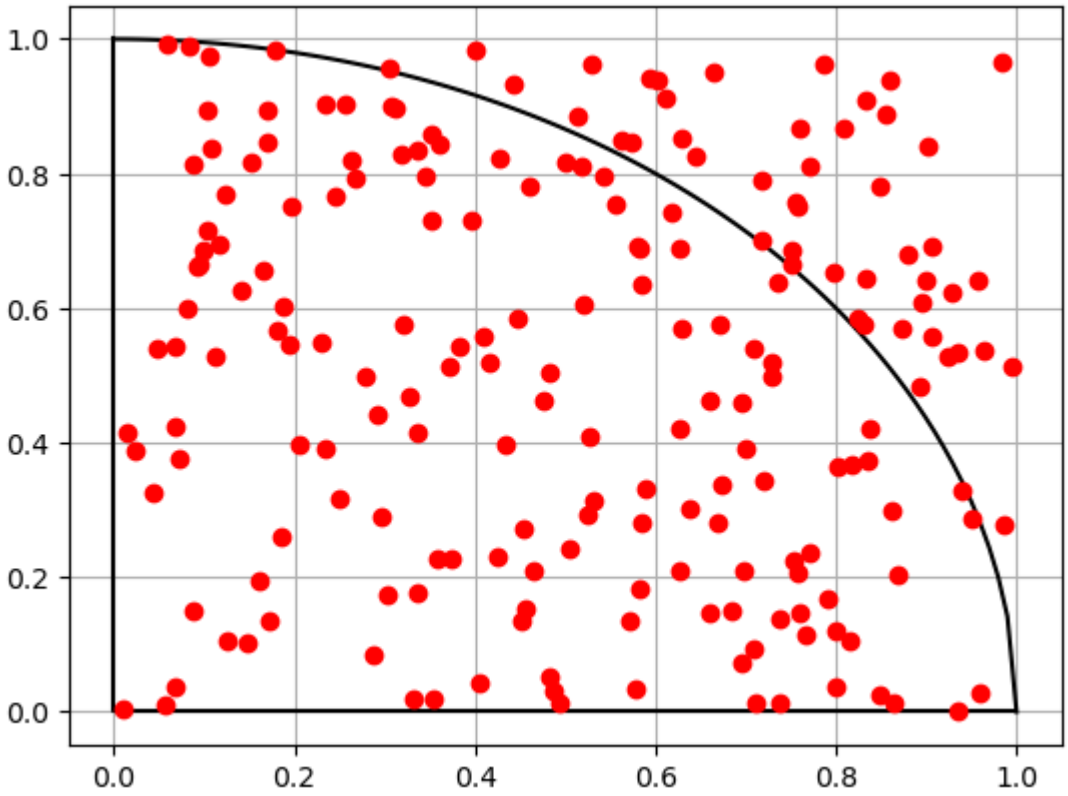
Sábado 11 de marzo de 2023

Estimación de pi

Esta famosa constante matemática $\pi = 3.14159265359$ podemos estimarla utilizando el método de Monte Carlo, el cual consiste en obtener una cantidad de muestras aleatorias para realizar un cálculo numérico. En este caso obtendremos muestras aleatorias para obtener una estimación de π.

La idea es simular puntos aleatorios (x, y) dentro de un cuarto de circunferencia de radio 1. El objetivo es contar la cantidad de puntos que hay en el cuarto de circunferencia, de tal forma de encontrar una relación entre el área del cuarto de circunferencia y el área total de muestreo.

En la siguiente figura se muestra una idea:



Sabemos que el área del espacio de muestreo es 1, mientras que el área del cuarto de circunferencia es $\pi/4$. Por lo tanto la razón entre las áreas es:

$$\frac{\text{area cuarto de circunferencia}}{\text{area total}} = \frac{\pi/4}{1}$$

Luego si tomamos una gran cantidad de muestras aleatorias, entonces:

$$\frac{\text{area cuarto de circunferencia}}{\text{area total}} = \frac{\text{cantidad de puntos dentro del cuarto de circunferencia}}{\text{cantidad de muestras totales}} =$$

Entonces podemos estimar π como:

$$\pi \approx \frac{4 \times \text{cantidad de puntos dentro del cuarto de circunferencia}}{\text{cantidad de muestras totales}}$$

A continuación se muestra un código que estima π utilizando list comprehension:

In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
from random import random
```

In [ ]:
```
def calcular_pi_list(n):
    np.random.seed(seed=9999)
    x = [random() for i in range(n)]
    y = [random() for i in range(n)]

    d = [x[i]**2 + y[i]**2 for i in range(n)]
    inside = 0
    for i in range(n):
        if d[i] < 1:
            inside += 1

    return 4*inside / n
```

Podemos calcular el tiempo que se demora estimar π con list comprehension utilizando 10000 muestras aleatorias

In [ ]:
```
#tiempo de función con lista
t1 = %timeit -o calcular_pi_list(10000)
```

Pregunta 1 Construya la función *calcular_pi_array(n)* que estime el valor de π utilizando numpy array y considerando n muestras aleatorias. Usted debe completar la siguiente función:

In [ ]:
```
#función a construir
def calcular_pi_array(n):
    np.random.seed(seed=9999)
    #acá va su código
    return 4*inside / n
```

Podemos calcular el tiempo que se demora estimar π con numpy array utilizando 10000 muestras aleatorias

In [ ]:
```
#tiempo de función con array
t2 = %timeit -o calcular_pi_array(10000)
```

Se calcula las veces que estimar π con array es mejor que con list comprehension.

In [ ]:
```
t1.average/t2.average
```

Esta función que acaba de realizar, se vectoriza.

In [ ]:
```
#se vectoriza la función
calcular_pi_array_vec = np.vectorize(calcular_pi_array)
```

Pregunta 2 El error exacto $E_e(n)$ utilizando n muestras aleatorias, viene dado por:

$$E_e(n) = |\pi - \pi_n|$$

donde π_n es la estimación de π utilizando n muestras aleatorias. Por otro lado, $E_a(n)$ es el error de aproximación de la estimación de π utilizando n muestras aleatorias:

$$E_a(n) = |\pi_{n_{max}} - \pi_n|$$

donde $\pi_{n_{max}}$ es la estimación de π utilizando la máxima cantidad de muestras aleatorias. Al graficar estos errores para cada n en escalas logarítmicas, se puede observar que tienen una tendencia lineal. Al ejecutar el siguiente código usted notará la distribución de los errores:

In [ ]:
```
#plotear la curva del error exacto y aproximado
n = np.logspace(2,5,50,dtype=int)
pi_n = calcular_pi_array_vec(n)

error_ex = np.abs(np.pi-pi_n)
plt.loglog(n,error_ex,"bo",label=r"$E_e(n)$")

error_ap = np.abs(pi_n[-1] - pi_n[:-1])
plt.loglog(n[:-1],error_ap,"rx",label=r"$E_a(n)$")

plt.xlabel(r"$n$")
plt.legend()
plt.grid()
plt.show()
```

Entonces el error viene dado por una función de potencia $E = c\, n^{\alpha}$. Aplicando logaritmo se obtiene que:

$$\log(E) = \log(c) + \alpha\, \log(n)$$

Se solicita entonces, obtener el valor de α mediante el uso de *regresión lineal*. Este valor es la pendiente de la recta de regresión que pasa por los puntos graficados anteriormente. Debe realizar este cálculo tanto para el error exacto como para el error de aproximación.

In [ ]:
```
#calcular pendiente de la curva de regresión
#para el error exacto y el error aproximado
```

Pregunta 3 Sea la integral,

$$I = \int_{2\pi}^{11\pi/4} g(t)\, dt = \int_{2\pi}^{11\pi/4} (3\,\sin(t) + \cos(t))\, dt$$

donde $2\pi \leq t \leq 11\pi/4$ y $0 \leq g(t) \leq 4$.

3.1 Utilizando el método de Monte Carlo aplicado anteriormente para la estimación de π, estime la integral I utilizando n muestras aleatorias.

In [ ]:
```
#calcular la integral de f(x) utilizando el método de Monte Carlo
def estimar_integral(n):
    np.random.seed(seed=9999)
    #acá va su código
    return e_int
```

3.2 Calcule el error de aproximación $E_a(n)$ para la estimación de la integral I.

In [ ]:
```
#cálculo del error de aproximación (recuerde que puede vectorizar la función ar
```

3.3 Calcule el valor de α utilizando *regresión lineal* para la recta de regresión del error de aproximación.

In [ ]:
```
#cálculo de alpha
```