Técnica de Monte-Carlo

Generar una introducción e implementar la resolución del problema utilizando Monte-Carlo.

1. Introducción

El algoritmo de MonteCarlo es aleatorio cuya salida puede ser incorrecta con una cierta probabilidad. Siempre se espera que la respuesta devuelta por un algoritmo determinista sea correcta, el algoritmo de Montecarlo con sesgo falso siempre es correcto cuando devuelve falso, un algoritmosesgado verdader siempre es correcto cuando devuelve verdadero. Esto describe algortimos con errores unilaterales, es posible que otros no tengan sesgo, entonces se dice que estos no tiene errores de dos caras. Para un algoritmo de Monte Carlo con errores unilaterales, la probabilidad de falla puede reducirse ejecutando el algoritmo varias veces.

2. Implementación

```
from sympy import integrate, init_printing
from sympy.abc import a,b,c,x
import math
import numpy as np
```

```
#Definicion de la funcion
fx = x**2
#Obtener el area por integrales
valor_integral = integrate(fx, (x,0,1))
valor_integral=float(valor_integral)
valor_integral
```

```
0.3333333333333333
```

```
#Generar los valores para el metodo de monte carlo
def generar_valores(n):

  valores_x=[]
  valores_y=[]
  for i in range(n):
    valor_x= np.random.random()
    valores_x.append(valor_x)
    valor_y= math.pow(valor_x,2)
    valores_y.append(valor_y)
```

```
valores_x=[]
valores_y=[]
N=100
for i in range(N):
  valor_x= np.random.random()
  valores_x.append(valor_x)
  valor_y= math.pow(valor_x,2)
  valores_y.append(valor_y)

#metodo_montecarlo
#generar numeros
valores_aleatorios=[]
for i in range(100):
  numero_ale = np.random.random()
  valores_aleatorios.append(numero_ale)


#validaciones
contador=0
for i in range(N):
  if valores_aleatorios[i]<=valores_y[i]:
    contador=contador+1
valor_montecarlo=contador/N
print("Area por integral: ",valor_integral, " - Area aproximada por Montecarlo: ",valor_monte
```

```
Area por integral:  0.3333333333333333  - Area aproximada por Montecarlo:  0.33
```

✓ 0s completed at 7:04 PM