UNIVERSIDAD NACIONAL DE SAN AGUSTIN
FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS
ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA

Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación
Aprobación: 2022/03/01 Código: GUIA-PRLE-001

INFORMACIÓN BÁSICA					
ASIGNATURA:	**Física Computacional.**				
TÍTULO DE LA PRÁCTICA:	**Práctica de Simulación de Monte Carlo.**				
NÚMERO DE PRÁCTICA:	**08**	AÑO LECTIVO:	**2022-A**	NRO. SEMESTRE:	**VII**
FECHA DE PRESENTACIÓN:	06/07/2022	HORA DE PRESENTACIÓN:	**19:19**		
Integrante(s): Alván Ventura Edsel Yael			NOTA		
DOCENTE(s): Danny Giancarlo Apaza Veliz.					

Práctica 8
Simulación de Monte Carlo
Física Computacional

Escrito por
Alván Ventura, Edsel Yael
ealvan@unsa.edu.pe

Profesor
Apaza Veliz, Danny Giancarlo
dapazav@unsa.edu.pe

06/07/2022

Aplicando la integración por método de Monte Carlo resuelva las siguientes integrales:

1 Problema 1

Calcule:

$$I[g(X)] = \int_0^1 e^{x^2}\, dx \tag{1}$$

1.1 Análisis

Para ver como funciona el método de Monte Carlo, se consideró como las integrales de cada función pueden ser aproximadas conforme a un número gigante de numeros aleatorios que estan entre los limites de la función.

Y cada vez que se eligen otros numeros aleatorios y se calculan con ellos la integral de la función, y el conjunto de todos esos intentos se ponen en un histograma para representar el numeros de respuestas diferentes que dan y cuantas dieron lo mismo, y por lo tanto ver la respuesta aproximada en el mayor numero de respuestas que apuntan hacia un mismo numero(que sería nuestra aproximación).

1.2 Programación

En esta sección se muestra la implementación realizada en *Python 3.x* y las herramientas usadas. Las herramientas usadas son:

- La librería *matplotlib* para hacer el histograma.
- La librería *scipy* para el calculo de números aleatorios.
- La librería estándar *math* para el número de Euler.

A continuación se muestra el código:

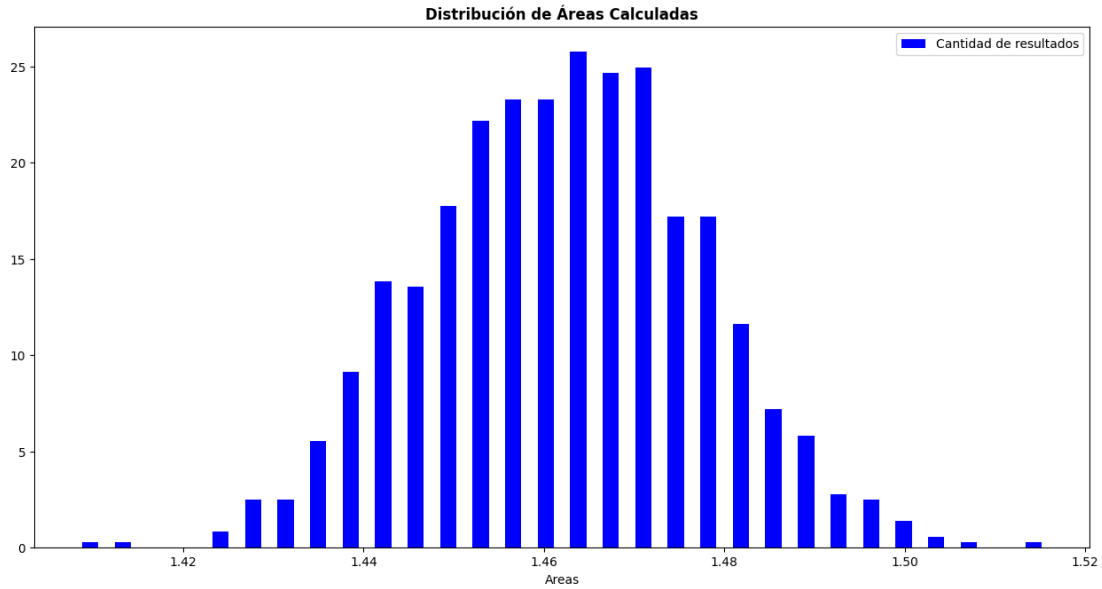
Archivo *MonteCarlo.py*

```
1  from scipy import random
2  import matplotlib.pyplot as plt
3  import math as m
4  #a = limite de inicio de la integral
5  #b = limite final
6  #f = la funcion a aplicar en la integral
7  #N = el rango de numeros aleatorios entre a y b
8  def calcularArea_MonteCarlo(a,b,f,N):
9      areas = []
10     barras = 30#numero de barras a mostrar
11     colors = ['tomato', 'blue', 'lime']#colores para el grafico
12     for i in range(N):
13         #esta funcion genera numeros aleatorios
14         #entre a y b, y lo hacen N veces
15         equis = random.uniform(a,b,N)#numeros aleatorios
16         integral = 0.0
17         for i in range(N):
18             integral += f(equis[i])
19
20         answer = (b-a)/float(N)*integral
21         #crear lista de todas las
22         #areas que nos dan, con diferentes
23         #numeros aleatorios
24         areas.append(answer)
25     print("La integral es: ",answer)
26     plt.title("Distribucion de Areas Calculadas",fontweight ="bold")
27
28     plt.hist(areas, barras, density = True,
29     histtype ='bar',
```

```
30      color = colors[1],
31      label="Cantidad de resultados",
32      rwidth = 0.5)
33      plt.legend(prop ={'size': 10})
34      plt.xlabel("Areas")
35      plt.show()
36
37  def main():
38      a = 0
39      b = 1
40      f = lambda x: m.e**(-1*x)
41      N = 10000
42      calcularArea_MonteCarlo(a,b,f,N)
43
44  if __name__ == "__main__":
45      main()
```

1.3 Resultados

A continuación se muestra la aproximación con la simulación de Monte Carlo y la distribución de los valores dados por el algoritmo y sus frecuencias:



Como se puede ver en la imágen anterior, las barras más altas son las aproximaciónes más certeras de la simulación de Monte Carlo.

A veces, por los numéros aleatorios tienden a estar a un lado del valor exacto de la integral, pero generalmente dan un valor aproximado conforme a la cantidad de intentos que se le da al algoritmo.

El valor aproximado a la integral es: 1.46579
Con 1000 iteraciones.

2 Problema 2

Calcule:

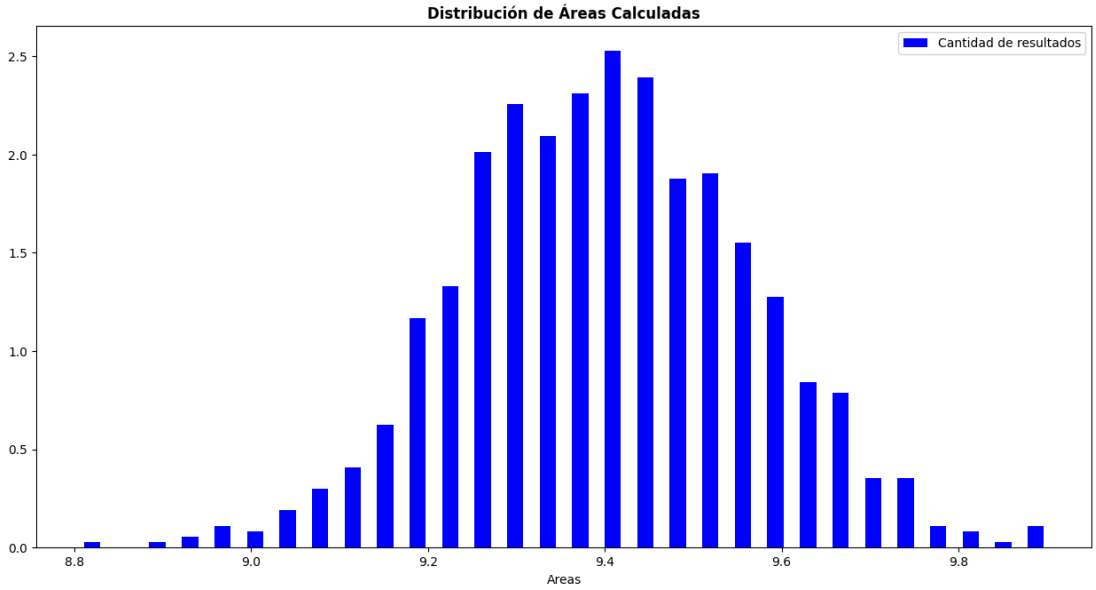
$$I[g(X)] = \int_{-1}^{1} e^x 4\, dx \tag{2}$$

2.1 Programación

```
1 def main():
2     a = -1
3     b = 1
4     f = lambda x: 4*m.e**x
5     N = 10000
6     calcularArea_MonteCarlo(a,b,f,N)
```

2.2 Resultados

A continuación se muestra la aproximación con la simulación de Monte Carlo y la distribución de los valores dados por el algoritmo y sus frecuencias:



Como se puede ver en la imágen anterior, las barras más altas son las aproximaciónes más certeras de la simulación de Monte Carlo.

El valor aproximado a la integral es: 9.4138
Que esta muy cerca del valor 9.401(valor más aproximado)
Con 10000 iteraciones.

3 Problema 3

Calcule:

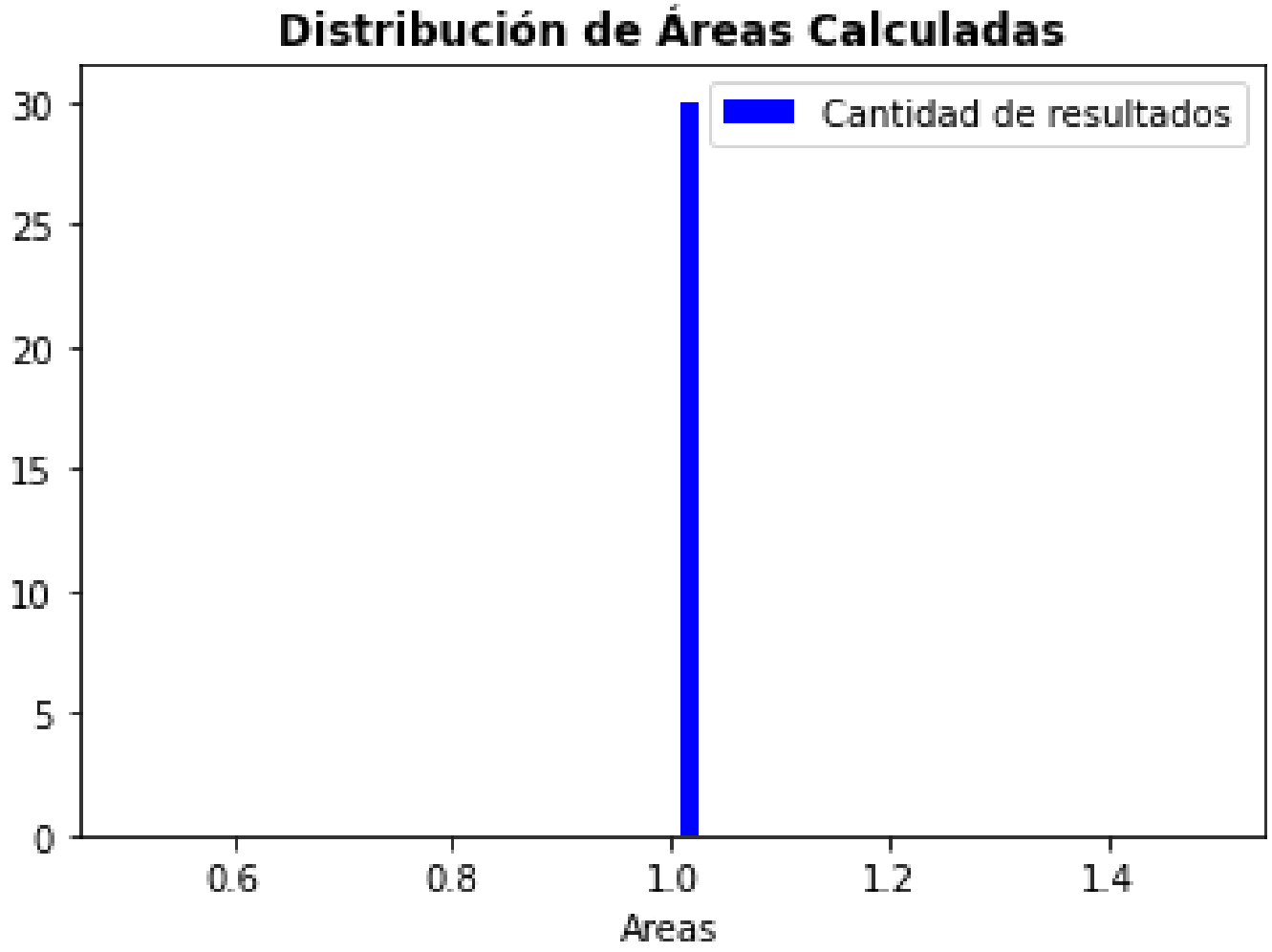
$$I[g(X)] = \int_0^1 (1 - e^2)^{\frac{1}{2}}\, dx \tag{3}$$

3.1 Programación****

```
1 def main():
2     a = 0
3     b = 1
4     f = lambda x: 1
5     N = 1000
6     calcularArea_MonteCarlo(a,b,f,N)
```

3.2 Resultados

A continuación se muestra la aproximación con la simulación de Monte Carlo y la distribución de los valores dados por el algoritmo y sus frecuencias:



En este caso, la distribución de frecuencias dio un valor de 1. Esto es porque, $(1-e^2)^{\frac{1}{2}}$ es una constante. Por lo que:

$$(1-e^2)^{\frac{1}{2}} \int_0^1 dx \tag{4}$$

Por lo que el resultado es: $(1-e^2)^{\frac{1}{2}} * [1-0]$

El valor aproximado a la integral es: 2.527658...
Con 1000 iteraciones.

4 Problema 4

Calcule:

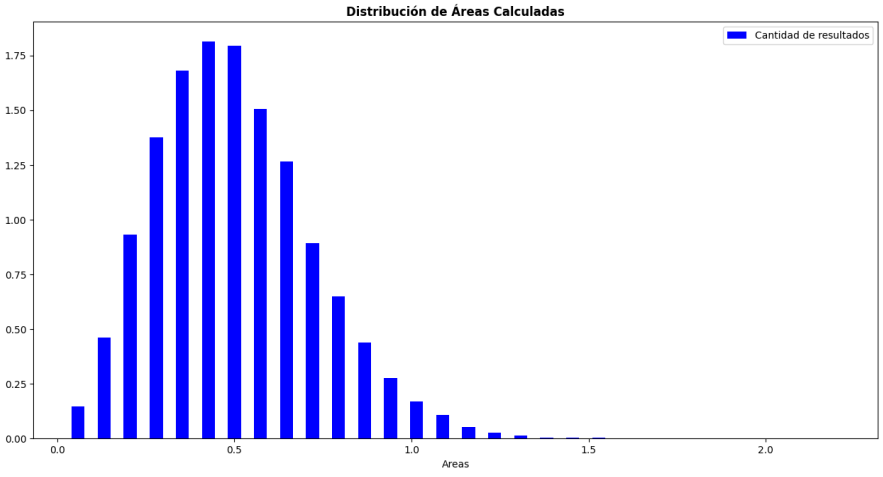
$$I[g(X)] = \int_0^\infty x(1+x^2)^{-2}\,dx \tag{5}$$

4.1 Programación

```
1 def main():
2     a = 0
3     b = 10000
4     f = lambda x: x*((1+x**2)**(-2))
5     N = 20000
6     calcularArea_MonteCarlo(a,b,f,N)
```

4.2 Resultados

A continuación se muestra la aproximación con la simulación de Monte Carlo y la distribución de los valores dados por el algoritmo y sus frecuencias:



Como se puede ver en la imágen anterior, las barras más altas son las aproximaciónes más certeras de la simulación de Monte Carlo.

El valor aproximado a la integral es: 0.5
Con 20000 iteraciones.

5 Problema 5

Calcule:

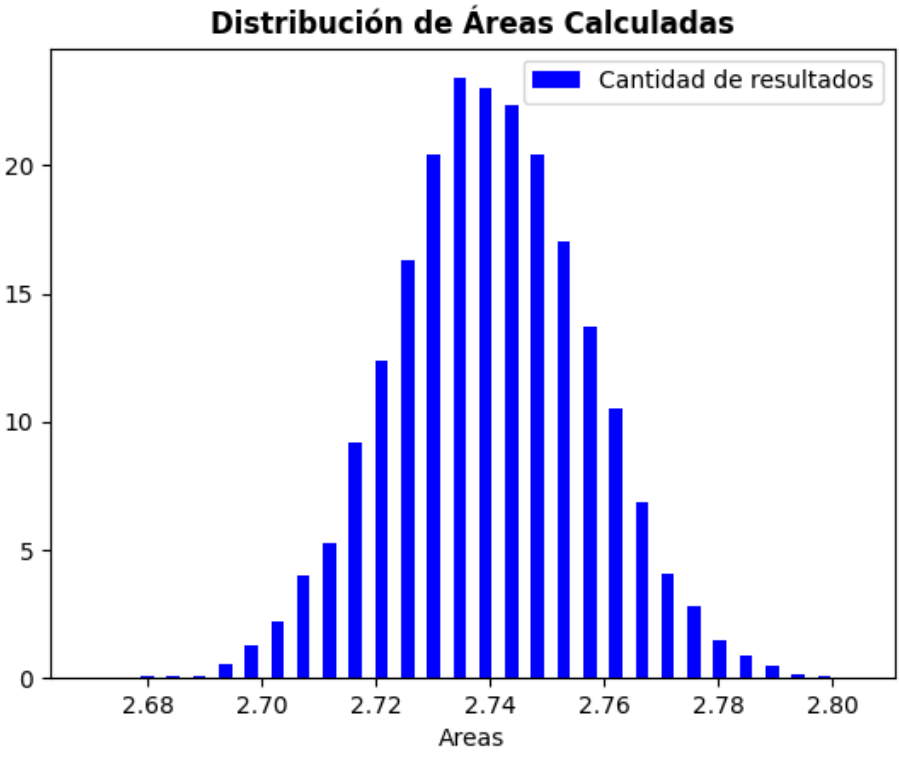
$$I[g(X)] = \int_0^1 e^{x+x^2}\, dx \tag{6}$$

5.1 Programación

```
1 def main():
2     a = 0
3     b = 1
4     f = lambda x: m.e**(-1*x)
5     N = 10000
6     calcularArea_MonteCarlo(a,b,f,N)
```

5.2 Resultados

A continuación se muestra la aproximación con la simulación de Monte Carlo y la distribución de los valores dados por el algoritmo y sus frecuencias:



Como se puede ver en la imágen anterior, las barras más altas son las aproximaciónes más certeras de la simulación de Monte Carlo.

El valor aproximado a la integral es: 2.7482
Que esta muy próximo de 2.7399(un valor más aproximado a la integral)
Con 10000 iteraciones.

6 Problema 6

Calcule:

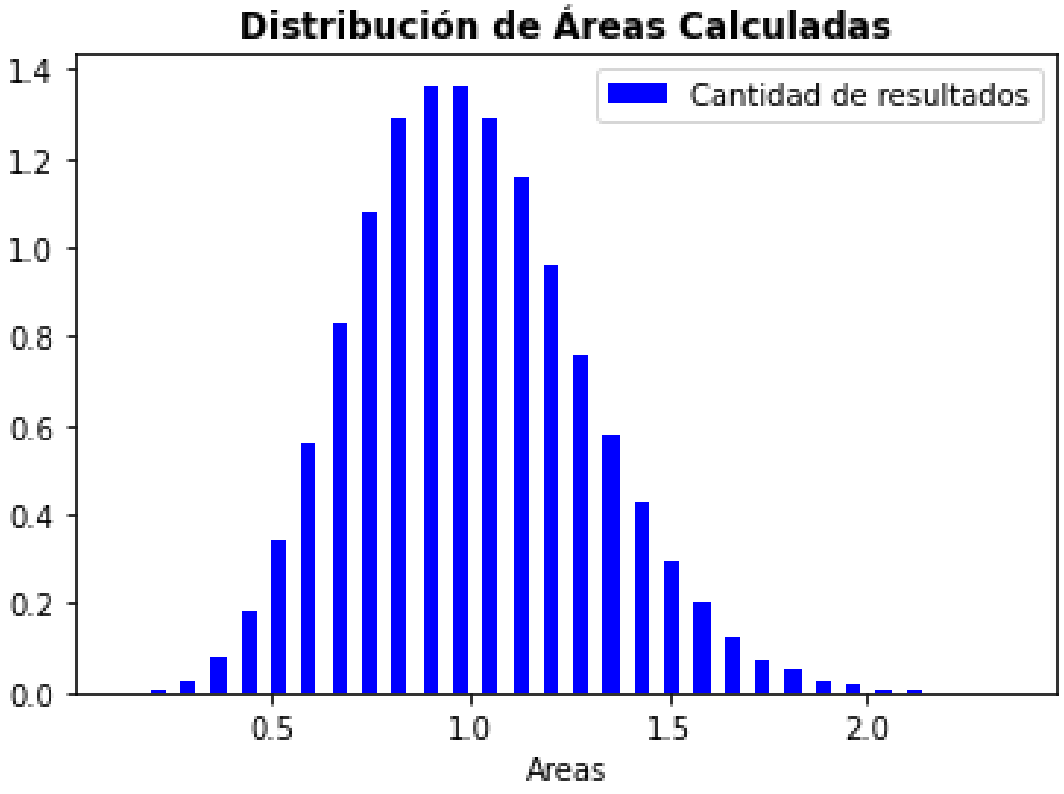
$$I[g(X)] = \int_0^{\infty} e^{-x}\, dx \tag{7}$$

6.1 Programación

```
1 def main():
2     a = 0
3     b = 100000
4     f = lambda x: m.e**(-1*x)#x*((1+x**2)**(-2))
5     N = 50000
6     calcularArea_MonteCarlo(a,b,f,N)
```

6.2 Resultados

A continuación se muestra la aproximación con la simulación de Monte Carlo y la distribución de los valores dados por el algoritmo y sus frecuencias:



El valor aproximado a la integral es: 0.91
Que esta muy próximo de 1(valor real de la integral)
Con 10000 iteraciones.

7 Anexos

En el siguiente link se encuentra los archivos de Python:
`https://drive.google.com/drive/folders/1WiiF428zxfC5vnRzmNjqNCvkJYgYGuyK?usp=sharing`