Práctica 5 + extra 1 + extra 2

December 26, 2023

Vamos a comentar la práctica 5. Para ello, se irán mostrando las partes del código junto a los resultados obtenidos.

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

Función pedida en la respectiva práctica, escrita como función en python:

[2]:
```
def f(x):

    return (np.sin(1/(x*(2-x))))**2
```

[3]:
```
maximo = int(input('Valor máximo de su función: '))
N = int(input('Número de pasos de Monte Carlo: '))
a = float(input('Valor mínimo de la variable independiente: '))
b = float(input('Valor máximo de la variable independiente: '))
```

```
Valor máximo de su función: 1
Número de pasos de Monte Carlo: 100000
Valor mínimo de la variable independiente: 0.001
Valor máximo de la variable independiente: 2
```

A continuación, se generan puntos, de forma aleatoria, y se reescalan al intervalo (a,b) y valor máximo de f(x) introducidos por el usuario.

[4]:
```
u = np.random.rand(N)

x = a + (b - a) * u  #reescalado para eje OX

v = np.random.rand(N)

y = maximo * v #reescalado para eje OY

fx = f(x) #evaluamos en f(x)
```

Se sigue con las funciones np.where() y np.count_nonzero(), las cuales nos devolverán los puntos y el número de elementos que cumplen y < fx, respectivamente.

[5]:
```
idx = np.where(y < fx)[0]
```

```
Ndebajo = np.count_nonzero(y < fx)
```

Se calcula área bajo el intervalo rectangular introducido, valor de la integral y su error asociado al método Monte Carlo.

[6]:
```
Arect = (b - a) * maximo

integral = Arect * Ndebajo / N

error = Arect * np.std(y < fx) / np.sqrt(N)

print(f'El valor de la integral para {N} pasos Monte Carlo es I = {integral:.
 ↪4f}')
print(f'Su error asociado es {error:.4f}')
```

```
El valor de la integral para 100000 pasos Monte Carlo es I = 1.4574
Su error asociado es 0.0028
```

Se le pide al usuario que introduzca un paso para la representación del valor de la integral y su error, en función del número de pasos Monte Carlo. Así, introducido un valor de 10000 pasos MC y un paso de 10 tendremos 1000 puntos para representar. Al presentar el código una forma vectorizada el hecho de trabajar con muchos valores no nos afecta significativamente en el tiempo de ejecución.

[7]:
```
paso = int(input(f'Introduzca un paso para representar el valor de la integral,␣
 ↪con sus errores, en función del número de pasos Monte Carlo. TENGA EN CUENTA␣
 ↪QUE EL VALOR DESDE EL QUE SE PARTE ES 100 Y SE LLEGARÁ HASTA UN VALOR DEL␣
 ↪ORDEN DE {N}: '))
# Generamos un array con los valores de los pasos Monte Carlo desde 100 hasta N␣
 ↪con el paso indicado
pasos = np.arange(100, N + 1, paso)
```

```
Introduzca un paso para representar el valor de la integral, con sus errores, en
función del número de pasos Monte Carlo. TENGA EN CUENTA QUE EL VALOR DESDE EL
QUE SE PARTE ES 100 Y SE LLEGARÁ HASTA UN VALOR DEL ORDEN DE 100000: 10
```

Ahora se calcula el valor de la integral de forma acumulada para el respectivo número de pasos Monte Carlo. Este paso ha sido crucial para reducir los tiempos de ejecución pues nos basta tomar nuestro vector con los valores y < fx y realizar la suma acumulada hasta el paso requerido.

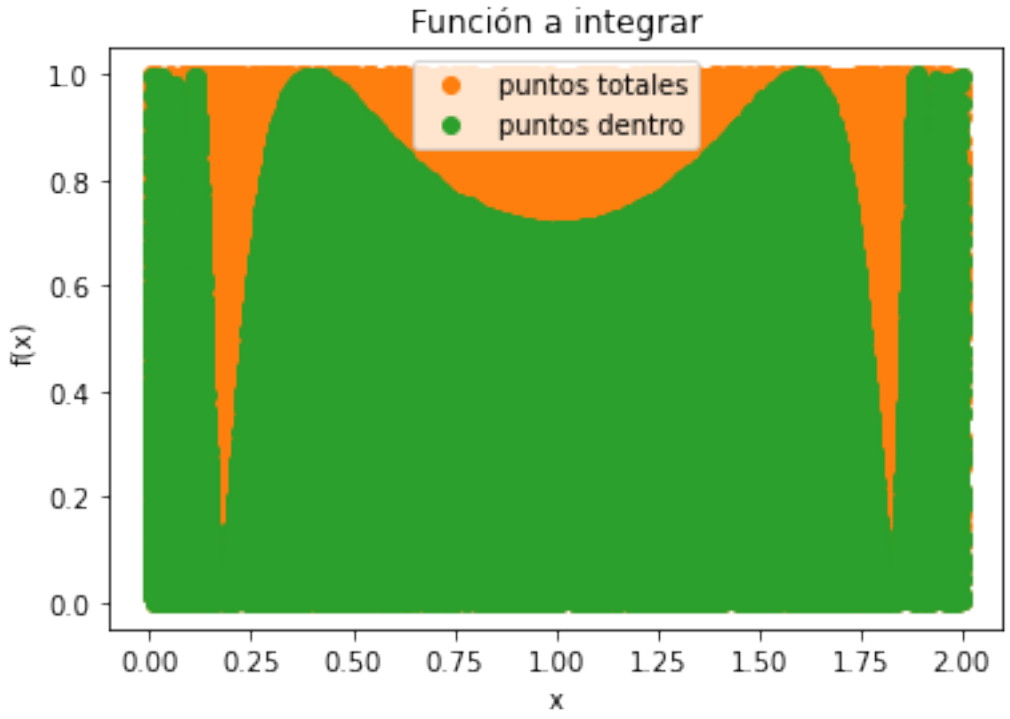
[8]:
```
I = Arect * np.cumsum(y < fx)[pasos - 1] / pasos

E = Arect * np.std(y < fx) / np.sqrt(pasos)
```

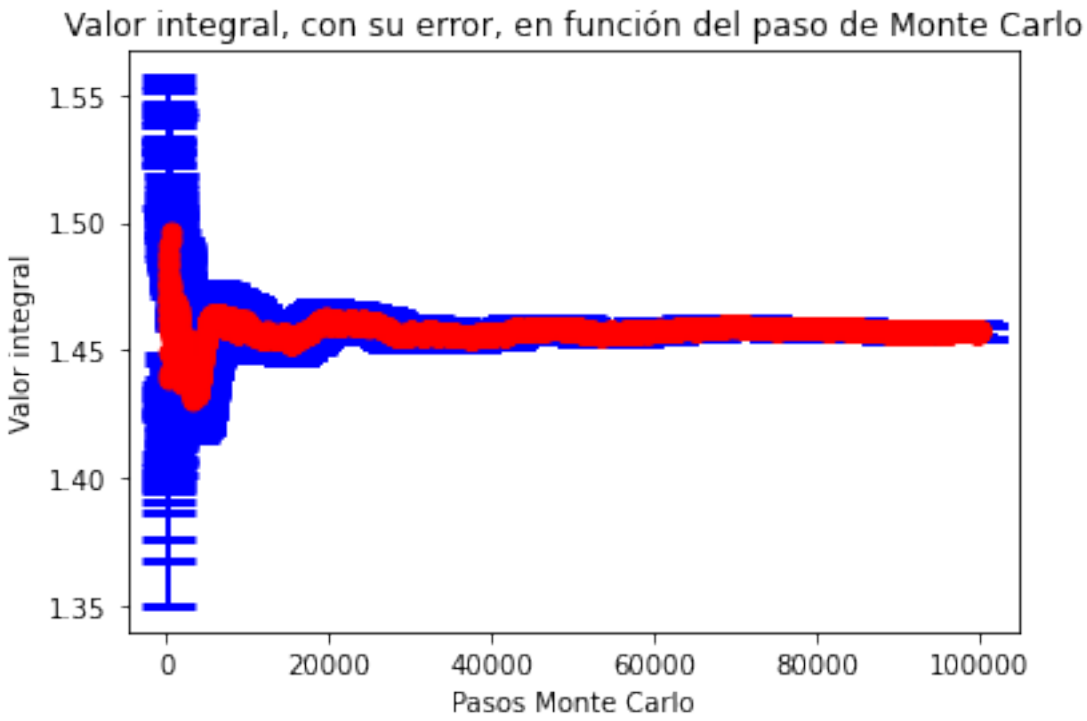
Ya por último, pintamos con plt las gráficas que se muestran.

[9]:
```
plt.figure(1)
plt.plot(np.linspace(0, 2, 1000), f(np.linspace(0.001, 2, 1000)))
plt.plot(x, y, 'o', label = 'puntos totales')
plt.plot(x[idx], y[idx], 'o', label = 'puntos dentro')
```

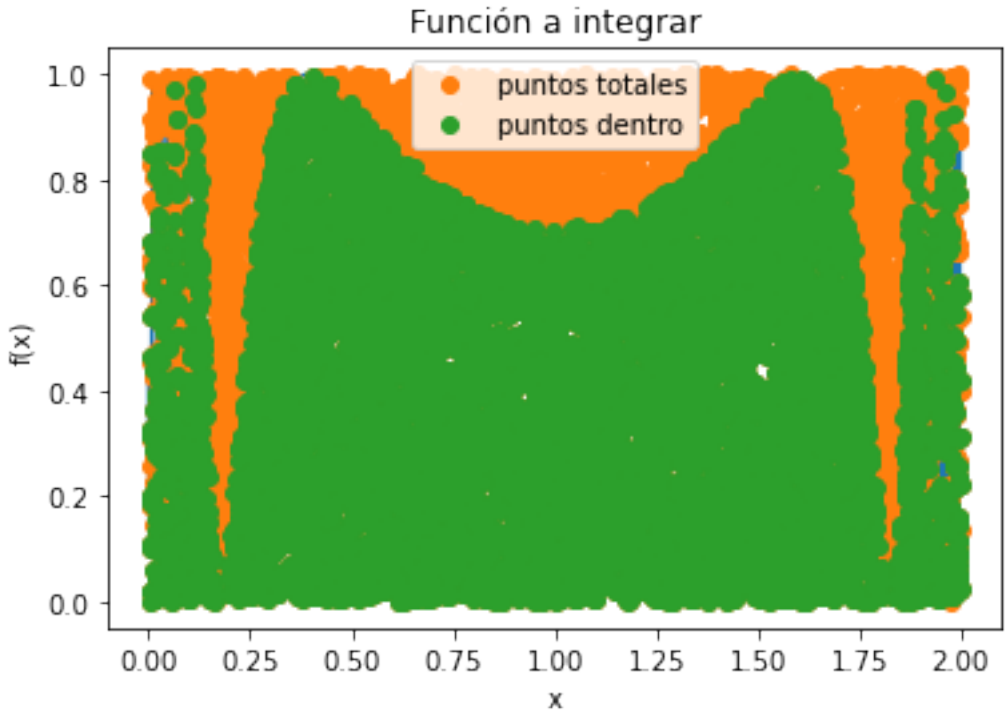
```
plt.xlabel('x')
plt.ylabel('f(x)')
plt.title('Función a integrar')
plt.legend()
plt.show()

plt.figure(2)
plt.errorbar(pasos, I, yerr = E, fmt="--ro", ms = 7, ecolor='b', elinewidth=2,␣
 ↪capsize=10, capthick=3)
plt.title('Valor integral, con su error, en función del paso de Monte Carlo')
plt.xlabel('Pasos Monte Carlo')
plt.ylabel('Valor integral')
plt.show()
```

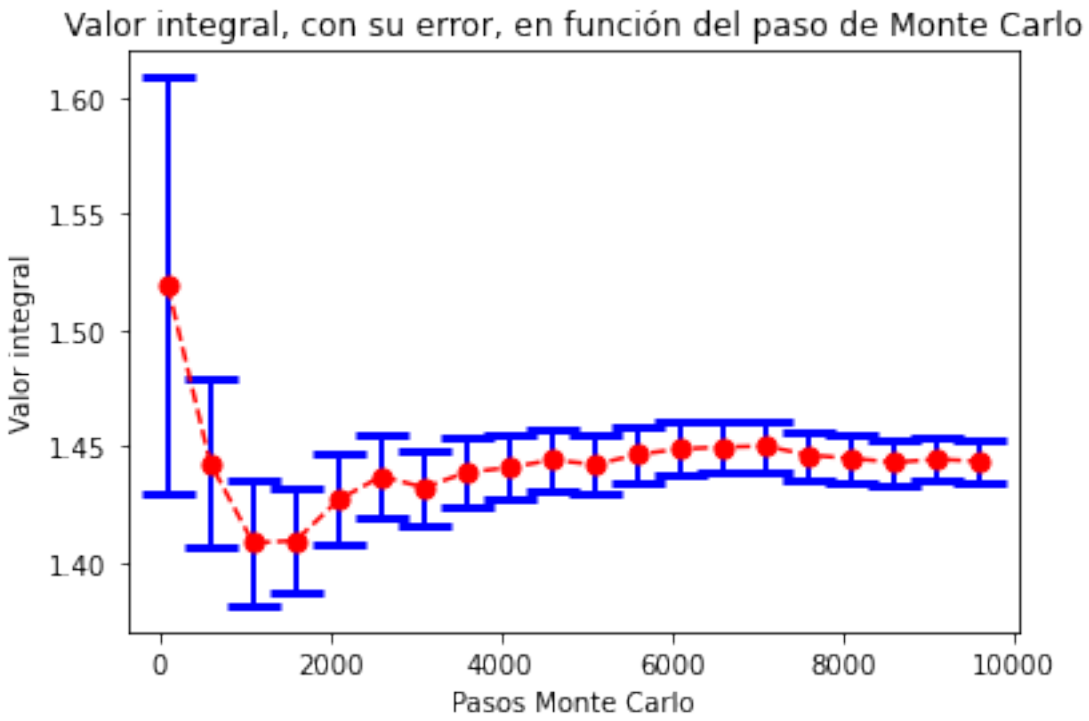
Se representan para los valores introducidos por el usuario: MC = 10000, Paso = 10.



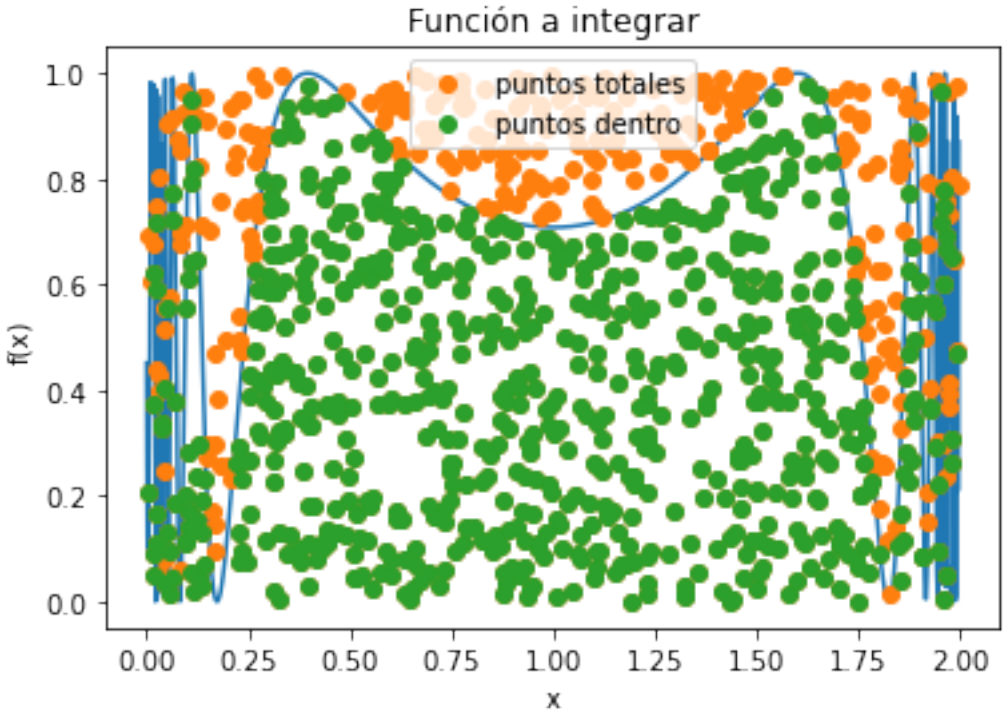


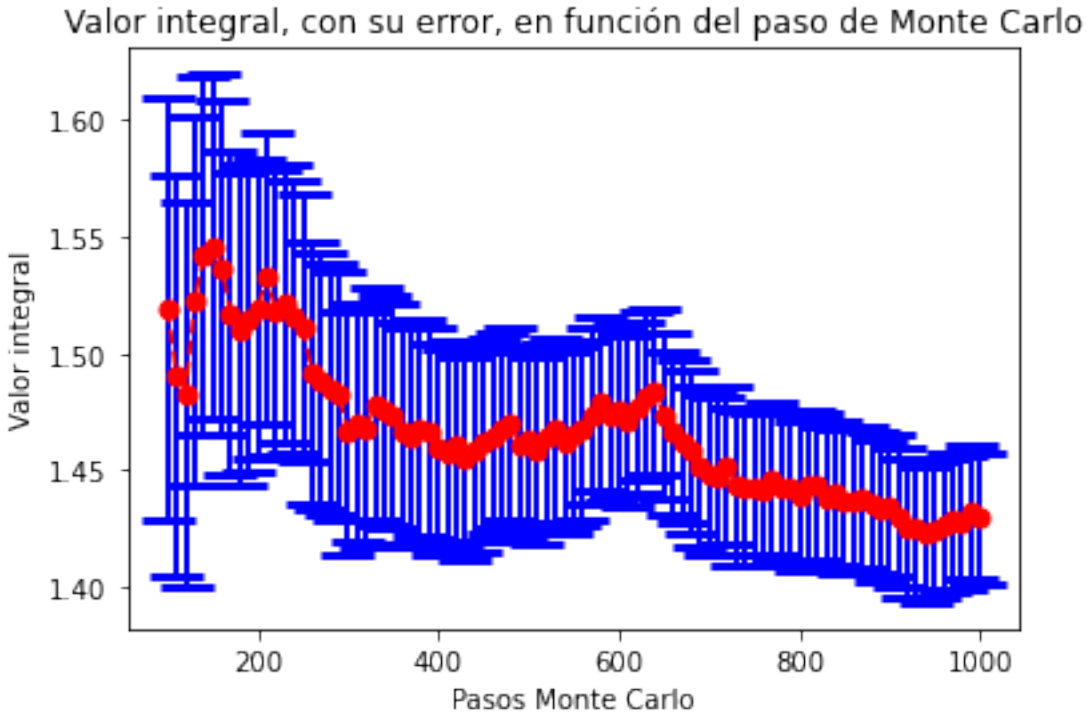
Se representan para los valores introducidos por el usuario: MC = 10000, Paso = 500.



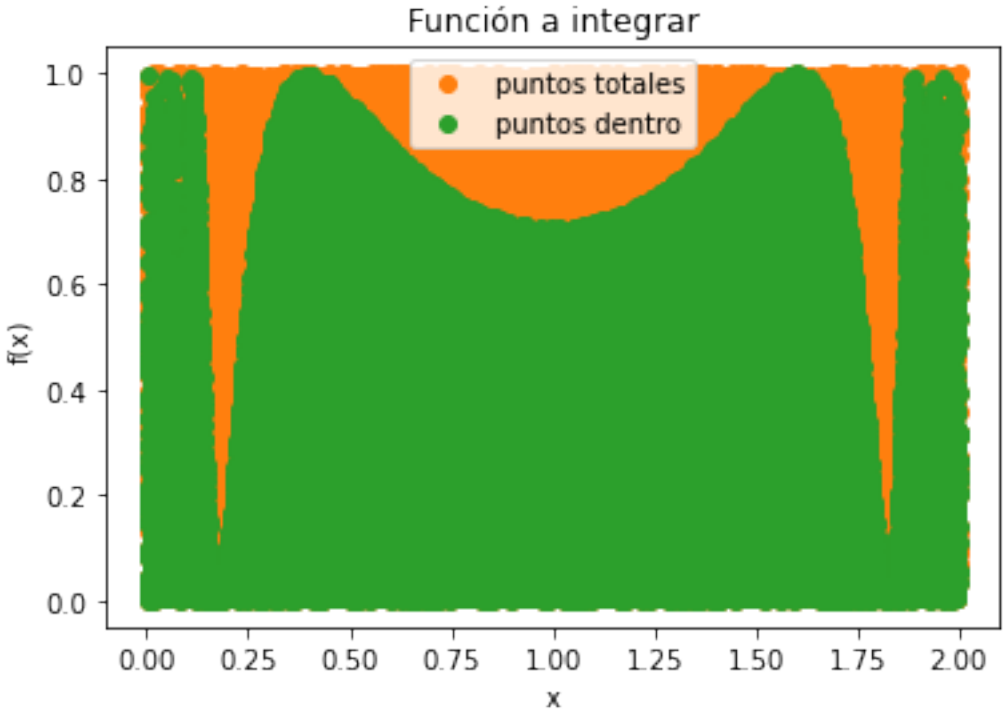


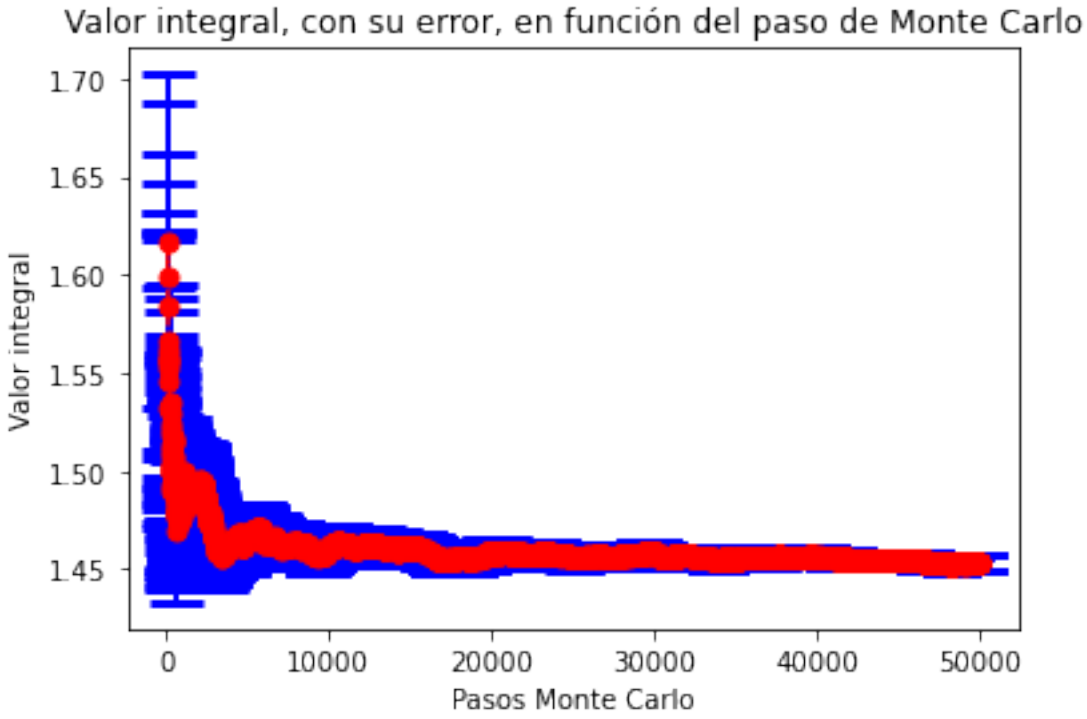
Se representan para los valores introducidos por el usuario: MC = 1000, Paso = 10.





Se representan para los valores introducidos por el usuario: MC = 50000, Paso = 10.





EXTRA 1

[3]:
```
import numpy as np

import matplotlib.pyplot as plt

# Definimos la función que queremos integrar
def f(x):
  return np.exp(-x**2) * np.sin(x) + 1 / (1 + x**2)
```

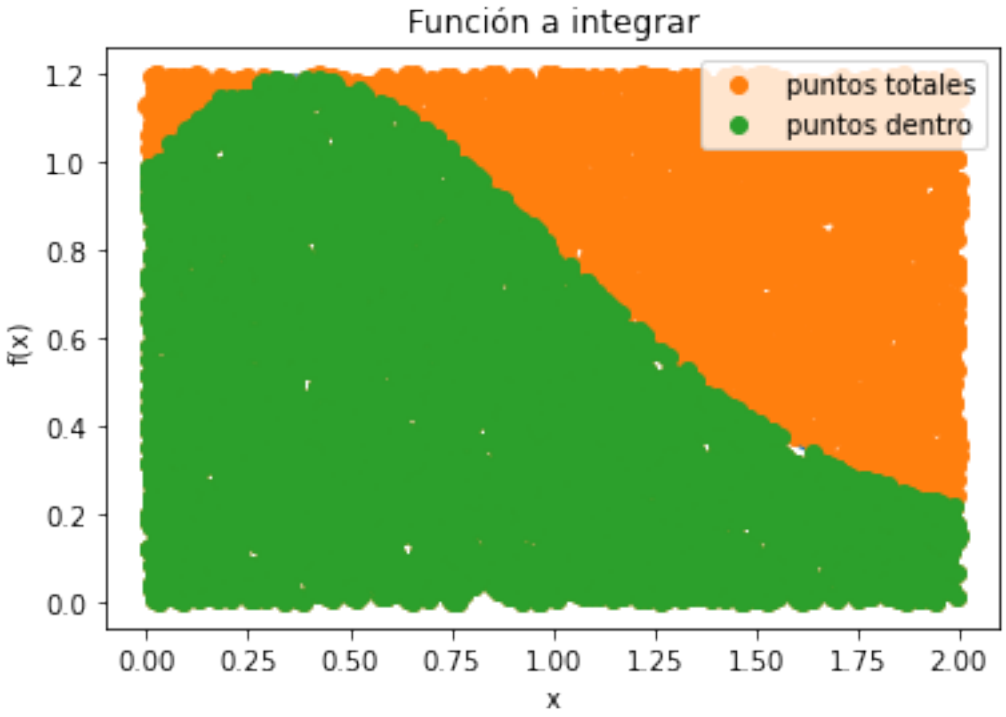
Ahora, lo único a tener en cuenta es que el plot se gráfica entre el a y el b introducido por el usuario. Es igual que en el caso anterior solo que aquí no se presenta problemas en el valor 0.

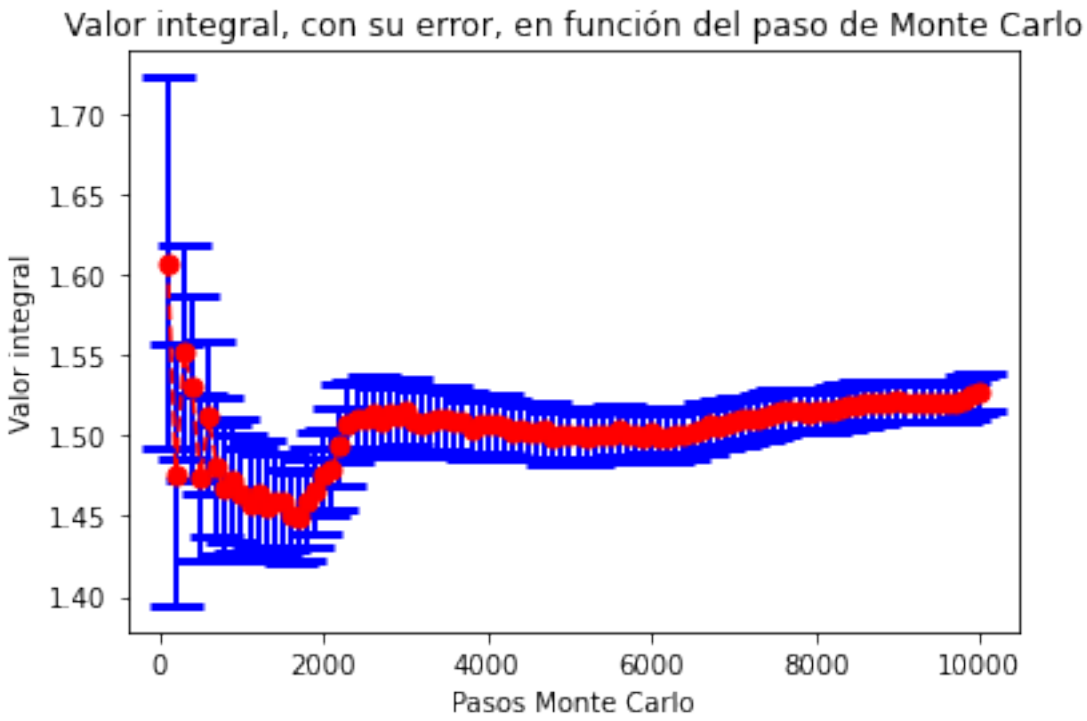
[4]:
```
# Representamos la función en el intervalo entre a y b
plt.figure(1)
plt.plot(np.linspace(a, b, 1000), f(np.linspace(a, b, 1000)))
plt.plot(x, y, 'o', label = 'puntos totales')
plt.plot(x[idx], y[idx], 'o', label = 'puntos dentro')
plt.xlabel('x')
plt.ylabel('f(x)')
plt.title('Función a integrar')
plt.legend()
plt.show()
```

```
# Representamos el valor de la integral y el error en función del número de␣
 ↪pasos Monte Carlo
plt.figure(2)
plt.errorbar(pasos, I, yerr = E, fmt="--ro", ms = 7, ecolor='b', elinewidth=2,␣
 ↪capsize=10, capthick=3)
plt.title('Valor integral, con su error, en función del paso de Monte Carlo')
plt.xlabel('Pasos Monte Carlo')
plt.ylabel('Valor integral')
plt.show()
```

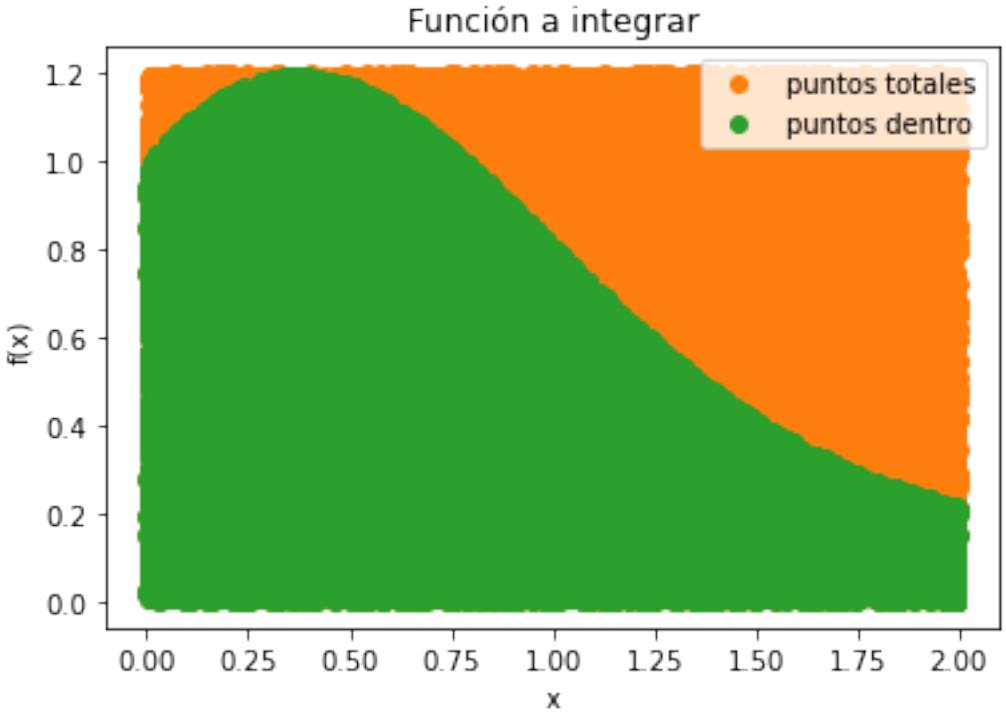
```
El valor de la integral para 10000 pasos Monte Carlo es I = 1.5274
Su error asociado es 0.0115
```

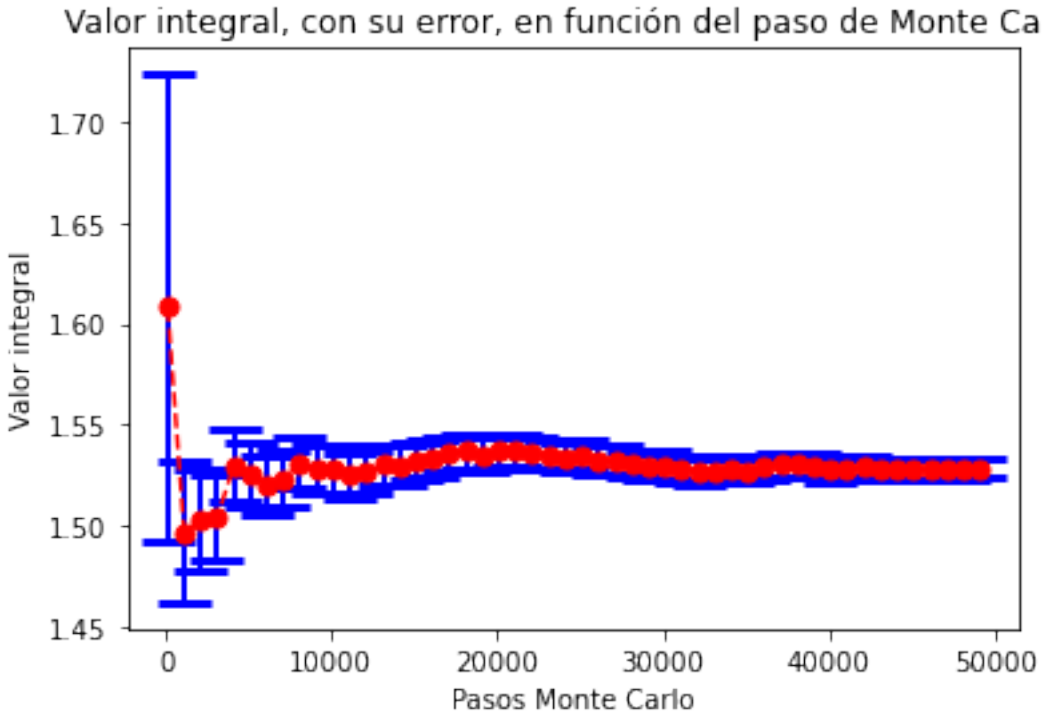
Se gráfica para los valores introducidos por terminal: MC = 10000, Paso = 100.



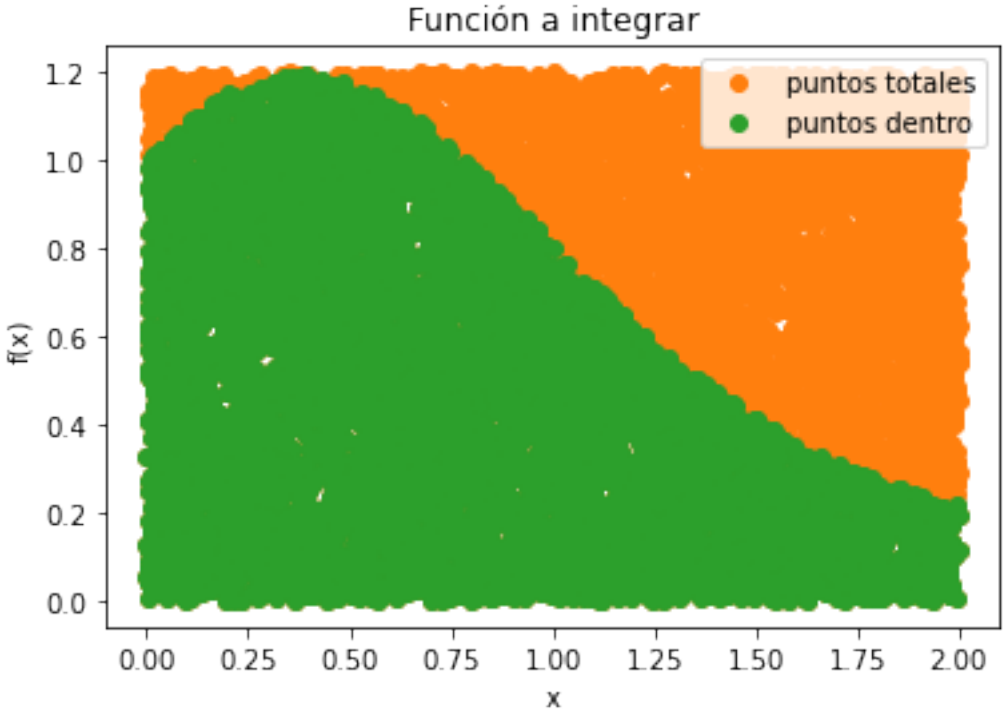


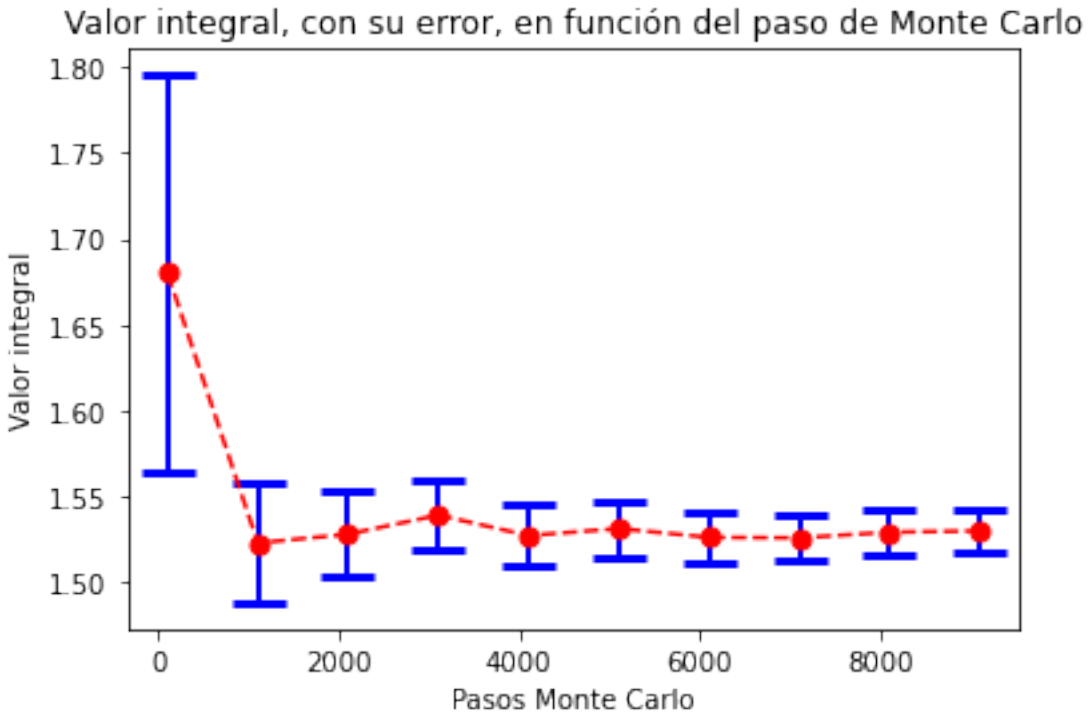
Se gráfica para los valores introducidos por terminal: MC = 50000, Paso = 1000.



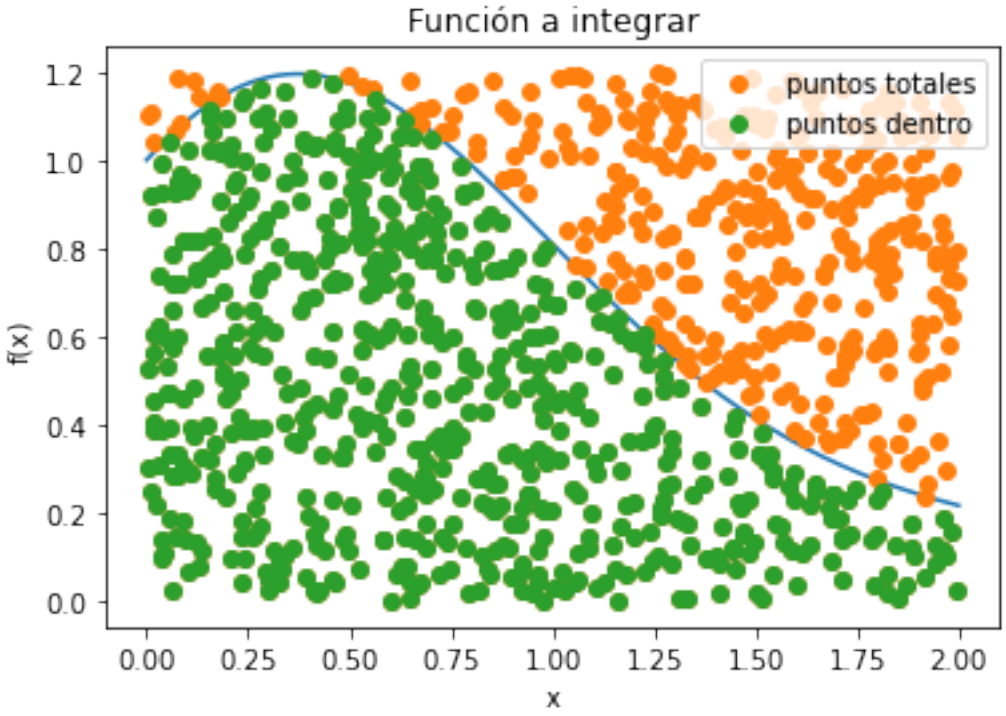


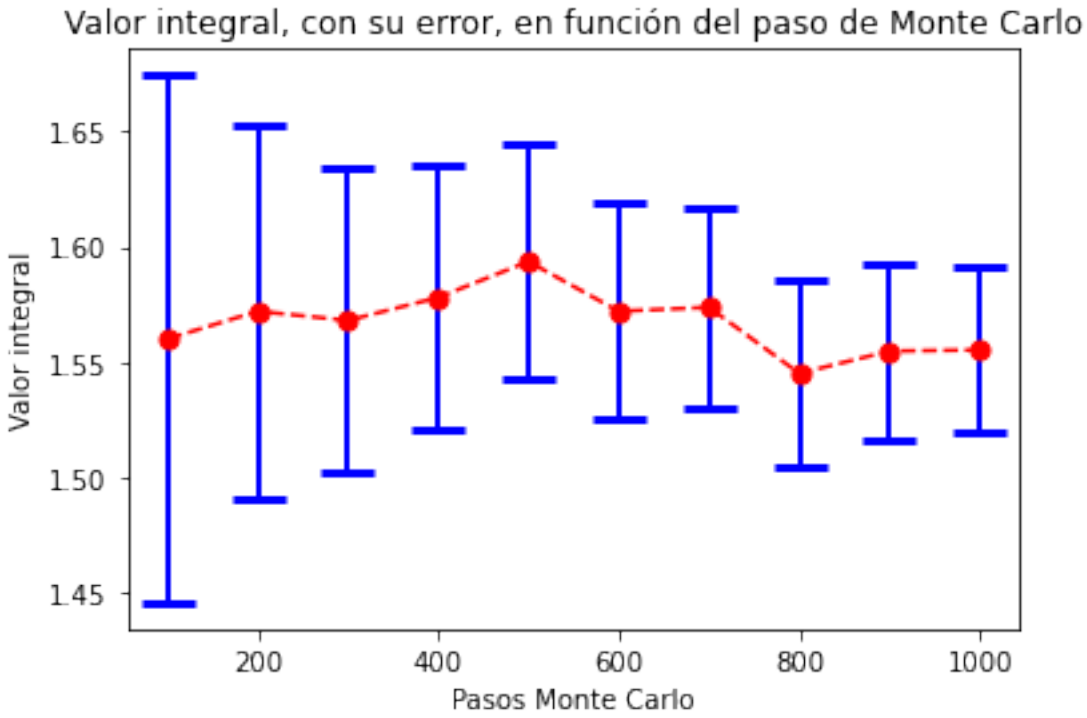
Se gráfica para los valores introducidos por terminal: MC = 10000, Paso = 1000.





Se gráfica para los valores introducidos por terminal: MC = 1000, Paso = 100.





EXTRA 2

Para este último extra, importamos una libreria especial de scipy.

[5]:
```
import numpy as np
```

[6]:
```
import scipy.special as sp
```

Definimos la función para el cálculo de la hiperesfera N-Dimensional. En nuestro caso se dará N = 10. Será necesario hacer uso de la función gamma.

[1]:
```
def V_exact(r, n):
  return (np.pi ** (n / 2) * r ** n) / sp.gamma(n / 2 + 1)
```

En este caso, estamos trabajando en un espacio topológico más abstracto que el euclídeo tridimensional pues ahora tenemos 10 dimensiones. Vamos a suponer que seguimos teniendo nuestro espacio dotado de la métrica euclídea. La dificultad de este caso será que no tenemos forma de imaginarnos la 10-esfera.

Se define una función para estimar el volumen, mediante Monte Carlo, de la hiperesfera de dimensión 10. En ella, se hallan puntos al azar dentro de un rango de 2r y para tantas dimensiones como queramos, n. Una vez hallados todos los puntos, se procede a calcular la distancia de cada uno de ellos al centro, si dicha norma (distancia) cae dentro del radio r = 1 de la hiperesfera pues no quedamos con él. La función devuelve el producto de la proporción de puntos dentro por el

volumen del hipercubo. El volumen de un hipercubo será la longitud de sus lados elevado a la dimensión (L^2). En este caso, la longitud del hipercubo que contendrá a la hiperesfera será 2r, por lo que su volumen será (2r)^n.

[2]:
```
def V_monte_carlo(r, n, N):
  points = np.random.uniform(-r, r, (N, n))

  distances = np.linalg.norm(points, axis=1)

  inside = np.count_nonzero(distances <= r)

  return (inside / N) * (2 * r) ** n
```

Por último, se definen los parámetros solicitados en el PDF de la práctica y se printean los resultados en la terminal.

[7]:
```
r = 1 # Radio de la hiperesfera
n = 10 # Dimensión de la hiperesfera
N = 100000 # Número de puntos aleatorios a generar

# Calcular el volumen exacto y el aproximado de la hiperesfera
V_e = V_exact(r, n)
V_a = V_monte_carlo(r, n, N)

# Mostrar los resultados
print("El volumen exacto de la hiperesfera es:", V_e)
print("El volumen aproximado de la hiperesfera es:", V_a)
```

```
El volumen exacto de la hiperesfera es: 2.550164039877345
El volumen aproximado de la hiperesfera es: 2.58048
```