Ejemplos del Método Monte Carlo Clásico

Ejemplo 1

Sea $\phi(x)$ la función de densidad $N(0,1)$. El cuantil al 95\% de esta distribución es $c_p = 1.65$, es decir, el valor c_p es tal que

$$\int_{-\infty}^{c_p} \phi(x)\,dx = 0.95.$$

Elabore un programa de cómputo que use el método clásico de Monte Carlo para corroborar este resultado. Considere que el intervalo sobre el que se lleva a cabo la integración es el intervalo acotado $(-4,\ 1.65)$.

Paso 1: Intervalo de integración

Se acota el dominio al intervalo

$$[a,b] = [-4,\ 1.65],$$

ya que la probabilidad de la normal estándar fuera de este rango es prácticamente nula.

Paso 2: Constante c

De acuerdo con el método de Monte Carlo clásico, necesitamos un valor $c > 0$ tal que

$$0 \le \phi(x) \le c, \quad \forall x \in [a,b].$$

Como la densidad normal estándar alcanza su máximo en $x = 0$, se tiene

$$c = \phi(0) = \frac{1}{\sqrt{2\pi}} \approx 0.399.$$

Paso 3: Muestreo

Generamos n pares (x_i, y_i) de manera independiente con:

$$x_i \sim U(a,b), \qquad y_i \sim U(0,c).$$

Paso 4: Contar los puntos bajo la curva

Definimos n_0 como el número de pares que satisfacen

$$y_i \le \phi(x_i).$$

Paso 5: Estimador Monte Carlo

Para n suficientemente grande (por ejemplo $n = 100{,}000$), la estimación se aproxima al valor exacto:

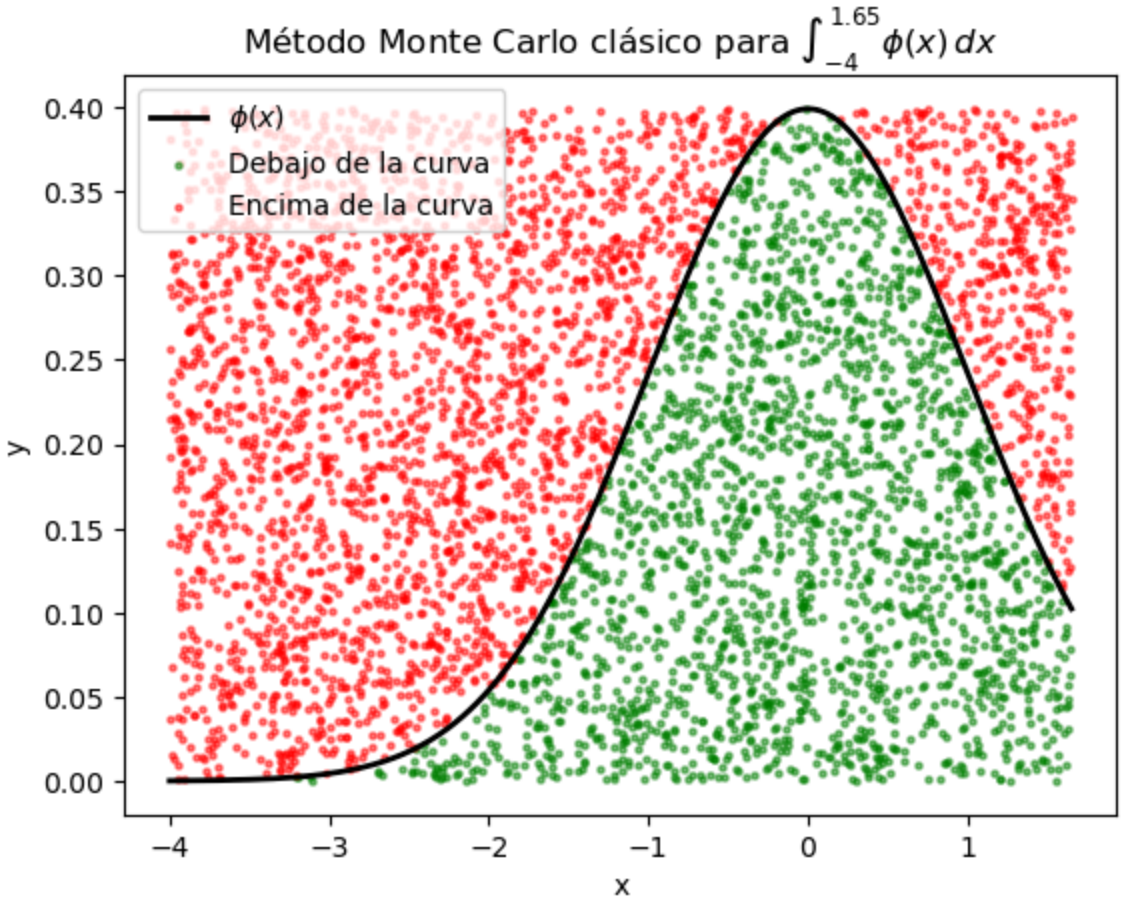
$$\int_{-\infty}^{1.65} \phi(x)\,dx \;\approx\; 0.95.$$

In [10]:
```python
import numpy as np

# Definimos la densidad normal estándar
def phi(x):
    return (1/np.sqrt(2*np.pi)) * np.exp(-x**2/2)

# Parámetros del intervalo
a, b = -4, 1.65
c = 1/np.sqrt(2*np.pi)   # valor máximo de la densidad normal en x=0
n = 100000               # número de simulaciones

# Generar muestras uniformes
x = np.random.uniform(a, b, n)
y = np.random.uniform(0, c, n)

# Contar cuántos puntos caen bajo la curva
n0 = np.sum(y <= phi(x))

# Estimación del área (integral aproximada)
integral = c * (b - a) * (n0 / n)

print(f"Estimación Monte Carlo = {integral:.5f}")
```

```
Estimación Monte Carlo = 0.95329
```

In [12]:
```python
import numpy as np
import matplotlib.pyplot as plt

# Definimos la densidad normal estándar
def phi(x):
    return (1/np.sqrt(2*np.pi)) * np.exp(-x**2/2)

# Parámetros
a, b = -4, 1.65
c = 1/np.sqrt(2*np.pi)
n = 5000   # menos puntos para que la gráfica sea clara

# Generar muestras
x = np.random.uniform(a, b, n)
y = np.random.uniform(0, c, n)

# Determinar cuáles puntos caen debajo de la curva
mask = y <= phi(x)

# Graficar la curva
xx = np.linspace(a, b, 500)
plt.plot(xx, phi(xx), 'k', linewidth=2, label=r'$\phi(x)$')

# Graficar puntos Monte Carlo
plt.scatter(x[mask], y[mask], color="green", s=5, alpha=0.5, label="Debajo de la curva
plt.scatter(x[~mask], y[~mask], color="red", s=5, alpha=0.5, label="Encima de la curva

# Ajustes de la gráfica
plt.title("Método Monte Carlo clásico para $\int_{-4}^{1.65} \phi(x)\,dx$")
plt.xlabel("x")
```

```
plt.ylabel("y")
plt.legend()
plt.show()
```



Ejemplo 2:

Elabore un programa de cómputo que utilice el método clásico de integración de Monte Carlo para calcular el volumen de una esfera de radio 1. Para corroborar su resultado, recuerde que el volumen de una esfera de radio r es $(4/3)\pi r^3$

Queremos estimar el volumen de la esfera unitaria

$$B = \{(x, y, z) \in \mathbb{R}^3 : x^2 + y^2 + z^2 \leq 1\}.$$

Método

1. Consideremos el cubo $C = [-1, 1]^3$, de volumen $V_C = (2)^3 = 8$.
2. Generamos n puntos (x_i, y_i, z_i) distribuidos uniformemente en C.
3. Definimos n_0 como el número de puntos que caen dentro de la esfera, es decir:

$$x_i^2 + y_i^2 + z_i^2 \leq 1.$$

4. El volumen estimado de la esfera es

$$V \approx V_C \cdot \frac{n_0}{n}.$$

Recordemos que el volumen de una esfera de radio r es

$$V(r) = \frac{4}{3}\pi r^3.$$

En el caso $r = 1$, el volumen exacto es

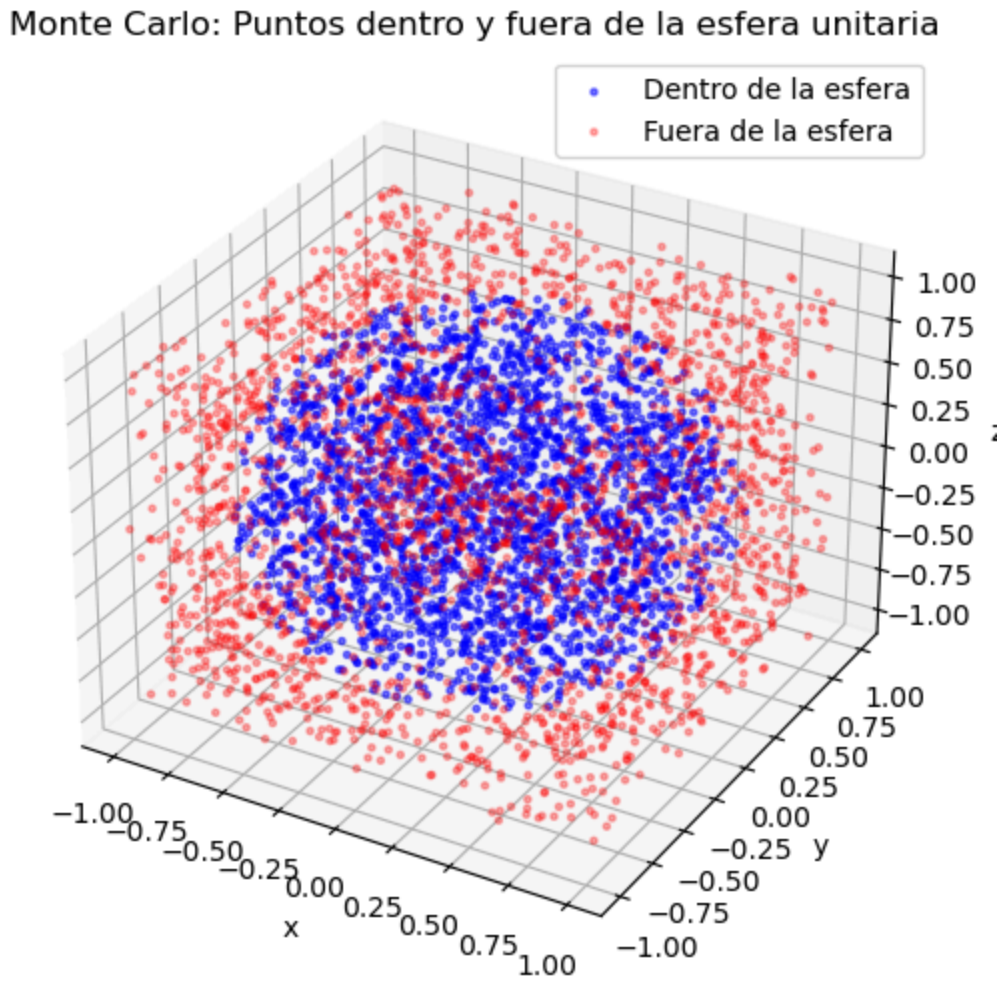
$$V(1) = \frac{4}{3}\pi \approx 4.18879.$$

In [13]:
```python
import numpy as np

# Parámetros
n = 1000000  # número de simulaciones
a, b = -1, 1
VC = (b - a)**3  # volumen del cubo = 8

# Generar puntos uniformes en el cubo
x = np.random.uniform(a, b, n)
y = np.random.uniform(a, b, n)
z = np.random.uniform(a, b, n)

# Contar los puntos dentro de la esfera unitaria
n0 = np.sum(x**2 + y**2 + z**2 <= 1)

# Estimación del volumen
V_estimado = VC * (n0 / n)

print(f"Volumen estimado = {V_estimado:.5f}")
print(f"Volumen exacto   = {4/3*np.pi:.5f}")
```

```
Volumen estimado = 4.19028
Volumen exacto   = 4.18879
```

In [14]:
```python
import numpy as np
import matplotlib.pyplot as plt

# Número de puntos (menor para visualizar mejor)
n = 5000
a, b = -1, 1

# Generar puntos uniformes en el cubo
x = np.random.uniform(a, b, n)
y = np.random.uniform(a, b, n)
z = np.random.uniform(a, b, n)

# Clasificar dentro y fuera de la esfera
inside = x**2 + y**2 + z**2 <= 1

# Crear figura 3D
fig = plt.figure(figsize=(8, 6))
ax = fig.add_subplot(111, projection='3d')

# Graficar puntos
ax.scatter(x[inside], y[inside], z[inside], color="blue", s=5, alpha=0.5, label="Dentr
ax.scatter(x[~inside], y[~inside], z[~inside], color="red", s=5, alpha=0.3, label="Fue

# Ajustes del gráfico
```

```
ax.set_xlabel("x")
ax.set_ylabel("y")
ax.set_zlabel("z")
ax.set_title("Monte Carlo: Puntos dentro y fuera de la esfera unitaria")
ax.legend()

plt.show()
```



In [ ]: