Método de Aceptación y Rechazo

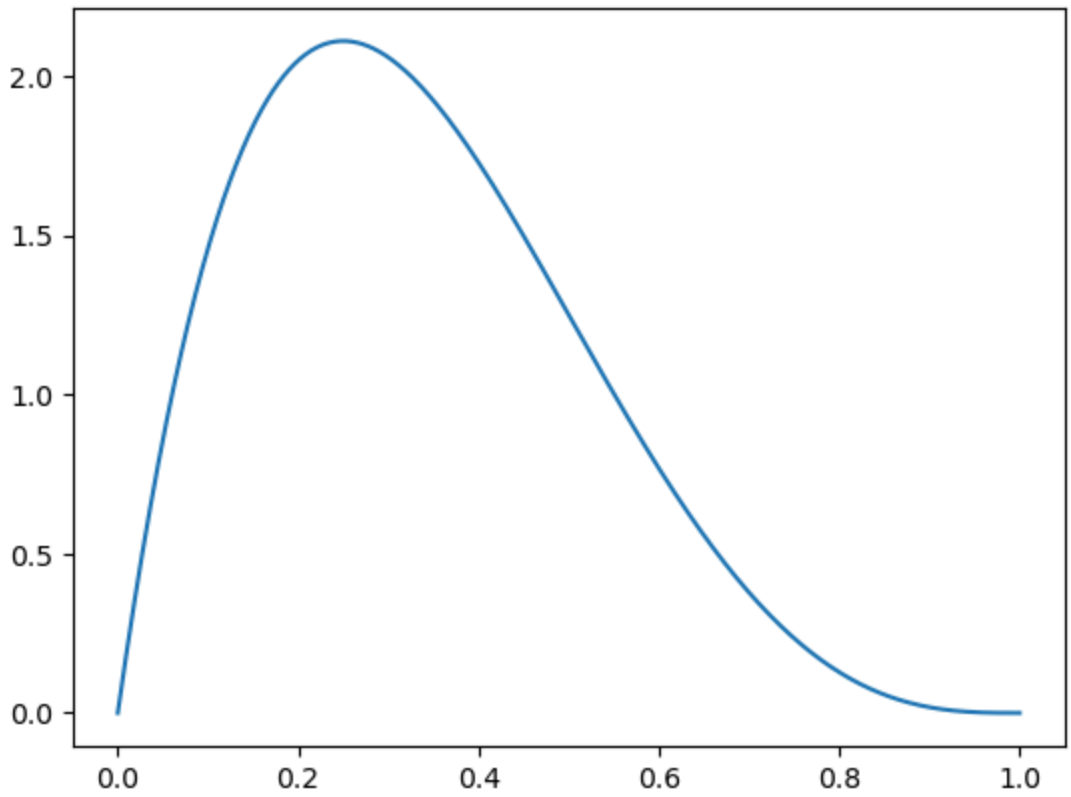
In [78]:
```
import numpy as np
import math
import matplotlib.pyplot as plt
```

In [79]:
```
# Asignar el valor
c = 135/64
c
```

Out[79]:
```
2.109375
```

In [80]:
```
def h(x):
    return 20*x*(1-x)**3
```

In [81]:
```
# función de densidad de la beta(2,4)
def f(x):
    return 20*x*(1-x)**3
#Creamos una partición del intervalo (0,1)
x_value = np.linspace(0,1,1000)
plt.plot(x_value,f(x_value))
plt.show()
```



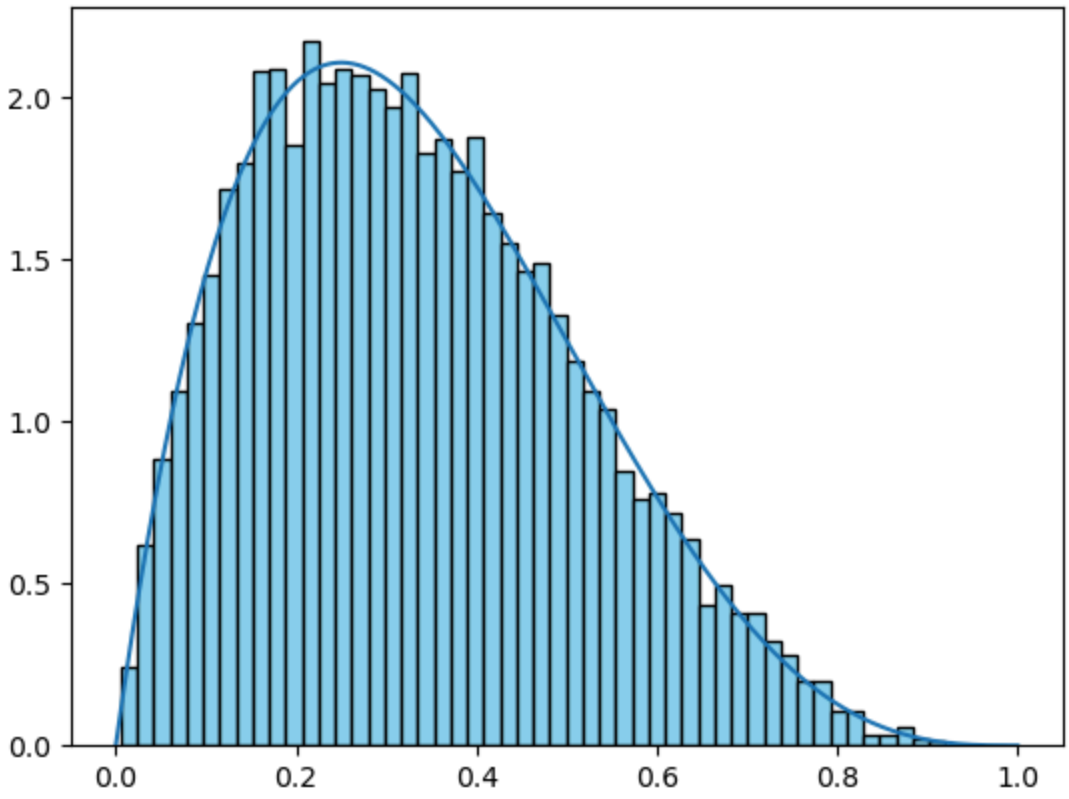
In [82]:
```
def rbeta(k):  #k numero de simulaciones
    sim = []
    cont = []
    for _ in range(k):
        y = np.random.random() #generamos una u(0,1)
```

```
        u = np.random.random() #generamos u(0,1)
        contador = 1
        while u>h(y)/c:
            y = np.random.random() #generamos una u(0,1)
            u = np.random.random() #generamos u(0,1)
            contador +=  1
        x = y
        sim.append(x)
        cont.append(contador)
    return np.array(sim),np.array(cont)
```

In [83]:
```
x,contador = rbeta(1000000)
print(contador.mean())
print(135/64)
```

```
2.109378
2.109375
```

In [84]:
```
x,contador = rbeta(10000)
x_value = np.linspace(0,1,1000)
plt.plot(x_value,f(x_value))
plt.hist(x,density = True, bins = 50, edgecolor="black", color = "skyblue")
plt.show()
```



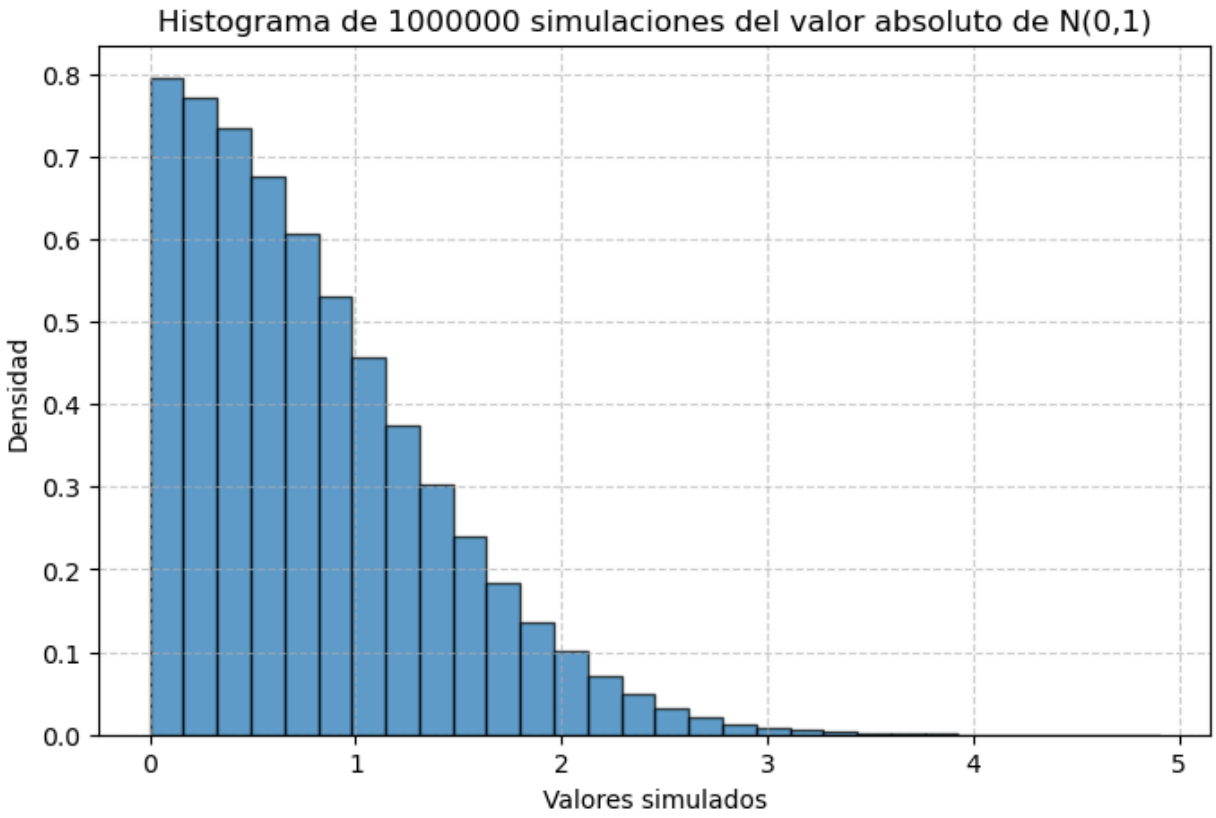
Simulación de la distribución normal

1. Simulación del $|Z|$ valor absoluto de una distribución normal estándar

In [85]:
```python
#Paso 1
def rnorm1(k):
    sim = []
    cont = []
    for _ in range(k):
        u1 = np.random.random()
        y = -np.log(u1)
        u2 = np.random.random()
        contador = 1
        while u2 > np.exp(-0.5*(y-1)**2):
            u1 = np.random.random()
            y = -np.log(u1)
            u2 = np.random.random()
            contador += 1
        x = y
        sim.append(x)
        cont.append(contador)
    return np.array(sim), np.array(cont)
```

In [86]:
```python
#Simulaciones de |Z|
k = 1000000
simulaciones, contadores = rnorm1(k)

# Cálculos
contadores_real = math.sqrt(2 * math.e / math.pi)
contadores_mean = np.mean(contadores)

print("Contadores (real):", contadores_real)
print("Contadores (mean de simulaciones):", contadores_mean)
# Histograma mejorado
plt.figure(figsize=(8,5))
plt.hist(simulaciones, bins=30, density=True, edgecolor="black", alpha=0.7)
plt.title(f"Histograma de {k} simulaciones del valor absoluto de N(0,1)")
plt.xlabel("Valores simulados")
plt.ylabel("Densidad")
plt.grid(True, linestyle="--", alpha=0.6)
plt.show()
```

```
Contadores (real): 1.315489246958914
Contadores (mean de simulaciones): 1.317149
```



Simulación de una variable aleatoria normal estándar

Algoritmo para simular una normal estándar Z apartir de $|Z|$:

1. Genera $U \sim \text{Unif}(0,1)$ de forma pseudoaleatoria.
2. Generar $Y = |Z|$
3. Asigna el signo:

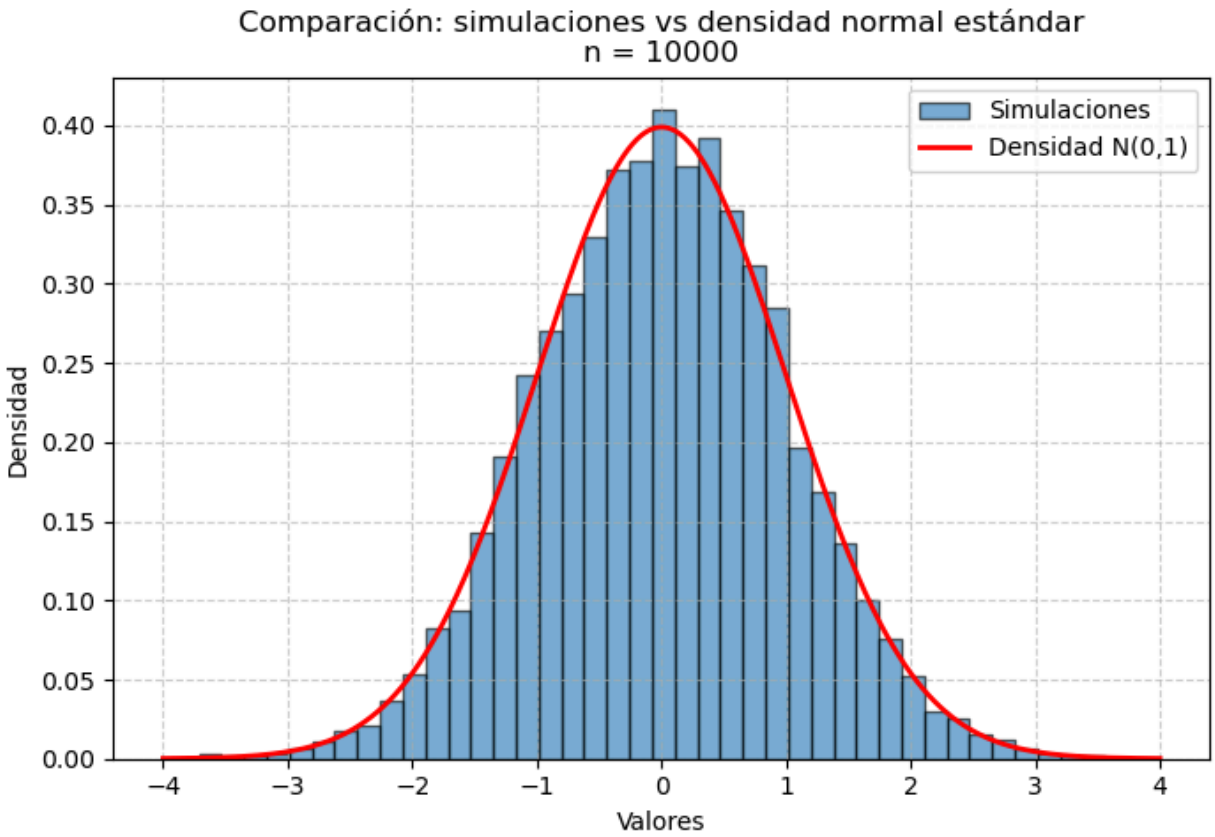
$$Z = \begin{cases} Y, & \text{si } U < \frac{1}{2}, \\ -Y, & \text{si } U \geq \frac{1}{2}. \end{cases}$$

In [87]:
```
# Nueva función: simula Z ~ N(0,1) a partir de rnorm1
def rnorm_stand(k):
    # Paso 1: simular |Z|
    absZ, contadores = rnorm1(k)
    # Paso 2: generar signos aleatorios ±1 con igual probabilidad
    signos = np.where(np.random.rand(k) < 0.5, 1, -1)
    # Paso 3: asignar signo
    Z = absZ * signos
    return Z, contadores
```

In [88]:
```
# Ejemplo
k = 10000
simulaciones, contadores = rnorm_stand(k)

#estadisticas
print("Número de simulaciones:", len(simulaciones))
```

```
print("Media aproximada:", np.mean(simulaciones))
print("Varianza aproximada:", np.var(simulaciones))

#contadores
print("Contadores (real):", contadores_real)
print("Contadores (mean de simulaciones):", contadores_mean)

# Histograma
plt.figure(figsize=(8,5))
plt.hist(simulaciones, bins=40, density=True, alpha=0.6, edgecolor="black", label="Sim

# Densidad teórica de la normal estándar
x = np.linspace(-4, 4, 500)
f = (1/np.sqrt(2*math.pi)) * np.exp(-x**2/2)
plt.plot(x, f, "r-", linewidth=2, label="Densidad N(0,1)")

# Etiquetas y leyenda
plt.title(f"Comparación: simulaciones vs densidad normal estándar\nn = {k}")
plt.xlabel("Valores")
plt.ylabel("Densidad")
plt.grid(True, linestyle="--", alpha=0.6)
plt.legend()
plt.show()
```

```
Número de simulaciones: 10000
Media aproximada: 0.00450161796748792
Varianza aproximada: 1.0015959229269338
Contadores (real): 1.315489246958914
Contadores (mean de simulaciones): 1.317149
```

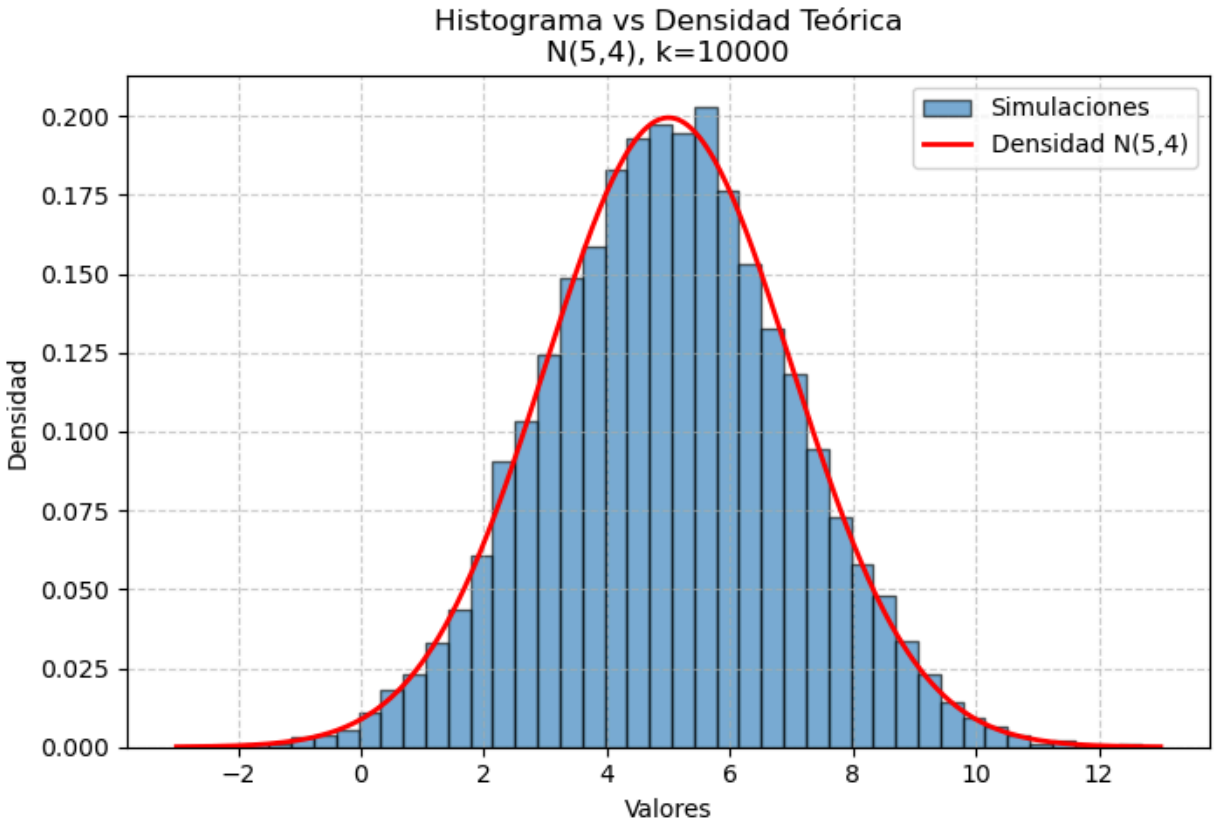


Simulación de una distribución normal X

In [89]:
```python
def rnorm(mu,std, k):
    z, contadores = rnorm_stand(k)
    x = z*std + mu
    return x, contadores
```

In [90]:
```python
# Ejemplo:
mu, sigma, k = 5, 2, 10000
simulaciones, contadores = rnorm(mu, sigma, k)

print("Número de simulaciones:", contadores)
print("Media aproximada:", np.mean(simulaciones))
print("Varianza aproximada:", np.var(simulaciones))


#contadores
print("Contadores (real):", contadores_real)
print("Contadores (mean de simulaciones):", contadores_mean)

# Histograma normalizado
plt.figure(figsize=(8,5))
plt.hist(simulaciones, bins=40, density=True, alpha=0.6, edgecolor="black", label="Sim

# Densidad teórica N(mu, sigma^2)
x = np.linspace(mu - 4*sigma, mu + 4*sigma, 500)
f = (1/(sigma*np.sqrt(2*math.pi))) * np.exp(-((x-mu)**2)/(2*sigma**2))
plt.plot(x, f, "r-", linewidth=2, label=f"Densidad N({mu},{sigma**2})")

# Etiquetas
plt.title(f"Histograma vs Densidad Teórica\nN({mu},{sigma**2}), k={k}")
plt.xlabel("Valores")
plt.ylabel("Densidad")
plt.grid(True, linestyle="--", alpha=0.6)
plt.legend()
plt.show()
```

```
Número de simulaciones: [1 2 1 ... 2 1 1]
Media aproximada: 5.00773033226667
Varianza aproximada: 4.0423921664991935
Contadores (real): 1.315489246958914
Contadores (mean de simulaciones): 1.317149
```


Histograma vs Densidad Teórica
N(5,4), k=10000
Simulaciones
Densidad N(5,4)
Densidad
Valores
0.000
0.025
0.050
0.075
0.100
0.125
0.150
0.175
0.200
−2
0
2
4
6
8
10
12