$$E(\Theta_1) = c(b-a)\,E(N_H/N)$$

$$= c(b-a)\,E(N_H)/N = pc(b-a) = I$$

$\Rightarrow \Theta_1$ es un estimador insesgado de $I$

$$\mathrm{var}(\Theta_1) = [c(b-a)]^2\,\mathrm{var}(\hat{p})$$

$$= [c(b-a)]^2 \frac{I}{N} p(1-p)$$

$$= \frac{I}{N}[c(b-a) - I]$$

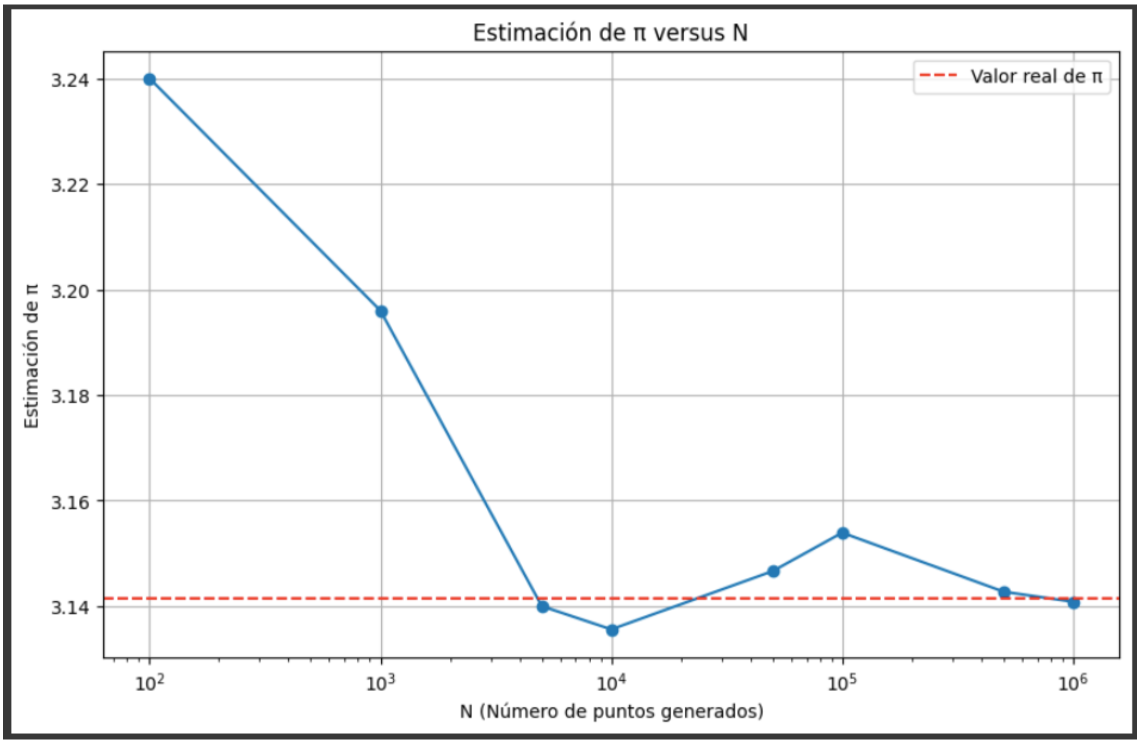
$$\sigma_{\Theta_1} = \sqrt{\mathrm{var}(\Theta_1)} = N^{-1/2}\{I[c(b-a) - I]\}^{1/2}$$

La fórmula para hallar $N$: $N \geq \frac{(1-p)\,p\,[c(b-a)]^2}{(1-\alpha)\varepsilon^2}$

```
import random

def hit_or_miss_monte_carlo(g, a, b, c, d, N):
    count_hits = 0
    for _ in range(N):
        x = random.uniform(a, b)
        y = random.uniform(c, d)
        if 0 <= y <= g(x):
            count_hits += 1
    area_rectangle = (b - a) * (d - c)
    area_region_under_curve = (count_hits / N) * area_rectangle
    return area_region_under_curve

# Solicitar al usuario que ingrese la función g(x)
def get_g_function():
    g_expression = input("Ingrese la función g(x) como una expresión
algebraica (por ejemplo, x**2): ")
    return lambda x: eval(g_expression)

def main():
    # Ingresar los límites de la región rectangular
    a = float(input("Ingrese el límite inferior del intervalo en x:
"))
    b = float(input("Ingrese el límite superior del intervalo en x:
"))
    c = float(input("Ingrese el límite inferior del intervalo en y:
"))
    d = float(input("Ingrese el límite superior del intervalo en y:
"))
    N = int(input("Ingrese el número de puntos generados: "))

    # Obtener la función g(x)
    g = get_g_function()

    # Calcular el área estimada bajo la curva
    estimated_area_under_curve = hit_or_miss_monte_carlo(g, a, b, c,
d, N)
```

```python
    print("Estimación del área bajo la curva utilizando el Método Monte Carlo de Acierto y Error:", estimated_area_under_curve)

if __name__ == "__main__":
    main()
```

```python
import random
import matplotlib.pyplot as plt

def estimate_pi(N):
    inside_circle = 0
    for _ in range(N):
        x = random.uniform(-1, 1)
        y = random.uniform(-1, 1)
        if x**2 + y**2 <= 1:
            inside_circle += 1
    return 4 * inside_circle / N

# Realizar múltiples estimaciones para diferentes valores de N
N_values = [100, 1000, 5000, 10000, 50000, 100000, 500000, 1000000]
pi_estimates = [estimate_pi(N) for N in N_values]

# Graficar el estimador de pi versus N
plt.figure(figsize=(10, 6))
plt.plot(N_values, pi_estimates, marker='o')
plt.axhline(y=3.141592653589793, color='r', linestyle='--', label='Valor real de π')
plt.xscale('log')
plt.xlabel('N (Número de puntos generados)')
plt.ylabel('Estimación de π')
plt.title('Estimación de π versus N')
plt.legend()
plt.grid(True)
plt.show()
```



```
Ingrese el límite inferior del intervalo en x: 0
Ingrese el límite superior del intervalo en x: 1
Ingrese el límite inferior del intervalo en y: 2.71828
Ingrese el límite superior del intervalo en y: 15.15426
Ingrese el número de puntos generados: 10000
Ingrese la función g(x) como una expresión algebraica (por ejemplo, x**2): 2.71828**(2.71828**x)
Estimación del área bajo la curva utilizando el Método Monte Carlo de Acierto y Error: 3.59399822
```

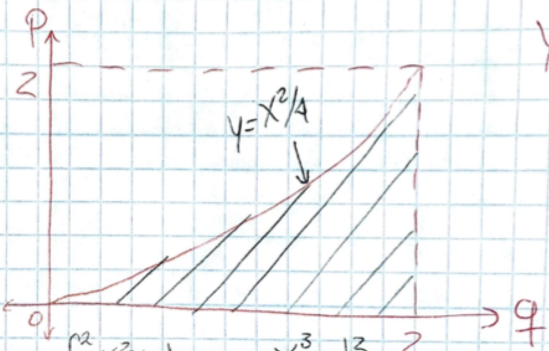
```
Ingrese el límite inferior del intervalo en x: 0
Ingrese el límite superior del intervalo en x: 1
Ingrese el límite inferior del intervalo en y: 0
Ingrese el límite superior del intervalo en y: 1
Ingrese el número de puntos generados: 1000
Ingrese la función g(x) como una expresión algebraica (por ejemplo, x**2): (1-x**2)**1.5
Estimación del área bajo la curva utilizando el Método Monte Carlo de Acierto y Error: 0.593
```

```python
import random

def real_roots_probability(num_simulations):
    real_roots_count = 0

    for _ in range(num_simulations):
        p = random.uniform(0, 2)
        q = random.uniform(0, 2)

        discriminant = p**2 - 4*q

        if discriminant >= 0:
            real_roots_count += 1

    return real_roots_count / num_simulations

num_simulations = 1000000
probability = real_roots_probability(num_simulations)
print("Probabilidad de que las raíces sean números reales:", probability)
```

```
Probabilidad de que las raíces sean números reales: 0.166959
```

Teóricamente calculamos el discriminante $I = p^2 - 4q$, y este debe ser mayor o igual a 0, lo que implica que $q \leq \frac{p^2}{4}$.

$$y \leq \frac{x^2}{4}$$




$$\Rightarrow p = \frac{\text{Área Sombreada}}{\text{Área Total}} = \frac{\int_0^2 x^2/4 \, dx}{4} = \frac{x^3/12 \Big|_0^2}{4} = \frac{8/12}{4} = \frac{2}{12} = \frac{1}{6}$$

Como vemos, la simulación es muy próxima al valor teórico.