Codice Python Es15

```
import matplotlib.pyplot as plt
import numpy as np

# Parametri generali
epsilon = 0.01
confidence = 0.95
z_alpha = 1.96

# Funzione per stimatore Monte Carlo e deviazione standard
def monte_carlo_integration(f, domain_sampler, domain_volume, N):
    samples = domain_sampler(N)
    values = f(samples)
    estimate = domain_volume * np.mean(values)
    std_dev = domain_volume * np.std(values, ddof=1) / np.sqrt(N)
    return estimate, std_dev

# Funzione per determinare Nmax dato uno stimatore e campionatore
def estimate_nmax(f, domain_sampler, domain_volume, initial_N=100000):
    samples = domain_sampler(initial_N)
    values = f(samples)
    sigma_hat = domain_volume * np.std(values, ddof=1)
    Nmax = int(np.ceil((z_alpha * sigma_hat / epsilon)**2))
    return Nmax, sigma_hat

# Caso (a): Stima di π usando l'area del cerchio
def f_pi(samples):
    x, y = samples[:, 0], samples[:, 1]
    return (x**2 + y**2) <= 1

def sampler_pi(N):
    return np.random.uniform(-1, 1, size=(N, 2))

V_pi = 4  # area del quadrato [-1,1]^2
```

```
# Caso (b): Volume della sfera tridimensionale
def f_sfera(samples):
    x, y, z = samples[:, 0], samples[:, 1], samples[:, 2]
    return (x**2 + y**2 + z**2) <= 1

def sampler_sfera(N):
    return np.random.uniform(-1, 1, size=(N, 3))

V_sfera = 8  # volume del cubo [-1,1]^3

# Caso (c): Volume corpo A
def f_corpo(samples):
    x, y, z = samples[:, 0], samples[:, 1], samples[:, 2]
    r = np.sqrt(x**2 + y**2)
    return r <= 1 + np.sin(np.pi * z)

def sampler_corpo(N):
    return np.random.uniform(-2, 2, size=(N, 3))

V_corpo = 64  # volume del cubo [-2,2]^3

# Funzione per generare grafico dell'andamento delle stime
def plot_convergence(f, sampler, V, Nmax, true_value, title, filename):
    Ns = np.logspace(3, np.log10(Nmax), num=50, dtype=int)
    estimates = []
    errors = []

    for N in Ns:
        est, std = monte_carlo_integration(f, sampler, V, N)
        estimates.append(est)
        errors.append(std)

    estimates = np.array(estimates)
    errors = np.array(errors)
    plt.figure(figsize=(10, 6))
    plt.plot(Ns, estimates, label='Stima Monte Carlo')
    plt.fill_between(Ns, estimates - z_alpha * errors, estimates + z_alpha * er
rors, color='gray', alpha=0.2, label='Intervallo di confidenza 95%')
```

```
    plt.axhline(y=true_value, color='red', linestyle='--', label='Valore vero')
    plt.xscale('log')
    plt.xlabel('Numero di campioni (N)')
    plt.ylabel('Stima')
    plt.title(title)
    plt.legend()
    plt.grid(True)
    plt.savefig(filename, format="pdf")
    plt.close()

# Calcolo Nmax per ciascun caso
Nmax_pi, _ = estimate_nmax(f_pi, sampler_pi, V_pi)
Nmax_sfera, _ = estimate_nmax(f_sfera, sampler_sfera, V_sfera)
Nmax_corpo, _ = estimate_nmax(f_corpo, sampler_corpo, V_corpo)

# Generazione dei grafici
plot_convergence(f_pi, sampler_pi, V_pi, Nmax_pi, np.pi, "Stima di π", "stim
a_pi.pdf")
plot_convergence(f_sfera, sampler_sfera, V_sfera, Nmax_sfera, (4/3)*np.pi,
"Volume della sfera unità", "volume_sfera.pdf")
# Per il caso (c), il valore vero non è noto; utilizziamo la stima con Nmax co
me riferimento
est_corpo, _ = monte_carlo_integration(f_corpo, sampler_corpo, V_corpo, N
max_corpo)
plot_convergence(f_corpo, sampler_corpo, V_corpo, Nmax_corpo, est_corp
o, "Volume del corpo A", "volume_corpo.pdf")
```