```python
import numpy as np
import matplotlib.pyplot as plt
from scipy.linalg import cholesky
from scipy.stats import norm


def generate_gp_realizations(m, L, alpha_mean_func=None):
    """
    Generate m realizations of a Gaussian process

    Parameters:
    m: number of realizations
    L: length scale parameter
    alpha_mean_func: function to compute mean at grid points
    """

    x = np.array([-1 + (i-1) * 0.02 for i in range(1, 102)])  # 101 points
    n = len(x)

    if alpha_mean_func is None:
        alpha_true = np.cos(np.pi * x)
    else:
        alpha_true = alpha_mean_func(x)

    # Build covariance matrix
    C = np.zeros((n, n))
    for i in range(n):
        for j in range(n):
            C[i, j] = 1/(2*L) * np.exp(-np.abs(x[i] - x[j]) / L)

    # small diagonal for numerical stability?
    C += 1e-12 * np.eye(n)

    L_chol = cholesky(C, lower=True)

    realizations = np.zeros((m, n))
    for k in range(m):
        z = np.random.randn(n)
        # Transform to have covariance C
        realization = alpha_true + L_chol @ z
        realizations[k, :] = realization

    return x, alpha_true, C, realizations

def plot_5_realizations(L=1):
    """Generate and plot 5 realizations"""
    x, alpha_true, C, realizations = generate_gp_realizations(m=5, L=L)

    plt.figure(figsize=(12, 8))

    # Plot each realization
    for i in range(5):
        plt.plot(x, realizations[i, :], alpha=0.7, linewidth=1.5,
                 label=f'Realization {i+1}')

    # Plot true mean
    plt.plot(x, alpha_true, 'k-', linewidth=3, label='True mean $\\alpha(x) = \cos(\pi x)$')

    # Plot theoretical 99.7% interval (B±3 std dev)
    std_dev = np.sqrt(np.diag(C))  # sqrt of diagonal elements
    plt.fill_between(x, alpha_true - 3*std_dev, alpha_true + 3*std_dev,
                     alpha=0.2, color='gray', label='Theoretical 99.7% interval')

    plt.xlabel('$x$', fontsize=14)
    plt.ylabel('$\\alpha(x)$', fontsize=14)
    plt.title(f'5 Realizations of Gaussian Process with $L={L}$', fontsize=16)
    plt.grid(True, alpha=0.3)
    plt.legend(loc='upper right', fontsize=10)
    plt.xlim(-1, 1)
    plt.tight_layout()
    plt.show()

    return x, alpha_true, C, realizations

def plot_500_realizations(L=1):
    """Generate 500 realizations and compute empirical statistics"""
    x, alpha_true, C, realizations = generate_gp_realizations(m=500, L=L)

    # Compute empirical statistics
    empirical_mean = np.mean(realizations, axis=0)
    empirical_std = np.std(realizations, axis=0)
    assert empirical_std.shape == (101,)
    assert empirical_std[1]!=empirical_std[2]
    # Theoretical values
    theoretical_std = np.sqrt(np.diag(C))  # sqrt(C(x_i, x_i))

    plt.figure(figsize=(14, 10))

    # Plot true mean
    plt.plot(x, alpha_true, 'k-', linewidth=3,
             label='True mean $\\alpha(x) = \cos(\pi x)$')

    # Plot theoretical 99.7% interval
    plt.plot(x, alpha_true - 3*theoretical_std, 'b--', linewidth=2,
             alpha=0.7, label='Theoretical: $\\alpha(x) \\pm 3\\sqrt{C(x,x)}$')
    plt.plot(x, alpha_true + 3*theoretical_std, 'b--', linewidth=2, alpha=0.7)

    # Plot empirical mean
    plt.plot(x, empirical_mean, 'r-', linewidth=2,
             label='Empirical mean $\\mu_x$')

    # Plot empirical 99.7% interval
    plt.plot(x, empirical_mean - 3*empirical_std, 'g--', linewidth=2,
             alpha=0.7, label='Empirical: $\\mu_x \\pm 3\\sigma_x$')
    plt.plot(x, empirical_mean + 3*empirical_std, 'g--', linewidth=2, alpha=0.7)

    # Fill between for better visualization
    plt.fill_between(x, alpha_true - 3*theoretical_std, alpha_true + 3*theoretical_std,
                     alpha=0.15, color='blue', label='Theoretical interval area')
    plt.fill_between(x, empirical_mean - 3*empirical_std, empirical_mean + 3*empirical_std,
                     alpha=0.1, color='green', label='Empirical interval area')

    plt.xlabel('$x$', fontsize=14)
    plt.ylabel('$\\alpha(x)$', fontsize=14)
    plt.title(f'Empirical vs Theoretical Statistics (L={L}, m=500 realizations)', fontsize=16)
    plt.grid(True, alpha=0.3)
    plt.legend(loc='upper right', fontsize=10)
    plt.xlim(-1, 1)
    plt.ylim(-3, 3)
    plt.tight_layout()
    plt.show()

    # Compute and print quantitative comparisons
```

```
    print("="*60)
    print(f"QUANTITATIVE ANALYSIS (L={L}, m=500 realizations):")
    print("="*60)

    # Mean squared error between true and empirical mean
    mse_mean = np.mean((alpha_true - empirical_mean)**2)
    print(f"Mean Squared Error (True vs Empirical mean): {mse_mean:.6f}")

    # Average absolute difference in standard deviations
    avg_std_diff = np.mean(np.abs(theoretical_std - empirical_std))
    print(f"Average absolute difference in std dev: {avg_std_diff:.6f}")

    # Correlation between theoretical and empirical intervals width
    theoretical_interval_width = 6 * theoretical_std
    empirical_interval_width = 6 * empirical_std
    correlation = np.corrcoef(theoretical_interval_width, empirical_interval_width)[0, 1]
    print(f"Correlation between interval widths: {correlation:.6f}")

    # Maximum pointwise error in mean
    max_abs_error_mean = np.max(np.abs(alpha_true - empirical_mean))
    print(f"Maximum absolute error in mean: {max_abs_error_mean:.6f}")

    # Check if empirical mean is within theoretical confidence band
    within_band = np.all(np.abs(alpha_true - empirical_mean) <= 3*theoretical_std/np.sqrt(500))
    print(f"Empirical mean within theoretical confidence band? {within_band}")

    return x, alpha_true, empirical_mean, theoretical_std, empirical_std, realizations, C

def additional_analysis(L=1):
    """Additional analysis of convergence"""
    m_values = [10, 50, 100, 200, 500, 1000]
    mse_values = []

    for m in m_values:
        x, alpha_true, C, realizations = generate_gp_realizations(m=m, L=L)
        empirical_mean = np.mean(realizations, axis=0)
        mse = np.mean((alpha_true - empirical_mean)**2)
        mse_values.append(mse)

    # Plot convergence of empirical mean
    plt.figure(figsize=(10, 6))
    plt.plot(m_values, mse_values, 'bo-', linewidth=2, markersize=8)
    plt.plot(m_values, 1/np.array(m_values), 'r--', linewidth=2,
             label='Theoretical $O(1/m)$ convergence')
    plt.xlabel('Number of realizations $m$', fontsize=14)
    plt.ylabel('Mean Squared Error', fontsize=14)
    plt.title(f'Convergence of Empirical Mean to True Mean (L={L})', fontsize=16)
    plt.grid(True, alpha=0.3)
    plt.xscale('log')
    plt.yscale('log')
    plt.legend(fontsize=12)
    plt.tight_layout()
    plt.show()

    print("\n" + "="*60)
    print("CONVERGENCE ANALYSIS:")
    print("="*60)
    for m, mse in zip(m_values, mse_values):
        print(f"m = {m:4d}: MSE = {mse:.6f}")

def plot_histogram_at_x51(x, alpha_true,  realizations,C, L=1, m=500):
    """Plot histogram of Oŧ(xв,…в,Ґ) at x=0"""
    if x is None or alpha_true is None or realizations is None or C is None:
        print("Error: Missing data for histogram plot. Rerunning 500 realizations first.")
        x, alpha_true, C, realizations = generate_gp_realizations(m=m, L=L)

    # xв,…в,Ґ corresponds to index 50 (0-based) since xв,Ґ = -1
    # x = -1 + (51-1)*0.02 = -1 + 50*0.02 = -1 + 1 = 0
    idx_51 = 50
    x_51 = x[idx_51]
    alpha_x51 = alpha_true[idx_51]  # cos(ПЂ*0) = 1

    # Extract values at xв,…в,Ґ across all realizations
    values_at_x51 = realizations[:, idx_51]

    # Theoretical Gaussian distribution
    mu_theoretical = alpha_x51  # cos(ПЂ*0) = 1
    sigma_theoretical = np.sqrt(C[idx_51, idx_51])  # sqrt(1) = 1


    # Plot 1: Histogram with fitted Gaussian
    plt.figure(figsize=(10, 6))

    # Plot histogram with fitted Gaussian
    n, bins, patches = plt.hist(values_at_x51, bins=30, density=True,
                                alpha=0.7, edgecolor='black', linewidth=0.5,
                                label=f'Empirical (m={m})')

    # Overlay theoretical Gaussian PDF
    x_pdf = np.linspace(mu_theoretical - 4*sigma_theoretical,
                        mu_theoretical + 4*sigma_theoretical, 1000)
    pdf_theoretical = norm.pdf(x_pdf, loc=mu_theoretical, scale=sigma_theoretical)
    plt.plot(x_pdf, pdf_theoretical, 'r-', linewidth=2.5,
             label='Theoretical $N(\\alpha(x_{51}), C(x_{51},x_{51}))$')

    # Add vertical line at theoretical mean
    plt.axvline(x=mu_theoretical, color='green', linestyle='--', linewidth=2,
                label=f'True mean: $\\alpha(x_{{51}}) = {mu_theoretical:.2f}$')

    plt.xlabel('$\\alpha(x_{51})$ at $x=0$', fontsize=14)
    plt.ylabel('Probability Density', fontsize=14)
    plt.title(f'Histogram of $\\alpha(x_{{51}})$ at $x=0$ (L={L})', fontsize=16)
    plt.grid(True, alpha=0.3)
    plt.legend(fontsize=11)



    # Quantitative comparison
    empirical_mean = np.mean(values_at_x51)
    empirical_std = np.std(values_at_x51)

    print("="*70)
    print(f"ANALYSIS AT xв,…в,Ґ = 0 (L={L}, m={m}):")
    print("="*70)
    print(f"Theoretical distribution: N(Oj={mu_theoretical:.4f}, Пѓ={sigma_theoretical:.4f})")
    print(f"Empirical mean: {empirical_mean:.4f}")
    print(f"Empirical std: {empirical_std:.4f}")
    print(f"Absolute error in mean: {abs(empirical_mean - mu_theoretical):.4f}")
    print(f"Relative error in std: {abs(empirical_std - sigma_theoretical)/sigma_theoretical:.4f}")

    # Kolmogorov-Smirnov test for goodness of fit
    from scipy.stats import kstest
    ks_stat, ks_pvalue = kstest(values_at_x51, 'norm',
```

```
                                args=(mu_theoretical, sigma_theoretical))
    print(f"K-S test statistic: {ks_stat:.4f}")
    print(f"K-S test p-value: {ks_pvalue:.4f}")

    if ks_pvalue > 0.05:
        print("data consistent with Gaussian")
    else:
        print("data not consistent with Gaussian")

    return values_at_x51, mu_theoretical, sigma_theoretical

def explore_different_L_values():
    """Explore effect of different L values"""
    m = 100  # Reduced for faster computation
    L_values = [0.1, 0.5, 1.0, 2.0, 5.0, 10.0]

    fig, axes = plt.subplots(2, 3, figsize=(15, 10))
    axes = axes.flatten()

    for idx, L in enumerate(L_values):
        x, alpha_true, C, realizations = generate_gp_realizations(m=m, L=L)

        # Plot a few realizations
        ax = axes[idx]
        for i in range(min(10, m)):
            ax.plot(x, realizations[i, :], alpha=0.4, linewidth=0.8)

        # Plot true mean
        ax.plot(x, alpha_true, 'k-', linewidth=2.5, label='True mean')

        # Plot theoretical B±3Пѓ interval
        theoretical_std = np.sqrt(np.diag(C))
        ax.fill_between(x, alpha_true - 3*theoretical_std, alpha_true + 3*theoretical_std,
                        alpha=0.2, color='gray', label='Theoretical 99.7% interval')

        ax.set_xlabel('$x$', fontsize=10)
        ax.set_ylabel('$\\alpha(x)$', fontsize=10)
        ax.set_title(f'L = {L}\nVar at x=0: {C[50,50]:.2f}', fontsize=12)
        ax.grid(True, alpha=0.3)
        ax.set_xlim(-1, 1)
        ax.set_ylim(-4, 4)

        # Add legend only to first subplot
        if idx == 0:
            ax.legend(fontsize=9, loc='upper right')

    plt.suptitle('Gaussian Process Realizations for Different L Values', fontsize=16, y=1.02)
    plt.tight_layout()
    plt.show()

    # Analysis of variance and correlation length
    print("\n" + "="*70)
    print("ANALYSIS OF L EFFECT ON VARIANCE AND CORRELATION:")
    print("="*70)

    for L in L_values:
        x, alpha_true, C, _ = generate_gp_realizations(m=1, L=L)  # Just get covariance matrix

        # Variance at a point (should always be 1 since C(x,x)=1)
        var_point = C[50, 50]

        # Correlation between neighboring points
        corr_neighbor = C[50, 51]  # correlation between x=0 and x=0.02

        # Correlation length (distance where correlation drops to 1/e)
        # Solve exp(-d/L) = 1/e => d = L
        correlation_length = L

        print(f"\nL = {L:.1f}:")
        print(f"  Variance at x=0: {var_point:.4f} (theoretical: 1.0000)")
        print(f"  Correlation(x=0, x=0.02): {corr_neighbor:.4f}")
        print(f"  Theoretical correlation length: {correlation_length:.4f}")
        print(f"  Distance for correlation to drop to 0.5: {-L*np.log(0.5):.4f}")

    # Additional plot: Realization smoothness vs L
    plt.figure(figsize=(12, 8))
def analyze_variance_trend():
    """Analyze how variance in realizations changes with L and across x positions"""
    m = 500
    L_values = [1.0, 2.0, 5.0, 1e1,  1e2]

    fig, (ax1, ax2) = plt.subplots(2, 1, figsize=(12, 10))

    for L in L_values:
        x, alpha_true, C, realizations = generate_gp_realizations(m=m, L=L)

        # Compute empirical variance across realizations at each x
        empirical_variance = np.var(realizations, axis=0)
        theoretical_variance = np.diag(C)  # Should be 1 for all x

        # Plot variance as function of x
        ax1.plot(x, empirical_variance, linewidth=2, label=f'L={L}')

        # Store data for histogram at x=0
        idx_51 = 50  # x=0 index
        values_at_x0 = realizations[:, idx_51]

        # Plot histogram of alpha at x=0
        ax2.hist(values_at_x0, bins=30, alpha=0.5, density=True,
                label=f'L={L} (Пѓ={np.std(values_at_x0):.3f})')

        # Print statistics
        print(f"L={L}: Empirical Пѓ at x=0 = {np.std(values_at_x0):.4f}, Theoretical = 1.0000")

    # Format variance plot

    ax1.set_xlabel('x', fontsize=12)
    ax1.set_ylabel('Empirical Variance', fontsize=12)
    ax1.set_title('Variance as Function of x for Different L Values', fontsize=14)
    ax1.grid(True, alpha=0.3)
    ax1.legend(loc='upper right', fontsize=10)
    ax1.set_xlim(-1, 1)

    # Format histogram plot
    x_pdf = np.linspace(-4, 4, 1000)
    pdf_theoretical = norm.pdf(x_pdf, loc=1, scale=1)  # Theoretical N(1,1)

    ax2.set_xlabel('Oұ(x=0)', fontsize=12)
    ax2.set_ylabel('Probability Density', fontsize=12)
    ax2.set_title('Histogram of Oұ(x=0) for Different L Values', fontsize=14)
    ax2.grid(True, alpha=0.3)
    ax2.legend(loc='best')
    ax2.set_xlim(-3, 5)
```

```
plt.tight_layout()
plt.show()

# Print statistical analysis
print("\n" + "="*70)
print("STATISTICAL ANALYSIS:")
print("="*70)
print("Empirical variance analysis:")
print("L Value | Mean Variance | Std of Variance ")
print("-"*60)

for L in L_values:
    x, alpha_true, C, realizations = generate_gp_realizations(m=m, L=L)
    empirical_variance = np.var(realizations, axis=0)

    mean_var = np.mean(empirical_variance)
    std_var = np.std(empirical_variance)
    max_dev = np.max(np.abs(empirical_variance - 1))

    print(f"L={L:<4} | {mean_var:.6f}     | {std_var:.6f}       | {max_dev:.6f}")
```