In [108]:

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

# Define the custom function f(x)
def custom_function(x):
    if 0 <= x <= np.pi:
        return 1
    elif -np.pi <= x < 0:
        return -2 * x / np.pi

# Fourier series coefficients
def fourier_coefficients(f, n):
    a = np.zeros(n)
    b = np.zeros(n)

    for i in range(n):
        a[i] = (1 / np.pi) * np.trapz([f(x) * np.cos(i * x) for x in np.linspace(-np.pi, np.pi, 1000)], dx=2 * n
        b[i] = (1 / np.pi) * np.trapz([f(x) * np.sin(i * x) for x in np.linspace(-np.pi, np.pi, 1000)], dx=2 * n

    return a, b

# Fourier series approximation
def fourier_series(x, a, b, n):
    series = a[0] / 2
    for i in range(1, n):
        series += a[i] * np.cos(i * x) + b[i] * np.sin(i * x)
    return series

# Find the lowest peak value to the left and the highest peak value to the right of x=0
def find_peaks(x_values, y_values, x_range):
    # Determine the range of indices within the specified x_range
    x_min, x_max = x_range
    x_min_idx = np.argmax(x_values >= x_min)
    x_max_idx = np.argmax(x_values >= x_max)

    # Find the peak within the specified range
    x_peak = x_values[x_min_idx:x_max_idx]
    y_peak = y_values[x_min_idx:x_max_idx]

    # Find the lowest peak to the left
    left_half_idx = len(y_peak) // 2
    x_peak_left = x_peak[:left_half_idx][np.argmin(y_peak[:left_half_idx])]
    y_peak_left = min(y_peak[:left_half_idx])

    # Find the highest peak to the right
    right_half_idx = len(y_peak) // 2
```