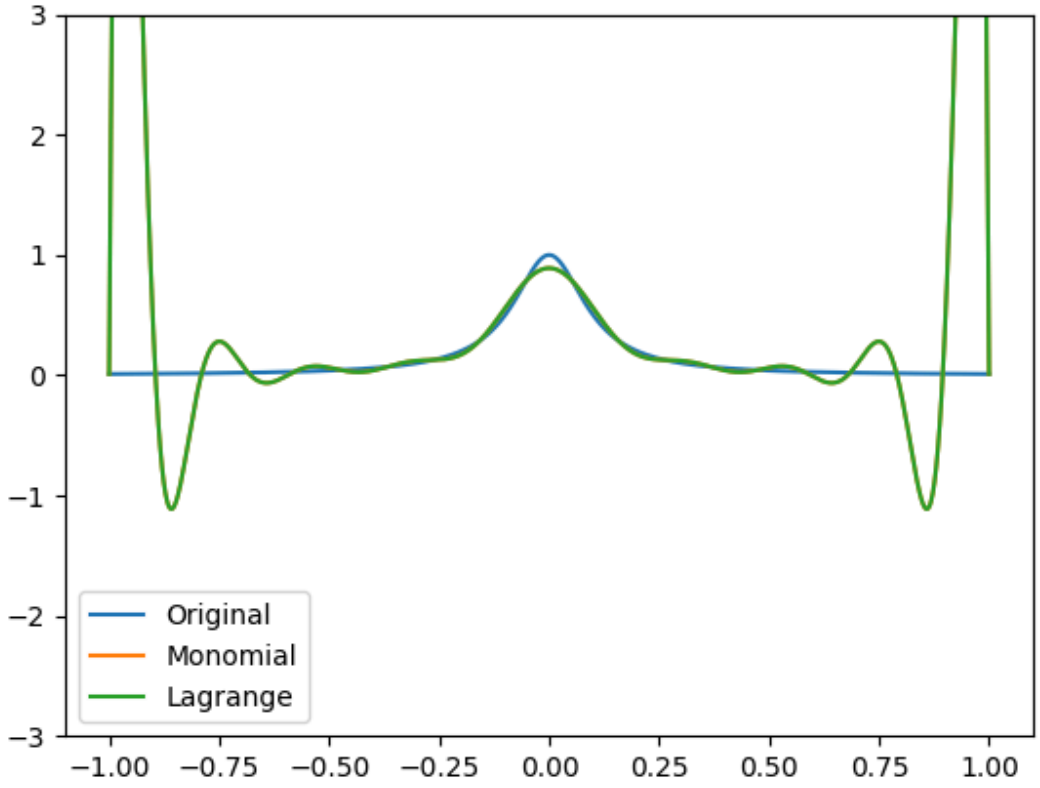
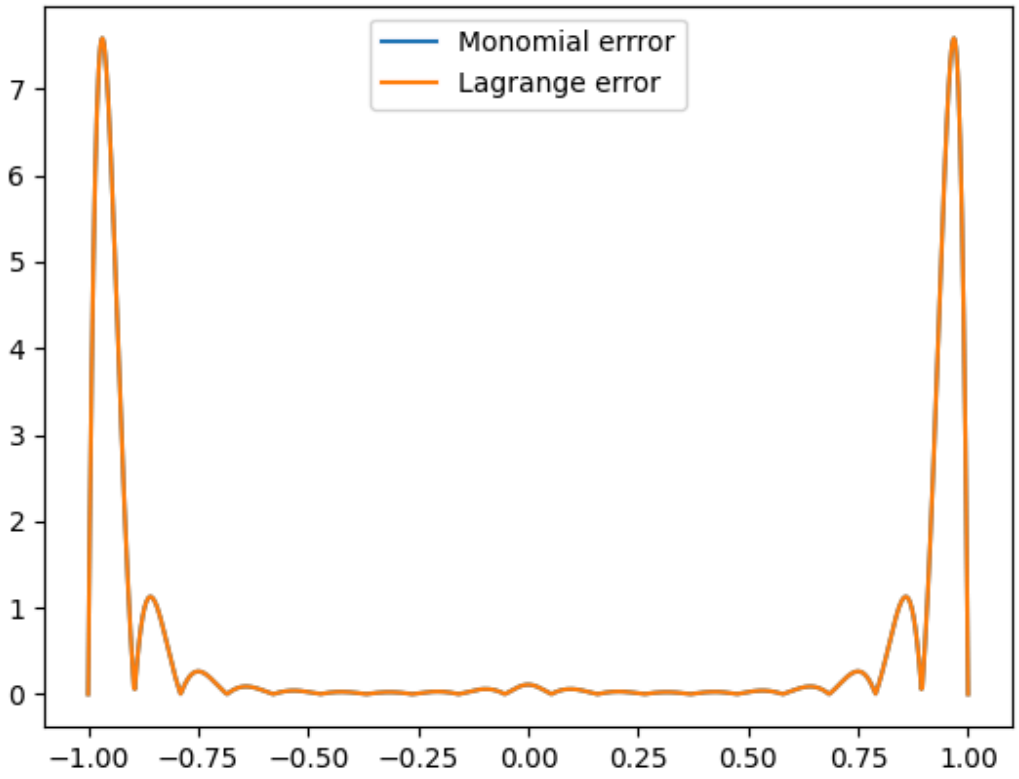

3
2
1
0
−1
−2
−3
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00
Original
Monomial
Lagrange




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

def p_monomial(x, f, n, interp_points):
    y_values = f(interp_points(n))

    basis_matrix = np.zeros((n + 1, n + 1))
    for row_idx in range(n + 1):
        for col_idx in range(n + 1):
            basis_matrix[row_idx][col_idx] = interp_points(n)[row_idx] **
col_idx

    coefficients = np.linalg.lstsq(basis_matrix, y_values, rcond=None)[0]

    p = np.zeros_like(x)
    for idx, coeff in enumerate(coefficients):
```

```
        p += coeff * x ** idx

    return p

def p_lagrangian(x, f, n, interp_points):
    interpolation_nodes = interp_points(n)

    lagrange_functions = []

    for k in range(0, n + 1):
        L_k_numerator = 1.0
        L_k_denominator = 1.0
        for i in range(0, n + 1):
            if (i != k):
                L_k_numerator *= (x - interpolation_nodes[i])
                L_k_denominator *= (interpolation_nodes[k] -
interpolation_nodes[i])

        L_k = L_k_numerator / L_k_denominator
        lagrange_functions.append(L_k)

    p = 0.0
    for i in range(0, len(lagrange_functions)):
        p += f(interpolation_nodes[i]) * lagrange_functions[i]

    return p


def main():
    def f(x):
        return 1.0 / (1 + (10 * x) ** 2)

    def interp_node1(N):

        x1 = np.zeros((N + 1))
        h = 2.0 / (N - 1)

        for i in range(0, N + 1):
            x1[i] = -1 + (i - 1) * h
```

```
        return x1

    x = np.linspace(-1, 1, 1000)

    N = 20

    y1 = f(x)
    y2 = p_monomial(x, f, N, interp_node1)
    y2err = np.abs(y2 - y1)
    y3 = p_lagrangian(x, f, N, interp_node1)
    y3err = np.abs(y3 - y1)

    plt.figure()
    plt.plot(x, y1)
    plt.plot(x, y2)
    plt.plot(x, y3)
    plt.ylim(-3, 3)
    plt.legend(["Original", "Monomial", "Lagrange"])
    plt.figure()
    plt.plot(x, y2err)
    plt.plot(x, y3err)
    plt.legend(["Monomial errror", "Lagrange error"])
    plt.show()


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