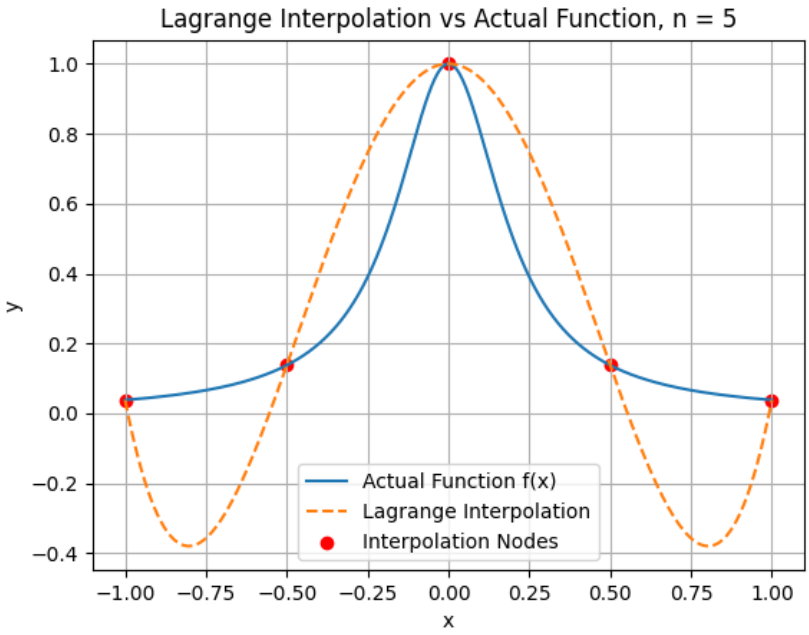
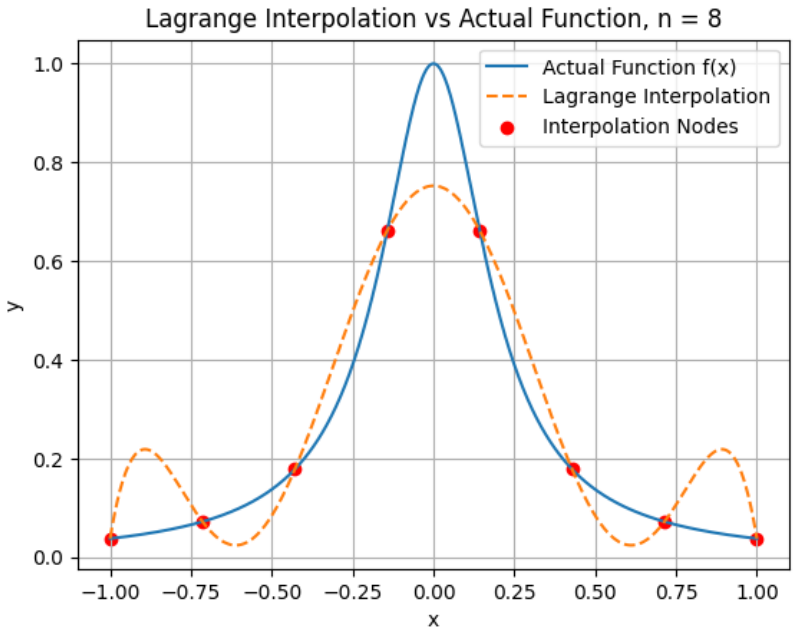
MATH 417 502
Homework 4

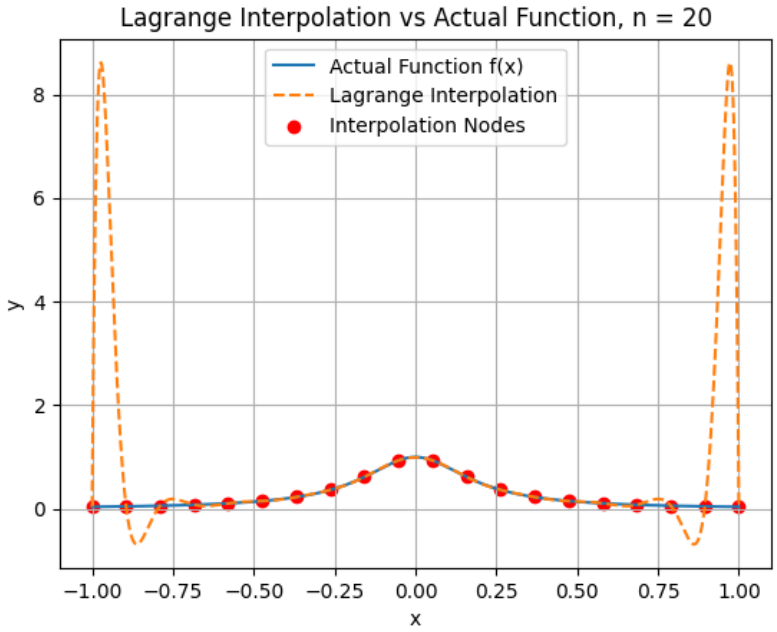
Keegan Smith

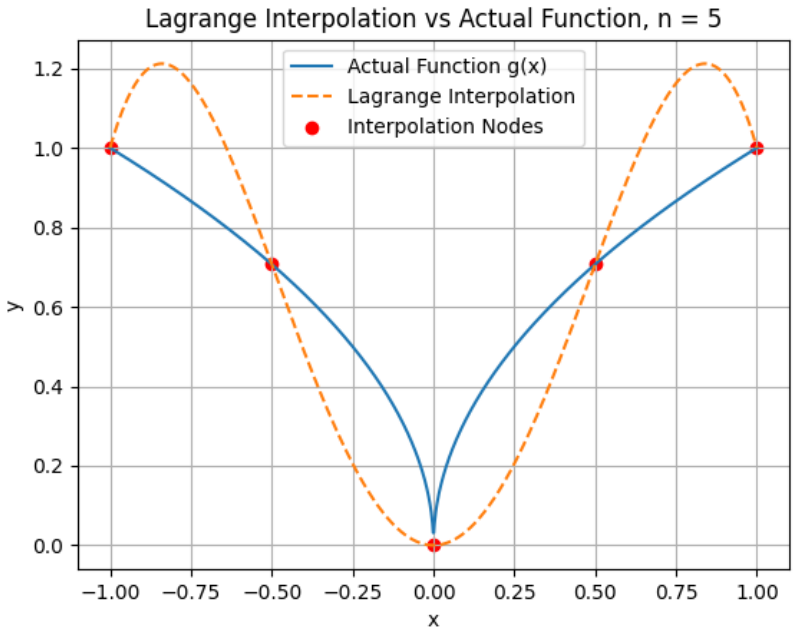
September 15, 2024

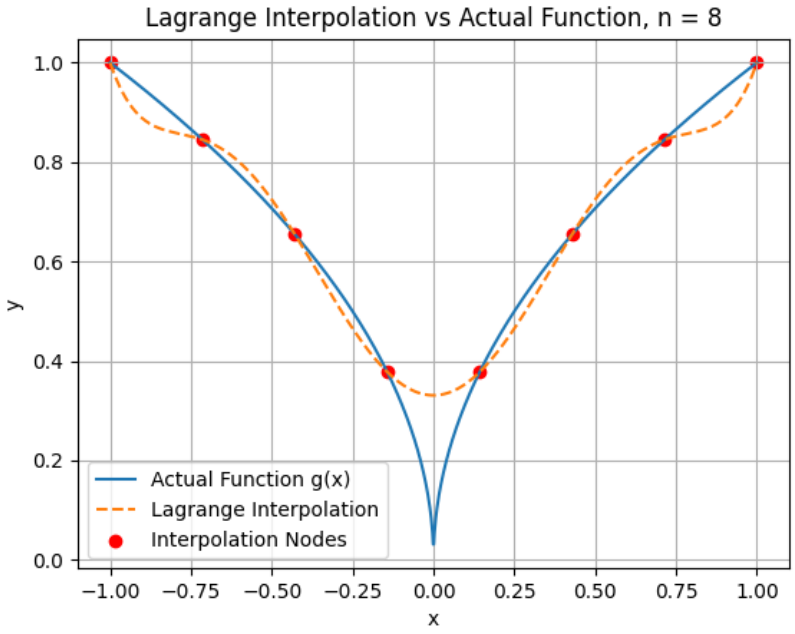
Problem 1

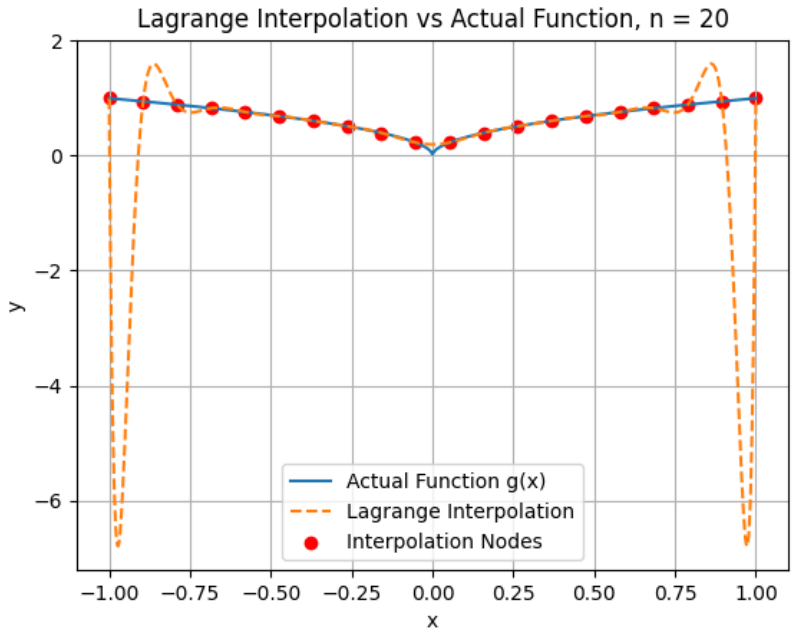



Lagrange Interpolation vs Actual Function, n = 8
Actual Function f(x)
Lagrange Interpolation
Interpolation Nodes
1.0
0.8
0.6
0.4
0.2
0.0
y
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00
x


Lagrange Interpolation vs Actual Function, n = 20
Actual Function f(x)
Lagrange Interpolation
Interpolation Nodes
8
6
4
2
0
y
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00
x


Lagrange Interpolation vs Actual Function, n = 5
Actual Function g(x)
Lagrange Interpolation
Interpolation Nodes
1.2
1.0
0.8
0.6
0.4
0.2
0.0
y
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00
x


Lagrange Interpolation vs Actual Function, n = 8
1.0
0.8
0.6
0.4
0.2
0.0
y
Actual Function g(x)
Lagrange Interpolation
Interpolation Nodes
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00
x



code:

```
import numpy as np
import matplotlib.pyplot as plt
def lagrange(x, points):
    lagrange_results = [];
    for i in range(0, len(points)):
        numerator = 1;
        denominator = 1;
        for j in range(0, len(points)):
            if(i == j):
                continue;
            numerator *= (x - points[j][0]);
            denominator *= (points[i][0] - points[j][0]);
        lagrange_results.append(numerator / denominator);
    result = 0;
    for i in range(0, len(points)):
        result += points[i][1] * lagrange_results[i]
    return result;

def f(x):
    return 1 / (1 + 25 * x**2);
def g(x):
    return (abs(x)) ** (1/2);
```

```
23
24 def get_x_coords(interval, num_points):
25     start = interval[0];
26     orig_start = start
27     end = interval[1];
28     result = [];
29     result.append(start);
30     for i in range(0, num_points - 1):
31         start += (end - orig_start) / (num_points - 1)
32         result.append(start)
33     return result;
34 def do_the_thing(my_function, n):
35     x_coords = get_x_coords([-1, 1], n);
36     actual_function_values = []
37     for x in x_coords:
38         actual_function_values.append([x, my_function(x)]);
39
40     x_plot = np.linspace(-1, 1, 1000)
41     y_actual = []
42     y_interp = []
43     for i in range(0, len(x_plot)):
44         y_actual.append(my_function(x_plot[i]));
45         y_interp.append(lagrange(x_plot[i],
               actual_function_values));
46
47     plt.plot(x_plot, y_actual, label=f'Actual Function {
           my_function.__name__}(x)')
48     plt.plot(x_plot, y_interp, '--', label='Lagrange
           Interpolation')
49     plt.scatter(x_coords, [my_function(x) for x in x_coords
           ], color='red', label='Interpolation Nodes')
50     plt.title(f'Lagrange Interpolation vs Actual Function, n
           = {n}')
51     plt.xlabel('x')
52     plt.ylabel('y')
53     plt.legend()
54     plt.grid(True)
55     plt.show()
56 def main():
57     functions = [f, g]
58     nums = [5, 8, 20]
59     for function in functions:
60         for num in nums:
61             do_the_thing(function, num)
62 if __name__ == "__main__":
63     main();
```