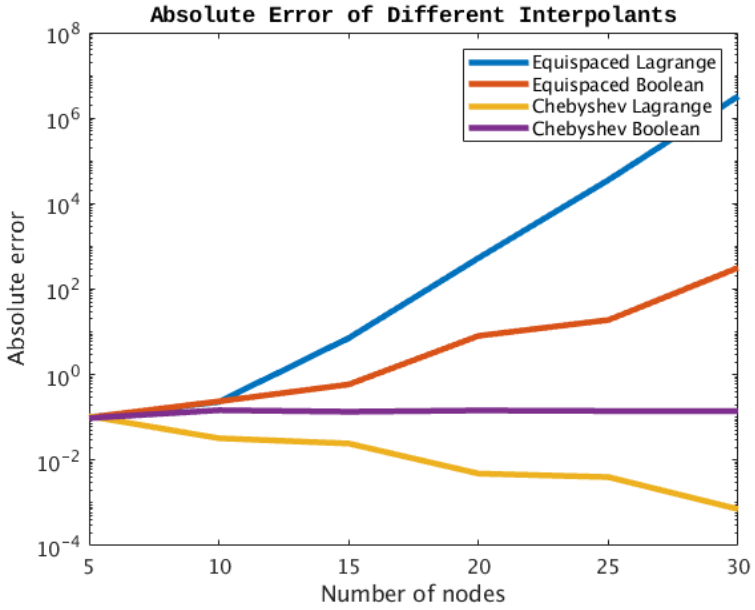
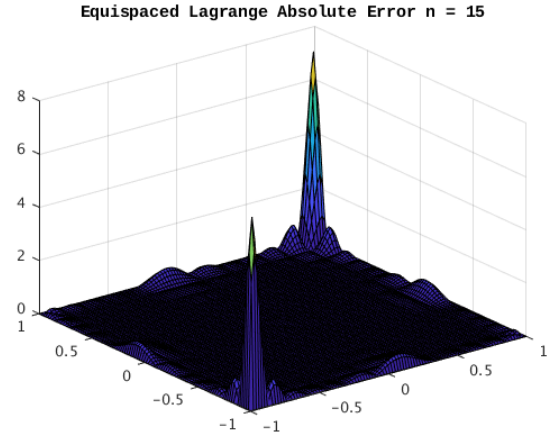
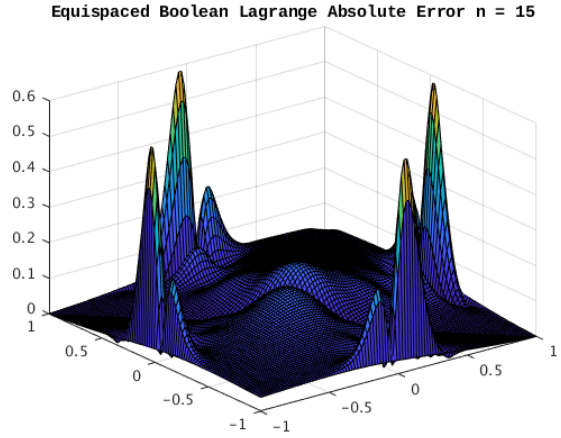
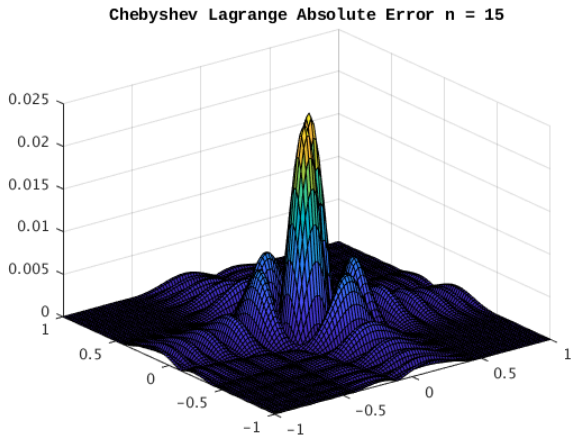
Problems

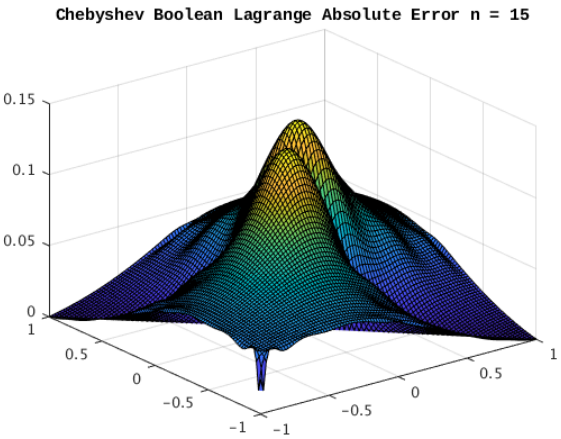
1. Consider the error plot and surface plots below comparing regular a regular 2D Lagrange interpolant vs a Boolean Sum Lagrange based interpolant











(a) From the error plot focusing on the equispaced curves, we can see that using a Boolean Sum Lagrange Interpolant works marginally better than a regular Lagrange based method. Although both errors are increasing as the number of nodes increases, the Boolean Sum is increasing a much slower rate.

(b) From the same error but now focusing on the Chebyshev a curves, we see a slightly different trend. Now the error in the Boolean Sum is remaining relatively constant with an increase in the number of nodes while the error is actually decreasing for the regular Lagrange interpolant.

(c) To understand the differences between the two methods, let's take a closer look at the sums we have to compute in each method. In the case of the regular Lagrange basis, we experience Runge at the boudnaries of our interval due to the Lagrange polynomials. So, using Chebyshev nodes pushes the error to the inside of the interval allowing us to actually have convergence of the Lagrange interpolant. Furthermore, the Lagrange interpolant uses the lagrange basis over our whole domain which pushes the error to the corners of our boundaries.

On the other hand, we have the Boolean Sum Lagrange interpolant which relies on the data along each x and y axis individually instead of having all of the cross terms. This splitting of the data causes the error to be focused near the middle of our boundaries. So, using Chebyshev nodes, the error is mitigated at the boundaries but the Runge at the middle of the boundaries was not bad to begin with so the error doesn't change much.

2. Prove the following result: Let $f \in C^2[a,b]$ with $f''(x) > 0$ for $a \leq x \leq b$. If $q_i^*(x) = a_0 + a_1 x$ is the linear minimax approximation to $f(x)$ on $[a,b]$, then

$$a_1 = \frac{f(b) - f(a)}{b-a}, \quad a_0 = \frac{f(a) + f(c)}{2} - \frac{a+c}{2} \cdot \frac{f(b) - f(a)}{b-a}$$

where c is the unique solution of

$$f'(c) = \frac{f(b) - f(a)}{b-a}.$$

Proof:

From Cauchy's Equioscillation Theorem, we know there exists a unique polynomial of the form $p_1^*(x) = a_0 + a_1 x$ such that the error $E(x) = f(x) - p_1^*(x)$ satisfies

$$E(a) = \rho \tag{1}$$
$$E(c) = -\rho \tag{2}$$
$$E(b) = \rho \tag{3}$$
$$E'(c) = 0 \tag{4}$$

where ρ is the maximum error on $[a,b]$ and $c \in (a,b)$. Now using (4), we have

$$E'(c) = f'(c) - a_1 = 0 \implies a_1 = f'(c).$$

Then, ((1) - (3)) implies

$$f(a) - f(b) + a_1(b-a) = 0 \implies \boxed{a_1 = \frac{f(b) - f(a)}{b-a}}$$

which implies c is the solution to

$$\boxed{f'(c) = \frac{f(b) - f(a)}{b - a}}.$$

Next, ((1) + (2)) implies

$$f(a) + f(c) - 2a_0 - a_1(a + c) = 0 \implies \boxed{a_0 = \frac{f(a) + f(c)}{2} - \frac{a + c}{2} \cdot \frac{f(b) - f(a)}{b - a}}.$$

Finally, the max error can be obtained from (1) as

$$\boxed{\rho = f(a) - \frac{f(a) + f(c)}{2} + \frac{f(b) - f(a)}{b - a}\left(\frac{a + c}{2} - a\right)}.$$

□

Code used

```
%%
% Comparison of lagrange interpolant and boolean sum lagrange
% Author: Caleb Jacobs
% Date last modified: 05-11-2021

%% Settings
f = @(x, y) (x + y) ./ (1 + 25 * (x.^2 + y.^2));
% f = @(x, y) exp(x + y);

errs = zeros(6, 4);
N = [5 : 5 : 30];

%% Part a
for i = 1 : length(N)
n = N(i);
x = linspace(-1, 1, n);
y = linspace(-1, 1, n);
[xtmp, ytmp] = meshgrid(x, y);
z = f(xtmp, ytmp);

xx = linspace(-1, 1, 100);
yy = linspace(-1, 1, 100);
[X, Y] = meshgrid(xx, yy);
Z = f(X, Y);
ZI = evalLagrange(X, Y, x, y, z);
ZB = evalBool(X, Y, x, y, f);

errs(i, 1:2) = [norm(abs(Z(:) - ZI(:)), 2), norm(abs(Z(:) - ZB(:)), 2)];

```

```
%% Generate error plots
f1 = figure(1);
surf(X, Y, abs(Z - ZI))
title(sprintf('Equispaced Lagrange Absolute Error n = %d', n))

f2 = figure(2);
surf(X, Y, abs(Z - ZB))
title(sprintf('Equispaced Boolean Lagrange Absolute Error n = %d', n))

%% Part b
% n = 30;
k = 1 : n;
x = cos((2*k - 1) * pi / (2 * n));
y = cos((2*k - 1) * pi / (2 * n));
[xtmp, ytmp] = meshgrid(x, y);
z = f(xtmp, ytmp);

xx = linspace(-1, 1, 100);
yy = linspace(-1, 1, 100);
[X, Y] = meshgrid(xx, yy);
Z = f(X, Y);
ZI = evalLagrange(X, Y, x, y, z);
ZB = evalBool(X, Y, x, y, f);

errs(i, 3:4) = [norm(abs(Z(:) - ZI(:)), 2), norm(abs(Z(:) - ZB(:)), 2)];

f3 = figure(3);
surf(X, Y, abs(Z - ZI))
title(sprintf('Chebyshev Lagrange Absolute Error n = %d', n))

f4 = figure(4);
surf(X, Y, abs(Z - ZB))
title(sprintf('Chebyshev Boolean Lagrange Absolute Error n = %d', n))

%% Save figures
pause

% saveas(f1, sprintf('images/EL%02d.png', n))
% saveas(f2, sprintf('images/CL%02d.png', n))
% saveas(f3, sprintf('images/EB%02d.png', n))
% saveas(f4, sprintf('images/CB%02d.png', n))
end

figure(10)
semilogy(N, errs, 'LineWidth', 3)
legend('Equispaced Lagrange', ...
       'Equispaced Boolean', ...
       'Chebyshev Lagrange', ...
       'Chebyshev Boolean')
title('Absolute Error of Different Interpolants')
xlabel('Number of nodes')
```

```
ylabel('Absolute error')


%% Lagrange basis
function val = L(x, xi, j)
    n = length(xi);

    val = 1;
    for i = 1 : j - 1
        val = val * (x - xi(i)) / (xi(j) - xi(i));
    end
    for i = j + 1 : n
        val = val * (x - xi(i)) / (xi(j) - xi(i));
    end
end

%% Evaluate lagrange interpolation
function Z = evalLagrange(X, Y, x, y, z)
    Z = zeros(size(X));
    X = X(:);
    Y = Y(:);

    for i = 1 : length(X)
        for j = 1 : length(y)
            for k = 1 : length(x)
               Z(i) = Z(i) + z(j, k) * L(X(i), x, k) * L(Y(i), y, j);
            end
        end
    end
end

%% Evaluate boolean lagrange interpolation
function Z = evalBool(X, Y, x, y, f)
    p = length(x);
    q = length(y);
    Z = zeros(size(X));
    X = X(:);
    Y = Y(:);

    for i = 1 : length(X)
        sx = 0;
        for j = 1 : p
            sx = sx + f(x(j), Y(i)) * L(X(i), x, j);
        end

        sy = 0;
        for k = 1 : q
            sy = sy + f(X(i), y(k)) * L(Y(i), y, k);
        end

        cor = 0;
```

```
132         for j = 1 : p
133             for k = 1 : q
134                 cor = f(x(j), y(k)) * L(X(i), x, j) * L(Y(i), y, k);
135             end
136         end
137 
138         Z(i) = sx + sy - cor;
139     end
140 end
```