APPM 5600 - Homework 10

Eappen Nelluvelil

November 5, 2021

1. In this problem, you will compare the performance of classic tensor product interpolation with the Boolean sum Lagrange interpolation.

   Let $f(x, y)$ denote the function we are interpolating. The Boolean sum Lagrange interpolation is built from data along curves where x is constant and y is constant. The two independent univariate interpolants are

   $$s_x(x, y) = \sum_{j=1}^{p} f(x_j, y)\, \ell_{x,j}(x)$$

   and

   $$s_y(x, y) = \sum_{k=1}^{q} f(x, y_k)\, \ell_{y,k}(y).$$

   These blend together the two curves. To construct the overall interpolant, we need to sum these and add a correction term. The final form of the interpolation polynomial is

   $$s(x, y) = s_x(x, y) + s_y(x, y) - \sum_{j=1}^{p} \sum_{k=1}^{q} f(x_j, y_k)\, \ell_{x,j}(x)\, \ell_{y,k}(y).$$

   Throughout, $\ell_{x,j}(x)$ denotes the j^{th} Lagrange polynomial in the x-direction, etc.

   (a) Consider $f(x, y) = \frac{x+y}{1+25(x^2+y^2)}$ on $[-1, 1] \times [-1, 1]$. Using an equispaced tensor product grid of interpolation nodes, compare the performance of the Boolean sum interpolation with the standard tensor product interpolation for `n = 5:5:30`.

   When using an equispaced tensor product grid of interpolation nodes, both the tensor product Lagrange interpolant and Boolean sum Lagrange interpolant perform poorly. This can be seen by analyzing a `semilogy` plot of the 2-norm of the errors between f and the two interpolants on a fine mesh on the domain $[-1, 1]$ (see attached code and plots). For large n, we see that there is no appreciable difference between the regular tensor product Lagrange interpolant and Boolean sum Lagrange interpolant.

   (b) Do the same problem, but use Chebyshev interpolation nodes. How does the performance of each technique change?

   When we use Chebyshev interpolation nodes and analyze the `semilogy` plot of the 2-norm of the errors between f and the two interpolants, both interpolants perform well, i.e., the 2-norm of the error decreases as we increase n. However, for large n, we see that the Boolean sum Lagrange interpolant achieves a lower 2-norm error than the regular tensor product Lagrange interpolant (see attached code and plots).

   (c) Provide an explanation for the observed results.

   We expect that using Chebyshev interpolation nodes will lead to better tensor product and Boolean sum Lagrange interpolants than using equispaced interpolation nodes. In one dimension, equispaced nodes are a poor choice since the associated Lebesgue constant grows rapidly, whereas the Chebyshev nodes are a

nearly optimal choice since the associated Lebesgue constant grows far more slowly. For similar reasons, equispaced nodes are also a poor choice for interpolating with, whereas Chebyshev nodes (although not optimal) are a better choice for interpolating with.

When using Chebyshev interpolation nodes, the Boolean sum Lagrange interpolant performs better than the regular tensor product Lagrange interpolant, particularly near the boundaries, because along each curve where x or y is held constant, the independent univariate interpolants do a better job of approximating f than a simple tensor product of the two.

2. Prove the following result: let $f \in \mathcal{C}^2([a,b])$, with $f''(x) > 0$ for $x \in [a,b]$. If $q_1^*(x) = a_0 + a_1 x$ is the linear minimax approximation to f on $[a,b]$, then

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

where c is the unique solution of

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

What is the error ρ?

Suppose that $f \in \mathcal{C}^2([a,b])$, with $f''(x) > 0$ for all $x \in [a,b]$. The linear minimax approximation to f on $[a,b]$ is given by $q_1^*(x) = a_0 + a_1 x$, where a_0 and a_1 are constants that are to be determined. We know that the linear minimax approximation is such that the maximum error, ρ, between f and q_1^* is achieved at three points on $[a,b]$: at a, b, and some point c, which is between a and b. Furthermore, the first degree minimax approximation of f is such that the error, $E(x) = f(x) - q_1^*(x)$, is maximized at a, b, and x^*, and the maximum error, ρ, equioscillates at a, b, and c. Furthermore, since the error changes sign at c, we have that $E'(c) = 0$. This leads to the following four equations in four unknowns:

$$f(a) - a_0 - a_1 a = \rho, \tag{1}$$
$$f(b) - a_0 - a_1 b = \rho, \tag{2}$$
$$f(c) - a_0 - a_1 c = -\rho, \tag{3}$$
$$f'(c) - a_1 = 0. \tag{4}$$

We can subtract the first equation from the second to obtain that

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

From the fourth equation, we have that

$$f'(c) - a_1 = 0 \implies f'(c) = a_1$$
$$\implies f'(c) = \frac{f(b) - f(a)}{b-a}.$$

Equating the first and third equations (after negating it), we have that

$$f(a) - a_0 - \left(\frac{f(b) - f(a)}{b-a}\right)a = -f(c) + a_0 + \left(\frac{f(b) - f(a)}{b-a}\right)c \implies \frac{f(a) + f(c)}{2} - \left(\frac{a+c}{2}\right)\left(\frac{f(b) - f(a)}{b-a}\right) = a_0.$$

Thus, we have that

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

where c is the unique solution of

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

Furthermore, the error, ρ, is given by

$$\begin{aligned}
\rho &= f(a) - \left(\frac{f(a) + f(c)}{2} - \left(\frac{a + c}{2} \right) \left(\frac{f(b) - f(a)}{b - a} \right) \right) - \left(\frac{f(b) - f(a)}{b - a} \right) a \\
&= f(a) - \left(\frac{f(a) + f(c)}{2} \right) + \left(\frac{a + c}{2} \right) \left(\frac{f(b) - f(a)}{b - a} \right) - \frac{2a}{2} \left(\frac{f(b) - f(a)}{b - a} \right) \\
&= f(a) - \left(\frac{f(a) + f(c)}{2} \right) + \left(\frac{c - a}{2} \right) \left(\frac{f(b) - f(a)}{b - a} \right).
\end{aligned}$$

```
% Homework 10
clc;
clear;
close all;

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

a = -1;
b = 1;

n = (5:5:30)';
n_eval = 100;

equispaced_nodes_errs = zeros(2, length(n));
chebyshev_nodes_errs  = zeros(2, length(n));

for i = 1:length(n)
    n_i = n(i);
    x_nodes = linspace(a, b, n_i);
    % x_nodes = cos((2*(0:n_i-1)+1)*pi/(2*(n_i)));
    y_nodes = x_nodes;
    [X, Y] = meshgrid(x_nodes, y_nodes);
    x = reshape(X, 1, (n_i)^2);
    y = reshape(Y, 1, (n_i)^2);
    f_xy_nodes = f(x, y);

    % Create the tensor grid Lagrange interpolant
    x_eval = linspace(a, b, n_eval);
    y_eval = x_eval;
    [XX, YY] = meshgrid(x_eval, y_eval);
    xx_eval = reshape(XX, 1, n_eval^2);
    yy_eval = reshape(YY, 1, n_eval^2);
    xy_eval = [xx_eval; yy_eval];
```

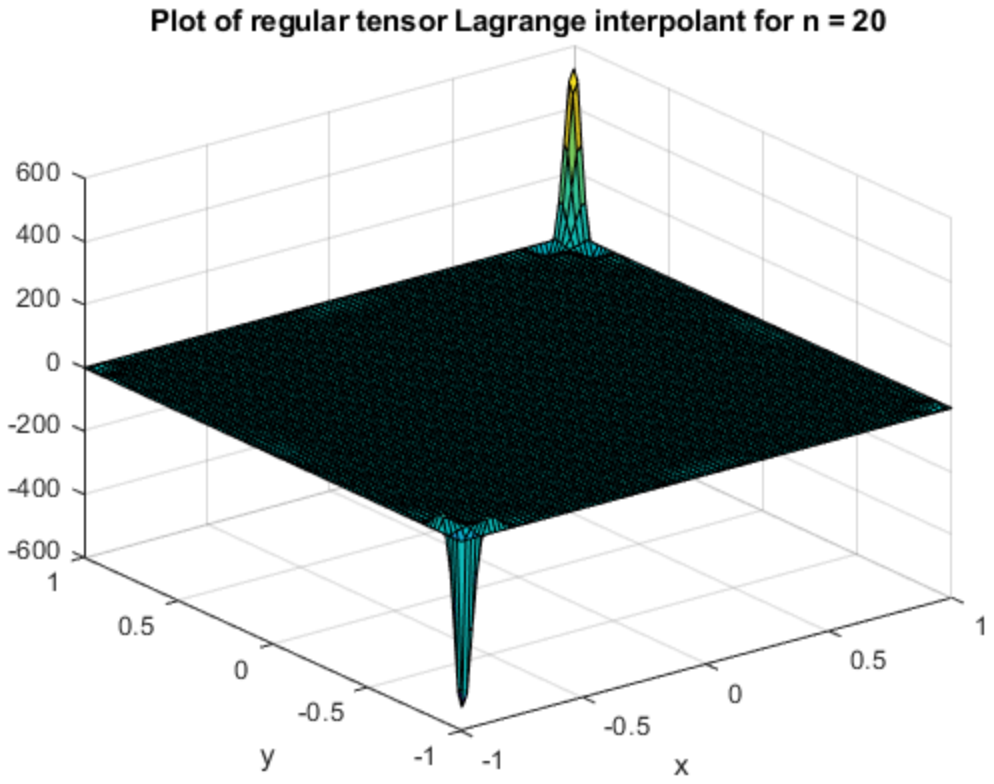
Regular tensor Lagrange interpolant

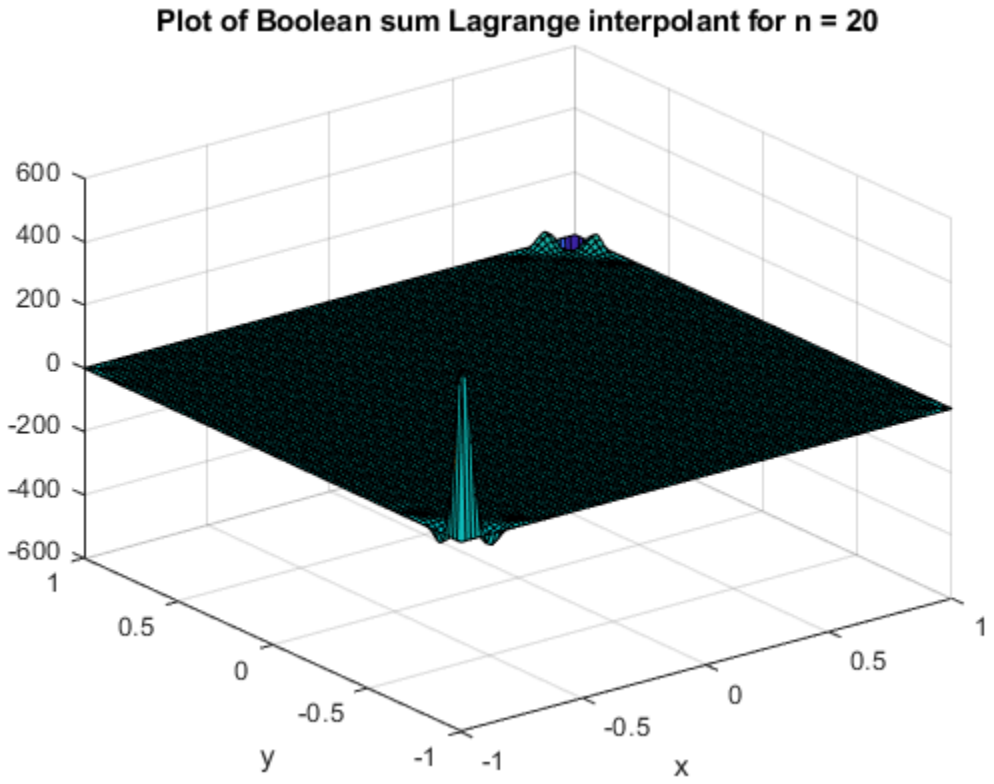
```
    z = tensorproduct2D_lagrange(xy_eval, x_nodes, y_nodes, f_xy_nodes);
    Z = reshape(z, n_eval, n_eval);

    if n_i == 20
        figure;
        surf(XX, YY, Z);
        xlabel("x");
        ylabel("y");
        title("Plot of regular tensor Lagrange interpolant for n = 20");
    end
```

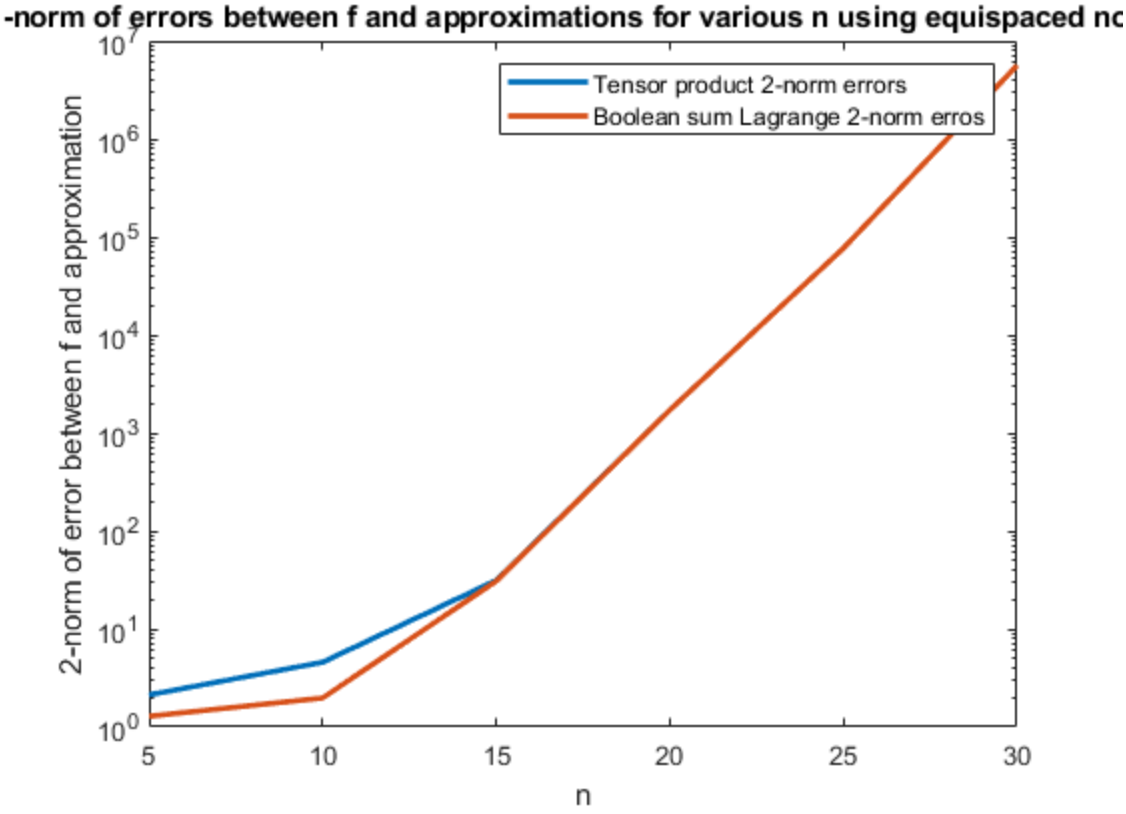
Boolean sum Lagrange interpolant

```
    s_x = zeros(length(xx_eval), 1);
    for k = 1:n_i
```

```
        s_x = s_x + f(x_nodes(k), yy_eval').*eval_lagrange(xx_eval', x_nodes,
 k);
    end

    s_y = zeros(length(yy_eval), 1);
    for k = 1:n_i
        s_y = s_y + f(xx_eval', y_nodes(k)).*eval_lagrange(yy_eval', y_nodes,
 k);
    end

    s = reshape(s_x + s_y, n_eval, n_eval) - Z;

    if n_i == 20
        figure;
        surf(XX, YY, s);
        xlabel("x");
        ylabel("y");
        title("Plot of Boolean sum Lagrange interpolant for n = 20");
    end

    % Display the error (in the 2-norm) between the interpolant and actual
    % function values
    f_xy_eval = f(xx_eval, yy_eval);
    % equispaced_nodes_errs(1, i) = norm(f_xy_eval - z');
    % equispaced_nodes_errs(2, i) = norm(f_xy_eval - reshape(s, 1, n_eval^2));
    chebyshev_nodes_errs(1, i) = norm(f_xy_eval - z');
    chebyshev_nodes_errs(2, i) = norm(f_xy_eval - reshape(s, 1, n_eval^2));
end

figure;
semilogy(n, chebyshev_nodes_errs(1, :), "LineWidth", 2, "DisplayName", "Tensor
 product 2-norm errors");
hold on;
semilogy(n, chebyshev_nodes_errs(2, :), "LineWidth",
 2, "DisplayName", "Boolean sum Lagrange 2-norm erros");
xlabel("n");
ylabel("2-norm of error between f and approximation");
title("2-norm of errors between f and approximations for various n using
 equispaced nodes");
% title("2-norm of errors between f and approximations for various n using
 Chebyshev nodes");
legend;

function val = eval_lagrange(x, x_nodes, j)
    val = ones(length(x), 1);
    for i = 1:length(x_nodes)
        if i ~= j
            val = val .* (x-x_nodes(i))/(x_nodes(j)-x_nodes(i));
        end
    end
end
```


Plot of regular tensor Lagrange interpolant for n = 20
600
400
200
0
-200
-400
-600
1
0.5
0
-0.5
-1
y
-1
-0.5
0
0.5
1
x


Plot of Boolean sum Lagrange interpolant for n = 20
600
400
200
0
-200
-400
-600
1
0.5
0
-0.5
-1
y
-1
-0.5
0
0.5
1
x



Published with MATLAB® R2021b

```
% Homework 10
clc;
clear;
close all;

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

a = -1;
b = 1;

n = (5:5:30)';
n_eval = 100;

equispaced_nodes_errs = zeros(2, length(n));
chebyshev_nodes_errs  = zeros(2, length(n));

for i = 1:length(n)
    n_i = n(i);
    % x_nodes = linspace(a, b, n_i);
    x_nodes = cos((2*(0:n_i-1)+1)*pi/(2*(n_i)));
    y_nodes = x_nodes;
    [X, Y] = meshgrid(x_nodes, y_nodes);
    x = reshape(X, 1, (n_i)^2);
    y = reshape(Y, 1, (n_i)^2);
    f_xy_nodes = f(x, y);

    % Create the tensor grid Lagrange interpolant
    x_eval = linspace(a, b, n_eval);
    y_eval = x_eval;
    [XX, YY] = meshgrid(x_eval, y_eval);
    xx_eval = reshape(XX, 1, n_eval^2);
    yy_eval = reshape(YY, 1, n_eval^2);
    xy_eval = [xx_eval; yy_eval];
```

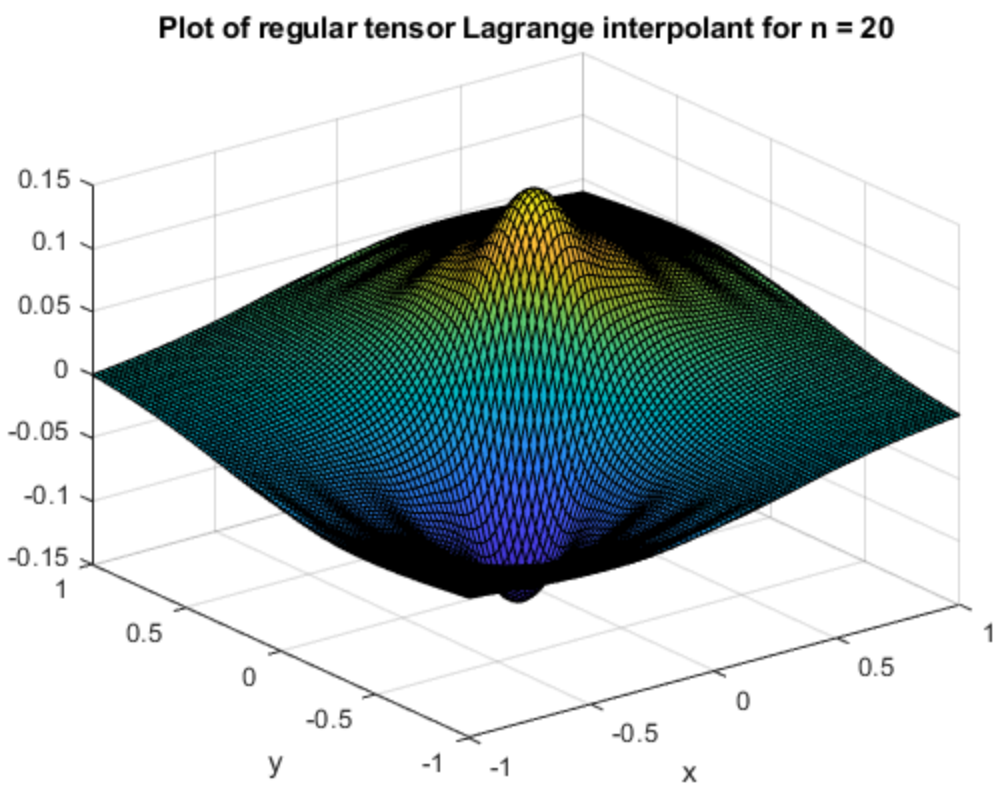
Regular tensor Lagrange interpolant

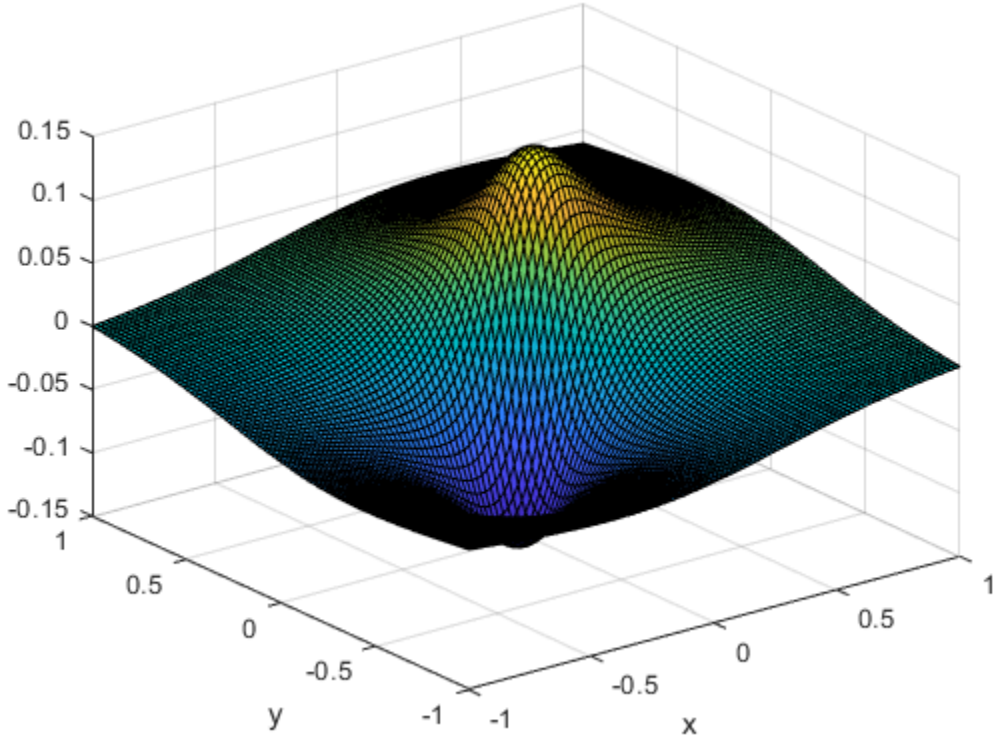
```
    z = tensorproduct2D_lagrange(xy_eval, x_nodes, y_nodes, f_xy_nodes);
    Z = reshape(z, n_eval, n_eval);

    if n_i == 20
        figure;
        surf(XX, YY, Z);
        xlabel("x");
        ylabel("y");
        title("Plot of regular tensor Lagrange interpolant for n = 20");
    end
```

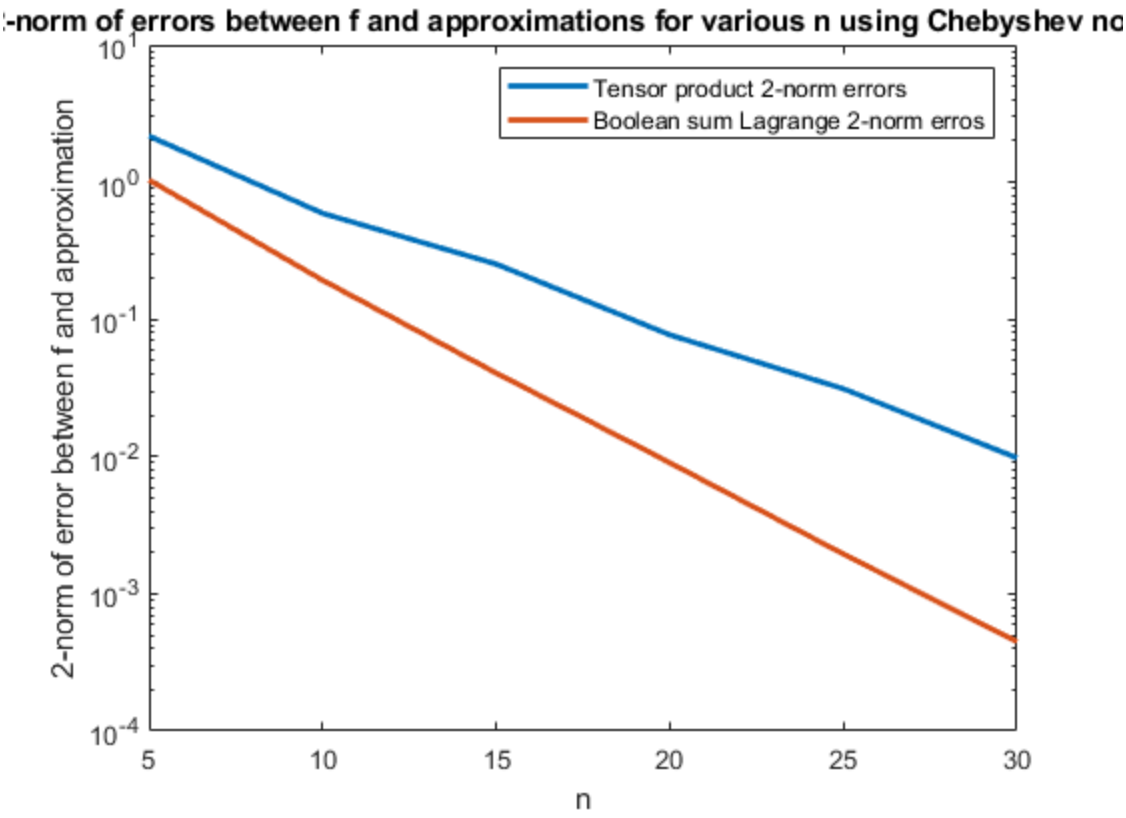
Boolean sum Lagrange interpolant

```
    s_x = zeros(length(xx_eval), 1);
    for k = 1:n_i
```

```
        s_x = s_x + f(x_nodes(k), yy_eval').*eval_lagrange(xx_eval', x_nodes,
 k);
    end

    s_y = zeros(length(yy_eval), 1);
    for k = 1:n_i
        s_y = s_y + f(xx_eval', y_nodes(k)).*eval_lagrange(yy_eval', y_nodes,
 k);
    end

    s = reshape(s_x + s_y, n_eval, n_eval) - Z;

    if n_i == 20
        figure;
        surf(XX, YY, s);
        xlabel("x");
        ylabel("y");
        title("Plot of Boolean sum Lagrange interpolant for n = 20");
    end

    % Display the error (in the 2-norm) between the interpolant and actual
    % function values
    f_xy_eval = f(xx_eval, yy_eval);
    % equispaced_nodes_errs(1, i) = norm(f_xy_eval - z');
    % equispaced_nodes_errs(2, i) = norm(f_xy_eval - reshape(s, 1, n_eval^2));
    chebyshev_nodes_errs(1, i) = norm(f_xy_eval - z');
    chebyshev_nodes_errs(2, i) = norm(f_xy_eval - reshape(s, 1, n_eval^2));
end

figure;
semilogy(n, chebyshev_nodes_errs(1, :), "LineWidth", 2, "DisplayName", "Tensor
 product 2-norm errors");
hold on;
semilogy(n, chebyshev_nodes_errs(2, :), "LineWidth",
 2, "DisplayName", "Boolean sum Lagrange 2-norm erros");
xlabel("n");
ylabel("2-norm of error between f and approximation");
% title("2-norm of errors between f and approximations for various n using
 equispaced nodes");
title("2-norm of errors between f and approximations for various n using
 Chebyshev nodes");
legend;

function val = eval_lagrange(x, x_nodes, j)
    val = ones(length(x), 1);
    for i = 1:length(x_nodes)
        if i ~= j
            val = val .* (x-x_nodes(i))/(x_nodes(j)-x_nodes(i));
        end
    end
end
```


Plot of regular tensor Lagrange interpolant for n = 20
0.15
0.1
0.05
0
-0.05
-0.1
-0.15
1
0.5
0
-0.5
-1
y
-1
-0.5
0
0.5
1
x


Plot of Boolean sum Lagrange interpolant for n = 20
0.15
0.1
0.05
0
-0.05
-0.1
-0.15
1
0.5
0
-0.5
-1
y
-1
-0.5
0
0.5
1
x



Published with MATLAB® R2021b

```
function L=lagrange1D(x,pointx)
n=size(pointx,2);
L2=ones(n,size(x,2));
   for i=1:n
      for j=1:n
         if (i~=j)
            L2(i,:)=L2(i,:).*(x-pointx(j))/(pointx(i)-pointx(j));
         end
      end
   end
   L = L2.';
end
```

```
Not enough input arguments.

Error in lagrange1D (line 2)
n=size(pointx,2);
```

Published with MATLAB® R2021b

```
function z = tensorproduct2D_lagrange(xy, pointx, pointy, pointz)

x = xy(1,:);
y = xy(2,:);
nxy = size(xy,2);

% create the 1D interpolation matrices
Lx = lagrange1D(x,pointx);
Ly = lagrange1D(y,pointy);
nxi = length(pointx);
nyi = length(pointy);

L = ones(nxy,nxi*nyi);

for i = 1:nxi
    L(:,(i-1)*nyi+(1:nyi)) = (Lx(:,i)*ones(1,nyi)).*Ly;
end

z = L*pointz';
end
```

```
Not enough input arguments.

Error in tensorproduct2D_lagrange (line 3)
x = xy(1,:);
```

Published with MATLAB® R2021b