Diagram, text, letter

Description automatically generated

Diagram, text, letter

Description automatically generated

The original Lagrange’s interpolation polynomial is given as  


The Barycentric form is given above. We will substitute phi(x) and wi into the formula as follows:



=> 

=> 

=> , equal to the original Lagrange’s interpolation polynomial and value at x of the polynomial p can be written as the formula above.

Diagram, text, letter

Description automatically generated

Lagrange’s interpolation polynomial



=> 

and we have => 

We can see that  is independent of i and can be treated as a constant, so we can put it outside of the sum as follows:

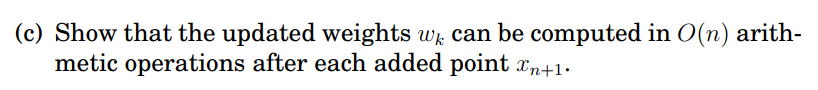
=>

Now we can try adding interpolation with the constant function 1 besides the original data. In other words, the interpolant of the constant function is itself: p(x) = yi => Both sides can be divided by p(x)

=> 

=> 

This is the derivation of the barycentric formula of the Lagrange’s interpolation polynomial



The formula of wk is given as: 

For each added point , the weights wk can be updated in a for-loop of range n



for k in range[0, n]:





end

=> The updated weights wk can be computed in O(n) arithmetic operations after adding new point

Text

Description automatically generated

The Matlab code for lagweights.m is  
% Compute the weights of the barycentric formula  
function [w] = lagweights(x)  
 w = ones([1,length(x)]);  
 for k = 1:length(x)  
 for j = 1:length(x)  
 if k ~= j  
 w(k) = w(k) \* 1/(x(k) - x(j));  
 end   
 end   
 end  
end  
The Matlab code for lagweights\_test.m is  
clc;  
w = lagweights([1,2,3]);  
disp(w)

Graphical user interface, text, application, chat or text message

Description automatically generated

Text

Description automatically generated

The Matlab code for specialsum.m is  
%{

x and z are arrays of size n and t is an array of size s. The

output has to be an array of size s. That is, t has the values where

the interpolation polynomial is evaluated

%}

function [sum] = specialsum(x,z,t)

sum = zeros([1,length(t)]);

for i = 1:length(x)

sum = sum + z(i)./(t - x(i));

end

end  
The Matlab code for specialsum\_test.m is  
clc;

S = specialsum((0:3),(1.5:0.5:3), -4:0);

disp(S)

Graphical user interface, text, application

Description automatically generated

Text

Description automatically generated

The Matlab code for lagpolint.m is  
% computes the barycentric form of p at points t

function [P] = lagpolint(X, T, fun)

Y=fun(X);

W=lagweights(X);

S1=specialsum(X, W.\*Y, T);

S2=specialsum(X, W, T);

P=S1./S2;

plot(X,Y,'.b',T,P,'r')

end

Text

Description automatically generated

The Matlab code for lagpolint\_test.m is  
%{ Test lagpolint.m by sampling from the function y = sqrt(|t|) on [?1; 1]. Try first 9 uniform points and then 101 Chebyshev points xj = -cos(j\*pi/n); j = 0; 1;...; 100 := n: Plot the polynomials. %}

clc;

fun = @(t)sqrt(abs(t)); % y = sqrt(|t|)

T=-1:0.002:1;

% 9 Chebyshev points

figure(1)

j=0:9;

X = -cos(j.\*pi/9);

P = lagpolint(X, T, fun); disp(P)

% 101 Chebyshev points

figure(2)

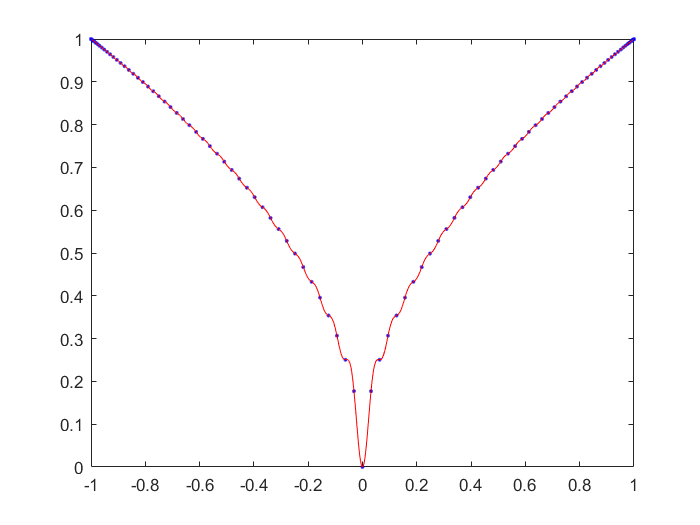
j=0:100;

X = -cos(j.\*pi/100);

P = lagpolint(X, T, fun); disp(P)

- Plotting the polynomial   
9 uniform Chebyshev points

Chart, line chart

Description automatically generated  
100 uniform Chebyshev points  


First values of interpolation polynomial P

A picture containing application

Description automatically generated

Text, letter

Description automatically generated

Text, letter

Description automatically generated

  
We have the following identities:  






=> 

=> =>



If P(x,y) is the interpolation polynomial, if we insert  into P, we should get

Analysis of

1. When x = xp, y = yq => 

2. When x = xj, j != p and y = yi, i can be anything in [0, n] => 

- In the identity , we know that j != p, and also u != p => There exists u such that u = j. In other words, there exists xj – xu = 0. A product of xj -xu thus will then be reduced to 0

=> 

3. When x = xi, i can be anything in [0, m] and y = yk, k != q => 

- In the identity , we know that k != q, and also v != q => There exists v such that v = k. In other words, there exists yk – yv = 0. A product of yk - yv thus will then be reduced to 0

=> 

Now we plut  into P(x,y)



=> 

Since we insert  into P(x,y) and we get, P(x,y) or P(s,t) is the interpolation polynomial

Text, letter

Description automatically generated

It is known in part (a) that



and 

First, we need to simplify the expression



P(x,y) can be derived as follows

 Now we can try adding interpolation with the constant function 1 besides the original data. The interpolant of the constant function is itself: p(x,y) = z\_xy => Both sides can be divided by p(x,y)

=> 

=>

=> 

Plugging back into P(x,y), we have the identity

(proven)

This is a generalization of the Lagrange’s barycentric formula to the 3-dimensional case

Text

Description automatically generated

The Matlab code for interpolsurf.m is

%{

Write a program interpolsurf.m such that given the grid

points as x and y, and the sampled values z = (f(xi; yj)) of R(m+1)x(n+1), computes the values of the polynomial P(s,t).

Test with f(s,t) = sin(s + t), m = 3, x a uniform partition of   
[0; pi],and n = 7, y a uniform partition of [0; 2pi]. Plot P(s,t), f(s,t), and the difference f - P

The interpolation grid is given as U x V, where size(U) = 51 is uniform partition on [0; pi] and size(V) = 101 is uniform partition on [0; 2pi]

%}

clc;

f = @(X,Y) sin(X + Y);

m = 3; n = 7;

X = (0:1/m:1) \* pi;

Y = (0:1/n:1) \* 2 \* pi;

[XX,YY] = meshgrid(X,Y);

ZZ = f(XX,YY);

U = (0:0.02:1) \* pi;

V = (0:0.01:1) \* 2 \* pi;

W1 = lagweights(X);

W2 = lagweights(Y);

% numerator

Suv = zeros([length(V) length(U)]);

for i = 1:length(V)

for j = 1:length(U)

for q = 1:length(Y)

for p = 1:length(X)

Suv(i,j) = Suv(i,j) + (W1(p) \* W2(q) \* ZZ(q, p))/((U(j) - X(p))\*(V(i) - Y(q)));

end

end

end

end

% denominator

SX = specialsum(X, W1, U);

SY = specialsum(Y, W2, V);

[SSX,SSY] = meshgrid(SX,SY);

Pst = Suv./(SSX .\* SSY);

[UU,VV] = meshgrid(U,V);

% Plotting f(UU,VV) or f(s,t)

fst = f(UU,VV);

figure(1)

surf(UU,VV,fst)

title("The function f(s,t)")

figure(2)

surf(UU,VV,Pst)

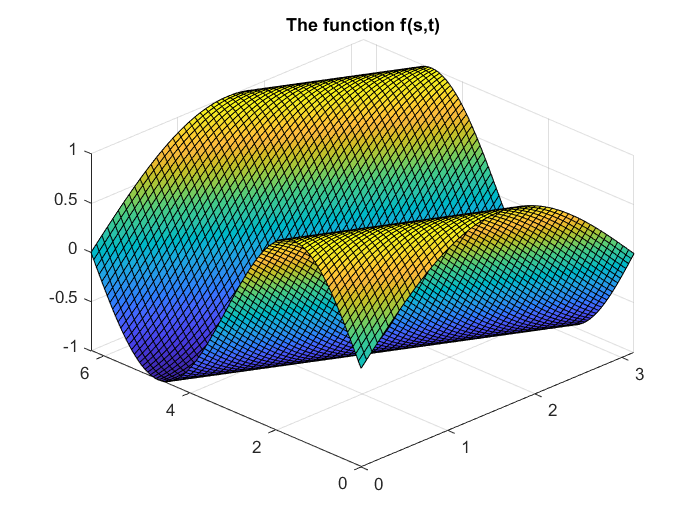
title("The interpolation P(s,t)")

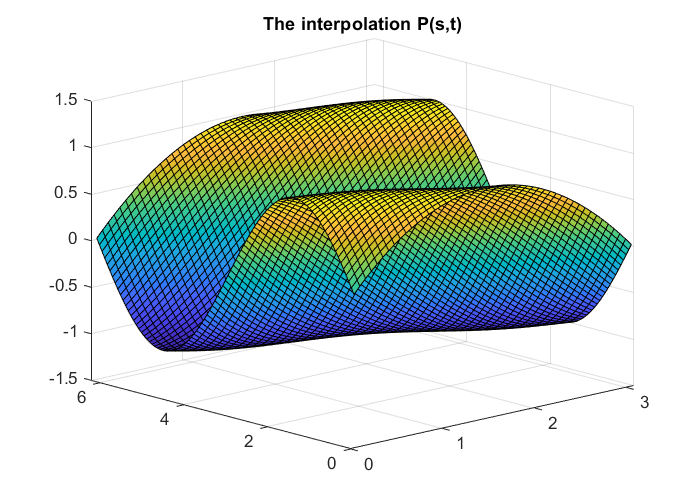
figure(3)

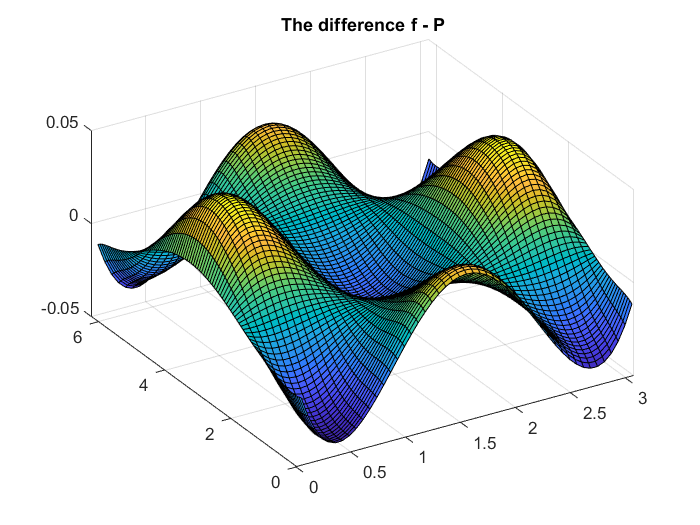
surf(UU,VV, Sf - Pst)

title("The difference f - P")

The original function f(s,t) = sin(s,t) on U and V



The interpolation function P(s,t) = sin(s,t) on U and V after using data from X and Y

The difference between the original function f(s,t) and the interpolation function P(s,t)   


We can observe that the errors is quite significant, varying from near -0.05 to near 0.05. To reduce the errors, we can add more datapoints as X,Y to increase the accuracy of the interpolation surface.