```
%
 ============================================================================
% AUTHOR ..... [Lishan Huang]
% UPDATED .... [Jan 23]
% Task 3
```

lagrange interpolation formula

slagrange.m

```
%
 ============================================================================
% AUTHOR ..... [Lishan Huang]
% UPDATED .... [Jan 23]
%
% Evaluate the Lagrange interpolation formula
%
% INPUT
% tau .... The vector of interpolation nodes (length n)
% rho .... The vector of values at the interpolation nodes (length n)
% x   .... A vector of values to evaluate the interpolating polynomial
 at (length 1 to many (probably not n!))
%
% OUTPUT
% T :
%
 ============================================================================
function F = lagrange(tau, rho, x)
%initialize F=0
t=tau;
p=rho;
F=0;
%create a for loop
    for k=1:length(t)
        %make L=1 before each loop where l means Lk in the function
        L=1;
        for i=1:length(t)
            if i~=k
                L=L.*(x-t(i))./(t(k)-t(i));
            end
        end
        F=F+p(k).*L;
    end
end
```

The first form of the barycentric formula

firstbaryeval.m

```
%
 ==========================================================================
% AUTHOR ..... [Lishan Huang]
% UPDATED .... [Jan 23]
%
%  Evaluate the first form of the barycentric formula
%
% INPUT
% tau .... The vector of interpolation nodes (length n)
% rho .... The vector of values at the interpolation nodes (length n)
% x   .... A vector of values to evaluate the interpolating polynomial
 at (length 1 to many (probably not n!))
%
% OUTPUT
% T :
%
 ==========================================================================

function T = firstbaryeval(tau, rho, x)
%initialize t p T and a
    t=tau;
    p=rho;
    T=0;
    a=1;
    %calculate w(x) and store in a
        for k=1:length(t)
            a=a.*(x-t(k));
        end
    %create for loop to add up the function
        for k=1:length(t)
            b=1;
            %for loop for b(k)
            for j=1:length(t)
               if j~=k
                   b=b.*((t(k)-t(j))).^-1;
               end
            end
            T=T+b.*p(k)./(x-t(k));
        end
    T=T.*a;
end
```

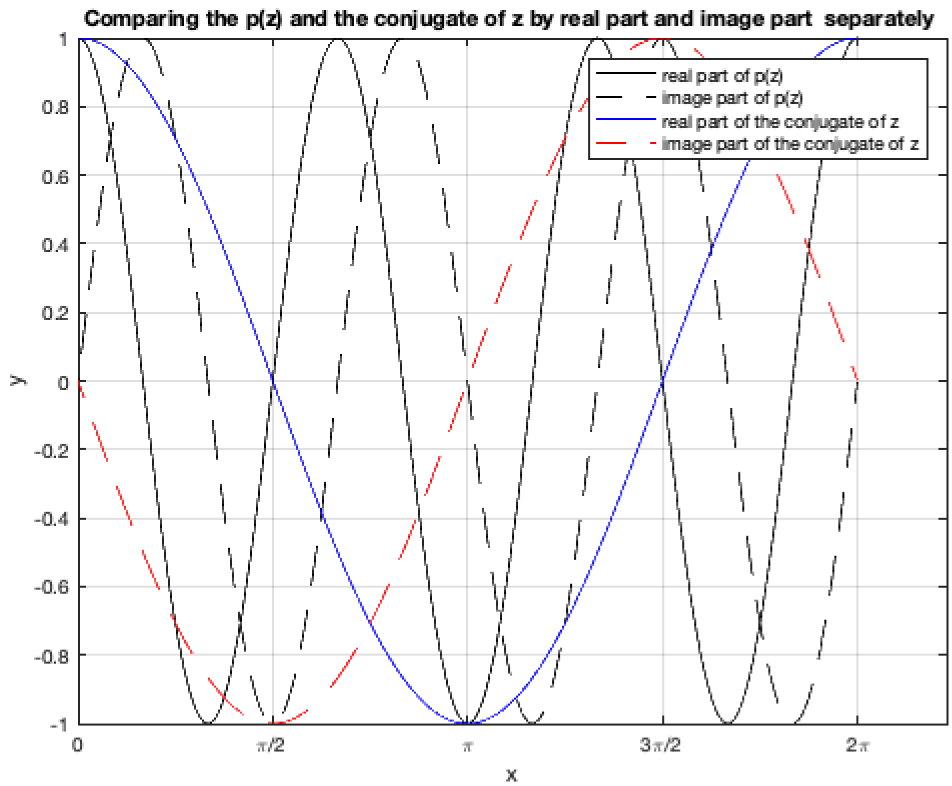
Script

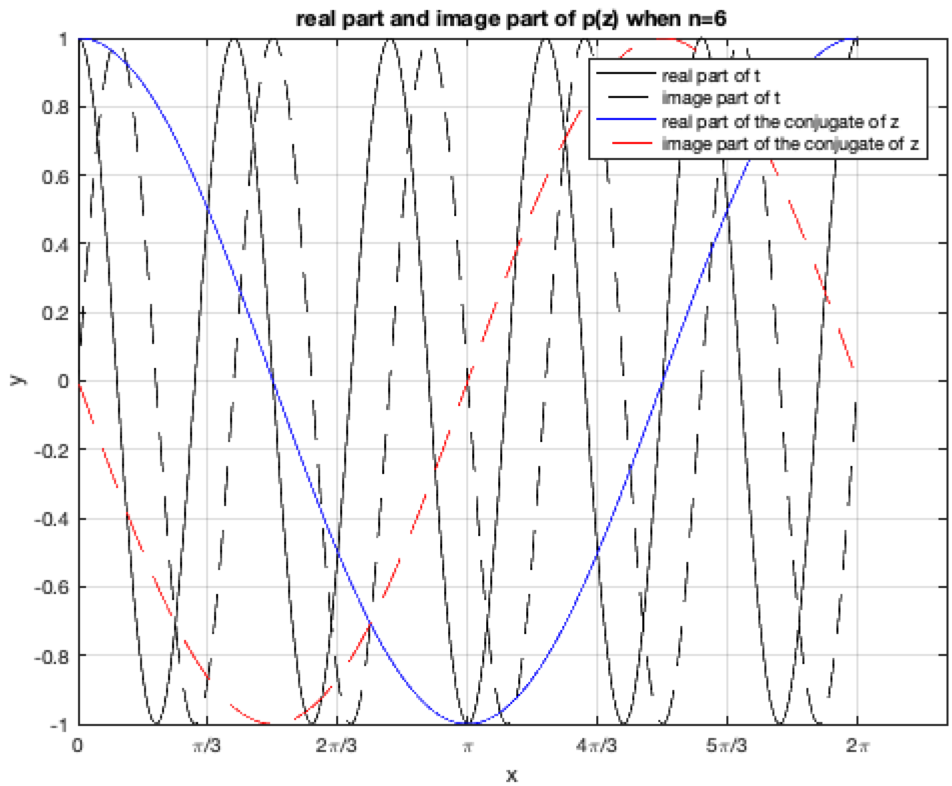
```
%The following is a code that using firstbaryeval function to
 interpolate
%complex numbers and plot the p(z) and the conjugate of z by real part
%and image part  separately
```

```
%create a vector with 2019 equal points between 0 and 2 pi
num1=linspace(0,2*pi,2019);
%create a vector with 5 equal points between 0 and 2 pi
num2=linspace(0,2*pi,5);
%define the function of z
z = @(x) cos(x)+1i*sin(x);
%define the function that inverse the image part of input
f = @(x) real(x) - 1i*imag(x);
%define tau and rho for firstbaryeval
tau=[1,1i,-1,-1i];
rho=[1,-1i,-1,1i];
%store the result of firstbaryeval in x
x=firstbaryeval(tau,rho,z(num1));
%plot the the real part of p(z)  versus ? in solid black and the
 imaginary
%part of p(z) versus ? in dashed balck
plot(num1,real(x),'-k',num1,imag(x),'--k');
grid on
xticks([0 pi/2 pi 3*pi/2 2*pi])
xticklabels({'0','\pi/2','\pi','3\pi/2','2\pi'})
%keep on this graph plot other line
hold on;
%plot the real part and imaginary part of the conjugate of z
plot(num1,real(f(z(num1))),'b',num1,imag(f(z(num1))),'--r')
%label axies
xlabel('x')
ylabel('y')
title('Comparing the p(z) and the conjugate of z by real part and
 image part  separately ')
%add the legend to the graph
legend('real part of p(z)','image part of p(z)','real part of the
 conjugate of z','image part of the conjugate of z')
%create a vector with 6 equal points between 0 and 5 (n-1 where n=6)
a=linspace(0,5,6);
%define the function
tau=exp(2*pi*1i*(a)/6);
%define tau and rho
rho=(f(tau));
%store result of firstbaryeval in t
t=firstbaryeval(tau,rho,z(num1));
figure(2)
%%plot the the real part of t  and the imaginary part of t
plot(num1,real(t),'-k',num1,imag(t),'--k');
grid on
xticks([0 pi/3 2*pi/3 pi 4*pi/3 5*pi/3 2*pi])
xticklabels({'0','\pi/3','2\pi/3','\pi','4\pi/3','5\pi/3','2\pi'})
title('real part and image part of p(z) when n=6')
xlabel('x')
ylabel('y')

hold on;
%plot the real part and imaginary part of the conjugate of z
plot(num1,real(f(z(num1))),'b',num1,imag(f(z(num1))),'--r')
```

```
legend('real part of t','image part of t','real part of the conjugate
 of z','image part of the conjugate of z')
```






```
% *Text Answers*
%  Do these curves agree with the polynomial interpolant at the points
 pi = 0
%, pi/2, pi, and 3pi/2? Do they agree only at those points?
%
%  Answer: Yes it agree with the polynomial interpolant at those
 points, and they agree at those points only, since intersection of
 corresponding point only occur in those points

%. What happens if you re-run your script with a higher number of
 points, say n = 6, and interpolate z
%at the points zk = exp(2pi*ik/n) for k = 0, 1, . . ., n - 1?
%
%  Answer: Since e^(ix)=cos(x)+i*sin(x), then e^(2*pi*k*i/
n)=cos(2*pi*k/ n)+i*sin(2*pi*k/ n) when k=0,1,2,3,4,5
 and n=6 ,then when k=1 :e^(2*pi*0*i/6)=cos(0)+i*sin(0)
 k=2 :e^(2*pi*1*i/6)=cos(2*pi*1/ 6)+i*sin(2*pi*1/ 6)=
 cos(pi /3)+i*sin(2*pi/ 6), it is obviously that this function is same
 as the previous z=cos(x)+i*sin(x) where x =linspace(0,2pi,6), and all
 of those interpolation points are intersect with f(z), but the curves
 are not intersection in those point only. In addition, the period of
 the interpolation curve is changed and it equals to 5 = n-1
```

Published with MATLAB® R2017b