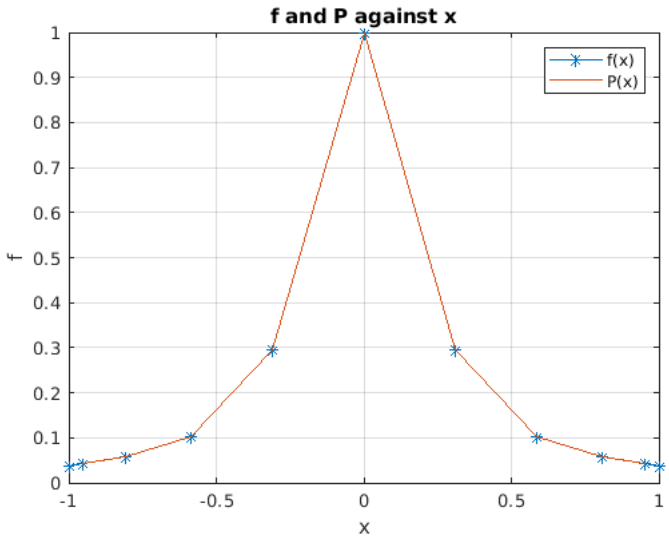
```
% The code approximates the function f(x) with a 10th polynomial.

clear all
close all

N = 10; h = 2/N; m = 300;

%function
f1 = @(x) 1./(1+25*x.^2);

x = zeros(1,N+1);
f = zeros(1,N+1);
for k = 0:N
   x(k+1) = -cos(k*pi/10);
   f(k+1) = f1(x(k+1));
end

xk = [];
PN = [];
for k = 1:m+1
     xk1 = -cos(k*pi/10); xk = [xk,xk1];
     pN = Barycentric(x,f,xk(k),N);
     PN = [PN,pN];
end

plot(x,f1(x),'-*'); grid on;
title('f and P against x');
xlabel('x');ylabel('f');
hold on
plot(xk,PN);
legend('f(x)','P(x)');

%compare
fprintf('Using the Chebyshev nodes its clear that the intepolant curve fits the data well, hence\n Chebyshev nodes approximate better than the ec

function pN = Barycentric(x,f,xx,N)

%weights
w = zeros(N+1,1);

numer = 0; %Numerator
demon = 0; %Denominator
for j = 1:N+1
    temp = 1;
    for k = 1:N+1
        if (k ~= j)
            temp = temp*(x(j) - x(k));
        end
    end
    w(j) = 1/temp;

    xdiff = xx-x(j);
    temp1 = w(j)./(xdiff);

    numer = numer + temp1*f(j);
    demon = demon + temp1;
end
pN = numer./demon;
end
```

```
Using the Chebyshev nodes its clear that the intepolant curve fits the data well, hence
 Chebyshev nodes approximate better than the equispaced points
```



Published with MATLAB® R2020a