ICE for interpolation

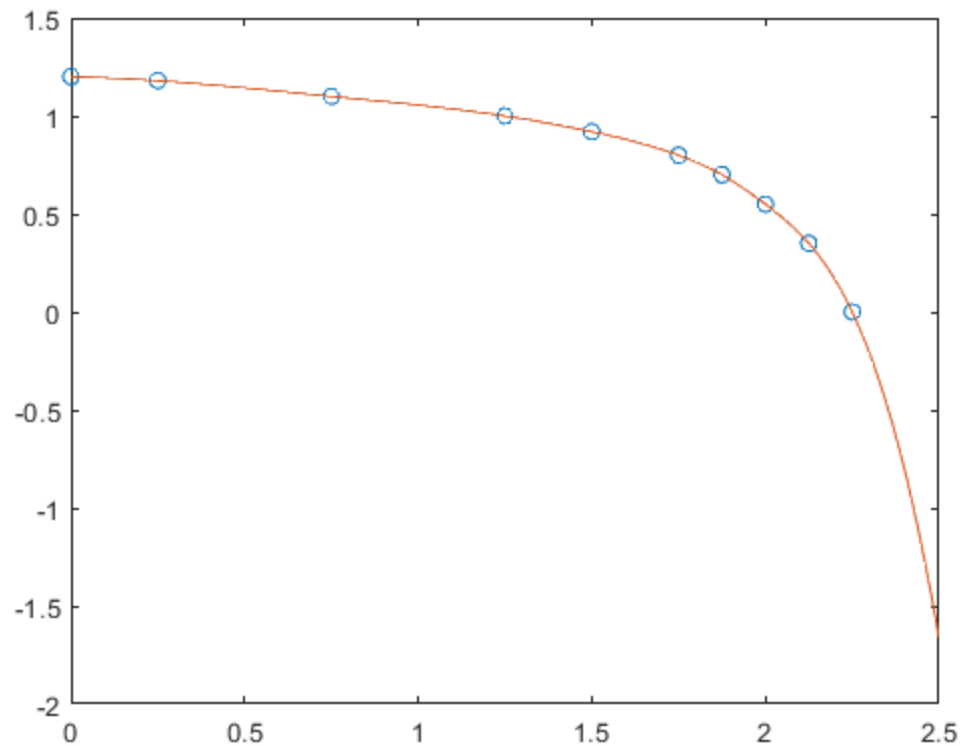## Table of Contents

Liu Leqi, 12011327

Problem 1

```
clear all; clc;
% given values
x = [0 0.25 0.75 1.25 1.5 1.75 1.875 2 2.125 2.25];
y = [1.2 1.18 1.1 1 0.92 0.8 0.7 0.55 0.35 0];
% plot the graph of (x,y)
plot(x,y,'o');
hold on;
% interpolation
xvals = 0:0.01:2.5; % interpolation points
yvals = interp1(x,y,xvals,'spline');
plot(xvals, yvals);
hold off;
```

Problem 2

```matlab
clear all; clc;
% given values
T = 0:8:40;
o = [14.621 11.843 9.870 8.418 7.305 6.413];
% (a) Linear interpolation
y1 = interp1(T,o,27,'linear');
fprintf("linear interpolation: %.2f \n", y1);
% (b) Newton's interpolating polynomial
y2 = NewtonInterp(T,o,27); % self-defined function, defined at the
 tail
fprintf("Newton interpolating polynomial: %.2f \n", y2);
% (c) Cubic splines
y3 = interp1(T,o,27,'spline');
fprintf("cubic spline: %.2f \n", y3);

linear interpolation: 8.00
```

Problem 3

```matlab
clear all; clc;
% given values
t = 0:pi/4:2*pi;
f = sin(t).^2;
```

```matlab
x = 0:0.1:2*pi; % interpolation points
% (a) seventh-order interpolating polynomial
y1 = NewtonInterp(t,f,x); % self-defined function, defined at the tail
plot(t,f,'o',x,y1);
hold on;
% (b) cubic spline
y2 = spline(t,f,x);
plot(x,y2);
legend("origin points","seven-order","cubic spline");
hold off;
```

function for Newton interpolation

X,Y are vectors of the coordinates of input points; x is vector of interpolation points;

```matlab
function y = NewtonInterp(X,Y,x)
n = length(X); m = length(x);
A = zeros(n,n); % matrix for difference quotient
A(:,1) = Y';
% calculate the difference quotients
for k = 1 : n-1
    for i = 1 : n - k
     A(i,k+1) = (A(i+1,k) - A(i,k))/(X(i+k) - X(i));
    end
end
% calculate the interpolation values at given points
y = zeros(1,m);
for i = 1 : n
    z = ones(1,m);
    for k = 1 : i-1
        z = z.*(x-X(k));
    end
    y = y + A(1,i)*z;
end
end
```

```
Newton interpolating polynomial: 7.97
cubic spline: 7.97
```

Published with MATLAB® R2021a