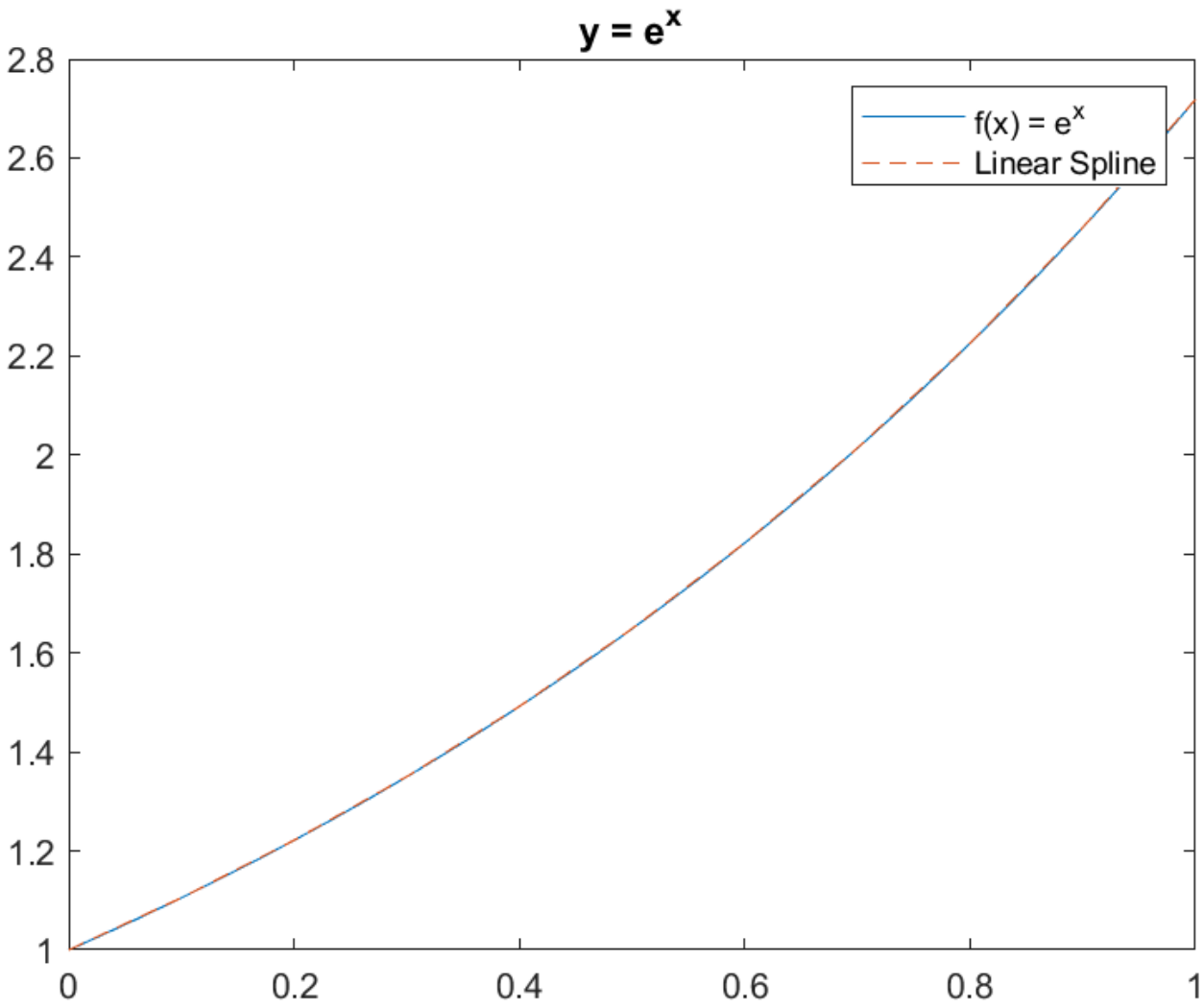
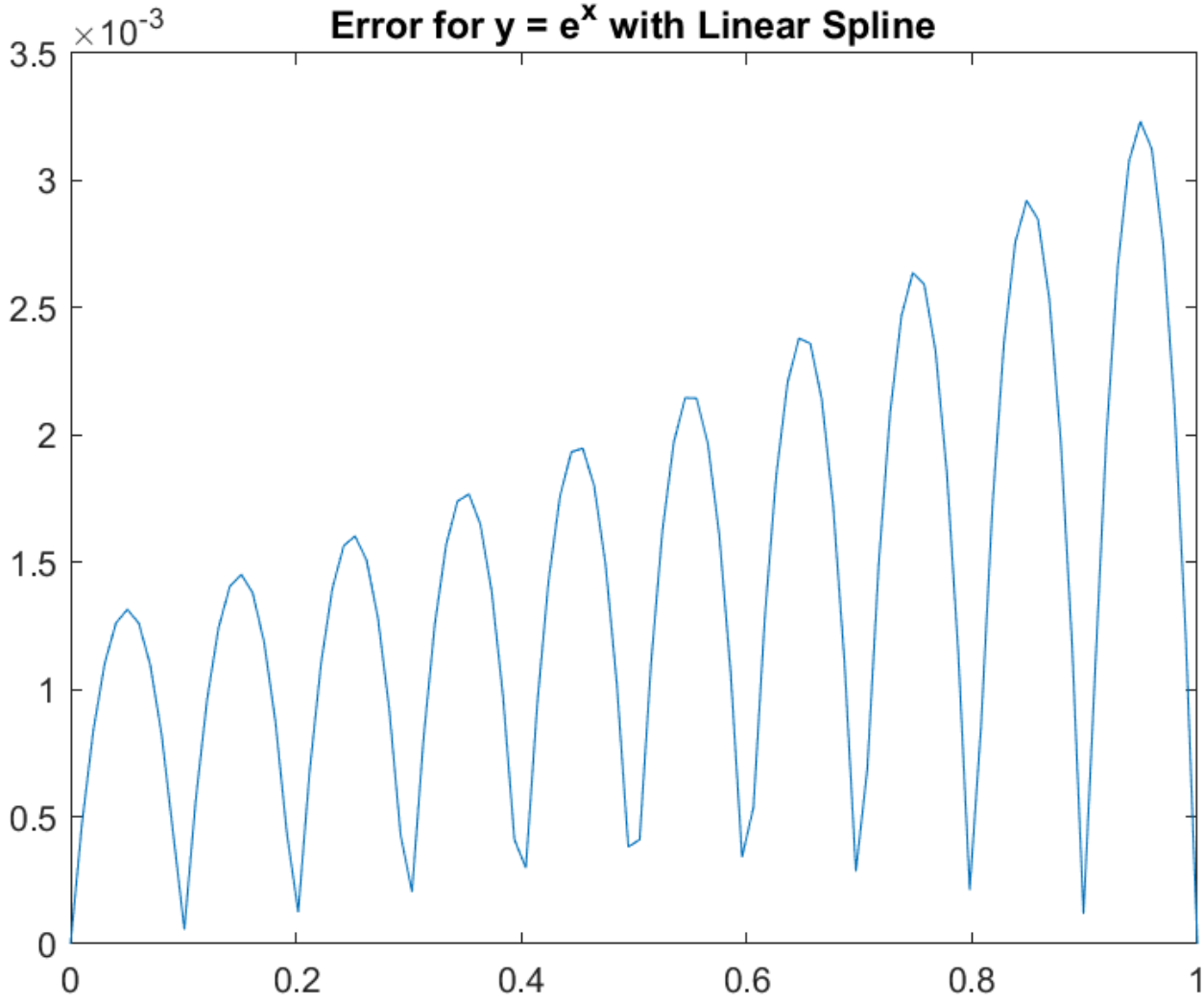
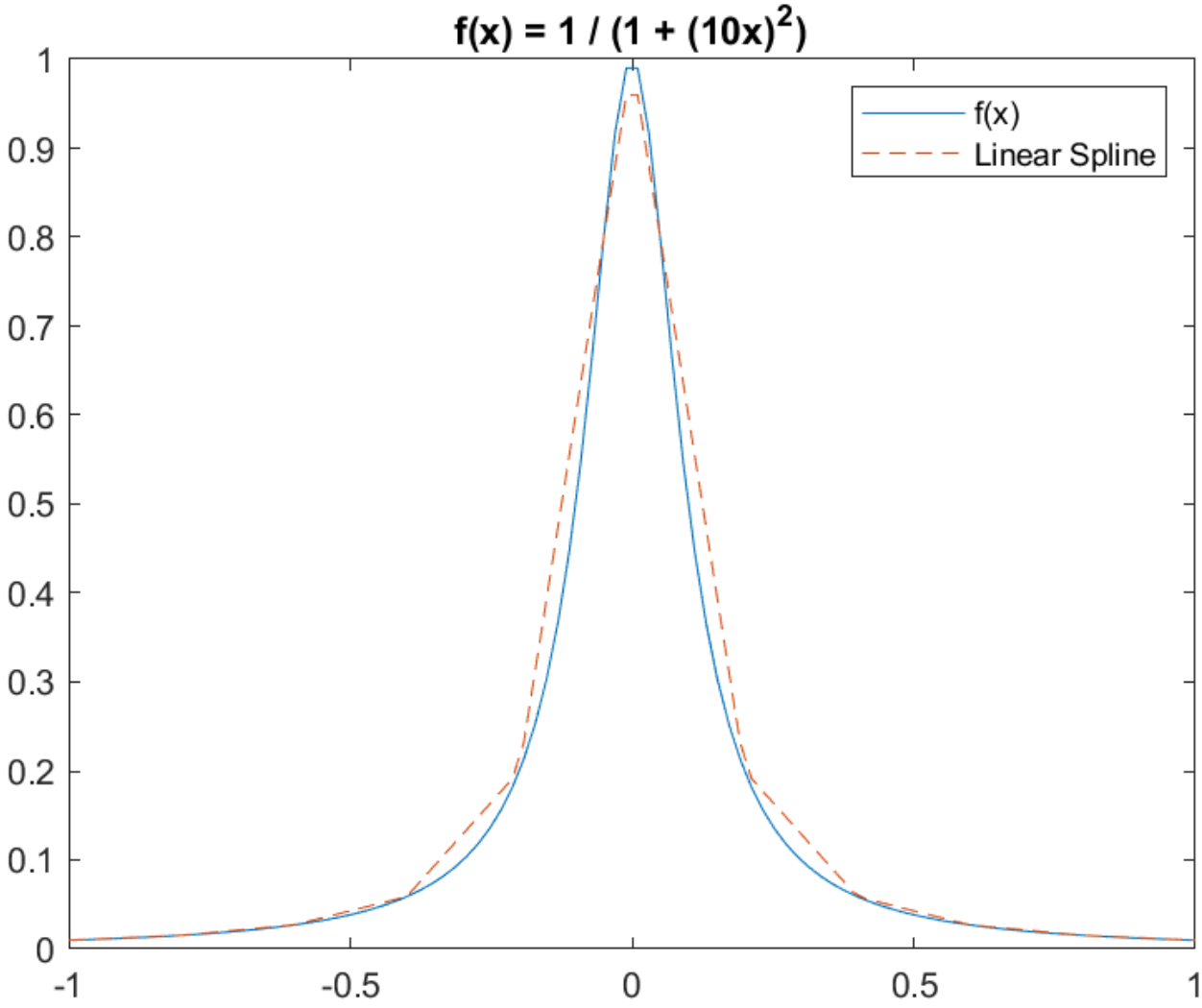
Lab 8

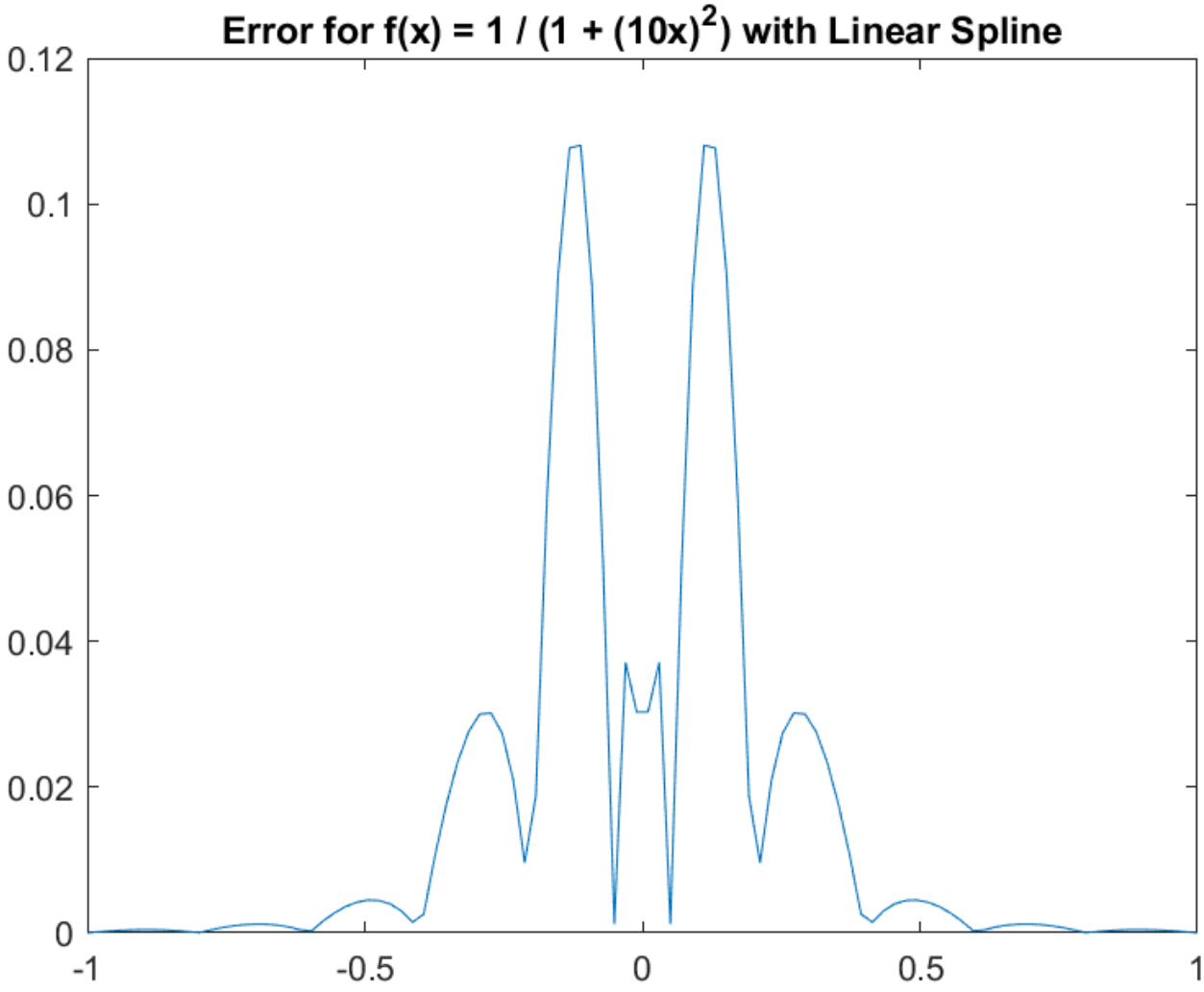
Ghulam Yahya Rajabi

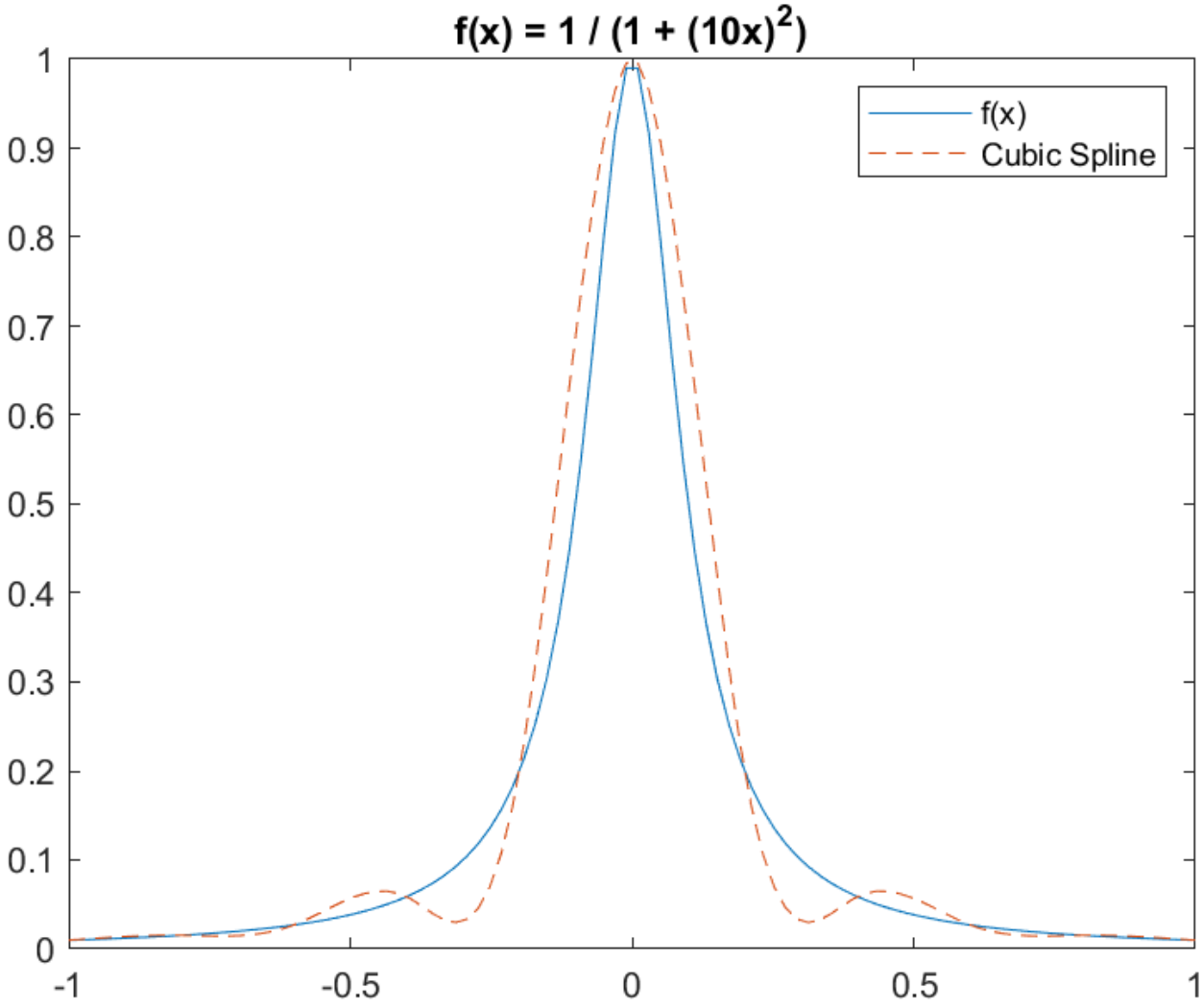
03/13/2024

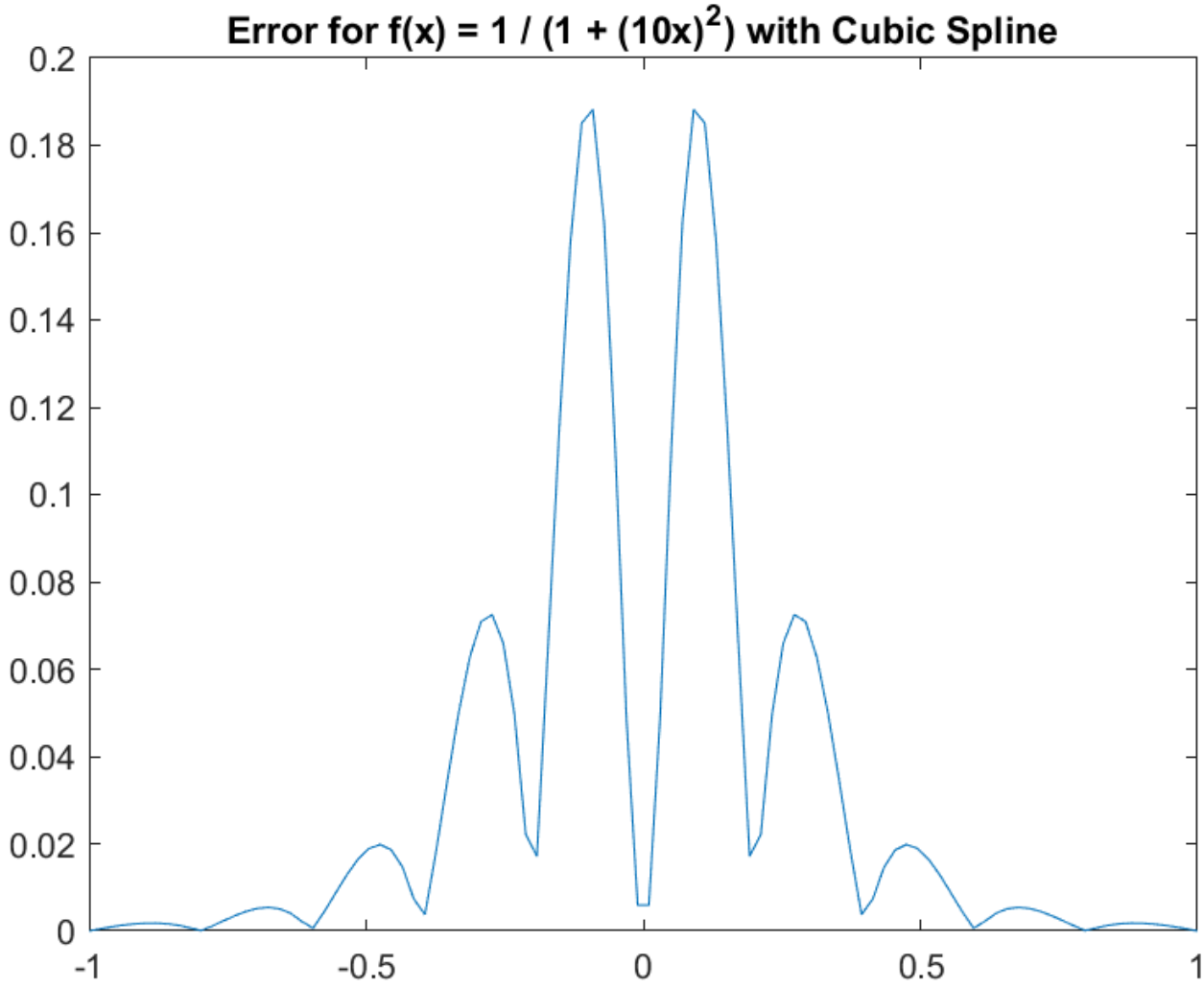



Error for y = e^x with Linear Spline
×10^-3
3.5
3
2.5
2
1.5
1
0.5
0
0
0.2
0.4
0.6
0.8
1


f(x) = 1 / (1 + (10x)^2)
f(x)
Linear Spline
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
-1
-0.5
0
0.5
1


Error for f(x) = 1 / (1 + (10x)^2) with Linear Spline
0.12
0.1
0.08
0.06
0.04
0.02
0
-1
-0.5
0
0.5
1


f(x) = 1 / (1 + (10x)^2)
f(x)
Cubic Spline
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
-1
-0.5
0
0.5
1



Appendix(I had a hard time doing this lab in python because I am still learning it so I did it in MATLAB, I hope you don't mind. I will upload a python version of it in github when I get it don**)**

```
clear
clearvars
clc
close all
func1 = @(x) exp(x);
start1 = 0;
end1 = 1;
numEval = 100;
xEval1 = linspace(start1, end1, numEval);
numIntervals1 = 10;
```

```
yEval1 = eval_linear_spline(xEval1, numEval, start1, end1, func1,
numIntervals1);
fEx1 = zeros(1, numEval);
for j1 = 1:numEval
   fEx1(j1) = func1(xEval1(j1));
end
figure();
plot(xEval1, fEx1);
hold on;
plot(xEval1, yEval1, 'LineStyle','--');
legend("f(x) = e^x", "Linear Spline");
title("y = e^x");
err1 = abs(yEval1 - fEx1);
figure();
plot(xEval1, err1);
title("Error for y = e^x with Linear Spline");
func2 = @(x) 1 ./ (1 + (10 .* x) .^ 2);
start2 = -1;
end2 = 1;
xEval2 = linspace(start2, end2, numEval);
y2Eval = eval_linear_spline(xEval2, numEval, start2, end2, func2,
numIntervals1);
gEx = zeros(1, numEval);
for j2 = 1:numEval
   gEx(j2) = func2(xEval2(j2));
end
figure();
plot(xEval2, gEx);
hold on;
plot(xEval2, y2Eval, 'LineStyle','--');
legend("f(x)", "Linear Spline");
title("f(x) = 1 / (1 + (10x)^2)");
err2 = abs(y2Eval - gEx);
figure();
plot(xEval2, err2);
title("Error for f(x) = 1 / (1 + (10x)^2) with Linear Spline");
xEval3 = linspace(start2, end2, numEval);
y3Eval = eval_cubic_spline(xEval3, numEval, start2, end2, func2,
numIntervals1);
figure();
plot(xEval3, gEx);
hold on;
plot(xEval3, y3Eval,'LineStyle','--');
legend("f(x)", "Cubic Spline");
title("f(x) = 1 / (1 + (10x)^2)")
err3 = abs(y3Eval - gEx);
figure();
plot(xEval3, err3);
title("Error for f(x) = 1 / (1 + (10x)^2) with Cubic Spline");
```

```
function yEval = eval_linear_spline(xEval, numEval, start, end_, func,
numIntervals)
   xIntervals = linspace(start, end_, numIntervals + 1);
   yEval = zeros(1, numEval);
   for jInt = 1:numIntervals
       ind = find(xEval >= xIntervals(jInt) & xEval <= xIntervals(jInt + 1));
       n = length(ind);
       a1 = xIntervals(jInt);
       fa1 = func(a1);
       b1 = xIntervals(jInt + 1);
       fb1 = func(b1);
       for kk = 1:n
           yEval(ind(kk)) = lineEval(a1, fa1, b1, fb1, xEval(ind(kk)));
       end
   end
end
function y = lineEval(x0, y0, x1, y1, x)
   m = (y1 - y0) / (x1 - x0);
   y = m * (x - x1) + y1;
end
function coeffList = cubicCoeff(start, end_, func, numIntervals)
   xIntervals = linspace(start, end_, numIntervals + 1);
   yIntervals = func(xIntervals);
   h = zeros(1, numIntervals);
   for i = 1:numIntervals
       h(i) = xIntervals(i + 1) - xIntervals(i);
   end
   alpha = zeros(1, numIntervals);
   for i = 2:numIntervals
       alpha(i) = (3 / h(i)) * (yIntervals(i + 1) - yIntervals(i)) - (3 / h(i -
1)) * (yIntervals(i) - yIntervals(i - 1));
   end
   l = zeros(1, numIntervals);
   l(1) = 0;
   mu = zeros(1, numIntervals);
   mu(1) = 0;
   z = zeros(1, numIntervals);
   z(1) = 0;
   for i = 2:numIntervals
       l(i) = 2 * (xIntervals(i + 1) - xIntervals(i - 1)) - (h(i - 1) * mu(i -
1));
       mu(i) = h(i) / l(i);
       z(i) = (alpha(i) - (h(i - 1) * z(i - 1))) / l(i);
   end
   c = zeros(1, numIntervals + 1);
   c(end) = 0;
   b = zeros(1, numIntervals);
   d = zeros(1, numIntervals);
   for j = numIntervals:-1:1
```

```
        c(j) = z(j) - (mu(j) * c(j + 1));
        b(j) = ((yIntervals(j + 1) - yIntervals(j)) / h(j)) - (h(j) * (c(j + 1)
+ 2 * c(j)) * (1 / 3));
        d(j) = (c(j + 1) - c(j)) / (3 * h(j));
    end
    coeffList = {yIntervals, b, c, d};
end
function yEval = eval_cubic_spline(xEval, numEval, start, end_, func,
numIntervals)
    coeffList = cubicCoeff(start, end_, func, numIntervals);
    yIntervals = coeffList{1};
    bList = coeffList{2};
    cList = coeffList{3};
    dList = coeffList{4};
    xIntervals = linspace(start, end_, numIntervals + 1);
    yEval = zeros(1, numEval);
    for jInt = 1:numIntervals
        ind = find(xEval >= xIntervals(jInt) & xEval <= xIntervals(jInt + 1));
        n = length(ind);
        for kk = 1:n
            yEval(ind(kk)) = cubicEval(yIntervals(jInt), bList(jInt),
cList(jInt), dList(jInt), xIntervals(jInt), xEval(ind(kk)));
        end
    end
end
function y = cubicEval(a, b, c, d, xj, x)
    y = a + b * (x - xj) + c * ((x - xj) .^ 2) + d * ((x - xj) .^ 3);
end
```