```matlab
% Anthony Galczak - Tristin Glunt
% agalczak@unm.edu - tglunt@unm.edu
% CS 375 - HW7

clc
close all

% Q1
f = @(x) 1 ./ (1 + 30 .* (x.^2));

% Get 11 equispaced points on [-1, 1] (x_s)
% Get 11 equispaced evaluations on [-1, 1] (y_s)
full_vals = linspace(-1, 1, 1001);
x_s = linspace(-1, 1, 11);
y_s = arrayfun(f,x_s);

full_ys = lagrange(full_vals, x_s, y_s);

full_real_ys = arrayfun(f, full_vals);

plot(full_vals, full_ys);
hold on
plot(full_vals, full_real_ys, 'LineWidth', 2);
plot(x_s, y_s, 'o');

title("Lagrange interpolation");
xlabel("x-axis");
ylabel("y-axis");
legend(["Approx","f(x)"],'Location','north');
hold off

% Q2b
n_s = [4, 8, 16, 32];
spline_list = {};

% Get all our splines for the n_s
for i = 1 : size(n_s, 2)
    n = n_s(i);
    x_s = linspace(-1,1,(n+1));
    y_s = arrayfun(f,x_s);

    z_s = spline3_coeff(x_s,y_s);

    spline_points = zeros(1,1001);

    for j = 1 : 1001
       spline_points(j) = eval_spline(full_vals(j), x_s, y_s, z_s);
    end

    spline_list{i} = spline_points;
end
```

```
% Plot all of our splines
figure
for i = 1 : size(n_s, 2)
    plot(full_vals, spline_list{i});
    hold on
end

plot(full_vals, full_real_ys, 'LineWidth', 2);
title("Natural Cubic Spline interpolation");
legend(["n=4","n=8","n=16","n=32","f(x)"]);
xlabel("x-axis");
ylabel("y-axis");
hold off

% Q2c
for i = 1:size(n_s, 2)
    error(i) = max(abs(full_real_ys - spline_list{i}));
end
figure
plot(n_s, error);
hold on
title("n's vs. Natural Cubic Spline interpolation error");
xlabel("n");
ylabel("y-axis");
legend(["error"]);
hold off

% Q3
x = 0.5;
h_s = [0.5, 0.1, 0.05, 0.025, 0.0125];
f_prime = @(x) (-60 * x) / ((1 + 30 * (x^2))^2);
actual_deriv = f_prime(x);

p_ks = [0];
cd_f = @(x,h) (f(x+h) - f(x-h)) / (2*h);

% Get errors
for i = 1 : size(h_s, 2)
    h = h_s(i);
    c_diffs(i) = cd_f(x,h);
    errors(i) = abs(actual_deriv - c_diffs(i));

    if i > 1
        numerator = log(errors(i) / errors(i-1));
        denominator = log(h_s(i) / h_s(i-1));
        p_ks(i) = numerator / denominator;
    end
end

% Make a table.
fprintf("Central Difference table\n");
T = table(h_s',c_diffs',errors',p_ks');
T.Properties.VariableNames = {'h' 'Approx' 'Error' 'pk_order'};
disp(T);
```

```
% Q3c - Richardson Extrapolation
p_ks = [0];

% Get errors
for i = 1 : size(h_s, 2)
    h = h_s(i);
    richard(i) = cd_f(x,h/2) + 1/3*(cd_f(x,h/2) - cd_f(x,h));
    errors(i) = abs(actual_deriv - richard(i));

    if i > 1
        numerator = log(errors(i) / errors(i-1));
        denominator = log(h_s(i) / h_s(i-1));
        p_ks(i) = numerator / denominator;
    end
end

% Make a table.
fprintf("Richardson Extrapolation table\n");
T = table(h_s',richard',errors',p_ks');
T.Properties.VariableNames = {'h' 'Approx' 'Error' 'pk_order'};
disp(T);

% Q4c
n_s = [10,20,40,80,160,320];

% The below is also a numerical integration... TODO: Ask?
actual_integral = integral(f, -1, 1);

for i = 1 : size(n_s, 2)
    n = n_s(i);
    trap_ints(i) =  sum(trap_int(f, -1, 1, n));
    errors(i) = abs(actual_integral - trap_ints(i));

    if i > 1
        numerator = log(errors(i) / errors(i-1));
        denominator = log(n_s(i) / n_s(i-1));
        p_ks(i) = numerator / denominator;
    end
end

fprintf("Composite Trapezoidal Integration table\n");
T = table(n_s',trap_ints',errors',p_ks');
T.Properties.VariableNames = {'n' 'Approx' 'Error' 'pk_order'};
disp(T);

% Lagrange for Q1.

% https://www.mathworks.com/matlabcentral/fileexchange/899-lagrange-
polynomial-interpolation
% Comments given to explain each step. Inspiration from Lecture 14,
 slide 4
function y0 = lagrange(x0, x, y)
```

```
n = length(x);
L = ones(n, length(x0));

% Calculates the coefficients.
% This loop corresponds to the product loop l_k
for i = 1:n
    for j = 1:n
        % If i equals j then don't put that difference
        % into the lagrange form.
        if (i ~= j)
            % Calculate the lagrange form. l_k
            L(i,:) = L(i,:).*(x0-x(j))/(x(i)-x(j));
        end
    end
end

% Multiply our coefficients from the Lagrange form by our
% evaluated x's. In this case it is our Chebyshev points.
% This returns our 1000 points interpolated by lagrange
% with x,y inputs.
% The below loop corresponds to p(x) = sum(l_k * y_k)
y0= 0;
for i = 1:n
    y0 = y0+y(i)*L(i,:);
end

end

function z = spline3_coeff(t,y)

    n = size(t,2) - 1; % n + 1
    u_s = zeros(1, (n+1));
    v_s = zeros(1, (n+1));
    u_s(1) = 1;
    u_s(n+1) = 1;
    v_s(1) = 0;
    v_s(n+1) = 0;

    % Loop over and get our u_s, v_s, etc.
    for i = 1 : n
        h = t(i+1) - t(i);
        h_s(i) = h;
        b = 1/h * (y(i+1) - y(i));
        b_s(i) = b;

        % We manually entered u and v for i == 1...
        if i ~= 1
            u_s(i) = 2 * (h + h_s(i-1));
            v_s(i) = 6 * (b - b_s(i-1));
        end

    end

    A = zeros((n+1),(n+1));
```

```
    temp_h_s1 = [h_s(1:(n-1)), 0];
    temp_h_s2 = [0, h_s(2:n)];
    A = diag(temp_h_s1, -1);
    A = A + diag(u_s, 0);
    A = A + diag(temp_h_s2, 1);

    z = A\v_s';

end

% Q2a,b function
function y_eval = eval_spline(x,t,y,z)

%Inputs:
% x: Scalar where spline is to be eval
% t: Vector of x-values of the data points
% y: Vector of y-values of the data points
% z: Vector of spline coefficients

%Outputs
%y_eval: scalar value giving S(x)

num_pts = length(y);

sp_index = num_pts-1;

for i = num_pts-1:-1:1
    if x >= t(i)
        sp_index = i;
        break;
    end
end
h = t(sp_index+1) - t(sp_index);
tmp = z(sp_index)/2 + (x - t(sp_index))*(z(sp_index+1) - z(sp_index))/
(6*h);
tmp = -(h/6)*(z(sp_index+1) + 2*z(sp_index)) + (y(sp_index+1) -
 y(sp_index))/h + (x-t(sp_index))*tmp;
y_eval = y(sp_index) + (x-t(sp_index))*tmp;

end

% Q4b function
function [sum] = trap_int(f,a,b,n)

    x_s = linspace(a,b,n+1);

    for i = 1 : n
       sum(i) =  1/2 * (x_s(i+1) - x_s(i)) * (f(x_s(i+1)) +
 f(x_s(i)));
    end
end
```

Central Difference table

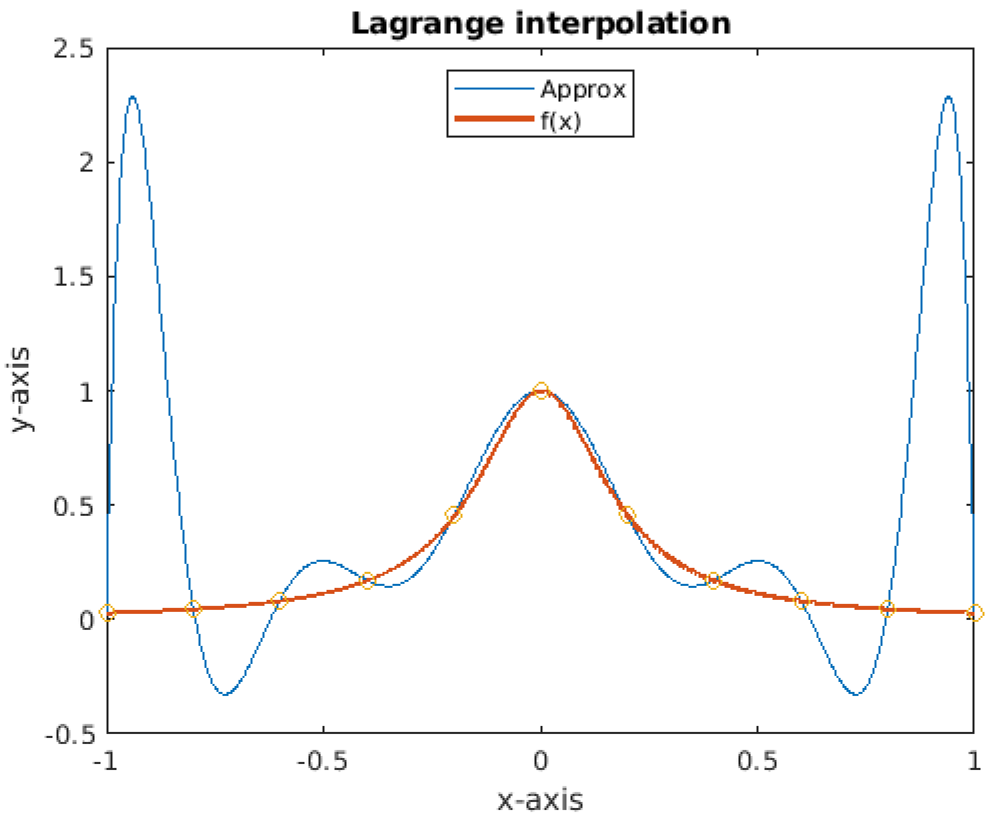
h	Approx	Error	pk_order
0.5	-0.96774	0.55252	0
0.1	-0.43834	0.023115	1.9721
0.05	-0.42087	0.0056468	2.0333
0.025	-0.41663	0.0014036	2.0084
0.0125	-0.41558	0.00035038	2.0021

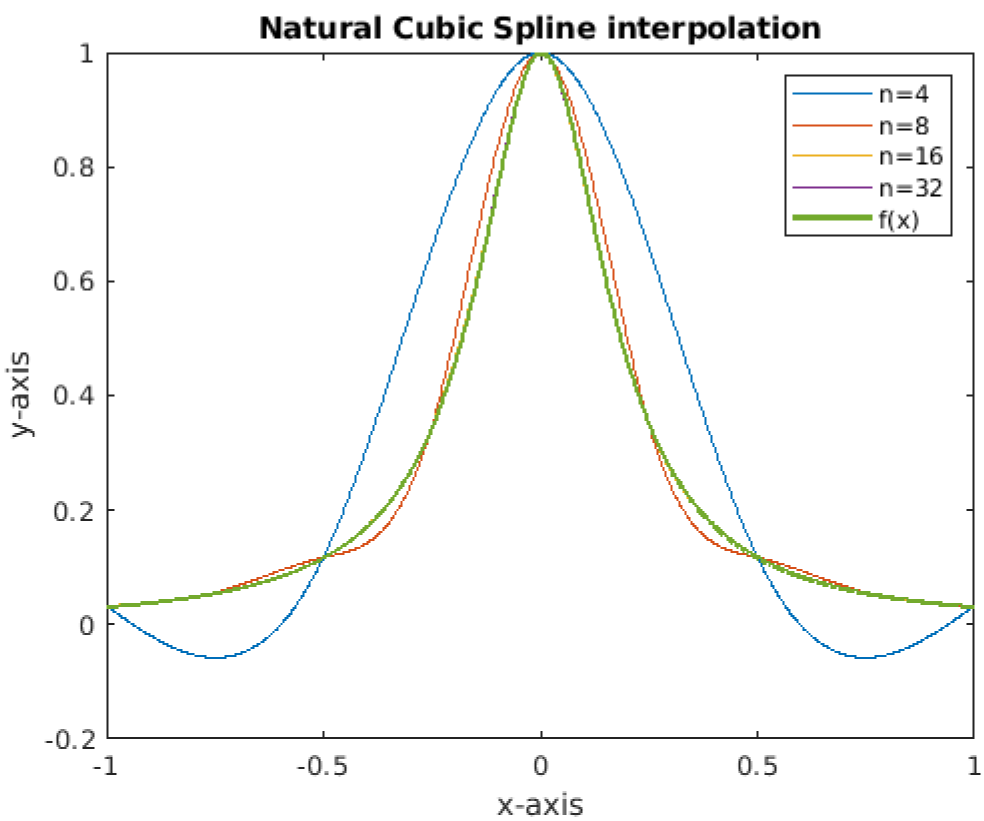
Richardson Extrapolation table

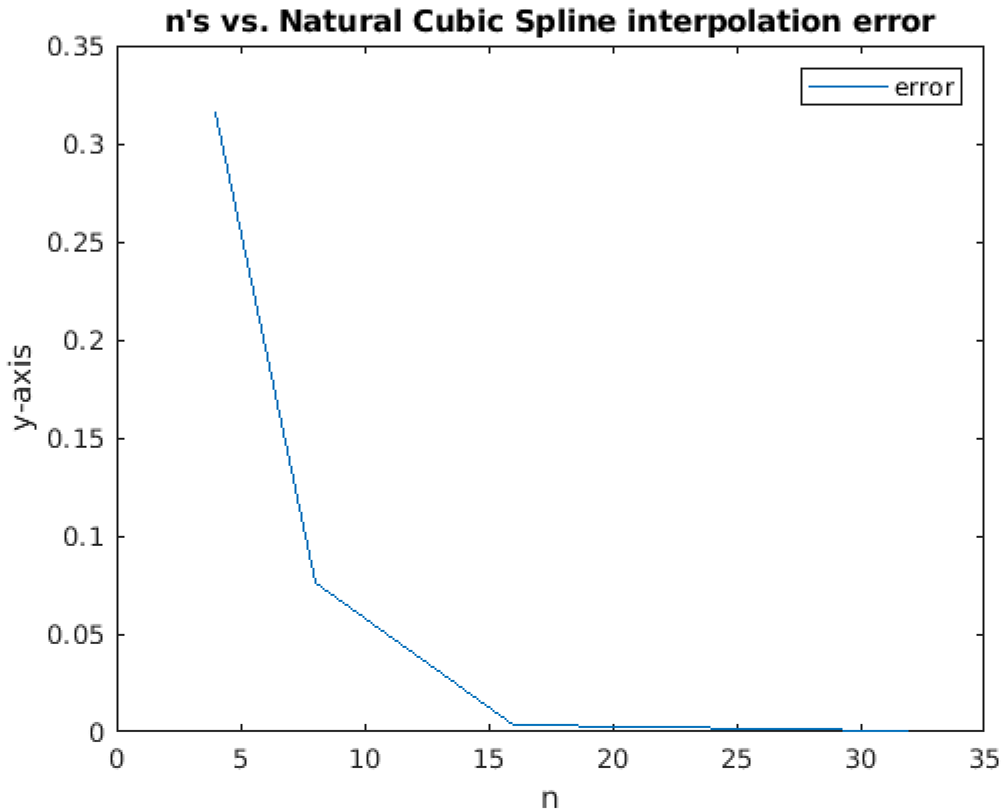
h	Approx	Error	pk_order
0.5	-0.45577	0.040547	0
0.1	-0.41505	0.00017599	3.3799
0.05	-0.41521	1.0868e-05	4.0173
0.025	-0.41522	6.7712e-07	4.0046
0.0125	-0.41522	4.2286e-08	4.0012

Composite Trapezoidal Integration table

n	Approx	Error	pk_order
10	0.51094	0.0033023	0
20	0.50754	9.1914e-05	-5.167
40	0.50761	2.6003e-05	-1.8216
80	0.50763	6.5029e-06	-1.9995
160	0.50763	1.6259e-06	-1.9999
320	0.50763	4.0647e-07	-2


Lagrange interpolation
Approx
f(x)
y-axis
x-axis
2.5
2
1.5
1
0.5
0
-0.5
-1
-0.5
0
0.5
1


Natural Cubic Spline interpolation
n=4
n=8
n=16
n=32
f(x)
y-axis
x-axis
1
0.8
0.6
0.4
0.2
0
-0.2
-1
-0.5
0
0.5
1



Published with MATLAB® R2018a