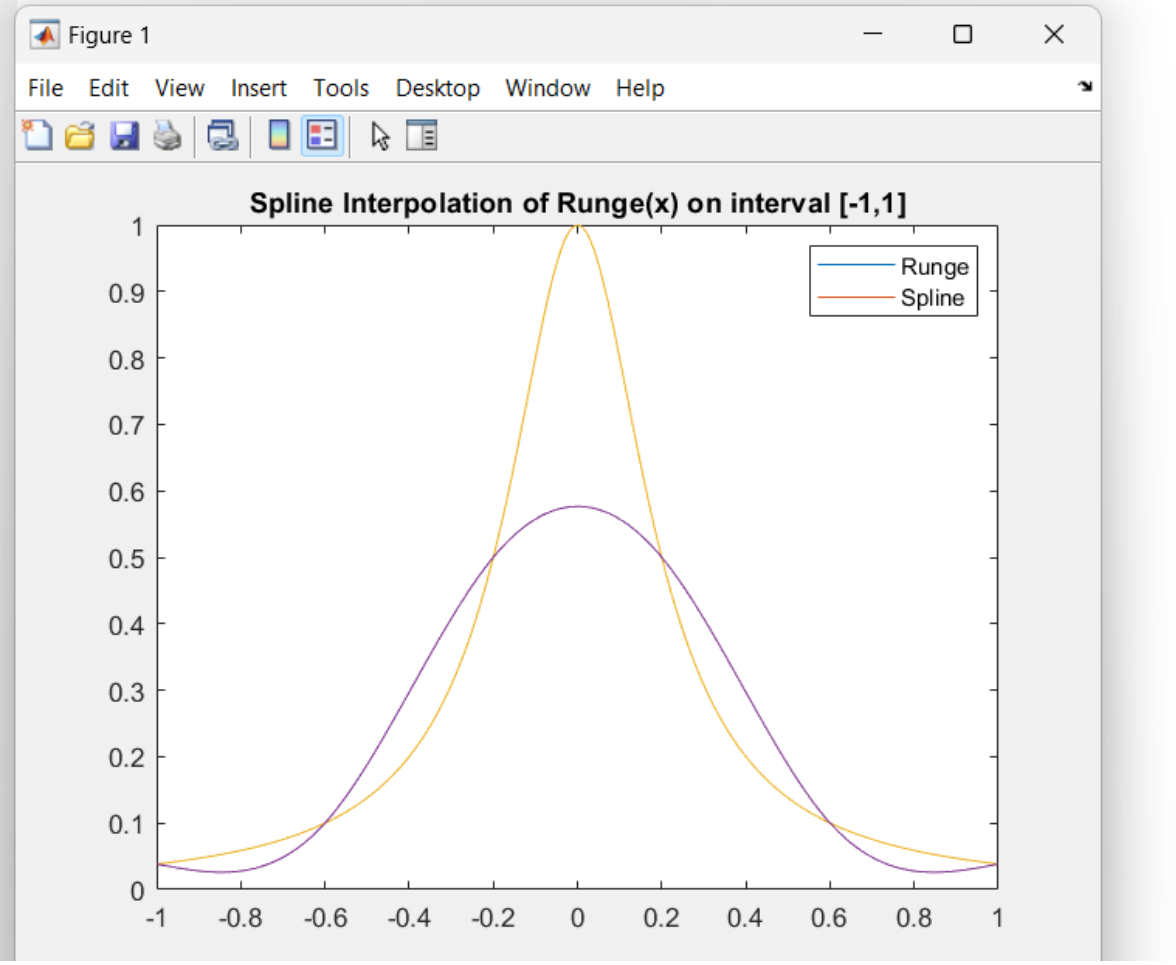
Problem 1.

(a – f) Initially using 5 knots to generate the spline, the maximum error of the spline approximation of the Runge function was .42

```
The Maximum Error: 0.423481781376518>>
```



Increasing the number of knots to n=4,8,16,32,64…, we observe:

```
>> P1_Natural_Cubic_Spline
For n = 4, error = 0.279311381158094
For n = 8, error = 0.056073644925168, convergence rate = 5.0
For n = 16, error = 0.003710967584725, convergence rate = 15.1
For n = 32, error = 0.000637335632232, convergence rate = 5.8
For n = 64, error = 0.000036066324368, convergence rate = 17.7
For n = 128, error = 0.000002053431999, convergence rate = 17.6
For n = 256, error = 0.000000159303183, convergence rate = 12.9
For n = 512, error = 0.000000009676896, convergence rate = 16.5
For n = 1024, error = 0.000000000331312, convergence rate = 29.2
For n = 2048, error = 0.000000000030329, convergence rate = 10.9
For n = 4096, error = 0.000000000002131, convergence rate = 14.2
```

The spline appears to converge quadratically to the runge function, as each time the size of the intervals are halfed, the error is atleast 4x smaller. The spline s appears to converge to f, and this agrees with the theoretical result that splines can converge to f.

Code:

P1_Natural_Cubic_Spline.m

```
% Goal: Create a cubic spline that interpolate a function (Runge)

% (0) Set up the interpolation problem
% (0.1) Sample the Runge function at (t_0,y_0)...(t_n,y_n)
for n=2.^(2:14)
    t = linspace(-1, 1, n+1);
    y = arrayfun(@Runge, t);

    % (1) Compute spline coeffients
    z = Spline_Coef(n, t, y);

    % (2) Evaluate the spline
    m = 200;
    X = linspace(-1, 1, m+1);
    Y_Runge = arrayfun(@Runge, X);
    Y_Spline = arrayfun(@(x) Spline_Eval(x, n, t, y, z), X);

    % (3) Printout
    plot(X, Y_Runge)
    hold on
    plot(X, Y_Spline)
    hold on
    legend('Runge','Spline')
    title("Spline Interpolation of Runge(x) on interval [-1,1]")

    error = max(abs(Y_Runge-Y_Spline));
    if n == 4
        fprintf("For n = %i, error = %.15f\n", n, error)
    else
        fprintf("For n = %i, error = %.15f, convergence rate = %3.1f\n", n,error, prev_error/error)
    end
    prev_error = error;
end

```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Function Definitions
function sum = Spline_Eval(x, n, t, y, z)
for i=n:-1:1
    if x - t(i) >= 0
        break
    end
end
h = t(i+1) - t(i);

sum = z(i+1)/(6*h) * (x - t(i))^3 ...
   +  z(i)/(6*h) * (t(i+1) - x)^3 + ...
          (y(i+1) / h - h/6 *z(i+1))*(x-t(i)) + (y(i) / h - h/6 * z(i)) *(t(i+1) - x);
end


function z = Spline_Coef(n, t, y)
    for i=1:n
        h(i) = t(i+1) - t(i);           % Length of ith interval
        b(i) = (y(i+1) - y(i))/h(i);    % Average slope in ith interval
    end

    u(2) = 2 * (h(1) + h(2));
    v(2) = 6 * (b(2)-b(1));

    for i=3:n
        u(i) = 2 * (h(i) + h(i-1)) - h(i-1)^2/u(i-1);
        v(i) = 6 * (b(i) - b(i-1)) - h(i-1)*v(i-1) / u(i-1);
    end

    z(n+1) = 0;
    z(1) = 0;
    for i = n:-1:2
        z(i) = (v(i) - h(i) * z(i+1)) / u(i);
    end
end

function y = Runge(x)
y = 1/(1 + 25*x^2);
end
```

Problem 2