```matlab
clear; clc
format short

x = [1 2 2.5 3 4 5];
f = [1 5 7 8 2 1];
n = length(x);

fp1 = 0; fpn = 0;

fprintf('The coefficients for a cubic spline with natural boundary conditions:\n')
[abcd] = myCubicSpline(x, f, 1, fp1, fpn);
coefficientsNatural = [abcd]

fprintf('\nThe coefficients for a cubic spline with clamped boundary conditions:\n')
[abcd] = myCubicSpline(x, f, 2, fp1, fpn);
coefficientsClamped = [abcd]

fprintf('\nThe coefficients for a cubic spline with not-a-knot boundary conditions:\n')
[abcd] = myCubicSpline(x, f, 3, fp1, fpn);
coefficientsNotknot = [abcd]
```

```matlab
% Clamped boundary conditions, where derivatives at the ends are 0
x11 = linspace(x(1), x(2), 1000);
f11 = @(x11) 1.0000 + 7.2183*(x11 - x(1)).^2 - 3.2183*(x11 - x(1)).^3;

x12 = linspace(x(2), x(3), 1000);
f12 = @(x12) 5.0000 + 4.7817*(x12 - x(2)) - 2.4367*(x12 - x(2)).^2 + 1.7467*(x12 - x(2)).^3;

x13 = linspace(x(3), x(4), 1000);
f13 = @(x13) 7.0000 + 3.6550*(x13 - x(3)) + 0.1834*(x13 - x(3)).^2 - 6.9869*(x13 - x(3)).^3;

x14 = linspace(x(4), x(5), 1000);
f14 = @(x14) 8.0000 - 1.4017*(x14 - x(4)) - 10.2969*(x14 - x(4)).^2 + 5.6987*(x14 - x(4)).^3;

x15 = linspace(x(5), x(6), 1000);
f15 = @(x15) 2.0000 - 4.8996*(x15 - x(5)) + 6.7991*(x15 - x(5)).^2 - 2.8996*(x15 - x(5)).^3;

% Natural boundary conditions
x21 = linspace(x(1), x(2), 1000);
f21 = @(x21) 1.0000 + 3.9710*(x21 - x(1)) + 0.0290*(x21 - x(1)).^3;
```

```matlab
x22 = linspace(x(2), x(3), 1000);
f22 = @(x22) 5.0000 + 4.0581*(x22 - x(2)) + 0.0871*(x22 - x(2)).^2 - 0.4066*(x22 - x(2)).^3;

x23 = linspace(x(3), x(4), 1000);
f23 = @(x23) 7.0000 + 3.8402*(x23 - x(3)) - 0.5228*(x23 - x(3)).^2 - 6.3154*(x23 - x(3)).^3;

x24 = linspace(x(4), x(5), 1000);
f24 = @(x24) 8.0000 - 1.4191*(x24 - x(4)) -9.9959*(x24 - x(4)).^2 + 5.4149*(x24 - x(4)).^3;

x25 = linspace(x(5), x(6), 1000);
f25 = @(x25) 2.0000 - 5.1660*(x25 - x(5)) + 6.2490*(x25 - x(5)).^2 - 2.0830*(x25 - x(5)).^3;

% Not-a-knot boundary conditions
x31 = linspace(x(1), x(2), 1000);
f31 = @(x31) 1.0000 + 3.4184*(x31 - x(1)) + 0.9694*(x31 - x(1)).^2 - 0.3878*(x31 - x(1)).^3;

x32 = linspace(x(2), x(3), 1000);
f32 = @(x32) 5.0000 + 4.1939*(x32 - x(2)) - 0.1939*(x32 - x(2)).^2 - 0.3878*(x32 - x(2)).^3;

x33 = linspace(x(3), x(4), 1000);
f33 = @(x33) 7.0000 + 3.7092*(x33 - x(3)) - 0.7755*(x33 - x(3)).^2 - 5.2857*(x33 - x(3)).^3;
```

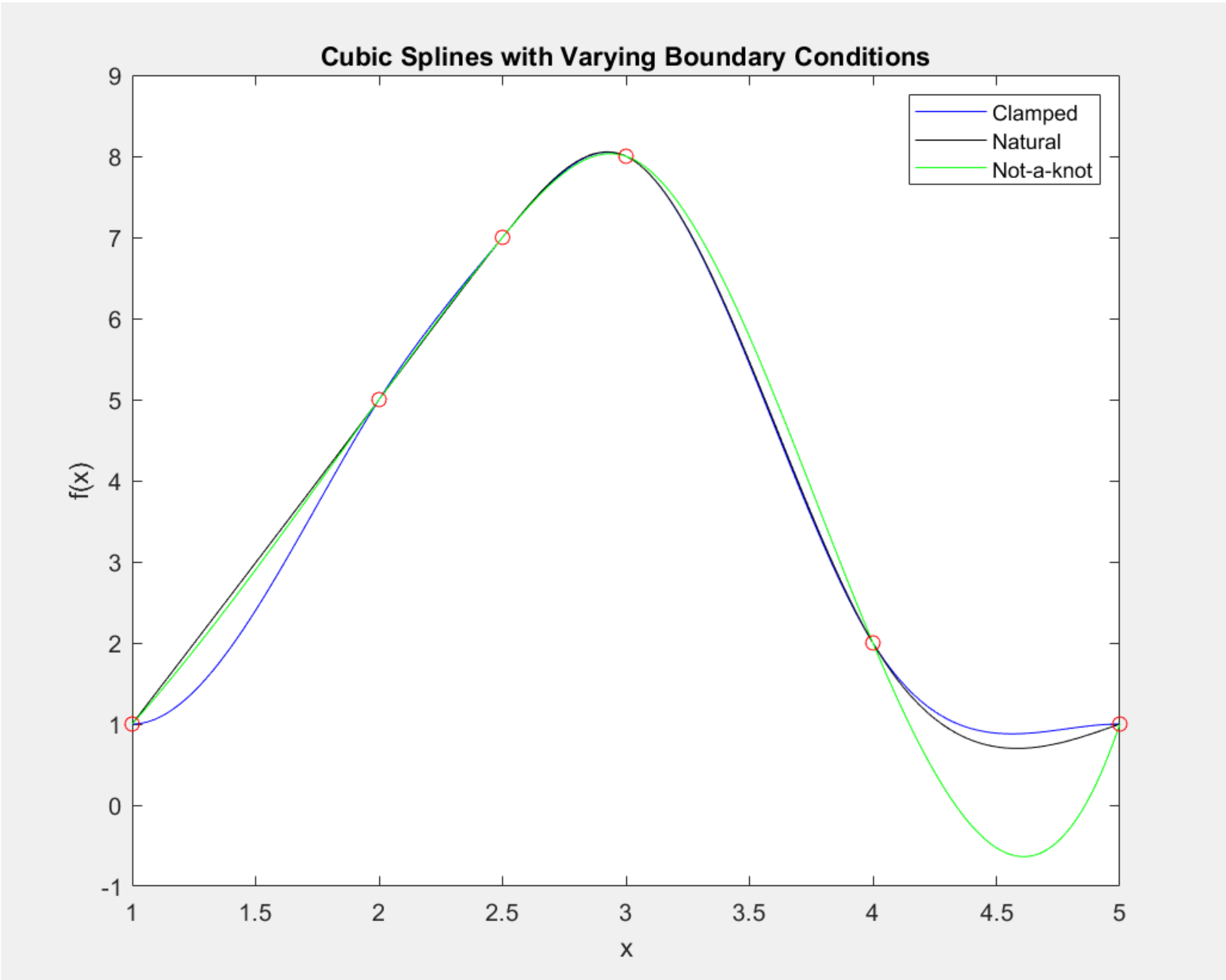
```matlab

x34 = linspace(x(4), x(5), 1000);
f34 = @(x34) 8.0000 - 1.0306*(x34 - x(4)) - 8.7041*(x34 - x(4)).^2 + 3.7347*(x34 - x(4)).^3;

x35 = linspace(x(5), x(6), 1000);
f35 = @(x35) 2.0000 - 7.2347*(x35 - x(5)) + 2.5000*(x35 - x(5)).^2 + 3.7347*(x35 - x(5)).^3;

plot(x11, f11(x11), 'b-')
hold on
plot(x21, f21(x21), 'k-')
plot(x31, f31(x31), 'g-')
plot(x12, f12(x12), 'b-')
plot(x13, f13(x13), 'b-')
plot(x14, f14(x14), 'b-')
plot(x15, f15(x15), 'b-')

plot(x22, f22(x22), 'k-')
plot(x23, f23(x23), 'k-')
plot(x24, f24(x24), 'k-')
plot(x25, f25(x25), 'k-')

plot(x32, f32(x32), 'g-')
plot(x33, f33(x33), 'g-')
plot(x34, f34(x34), 'g-')
plot(x35, f35(x35), 'g-')
plot(x, f, 'ro')
hold off
title('Cubic Splines with Varying Boundary Conditions')
xlabel('x')
ylabel('f(x)')

labelnat = "Natural";
labelclamp = "Clamped";
labelknot = "Not-a-knot";

legend({labelclamp, labelnat, labelknot})

function [abcd] = myCubicSpline(x, f, C, fp1, fpn)
    % 1.) n = number of data points
    % 2.) h = step sizes and Df = divided differences to the first order
    % 3.) preload left-hand side matrix A and right-hand side matrix r.
    % 4.) for i = 2:n - 1
    % 5.) fill in the values for matrix A accordingly from row 2 to row n -
    % 1. Row 2 and row n will be evaluated based on the boundary conditions
    % 6.) fill in the values for matrix r accordingly from row 2 to row n -
    % 1. Just as with matrix A, the values in row 1 and row n will be
    % evaluated based on boundary conditions
    % 7.) end of for-loop
    % 8.) if C == 1 (natural boundary condition)
    % 9.) coefficient matrix c = C\r
    % 10.) c(1) = 0, c(n) = 0
    % 11.) for j = 1:n - 1
    % 12.) solve for the values of coefficients b and d
    % 13.) end of for-loop
    % 14.) coefficient values of a are equal to function values
    % 15.) abcd = [a(1:n - 1), b, c(1:n - 1), d]
    % 16.) elseif C == 2 (clamped boundary conditions)
    % 17.) A(1, 1) = 2*h(1) and A(1, 2) = h(1)
```

```
% 18.) A(n, n - 1) = h(n - 1) and A(n, n) = 2*h(n - 1)
% 19.) r(1) = 3*Df(1) - 3*fp1 and r(n) = 3*fpn - 3*Df(n - 1)
% 20.) coefficient matrix c = C\r
% 21.) repeat steps 11 to 15 to solve for coefficient values of a, b,
% and d
% 22.) else (C == 3, Not-a-knot boundary condition)
% 23.) A(1, 1) = h(2), A(1, 2) = -(h(1) + h(2)), and A(1, 3) = h(1)
% 24.) A(n, n - 2) = h(n - 1), A(n, n - 1) = -(h(n - 2) + h(n - 1)),
% and A(n, n) = h(n - 2)
% 25.) r(1) = 0 and r(n) = 0
% 26.) coefficient matrix c = C\r
% 27.) repeat steps 11 to 15 to solve for coefficient values of a, b,
% and d
% 28.) end of if-elseif-else statement

n = length(x); % number of data points
h = x(2:n) - x(1:n - 1); % step sizes
Df = (f(2:n) - f(1:n - 1))./h; % divided differences

A = eye(n);
r = zeros(n, 1);


for i = 2:n - 1
    A(i, i - 1:i + 1) = [h(i - 1) 2*(h(i - 1) + h(i)) h(i)];
    r(i) = 3*(Df(i) - Df(i - 1));
end
if C == 1 % natural boundary conditions
    c = A\r;
    c(1) = 0; c(n) = 0;
    for j = 1:n - 1
        a(j) = f(j);
        d(j) = (c(j + 1) - c(j))/(3*h(j));
        b(j) = ((f(j + 1) - f(j))/h(j)) - (h(j)/3)*(2*c(j) + c(j + 1));
    end
    a = a(:); b = b(:); c = c(:); d = d(:);
    abcd = [a(1:n - 1), b, c(1:n - 1), d];
elseif C == 2 % clampled boundary conditions
    A(1, 1) = 2*h(1); A(1, 2) = h(1);
    A(n, n - 1) = h(n - 1); A(n, n) = 2*h(n - 1);
    r(1) = 3*Df(1) - 3*fp1; r(n) = 3*fpn - 3*Df(n - 1);


    c = A\r;
    for j = 1:n - 1
        a(j) = f(j);
        d(j) = (c(j + 1) - c(j))/(3*h(j));
        b(j) = ((f(j + 1) - f(j))/h(j)) - (h(j)/3)*(2*c(j) + c(j + 1));
    end
    a = a(:); b = b(:); c = c(:); d = d(:);
    abcd = [a(1:n - 1), b, c(1:n - 1), d];
else % not-a-knot boundary conditions
    A(1, 1) = h(2); A(1, 2) = -(h(1) + h(2)); A(1, 3) = h(1);
    A(n, n - 2) = h(n - 1); A(n, n - 1) = -(h(n - 2) + h(n - 1)); A(n, n) = h(n - 2);
    r(1) = 0; r(n) = 0;

    c = A\r;
    for j = 1:n - 1
        a(j) = f(j);
        d(j) = (c(j + 1) - c(j))/(3*h(j));
        b(j) = ((f(j + 1) - f(j))/h(j)) - (h(j)/3)*(2*c(j) + c(j + 1));
```

```
            end
            a = a(:); b = b(:); c = c(:); d = d(:);
            abcd = [a(1:n - 1), b, c(1:n - 1), d];
        end
    end
```

Command Window

```
The coefficients for a cubic spline with natural boundary conditions:

coefficientsNatural =

    1.0000    3.9710         0    0.0290
    5.0000    4.0581    0.0871   -0.4066
    7.0000    3.8402   -0.5228   -6.3154
    8.0000   -1.4191   -9.9959    5.4149
    2.0000   -5.1660    6.2490   -2.0830


The coefficients for a cubic spline with clamped boundary conditions:

coefficientsClamped =

    1.0000    0.0000    7.2183   -3.2183
    5.0000    4.7817   -2.4367    1.7467
    7.0000    3.6550    0.1834   -6.9869
    8.0000   -1.4017  -10.2969    5.6987
    2.0000   -4.8996    6.7991   -2.8996

The coefficients for a cubic spline with not-a-knot boundary conditions:

coefficientsNotknot =

    1.0000    3.4184    0.9694   -0.3878
    5.0000    4.1939   -0.1939   -0.3878
    7.0000    3.7092   -0.7755   -5.2857
    8.0000   -1.0306   -8.7041    3.7347
    2.0000   -7.2347    2.5000    3.7347

fx >>
```


Cubic Splines with Varying Boundary Conditions
Clamped
Natural
Not-a-knot
f(x)
x
9
8
7
6
5
4
3
2
1
0
-1
1
1.5
2
2.5
3
3.5
4
4.5
5