```
% example splines
%NOTE: In the slide is not used time normalization contrary to the
 example
tvals = [1 2 2.5 4];
qvals = [45 90 -45 45];
    % If initial and final velocities need to be imposed,
    % this needs modifications (slide 13).

    % tau = t - t1 --> tau = (t - t1) / h

    if exist('norm', 'var')
        normalize = norm;
    else
        normalize = 1;
    end
```

Calcs

```
    N = size(tvals, 2);
    t = sym ('t', [1, N], 'real');
    q = sym ('q', [1, N], 'real');
    v = sym ('v', [1, N], 'real');

    if ~exist('v1', 'var')
        v1 = 0;
        vn = 0;
    end

    v(1) = v1;
    v(N) = vn;

    h = t(2:N) - t(1:N-1)


 fprintf('_________________________________________________________________
\n');
    fprintf('First at all, we write the coefficients as function of
 velocities\n\n');
    for i = 1:N-1
        fprintf('Symbolic Cubic %d parameters:\n', i);

        a0 = q(i);
        if normalize == 1
            a1 = v(i) * h(i);
            a = inv([1, 1; 2/h(i), 3/h(i)]) * [q(i+1) - q(i) - v(i) *
 h(i); v(i+1) - v(i)];
        else
            a1 = v(i);
            a = inv([h(i)^2, h(i)^3; 2*h(i), 3*h(i)^2]) * [q(i+1) -
 q(i) - v(i) * h(i); v(i+1) - v(i)];
        end
```

```
        % For normalization use this one:
        a = simplify([a0; a1; a])
    end

 fprintf('__________________________******_____________________________
\n');

 fprintf('_________________________________________________________________
\n');
    fprintf('To impose continuity of the acceleration at the internal
 knots\n\n');

    % A is a N-2 x N-2 matrix
    % The same for all joints
    A = sym(zeros([N-2, N-2]));
    for i = 1:N-2
        if i-1 > 0
            A(i-1, i) = h(i-1);
        end
        A(i, i) = 2 * (h(i) + h(i+1));
        if i+1 <= N-2
            A(i+1, i) = h(i+2);
        end
    end

    % b changes for each joint
    % as it depends on q
    b = sym(zeros([N-2, 1]));
    for i = 1:N-2
        b(i) = 3 * (q(i+2) - q(i+1)) * h(i)/h(i+1) + 3 * (q(i+1) -
 q(i)) * h(i+1)/h(i);
    end
    b = b - [h(2) * v1; zeros([N-4,1]); h(N-2) * vn];

    A, b %a(h), b(...)


 fprintf('__________________________******_____________________________
\n');
```

h =

[t2 - t1, t3 - t2, t4 - t3]

First at all, we write the coefficients as function of velocities

Symbolic Cubic 1 parameters:

a =

q1
0

```
 3*q2 - 3*q1 + v2*(t1 - t2)
 2*q1 - 2*q2 - v2*(t1 - t2)

Symbolic Cubic 2 parameters:

a =

                                                            q2
                                                 -v2*(t2 - t3)
 3*q3 - 3*q2 + 3*v2*(t2 - t3) - (t2 - t3)*(v2 - v3)
 2*q2 - 2*q3 - 2*v2*(t2 - t3) + (t2 - t3)*(v2 - v3)

Symbolic Cubic 3 parameters:

a =

                          q3
                -v3*(t3 - t4)
 3*q4 - 3*q3 + 2*v3*(t3 - t4)
   2*q3 - 2*q4 - v3*(t3 - t4)
```

******

```
To impose continuity of the acceleration at the internal knots


A =

[ 2*t3 - 2*t1,     t2 - t1]
[     t4 - t3, 2*t4 - 2*t2]


b =

 - ((3*q1 - 3*q2)*(t2 - t3))/(t1 - t2) - ((3*q2 - 3*q3)*(t1 - t2))/(t2
 - t3)
 - ((3*q2 - 3*q3)*(t3 - t4))/(t2 - t3) - ((3*q3 - 3*q4)*(t2 - t3))/(t3
 - t4)
```

******

Numberical Results

Convert everything to numerical values

```
A = subs(A, t, tvals);
b = subs(b, t, tvals);
b = subs(b, q, qvals);
h = subs(h, t, tvals);

v1 = subs(v1, q, qvals);
v1 = subs(v1, t, tvals);
vn = subs(vn, q, qvals);
```

```
    vn = subs(vn, t, tvals);
    v = inv(A) * b; % [v2 ... v_N-1]
    v = eval([v1; v; vn])
    coeffs = [];
    for i = 1:N-1
        % fprintf('Numeric Cubic %d parameters:\n', i);
        a0 = qvals(i);

        if normalize == 1
            a1 = v(i) * h(i);
            a = inv([1, 1; 2/h(i), 3/h(i)]) * [qvals(i+1) - qvals(i) -
 v(i) * h(i); v(i+1) - v(i)];
        else
            a1 = v(i);
            a = inv([h(i)^2, h(i)^3; 2*h(i), 3*h(i)^2]) * [qvals(i+1)
 - qvals(i) - v(i) * h(i); v(i+1) - v(i)];
        end
        a = eval([a0, a1, a']);

        coeffs = [coeffs; a];
    end

 fprintf('_________________________________________________________________
\n');
    fprintf('Results\n\n');
    for i=1:size(v,1)
        fprintf('v%d = %f \n',i, v(i));
    end
    for i=1:size(coeffs, 1)
        fprintf('Symbolic Cubic %d parameters:\n', i);
        for j=1:size(coeffs,2)
            fprintf('a%d = %f \n',j-1, coeffs(i,j));
        end
    end


 fprintf('__________________________******_____________________________
\n');

    % Test using directly the function
    %   function [coeffs, v] = splines(tvals, qvals, v1, vn, norm)
    tvals = [1 2 2.5 4];
    qvals = [45 90 -45 45];
    [cofs, vels] = splines(tvals, qvals);


 fprintf('_________________________________________________________________
\n');
    fprintf('Results\n\n');
    for i=1:size(vels,1)
        fprintf('v%d = %f \n',i, vels(i));
    end
    for i=1:size(cofs, 1)
        fprintf('Symbolic Cubic %d parameters:\n', i);
```

```
        for j=1:size(cofs,2)
            fprintf('a%d = %f \n',j-1, cofs(i,j));
        end
    end


 fprintf('__________________________******_____________________________
\n');
% test spline_plot
spline_plot(tvals, cofs)
```

```
v =

         0
 -175.7143
 -215.3571
         0

_________________________________________________________________
Results

v1 = 0.000000
v2 = -175.714286
v3 = -215.357143
v4 = 0.000000
Symbolic Cubic 1 parameters:
a0 = 45.000000
a1 = 0.000000
a2 = 310.714286
a3 = -265.714286
Symbolic Cubic 2 parameters:
a0 = 90.000000
a1 = -87.857143
a2 = -121.607143
a3 = 74.464286
Symbolic Cubic 3 parameters:
a0 = -45.000000
a1 = -323.035714
a2 = 916.071429
a3 = -503.035714
__________________________******_____________________________

Undefined function 'splines' for input arguments of type 'double'.

Error in splines__spline_plot (line 126)
    [cofs, vels] = splines(tvals, qvals);
```

Published with MATLAB® R2020a