# MA5271 Assignment

Nguyen Ngoc Khanh - A0275047B

May 20, 2025

# Contents

# 1 MOTIVATION OF CUBIC SPLINE INTERPOLATION

Given $n+1$ data points $(x_1, y_1), (x_1, y_1), ..., (x_{n+1}, y_{n+1})$, we seek to construct a smooth curve that passes through all data points. The easiest method is linear interpolation, however linear interpolation does not produce a smooth curve. Another method is to use a polynomial of high degreewhich tends to produce unwanted oscillations.
Cubic spline interpolation is a method that sits in between, provides smooth curve of second order and does not oscillate too much (limited to third order polynomials)

# 2 DEFINITION OF CUBIC SPLINE INTERPOLATION

Given $n+1$ data points $(x_1, y_1), (x_1, y_1), ..., (x_{n+1}, y_{n+1})$, for every $i = 1, ..., n$, we choose a polynomial (spline)

$$f_i(x) = a_i(x - x_i)^3 + b_i(x - x_i)^2 + c_i(x - x_i) + d_i$$

for each interval $[x_{i-1}, x_i]$, so that it satisfies the following constraints

1. (goes through all points) $f_i(x_i) = y_i$ and $f_i(x_{i+1}) = y_{i+1}$, that is

$$\begin{aligned} y_i &= d_i \\ y_{i+1} &= (x_{i+1} - x_i)^3 a_i + (x_{i+1} - x_i)^2 b_i + (x_{i+1} - x_i) c_i + d_i \end{aligned}$$

   for every $i = 1, ..., n$

2. (continuity of first derivative) $f_i'(x_{i+1}) = f_{i+1}'(x_{i+1})$, that is

$$3(x_{i+1} - x_i)^2 a_i + 2(x_{i+1} - x_i) b_i + c_i = c_{i+1}$$

   for every $i = 1, ..., n$

3. (continuity of second derivative) $f_i''(x_{i+1}) = f_{i+1}''(x_{i+1})$, that is

$$6(x_{i+1} - x_i) a_i + 2b_i = 2b_{i+1}$$

   for every $i = 1, ..., n$

In total, we have $4n-2$ constraints and $4n$ variables. The other two constraints (boundary conditions) will be introduced in the next section

# 3 TWO METHODS FOR CONSTRUCTING CUBIC SPLINE INTERPOLATION

## 3.1 NATURAL SPLINE

Natural spline is used when there is no information about *derivative* of the curve at $x_1$ and $x_{n+1}$, that is to set

$$f_1''(x_1) = 0 \text{ and } f_n''(x_{n+1}) = 0$$

That induces two constraints

$$2b_1 = 0 \text{ and } 6(x_{n+1} - x_n)a_n + 2b_n = 0$$

## 3.2 CLAMPED SPLINE

Clamped spline is used when there is information about the *first derivative* of the curve at $x_1$ and $x_{n+1}$ that is to set

$$f_1'(x_1) = f'(x_1) \text{ and } f_n'(x_{n+1}) = f'(x_{n+1})$$

That induces two constraints

$$c_1 = f_1'(x_1) \text{ and } 3(x_{n+1} - x_n)^2 a_n + 2(x_{n+1} - x_n)b_n + c_n = f'(x_{n+1})$$

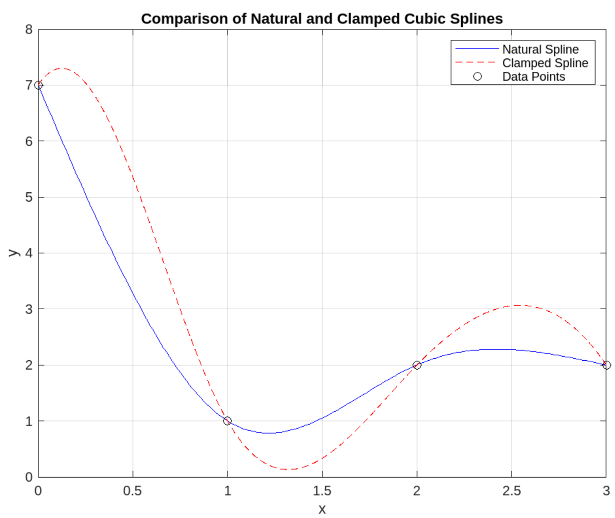
# 4 MATLAB CODE AND NUMERICAL EXAMPLES

## 4.1 NUMERICAL EXAMPLES

Consider the following data points

$$(0, 7), (1, 1), (2, 2), (3, 2)$$

with boundary conditions $f'(x_1) = f'(0) = 5$ and $f'(x_{n+1}) = f'(3) = -5$. The figure below describes the two curves constructed from natural cubic spline (not using boundary condition) and clamped spline.




## 4.2 MATLAB CODE

```
function coeffs = naive_cubic_spline(x, y, type, bc1, bc2)
    n = length(x) - 1;        % number of spline
    N = 4 * n;                % number of unknowns (a_i, b_i, c_i, d_i for each segment)
    A = zeros(N, N);          % solve for x in Ax = b
    b = zeros(N, 1);

    % indexing of a_i, b_i, c_i, d_i in A
```

```
ai = @(i) i;              % index of a_i
bi = @(i) n + i;          % index of b_i
ci = @(i) 2*n + i;        % index of c_i
di = @(i) 3*n + i;        % index of d_i

row = 1;

% goes through all points: f_i(x_i) = y_i and f_i(x_{i+1}) = y_{i+1}
for i = 1:n
    % x = x_i, h = x_i - x_i = 0
    h = 0;
    A(row, ai(i)) = h^3;
    A(row, bi(i)) = h^2;
    A(row, ci(i)) = h;
    A(row, di(i)) = 1;
    b(row) = y(i);
    row = row + 1;

    % x = x_{i+1}, h = x_{i+1} - x_i
    % note that i+1 -> i and i -> i-1 in the document because mathlab is 1-indexed
    h = x(i+1) - x(i);
    A(row, ai(i)) = h^3;
    A(row, bi(i)) = h^2;
    A(row, ci(i)) = h;
    A(row, di(i)) = 1;
    b(row) = y(i+1);
    row = row + 1;
end

% continuity of first derivative: f_i'(x_{i+1}) = f_{i+1}'(x_{i+1})
for i = 1:n-1
    h = x(i+1) - x(i);
    A(row, ai(i))   = 3*h^2;
    A(row, bi(i))   = 2*h;
    A(row, ci(i))   = 1;
    A(row, ci(i+1)) = -1;
    b(row) = 0;
    row = row + 1;
end

% continuity of second derivative: f_i''(x_{i+1}) = f_{i+1}''(x_{i+1})
for i = 1:n-1
    h = x(i+1) - x(i);
    A(row, ai(i))   = 6*h;
    A(row, bi(i))   = 2;
    A(row, bi(i+1)) = -2;
    b(row) = 0;
    row = row + 1;
end

% boundary conditions
h1 = x(2) - x(1);
hn = x(n+1) - x(n);

```

```
    if strcmp(type, 'natural')
        % natural spline: second derivative = 0 at endpoints
        A(row, ai(1)) = 0;       % 6*h1 = 0 because h = 0
        A(row, bi(1)) = 2;
        b(row) = 0;
        row = row + 1;

        A(row, ai(n)) = 6*hn;
        A(row, bi(n)) = 2;
        b(row) = 0;
        row = row + 1;

    elseif strcmp(type, 'clamped')
        % clamped spline: first derivative specified at endpoints
        A(row, ai(1)) = 0;       % 3*0^2
        A(row, bi(1)) = 0;       % 2*0
        A(row, ci(1)) = 1;       % f_1'(x_1) = c_1
        b(row) = bc1;
        row = row + 1;

        A(row, ai(n)) = 3*hn^2;
        A(row, bi(n)) = 2*hn;
        A(row, ci(n)) = 1;
        b(row) = bc2;
        row = row + 1;
    else
        error('unknown boundary condition: "natural" or "clamped"');
    end

    % solve for a_i, b_i, c_i, d_i
    xsol = A \ b;

    % get [a_i, b_i, c_i, d_i] from xsol
    coeffs = zeros(n, 4);
    for i = 1:n
        coeffs(i, :) = [xsol(ai(i)), xsol(bi(i)), xsol(ci(i)), xsol(di(i))];
    end
end


% EXAMPLE

x = [0 1 2 3];
y = [7 1 2 2];

% natural spline
coeffs_nat = naive_cubic_spline(x, y, 'natural');

% calculate natural spline
xq = linspace(x(1), x(end), 200);
yq_nat = zeros(size(xq));
for j = 1:length(xq)
    i = find(xq(j) >= x(1:end-1) & xq(j) <= x(2:end), 1);
    if isempty(i), i = length(x)-1; end
```

```matlab
    dx = xq(j) - x(i);
    a = coeffs_nat(i, 1);
    b = coeffs_nat(i, 2);
    c = coeffs_nat(i, 3);
    d = coeffs_nat(i, 4);
    yq_nat(j) = a*dx^3 + b*dx^2 + c*dx + d;
end

% clamped spline
bc_start = 5;   % f'(x_1)
bc_end = -5;    % f'(x_{n+1})
coeffs_clamped = naive_cubic_spline(x, y, 'clamped', bc_start, bc_end);

% calculate clamped spline
yq_clamp = zeros(size(xq));
for j = 1:length(xq)
    i = find(xq(j) >= x(1:end-1) & xq(j) <= x(2:end), 1);
    if isempty(i), i = length(x)-1; end
    dx = xq(j) - x(i);
    a = coeffs_clamped(i, 1);
    b = coeffs_clamped(i, 2);
    c = coeffs_clamped(i, 3);
    d = coeffs_clamped(i, 4);
    yq_clamp(j) = a*dx^3 + b*dx^2 + c*dx + d;
end

% splot
plot(xq, yq_nat, 'b-', 'DisplayName', 'Natural Spline'); hold on;
plot(xq, yq_clamp, 'r--', 'DisplayName', 'Clamped Spline');
plot(x, y, 'ko', 'DisplayName', 'Data Points');
legend;
title('Comparison of Natural and Clamped Cubic Splines');
xlabel('x');
ylabel('y');
grid on;
```