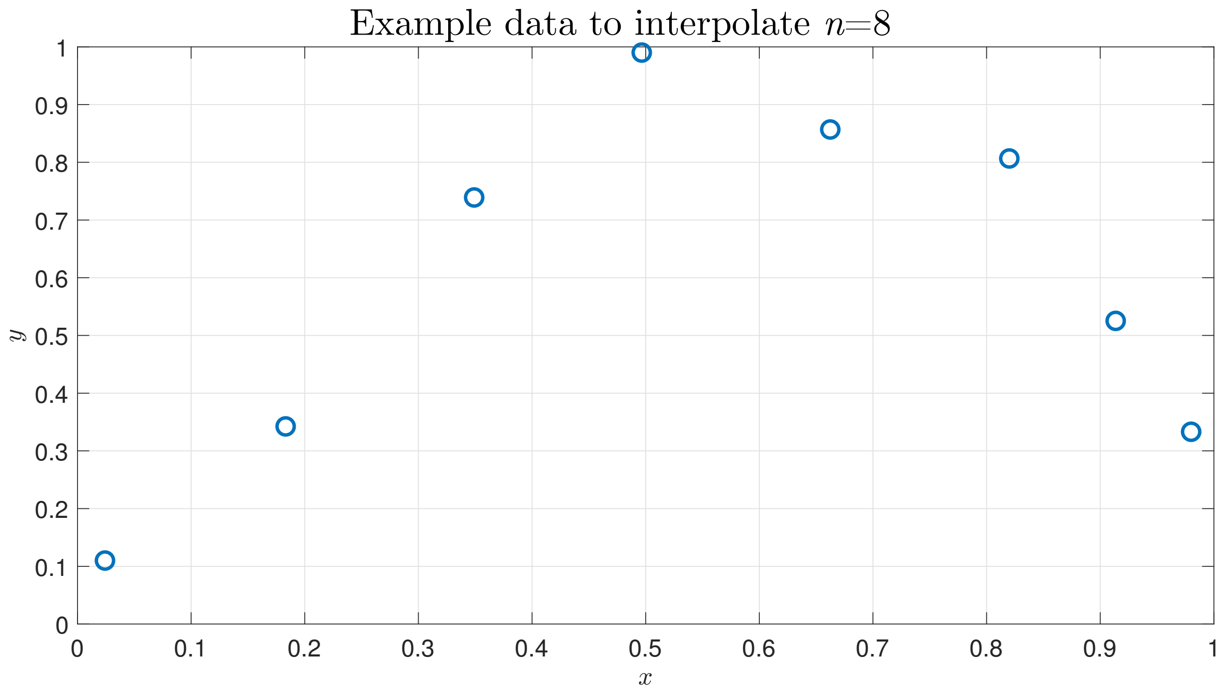
Cubic splines

- We now consider another way to approximate the derivatives of the data to use in a piecewise cubic Hermite interpolating polynomial.



Data:
(x_k, f_k),
$k = 1, \ldots, n$

Recall:

For each interval $[x_k, x_{k+1}]$, $k = 1, \ldots, n-1$, fit a cubic Hermite polynomial to the data (x_k, f_k)

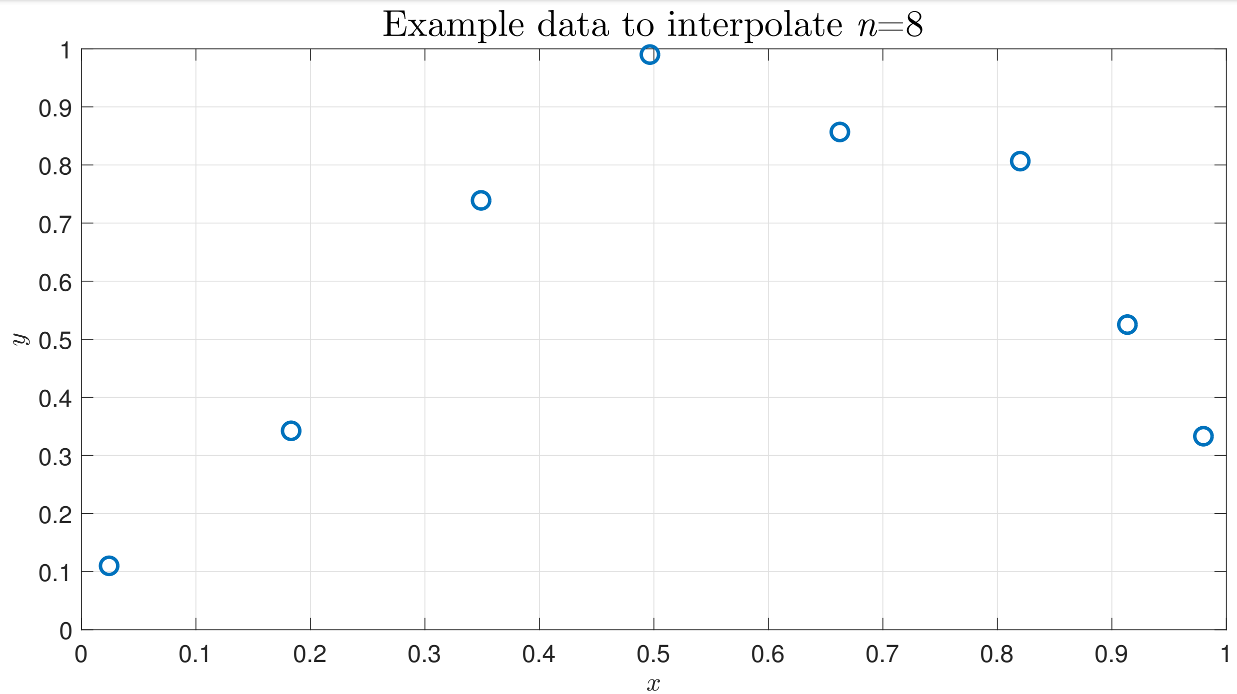
$$H_k(x) = b_k(x - x_k)^3 + c_k(x - x_k)^2 + d_k(x - x_k) + e_k, \quad x \in [x_k, x_{k+1}]$$

where

$$h_k = x_{k+1} - x_k \qquad c_k = \frac{3\delta_k - 2d_k - d_{k+1}}{h_k} \qquad e_k = f_k$$

$$\delta_k = \frac{f_{k+1} - f_k}{h_k} \qquad b_k = \frac{d_k - 2\delta_k + d_{k+1}}{h_k^2} \qquad d_k \approx f'_k$$

Cubic splines: main idea



Data:
(x_k, f_k),
$k = 1, \dots, n$

Recall: The piecewise cubic functions

$$H_k(x) = b_k(x - x_k)^3 + c_k(x - x_k)^2 + d_k(x - x_k) + e_k, \quad x \in [x_k, x_{k+1}]$$

are guaranteed to produce an interpolating function that is continuous over the entire domain and has a first derivative that is continuous regardless of how d_k are determined, i.e.

$$\begin{aligned} H_k(x_{k+1}) &= H_{k+1}(x_{k+1}) \\ H_k'(x_{k+1}) &= H_{k+1}'(x_{k+1}) \end{aligned} \qquad k = 1, 2, \dots, n-2$$

Idea: Enforce that the interpolating function also has a continuous second derivative:

$$H_k''(x_{k+1}) = H_{k+1}''(x_{k+1}), \quad k = 1, 2, \dots, n-2$$

Cubic splines: working through the algebra

Let's write down the system of equations that result from the constraint on the second derivative.

$$H_k''(x) = 6b_k(x - x_k) + 2c_k \quad \text{and} \quad H_{k+1}''(x) = 6b_{k+1}(x - x_{k+1}) + 2c_{k+1}$$

Setting $H_k''(x_{k+1}) = H_{k+1}''(x_{k+1})$, gives:

$$6b_k(x_{k+1} - x_k) + 2c_k = -6b_{k+1}(x_{k+1} - x_k) + 2c_{k+1}$$

$$k = 1, 2, \ldots, n-2$$

$$\Longrightarrow 6h_k(b_k + b_{k+1}) + 2(c_k - c_{k+1}) = 0$$

Now express everything in terms of the approximate derivatives d_k and the given data f_k using:

$$c_k = (3\delta_k - 2d_k - d_{k+1})/h_k \qquad c_{k+1} = (3\delta_{k+1} - 2d_{k+1} - d_{k+2})/h_{k+1} \qquad \delta_k = (f_{k+1} - f_k)/h_k$$

$$b_k = (d_k - 2\delta_k + d_{k+1})/h_k^2 \qquad b_{k+1} = (d_{k+1} - 2\delta_{k+1} + d_{k+2})/h_{k+1}^2 \qquad \delta_{k+1} = (f_{k+2} - f_{k+1})/h_{k+1}$$

After some algebra and simplification, one arrives at the system of equations:

$$h_{k+1}d_k + 2(h_k + h_{k+1})d_{k+1} + h_k d_{k+2} = 3(h_{k+1}\delta_k + h_k\delta_{k+1}), \quad k = 1, \ldots, n-2$$

This is a tridiagonal system of $n-2$ equations for n unknowns d_k, $k = 1, \ldots, n$.

$\Longrightarrow$ Need to add two additional constraints to determine all d_k

Cubic splines: end conditions

These constraints often involve imposing some conditions at the ends of domain, called end conditions.

Three popular choices for end conditions:

1. Natural end conditions:

$$H_1''(x_1) = 0 \quad \text{and} \quad H_{n-1}''(x_n) = 0$$

2. Knot-a-not end conditions

$$H_1'''(x_2) = H_2'''(x_2) \quad \text{and} \quad H_{n-2}'''(x_{n-1}) = H_{n-1}'''(x_{n-1})$$

3. Clamped end conditions

$$H_1'(x_1) = f_1' \quad \text{and} \quad H_{n-1}'(x_n) = f_n'$$

All of these options are available in the MATLAB `spline` command.