Lecture 17

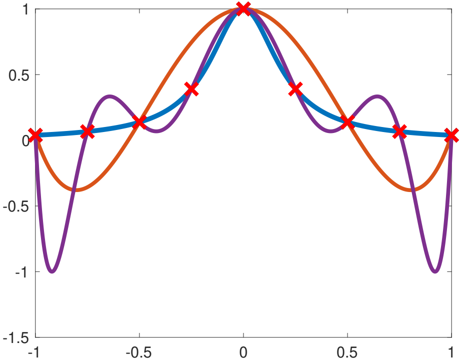
Chebychev Nodes & Splines

Owen L. Lewis

Department of Mathematics and Statistics
University of New Mexico

October 24, 2024

Review: Polynomial interpolation can be BAD!



Can show that, when using equispaced data points (for this f),

$$\lim_{n \to \infty} \left(\max_{-1 \leqslant x \leqslant 1} |f(x) - p(x)| = \infty \right)$$

Analysis of Interpolation Error: Equispaced Points

Theorem: Interpolation Error II

Let $|f^{(n+1)}(x)| \leqslant M$, then with the above,

$$|f(x) - p_n(x)| \leqslant \frac{Mh^{n+1}}{4(n+1)}$$

Up-shot

As n increases, h decreases, but M *might* grow too fast, causing error to explode.

Fixes

We have two options:

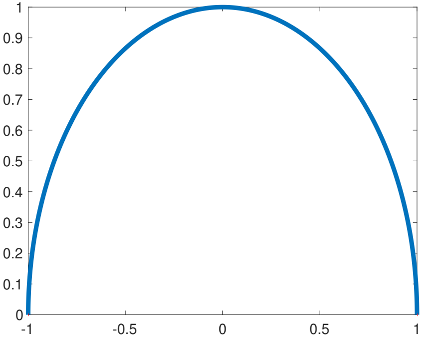
1. move the nodes: Chebychev nodes
2. piecewise polynomials (splines)

Option #1: Chebychev nodes in $[-1, 1]$

Option #2: piecewise polynomials...

Chebychev Nodes (First Kind)

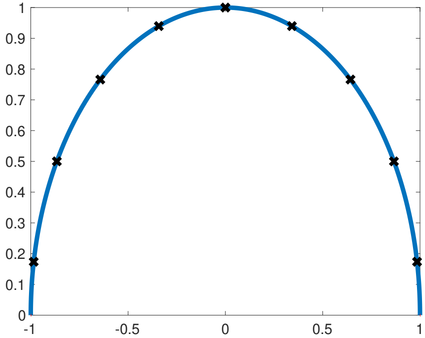
$$x_i = cos\left(\pi\frac{2i+1}{2n+2}\right), \quad i = 0, \ldots, n$$



Start with a semi-circle above the interval.

Chebychev Nodes (First Kind)

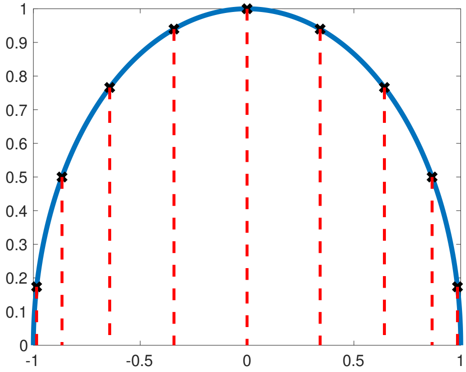
$$x_i = cos\left(\pi\frac{2i+1}{2n+2}\right), \quad i = 0, \ldots, n$$



Equally space points on that circle.

Chebychev Nodes (First Kind)

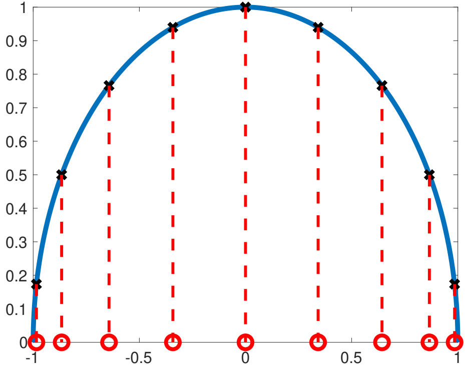
$$x_i = cos\left(\pi\frac{2i+1}{2n+2}\right), \quad i = 0, \ldots, n$$



Project down to the x axis

Chebychev Nodes (First Kind)

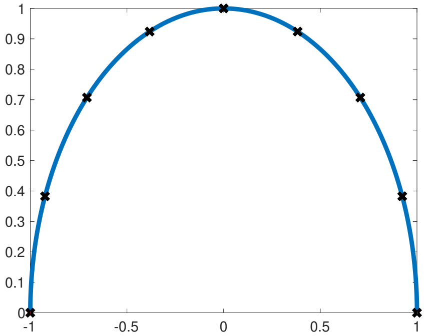
$$x_i = cos\left(\pi \frac{2i+1}{2n+2}\right), \quad i = 0, \ldots, n$$



Use *those* as your interpolation points.

Chebychev Nodes (Second Kind)

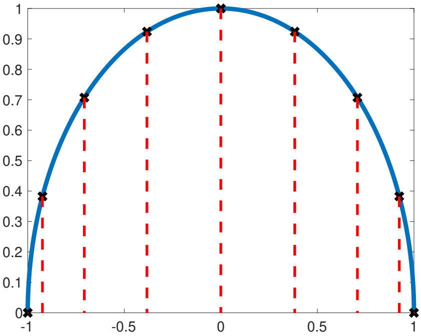
$$x_i = cos\left(\pi \frac{i}{n}\right), \quad i = 0, \dots, n$$



Equally space points on that circle.

Chebychev Nodes (Second Kind)

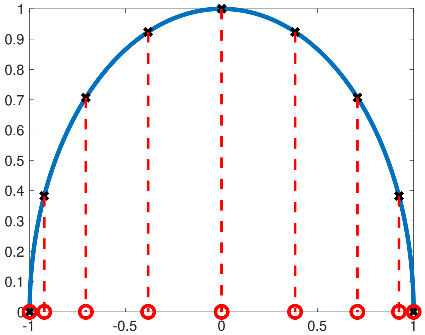
$$x_i = cos\left(\pi\frac{i}{n}\right), \quad i = 0, \dots, n$$



Project down to the x axis

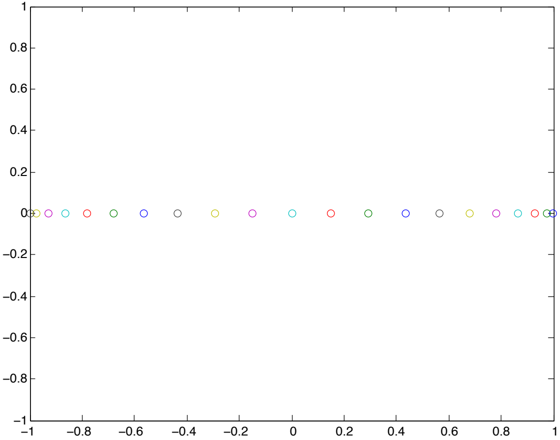
Chebychev Nodes (Second Kind)

$$x_i = cos\left(\pi\frac{i}{n}\right), \quad i = 0, \ldots, n$$



Use *those* as your interpolation points.

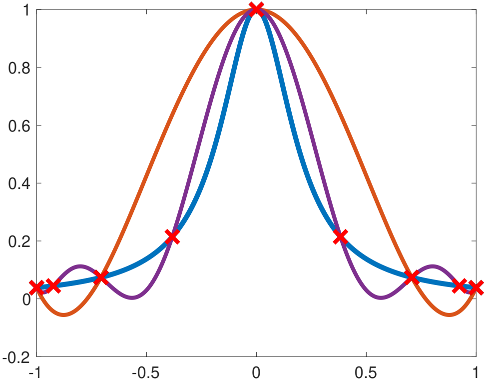
Chebychev Nodes (or Chebychev-Lobatto)



- Can obtain nodes from equidistant points on a circle projected down
- Nodes are non uniform and non nested

Chebychev Nodes

High degree polynomials using equispaced points suffer from many oscillations



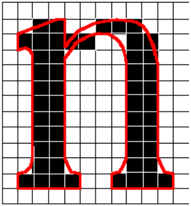
- Chebyshev bunches the points towards the ends of the interval
- This ”ties” the function down at the ends, and the error is distributed more evenly

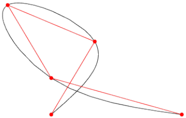
Why not Chebychev?

Chebychev points are “optimal” in that they minimize Runge phenomenon as n increases.

Unfortunately this presumes *we* get to choose where our “data” points lie on the x-axis.

Why Splines?





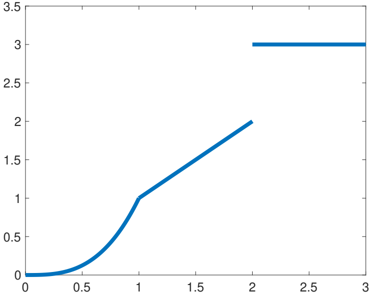
- Truetype fonts, postscript, metafonts
- Graphics surfaces
- Smooth surfaces are needed
- How do we interpolate smoothly a set of data?
- Keywords: Bezier Curves, splines, B-splines, NURBS
- Basic tool: piecewise interpolation

Piecewise Polynomial

A function $f(x)$ is considered a piecewise polynomial on $[a, b]$ if there exists a (finite) partition P of $[a, b]$ such that $f(x)$ is a polynomial on each $[t_i, t_{i+1}] \in P$.

Example

$$f(x) = \begin{cases} x^3 & x \in [0, 1] \\ x & x \in (1, 2) \\ 3 & x \in [2, 3] \end{cases}$$

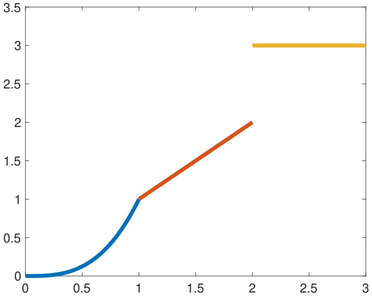


Piecewise Polynomial

A function $f(x)$ is considered a piecewise polynomial on $[a, b]$ if there exists a (finite) partition P of $[a, b]$ such that $f(x)$ is a polynomial on each $[t_i, t_{i+1}] \in P$.

Example

$$f(x) = \begin{cases} x^3 & x \in [0, 1] \\ x & x \in (1, 2) \\ 3 & x \in [2, 3] \end{cases}$$

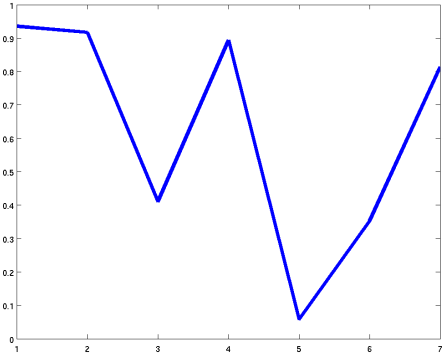


What do we want?

- We would like the piecewise polynomial to do two things
  1. Interpolate (or be close to) some set of data points
  2. Look nice (smooth)
- One option is called a *spline*

Splines

- A *spline* is a piecewise polynomial with a certain level of smoothness.
- Take Matlab: `plot(1:7,rand(7,1))`
- This is linear and continuous, but not very smooth
- The function changes behavior at *knots* $t_0, \ldots, t_n$



Degree 1 spline

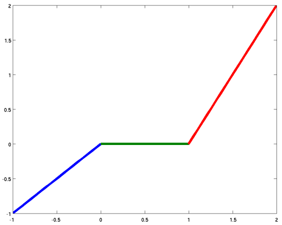
definition

A function $S(x)$ is a spline of degree 1 if:

1. The domain of $S(x)$ is an interval $[a, b]$
2. $S(x)$ is continuous on $[a, b]$
3. There is a partition $a = t_0 < t_1 < \cdots < t_n = b$ such that $S(x)$ is linear on each subinterval $[t_i, t_{i+1}]$.

Example

$$S(x) = \begin{cases} x & x \in [-1, 0] \\ 1 & x \in (0, 1) \\ 2x - 2 & x \in [1, 2] \end{cases}$$



Notes

Degree 1 spline

Given data $t_0, \ldots, t_n$ and $y_0, \ldots, y_n$, how do we form a spline?

We need two things to hold in the interval $[a, b] = [t_0, t_n]$:

1. $S(t_i) = y_i$ for $i = 0, \ldots, n$
2. $S_i(x) = a_i x + b_i$ for $i = 0, \ldots, n$

Write $S_i(x)$ in point-slope form

$$\begin{aligned} S_i(x) &= y_i + m_i(x - t_i) \\ &= y_i + \frac{y_{i+1} - y_i}{t_{i+1} - t_i}(x - t_i) \end{aligned}$$

Done.

Degree 1 spline

```
input t, y vectors of data
input evaluation location x
find interval i with x ∈ [t_i, t_{i+1}]
S = y_i + (x-t_i) m_i
```

Degree 1 spline

Interesting:

- Input $n+1$ data points $t_0, \ldots, t_n, y_0, \ldots, y_n$
- In each interval we have $S_i(x) = a_i x + b_i$
- Two unknowns per interval $[t_i, t_{i+1}]$
- Or, $2n$ total unknowns
- The $n+1$ pieces of input, require that $S(t_i) = y_i$. This provides 2 constraints per interval
- Or $2n$ total constraints

This is a well-defined problem.

Degree 2 splines

Definition

A function $S(x)$ is a spline of degree 2 if:

1. The domain of $S(x)$ is an interval $[a, b]$
2. $S(x)$ is continuous on $[a, b]$
3. $S'(x)$ is continuous on $[a, b]$
4. There is a partition $a = t_0 < t_1 < \cdots < t_n = b$ such that $S(x)$ is quadratic on each subinterval $[t_i, t_{i+1}]$.

Degree 2 splines

$$S(x) = \begin{cases} S_0(x) & x \in [t_0, t_1] \\ S_1(x) & x \in [t_1, t_2] \\ \vdots & \vdots \\ S_{n-1}(x) & x \in [t_{n-1}, t_n] \end{cases}$$

for each i we have

$$S_i(x) = a_i x^2 + b_i x + c_i$$

What are a_i, b_i, c_i?

Degree 3 spline: cubic spline

definition

A function $S(x)$ is a spline of degree 3 if:

1. The domain of $S(x)$ is an interval $[a, b]$
2. $S(x)$ is continuous on $[a, b]$
3. $S'(x)$ is continuous on $[a, b]$
4. $S''(x)$ is continuous on $[a, b]$
5. There is a partition $a = t_0 < t_1 < \cdots < t_n = b$ such that $S(x)$ is cubic on each subinterval $[t_i, t_{i+1}]$.

Degree 3 spline: cubic spline

In each interval $[t_i, t_{i+1}]$, $S(x)$ looks like

$$S_i(x) = a_{0,i} + a_{1,i}x + a_{2,i}x^2 + a_{3,i}x^3$$

- n intervals, $n + 1$ data points, 4 unknowns per interval
- $4n$ unknowns

Degree 3 spline: cubic spline

In each interval $[t_i, t_{i+1}]$, $S(x)$ looks like

$$S_i(x) = a_{0,i} + a_{1,i}x + a_{2,i}x^2 + a_{3,i}x^3$$

- n intervals, $n + 1$ data points, 4 unknowns per interval
- $4n$ unknowns
- $2n$ constraints by continuity

Degree 3 spline: cubic spline

In each interval $[t_i, t_{i+1}]$, $S(x)$ looks like

$$S_i(x) = a_{0,i} + a_{1,i}x + a_{2,i}x^2 + a_{3,i}x^3$$

- n intervals, $n+1$ data points, 4 unknowns per interval
- $4n$ unknowns
- $2n$ constraints by continuity
- $n-1$ constraints by continuity of $S'(x)$

Degree 3 spline: cubic spline

In each interval $[t_i, t_{i+1}]$, $S(x)$ looks like

$$S_i(x) = a_{0,i} + a_{1,i}x + a_{2,i}x^2 + a_{3,i}x^3$$

- n intervals, $n+1$ data points, 4 unknowns per interval
- $4n$ unknowns
- $2n$ constraints by continuity
- $n-1$ constraints by continuity of $S'(x)$
- $n-1$ constraints by continuity of $S''(x)$

Degree 3 spline: cubic spline

In each interval $[t_i, t_{i+1}]$, $S(x)$ looks like

$$S_i(x) = a_{0,i} + a_{1,i}x + a_{2,i}x^2 + a_{3,i}x^3$$

- n intervals, $n+1$ data points, 4 unknowns per interval
- $4n$ unknowns
- $2n$ constraints by continuity
- $n-1$ constraints by continuity of $S'(x)$
- $n-1$ constraints by continuity of $S''(x)$
- $4n-2$ total constraints

Degree 3 spline: cubic spline

In each interval $[t_i, t_{i+1}]$, $S(x)$ looks like

$$S_i(x) = a_{0,i} + a_{1,i}x + a_{2,i}x^2 + a_{3,i}x^3$$

- n intervals, $n+1$ data points, 4 unknowns per interval
- $4n$ unknowns
- $2n$ constraints by continuity
- $n-1$ constraints by continuity of $S'(x)$
- $n-1$ constraints by continuity of $S''(x)$
- $4n-2$ total constraints
- This leaves 2 extra degrees of freedom. The cubic spline is not yet unique!

Degree 3 spline: cubic spline

Some options:

- Natural cubic spline: $S''(t_0) = S''(t_n) = 0$
- Clamped: $S'(t_0) = a$, $S'(t_n) = b$ (User input)
- Periodic: S' and S'' are periodic at the ends: $S'(t_0) = S'(t_n)$ and $S''(t_0) = S''(t_n)$

Natural cubic spline

How do we find $a_{0,i}, a_{1,i}, a_{2,i}, a_{3,i}$ for each i?

First, we re-write the cubic polynomials, then do the following:

1. Write out matching conditions.
2. Differentiate twice, writing matching conditions as we go.
3. Write unknown coefficients in terms of second derivatives
4. Algebra
5. Find a linear system to solve for second derivatives.

Other ways?

Other books/professors do this slightly differently

Consider knots $t_0, \ldots, t_n$. Follow the following steps:

1. Assume we knew $S''(t_i)$ for each i
2. $S_i''(x)$ is linear, so construct it as we did for linear splines
3. Get $S_i(x)$ by integrating $S_i''(x)$ twice
4. Impose continuity on $S_i(x)$
5. Differentiate $S_i(x)$ to impose continuity on $S'(x)$

Natural cubic spline

"Center" each cubic at its own left endpoint

$$S_i(x) = a_i + b_i(x - t_i) + c_i(x - t_i)^2 + d_i(x - t_i)^3.$$

1. Match data to knots.
2. Differentiate S_i and match derivative at "inner" knots.
3. Differentiate again and match second derivative at "inner" knots.
4. Much algebra.

Keep the count

For $n+1$ knots enumerated t_i, $i = 0, 1, \dots n$.
We have n sub-intervals $[t_i, t_i + 1]$, $i = 0, 1, \dots n-1$.
Each gets its own cubic.

$$S_i(x) = a_i + b_i(x - t_i) + c_i(x - t_i)^2 + d_i(x - t_i)^3.$$

Natural cubic spline: Step 1

Getting Value Correct

At the left endpoints:

$$S_i(t_i) = y_i$$

$$\Rightarrow a_i + b_i(t_i - t_i) + c_i(t_i - t_i)^2 + d_i(t_i - t_i)^3 = y_i$$

$$\Rightarrow a_i = y_i.$$

For $i = 0, 1, \ldots n - 1$.

We know the a_i's.

Natural cubic spline: Step 1

Getting Value Correct

At the right endpoints:

$$
\begin{aligned}
S_i(t_{i+1}) &= y_{i+1} \\
\Rightarrow y_i + b_i(t_{i+1} - t_i) + c_i(t_{i+1} - t_i)^2 + d_i(t_{i+1} - t_i)^3 &= y_{i+1} \\
\Rightarrow b_i\delta_i + c_i\delta_i^2 + d_i\delta_i^3 &= \Delta_i \qquad (1)
\end{aligned}
$$

For $i = 0, 1, \ldots n - 1$.

Where we've defined $\delta_i = t_{i+1} - t_i$ and $\Delta_i = y_{i+1} - y_i$.

Natural cubic spline: Step 2

Getting the derivatives right

$S_i'(x)$ is a parabola

$$S_i'(x) = b_i + 2c_i(x - t_i) + 3d_i(x - t_i)^2.$$

Match derivative where sub-intervals meet:

$$S_i'(t_{i+1}) = S_{i+1}'(t_{i+1})$$

$$\Rightarrow b_i + 2c_i(t_{i+1} - t_i) + 3d_i(t_{i+1} - t_i)^2 = b_{i+1} + 2c_{i+1}(t_{i+1} - t_{i+1}) + 3d_{i+1}(t_{i+1} - t_{i+1})^2$$
$$\Rightarrow b_i + 2c_i\delta_i + 3d_i\delta_i^2 = b_{i+1}. \tag{2}$$

For $i = 0, 1, \ldots n - 2$.

Natural cubic spline: Step 3

Getting the convexity right

$S_i''(x)$ is a line

$$S_i''(x) = 2c_i + 6d_i(x - t_i).$$

Match second derivative where sub-intervals meet:

$$S_i''(t_{i+1}) = S_{i+1}''(t_{i+1})$$

$$\Rightarrow 2c_i + 6d_i(t_{i+1} - t_i) = 2c_{i+1} + 6d_{i+1}(t_{i+1} - t_{i+1})$$

$$\Rightarrow 2c_i + 6d_i\delta_i = 2c_{i+1}. \tag{3}$$

For $i = 0, 1, \ldots n - 2$.

Natural cubic spline: Step 4

Time for some algebra

From Equation 3, we can say

$$d_i = \frac{c_{i+1} - c_i}{3\delta_i}. \tag{4}$$

From Equation 1, we get

$$b_i = \frac{\Delta_i}{\delta_i} - c_i\delta_i - d_i\delta_i^2.$$

Combining with Equation 4 gives

$$b_i = \frac{\Delta_i}{\delta_i} - \frac{\delta_i}{3}(2c_i + c_{i+1}). \tag{5}$$

Natural cubic spline: Step 4

Time for some algebra

Finally, combining Equations 2, 4, and 5 gives

$$\delta_i c_i + 2(\delta_i + \delta_{i+1})c_{i+1} + \delta_{i+1}c_{i+2} = 3\left(\frac{\Delta_{i+1}}{\delta_{i+1}} - \frac{\Delta_i}{\delta_i}\right) \tag{6}$$

for $i = 0, 1, \ldots n-2$.
This seems complex, write it out...

$$\delta_0 c_0 + 2(\delta_0 + \delta_1)c_1 + \delta_1 c_2 = 3\left(\frac{\Delta_1}{\delta_1} - \frac{\Delta_0}{\delta_0}\right)$$

$$\vdots$$

$$\delta_{n-2}c_{n-2} + 2(\delta_{n-2} + \delta_{n-1})c_{n-1} + \delta_{n-1}c_n = 3\left(\frac{\Delta_{n-1}}{\delta_{n-1}} - \frac{\Delta_{n-2}}{\delta_{n-2}}\right)$$

Natural cubic spline: Step 4

Time for some algebra

We have $n - 2$ equations, we need more.

Zero derivative at the very left:

$$S_0''(t_0) = 0,$$

$$\Rightarrow 2c_0 = 0.$$

Zero derivative at the very right. Define

$$S''(t_n) = 2c_n = 0.$$

Natural cubic spline: Step 4

One big matrix

$$
\begin{bmatrix}
1 & 0 & 0 & & & \\
\delta_0 & 2(\delta_0+\delta_1) & \delta_1 & \ddots & & \\
0 & \delta_1 & 2(\delta_1+\delta_2) & \delta_2 & \ddots & \\
 & \ddots & \ddots & \ddots & \ddots & \\
 & & & \delta_{n-2} & 2(\delta_{n-2}+\delta_{n-1}) & \delta_{n-1} \\
 & & & 0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
c_0 \\ c_1 \\ \vdots \\ c_{n-1} \\ c_n
\end{bmatrix}
=
\begin{bmatrix}
0 \\
3\left(\frac{\Delta_1}{\delta_1}-\frac{\Delta_0}{\delta_0}\right) \\
\vdots \\
3\left(\frac{\Delta_{n-1}}{\delta_{n-1}}-\frac{\Delta_{n-2}}{\delta_{n-2}}\right) \\
0
\end{bmatrix}
\tag{7}
$$

Note: we don't need c_n for anything, it is just a convenience to make the matrix pattern nice.

Finding the spline

See Chapter 3.4 (page 176 in first edition)

```
input t, y vectors of data
calculate δ and Δ.
form right-hand-side vector b and matrix L
c = L\b
for i = 0, 1, ... n − 1
        b_i = Δ_i/δ_i − δ_i(2c_i + c_{i+1})/3
        d_i = (c_{i+1} + c_i)/(3δ_i)
end
S_i(x) = y_i + b_i(x − t_i) + c_i(x − t_i)^2 + d_i(x − t_i)^3
```