Numerical Mathematics & Computing, 7 Ed.
§6.1 First-Degree and Second-Degree Splines

Ward Cheney & David Kincaid

http:www.cengagebrain.com
http://www.ma.utexas.edu/CNA/NMC7

Recaps of Chapter 4 Polynomial Interpolation

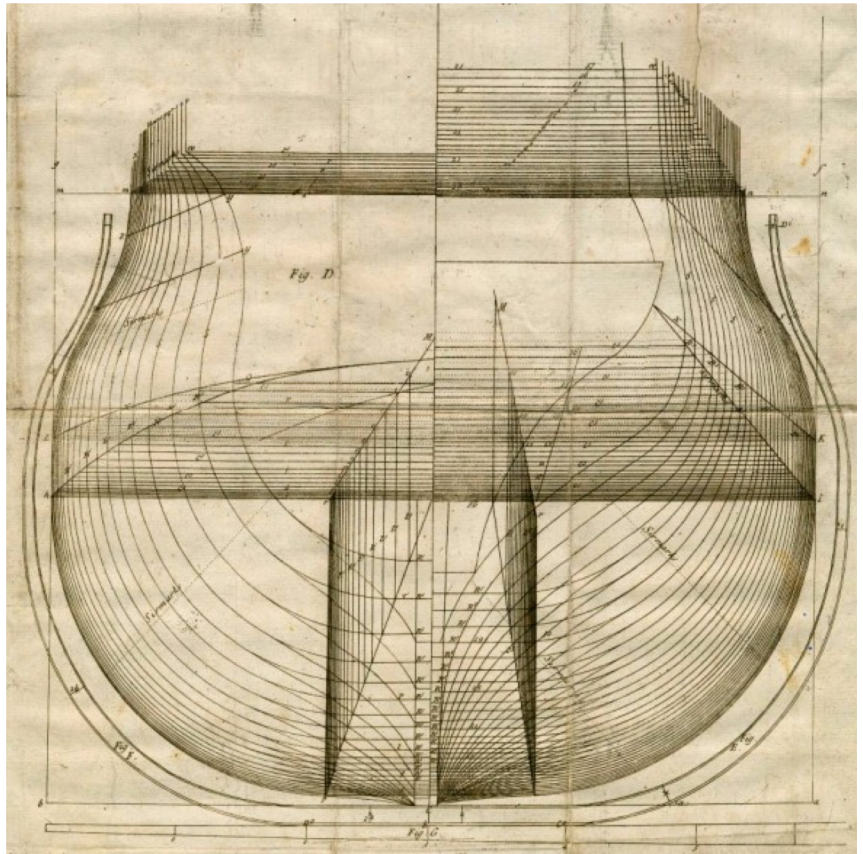
- Use a polynomial *p(x)* passing through a table of n points as a representation of the function from which the table arose.

- When *n* increases, *p(x)* exhibits wild oscillations.

- Polynomial interpolation of high degree with many nodes is a risky operation.

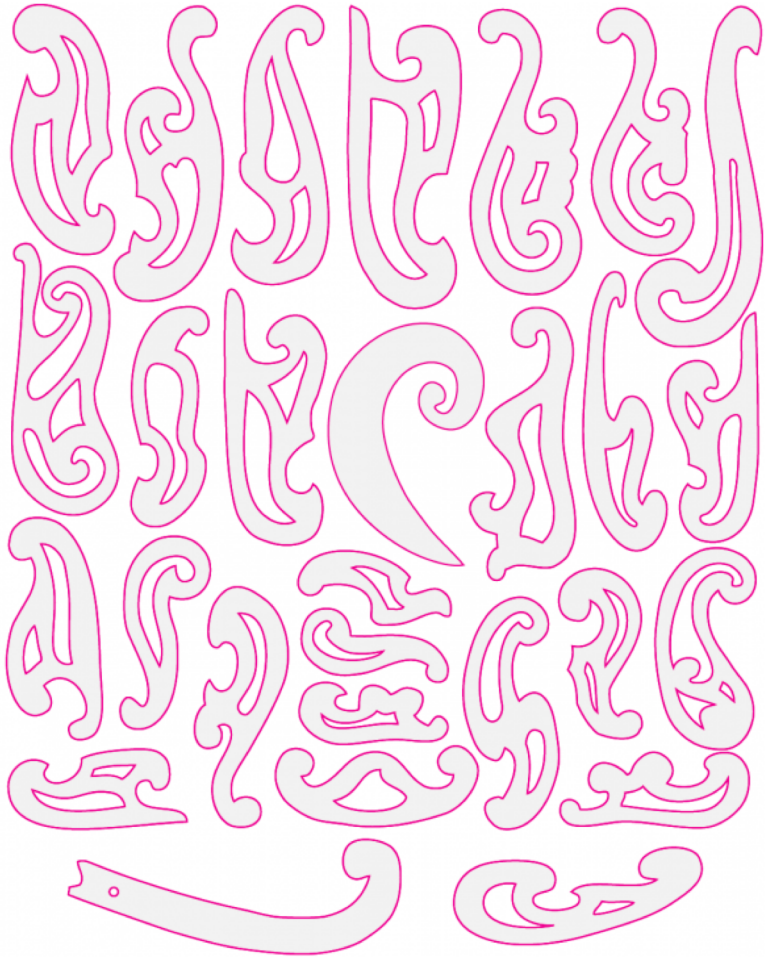
History of Spline Functions

- The history of spline functions is rooted in the work of draftsmen, who often needed to draw a gently turning curve between points on a drawing.



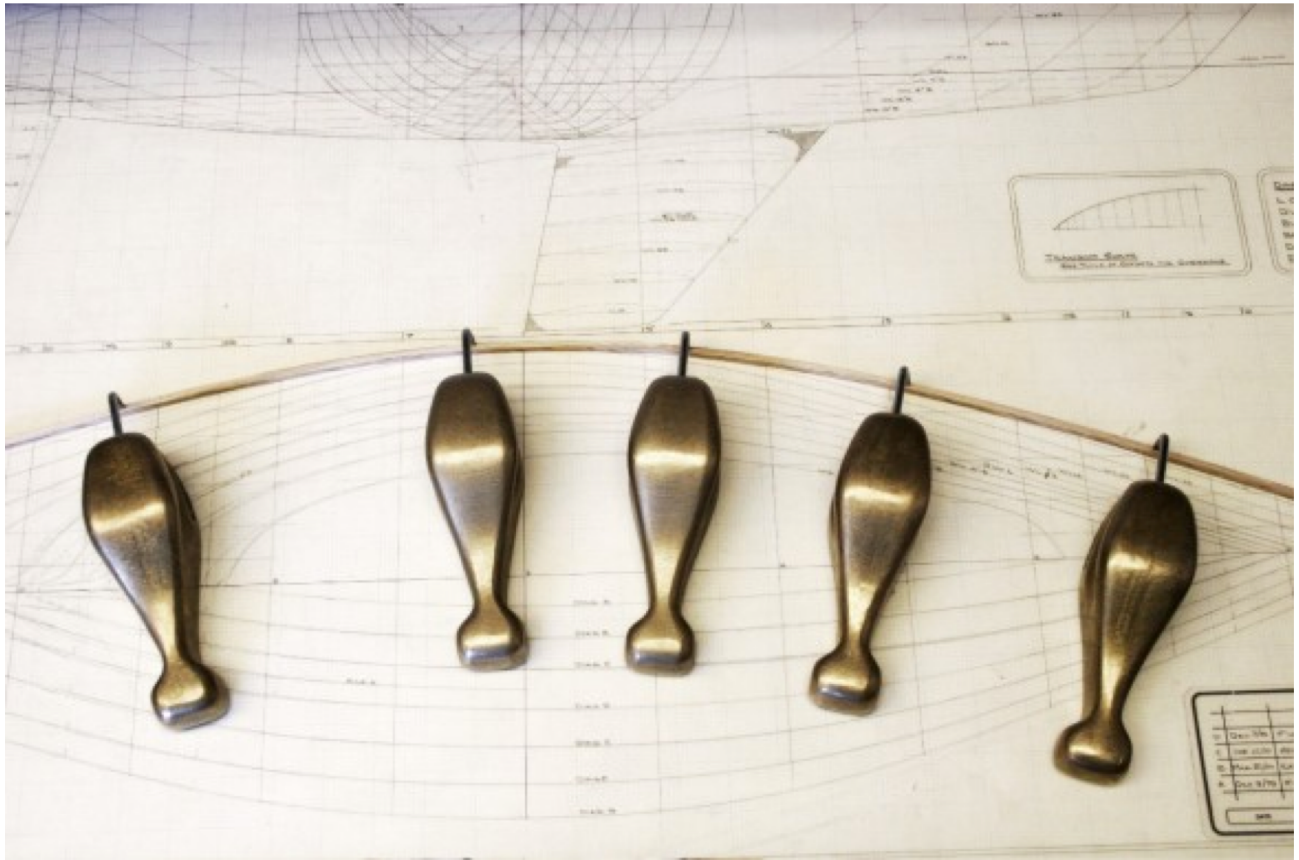
A ship's body plan comprised of superimposed construction geometry.

- This process is called fairing and can be accomplished with a number of ad hoc devices, such as the French curve, made of plastic and presenting a number of curves of different curvature for the draftsman to select.



Set of French Curves devised by the German kinematician and geometer Ludwig Burmester, as illustrated in the Lexikon der gesamten Technik (dictionary of technology) from 1904 by Otto Lueger.

- Long strips of wood were also used, being made to pass through the control points by weights laid on the draftsman's table and attached to the strips.
- The weights were called **ducks** and the strips of wood were called **splines**, even as early as 1891.



Hooked weights, called "ducks," accurately secure a spline – here, no more than a thin strip of balsa – for tracing the hull of a sailing vessel. Source: Edson International (with kind permission).

- The elastic nature of the wooden strips allowed them to bend only a little while still passing through the prescribed points.

- The mathematical theory of these curves owes much to the early investigators, particularly Isaac Schoenberg, a Romanian-American mathematician, in the 1940s and 1950s.

- During 1943–1945, Isaac Schoenberg initiated the work for which he is most famous, the theory of splines.

Isaac Jacob Schoenberg (April 21, 1903 – February 21, 1990)

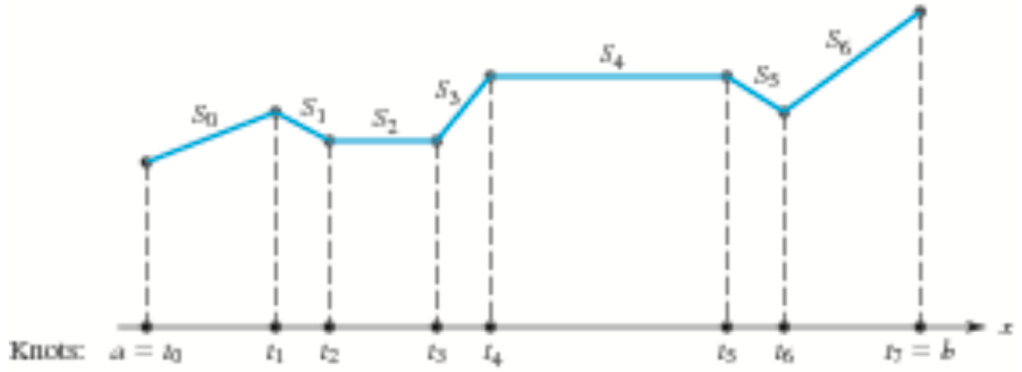
Spline Functions

- The word "spline" was originally an East Anglian dialect word.

- Schoenberg is probably the first one used the word "spline" in connection with smooth, piecewise polynomial approximation.

Spline function

- A **spline function** is a function that consists of polynomial pieces joined together with certain smoothness conditions.
- A simple example is the **polygonal** function (or spline of degree 1), whose pieces are linear polynomials joined together to achieve continuity.
- The points $t_0, t_1, \ldots, t_n$ at which the function changes its character are termed **knots** in the theory of splines.
- Thus, the spline function shown in the next figure has eight knots.



First-degree spline function

First-Degree Spline

- Such a function appears somewhat **complicated** when defined in explicit terms.
- We are forced to write

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

where

$$S_i(x) = a_i x + b_i \tag{2}$$

because each piece of $S(x)$ is a **linear polynomial**.

- Such a function $S(x)$ is **piecewise linear**.
- If the knots $t_0, t_1, \ldots, t_n$ were given and if the coefficients a_0, b_0, a_1, b_1, $\ldots$, a_{n-1}, b_{n-1} were all known, then the evaluation of $S(x)$ at a specific x would proceed as follows:
  - Determine the interval that contains x.
  - Use the appropriate linear function for that interval.

Spline of Degree 1

- If the function S defined by (1) is continuous, we call it a **first-degree spline**.
- It is characterized by the following three properties.

Definition 1

A function S is called a **spline of degree 1** *if:*

- *The domain of S is an interval $[a, b]$.*
- *S is continuous on $[a, b]$.*
- *There is a partitioning of the interval $a = t_0 < t_1 < \cdots < t_n = b$ such that S is a linear polynomial on each subinterval $[t_i, t_{i+1}]$.*

- Outside the interval $[a, b]$, $S(x)$ is usually defined to be the **same function**
  - On the **left of** a as it is on the **leftmost subinterval** $[t_0, t_1]$: $S(x) = S_0(x)$ when $x < a$
  - On the **right of** b as it is on the **rightmost subinterval** $[t_{n-1}, t_n]$: $S(x) = S_{n-1}(x)$ when $x > b$
- **Continuity** of a function f at a point s can be defined by the condition

$$\lim_{x \to s^-} f(x) = f(s) = \lim_{x \to s^+} f(x)$$

Example 1

Determine whether this function is a first-degree spline function:

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

- The function is obviously **piecewise linear**, but is **not** a spline of degree 1, because it is **discontinuous** at $x = 0$.
- Notice that

$$\lim_{x \to 0^-} S(x) = \lim_{x \to 0} (1 - x) = 1$$

$$\lim_{x \to 0^+} S(x) = \lim_{x \to 0} x = 0$$

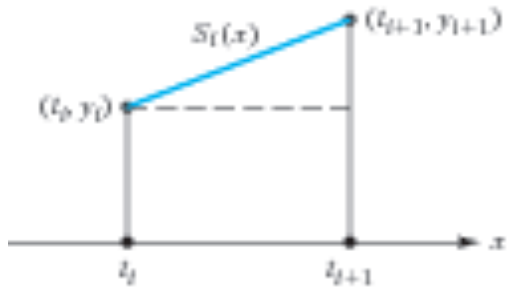
- The spline functions of degree 1 can be used for **interpolation**.
- Suppose the following table of function values is given:

x	t_0	t_1	$\cdots$	t_n
y	y_0	y_1	$\cdots$	y_n

- There is **no loss of generality** in supposing that

$$t_0 < t_1 < \cdots < t_n$$

because this is only a matter of labeling the knots.



First-degree spline: linear $S_i(x)$

- By referring to the figure and using the **point-slope form of a line**, we obtain

$$S_i(x) = y_i + m_i(x - t_i) \tag{3}$$

on the interval $[t_i, t_{i+1}]$, where m_i is the **slope** of the line and is therefore given by the formula

$$m_i = \frac{y_{i+1} - y_i}{t_{i+1} - t_i}$$

- The form of (3) is better than that of (2) for the practical evaluation of $S(x)$ because some of the quantities $x - t_i$ must be computed in any case simply to **determine which subinterval contains** x.
- If $t_0 \leq x \leq t_n$ then the interval $[t_i, t_{i+1}]$ containing x is characterized by the fact that $x - t_i$ is the **first** of the quantities $x - t_{n-1}, x - t_{n-2}, \ldots, x - t_0$ that is **nonnegative**.

- The following is a function procedure that utilizes $n+1$ table values (t_i, y_i) in linear arrays (t_i) and (y_i), assuming that $a = t_0 < t_1 < \cdots < t_n = b$.
- Given an x value, the routine returns $S(x)$ using (1) and (3).
- If $x < t_0$, then
$$S(x) = y_0 + m_0(x - t_0)$$
- If $x > t_n$, then
$$S(x) = y_{n-1} + m_{n-1}(x - t_{n-1})$$

real function Spline1 $(n, (t_i), (y_i), x)$
integer i, n; **real** x; **real array** $(t_i)_{0:n}, (y_i)_{0:n}$
for $i = n - 1$ **to** 0 **step** -1
  if $x - t_i \geq 0$ **then** exit loop
Spline1 $\leftarrow y_i + (x - t_i)[(y_{i+1} - y_i)/(t_{i+1} - t_i)]$
end function Spline1

Modulus of Continuity

- To assess the **goodness of fit** when we interpolate a function with a first-degree spline, it is useful to have something called the **modulus of continuity** of a function f.
- Suppose f is defined on an interval $[a, b]$.
- The **modulus of continuity** of f is

$$\omega(f; h) = \sup\{|f(u) - f(v)|: a \leq u \leq v \leq b, |u - v| \leq h\}$$

- Here, sup is the **supremum**, which is the least upper bound of the given set of real numbers.

- The quantity $\omega(f;h)$ measures how much f can change over a small interval of width h.
- If f is continuous on $[a,b]$, then it is uniformly continuous, and $\omega(f;h)$ will tend to zero as h tends to zero.
- If f is not continuous, $\omega(f;h)$ will not tend to zero.
- If f is differentiable on (a,b) (in addition to being continuous on $[a,b]$) and if $f'(x)$ is bounded on (a,b), then the Mean Value Theorem can be used to get an estimate of the modulus of continuity:
- If u and v are as described in the definition of $\omega(f;h)$, then

$$|f(u)-f(v)| = |f'(c)(u-v)| \leq M_1|u-v| \leq M_1 h$$

- Here, M_1 denotes the maximum of $|f'(x)|$ as x runs over (a,b).
- For example, if $f(x)=x^3$ and $[a,b]=[1,4]$, then we find that $\omega(f;h) \leq 48h$.

First-Degree Polynomial Accuracy Theorem

Theorem 1

If p is the first-degree polynomial that interpolates a function f at the endpoints of an interval $[a, b]$, then with $h = b - a$, we have

$$|f(x) - p(x)| \leq \omega(f; h) \quad (a \leq x \leq b)$$

First-Degree Spline Accuracy Theorem

- From this basic result, one can easily prove the following one, simply by applying the basic inequality to each subinterval.

Theorem 2

Let p be a first-degree spline having knots $a = x_0 < x_1 < \cdots < x_n = b$. If p interpolates a function f at these knots, then with $h = \max_i(x_i - x_{i-1})$, we have

$$|f(x) - p(x)| \leq \omega(f; h) \quad (a \leq x \leq b)$$

- The first theorem tells us that if more knots are inserted in such a way that the maximum spacing h goes to zero, then the corresponding first-degree spline will converge uniformly to f .

- Recall that this type of result is conspicuously lacking in the polynomial interpolation theory.

- In that situation, raising the degree and making the nodes fill up the interval will not necessarily ensure that convergence takes place for an arbitrary continuous function.

Spline of Degree 2

Definition 2

A function Q is called a **spline of degree 2** *if:*

- *The domain of Q is an interval $[a, b]$.*
- *Q and Q' are continuous on $[a, b]$.*
- *There are points t_i (called* **knots***) such that $a = t_0 < t_1 < \cdots < t_n = b$ and Q is a polynomial of degree at most 2 on each subinterval $[t_i, t_{i+1}]$.*

- In brief, a **quadratic spline** is a continuously differentiable piecewise quadratic function, where **quadratic** includes all linear combinations of the basic functions $x \mapsto 1, x, x^2$.

Example 2

Determine whether the following function is a quadratic spline:

$$Q(x) = \begin{cases} x^2 & (-10 \leq x \leq 0) \\ -x^2 & (0 \leq x \leq 1) \\ 1 - 2x & (1 \leq x \leq 20) \end{cases}$$

- The function is obviously **piecewise quadratic**.
- Whether Q and Q' are continuous at the interior knots can be determined as follows:

$$\lim_{x\to 0^-} Q(x) = \lim_{x\to 0^-} x^2 = 0, \qquad \lim_{x\to 0^+} Q(x) = \lim_{x\to 0^+} (-x^2) = 0$$

$$\lim_{x\to 1^-} Q(x) = \lim_{x\to 1^-} (-x^2) = -1, \qquad \lim_{x\to 1^+} Q(x) = \lim_{x\to 1^+} (1-2x) = -1$$

$$\lim_{x\to 0^-} Q'(x) = \lim_{x\to 0^-} 2x = 0, \qquad \lim_{x\to 0^+} Q'(x) = \lim_{x\to 0^+} (-2x) = 0$$

$$\lim_{x\to 1^-} Q'(x) = \lim_{x\to 1^-} (-2x) = -2, \qquad \lim_{x\to 1^+} Q'(x) = \lim_{x\to 1^+} (-2) = -2$$

- Consequently, $Q(x)$ is a **quadratic spline**.

Interpolating Quadratic Spline Q(x)

- Quadratic splines are **rarely** used for interpolation, and the discussion here is provided only as preparation for the study of cubic splines.

- Suppose that a table of values has been given:

x	t_0	t_1	t_2	$\cdots$	t_n
y	y_0	y_1	y_2	$\cdots$	y_n

- A quadratic spline, as just described, consists of n separate quadratic functions $x \mapsto a_i x^2 + b_i x + c_i$, one for each subinterval created by the $n+1$ knots. Thus, we start with $3n$ coefficients.
- On each subinterval $[t_i, t_{i+1}]$, the quadratic spline function Q_i must satisfy the interpolation conditions $Q_i(t_i) = y_i$ and $Q_i(t_{i+1}) = y_{i+1}$.
- Since there are n such subintervals, this imposes $2n$ conditions.
- The continuity of Q does **not** add any additional conditions. (Why?)
- However, the continuity of Q' at each of the interior knots gives $n-1$ more conditions.
- Thus, we have $2n + n - 1 = 3n - 1$ conditions, or **one condition short** of the $3n$ conditions required.
- There are a variety of ways to impose this additional condition; for example, $Q'(t_0) = 0$ or $Q_0'' = 0$.

- We now derive the equations for the **interpolating quadratic spline**, $Q(x)$.
- The value of $Q'(t_0)$ is prescribed as the additional condition.
- We seek a **piecewise quadratic function**

$$Q(x) = \begin{cases} Q_0(x) & (t_0 \leq x \leq t_1) \\ Q_1(x) & (t_1 \leq x \leq t_2) \\ \vdots & \vdots \\ Q_{n-1}(x) & (t_{n-1} \leq x \leq t_n) \end{cases} \tag{4}$$

- Which is **continuously differentiable** on the entire interval $[t_0, t_n]$ and which interpolates the table; that is, $Q(t_i) = y_i$ for $0 \leq i \leq n$.

- Since Q' is continuous, we can put $z_i \equiv Q'(t_i)$.
- At present, we do not know the correct values of z_i; nevertheless, the following must be the formula for Q_i:

$$Q_i(x) = \frac{z_{i+1} - z_i}{2(t_{i+1} - t_i)}(x - t_i)^2 + z_i(x - t_i) + y_i \qquad (5)$$

- To see that this is correct, verify that

$$Q_i(t_i) = y_i, \qquad Q_i'(t_i) = z_i, \qquad Q_i'(t_{i+1}) = z_{i+1}$$

- These three conditions define the function Q_i uniquely on $[t_i, t_{i+1}]$ as given in (5).

- Now, for the quadratic spline function Q to be continuous and to interpolate the table of data, it is necessary and sufficient that

$$Q_i(t_{i+1}) = y_{i+1} \qquad \text{for } i = 0, 1, \ldots, n-1$$

  in (5).
- When this equation is written out in detail and simplified, the result is

$$z_{i+1} = -z_i + 2\left(\frac{y_{i+1} - y_i}{t_{i+1} - t_i}\right) \quad (0 \le i \le n-1) \tag{6}$$

- This equation can be used to obtain the vector $[z_0, z_1, \ldots, z_n]^T$, starting with an arbitrary value for z_0.

Quadratic Spline Interpolation at the Knots

- We summarize with an algorithm:

Algorithm 1

1. *Determine* $[z_0, z_1, \ldots, z_n]^T$ *by selecting* z_0 *arbitrarily and computing* $z_1, z_2, \ldots, z_n$ *recursively by (6).*
2. *The quadratic spline interpolating function* Q *is given by (4) and (5).*

Example 3

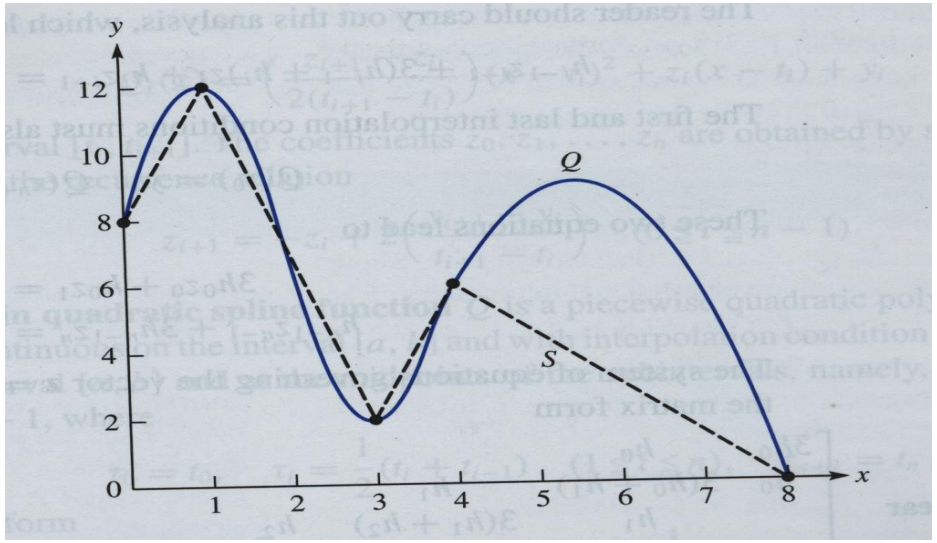
For the five data points

$$(0, 8),\ (1, 12),\ (3, 2),\ (4, 6),\ (8, 0)$$

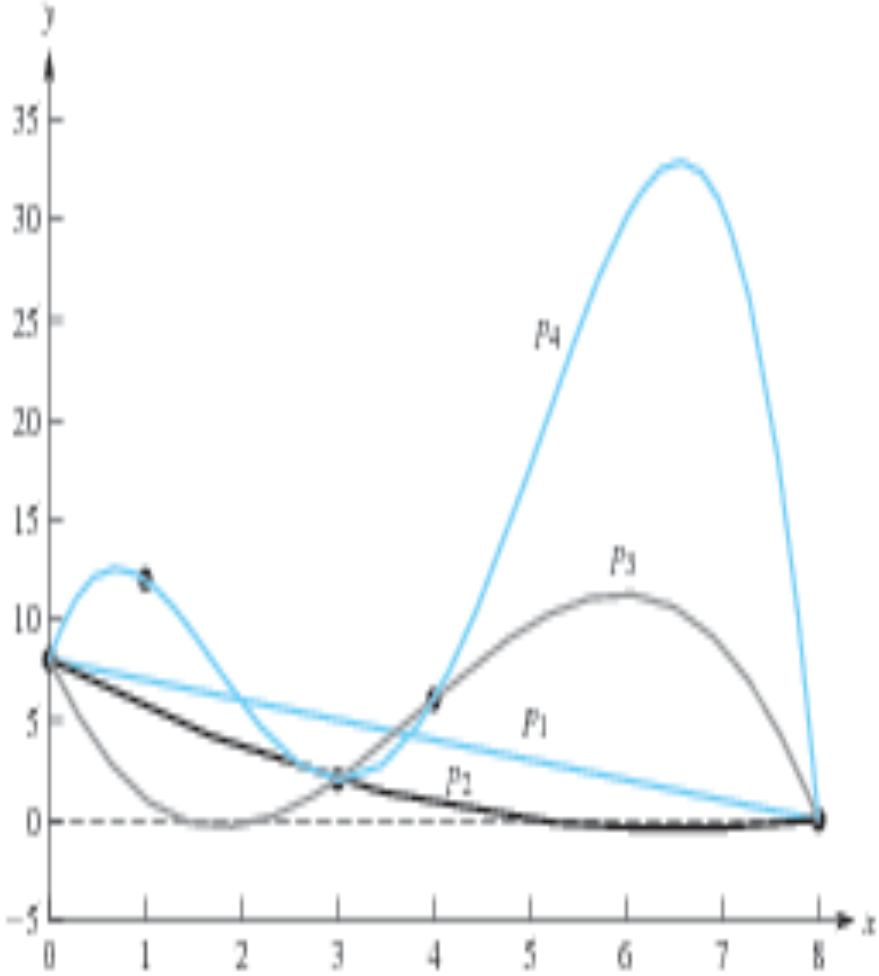
construct the linear spline S and the quadratic spline Q.

- A figure illustrates graphically these two low order spline curves.

- They fit **better than the interpolating polynomials** with regard to reduced oscillations!



First-degree and second-degree spline functions



Interpolant polynomials over data points

Drawbacks of the first- and second-degree splines

- Their low-order derivatives are discontinuous.
- In the case of the first-degree spline (or polygonal line), this lack of smoothness is immediately evident because the slope of the spline may change abruptly from one value to another at each knot.
- For the quadratic spline, the discontinuity is in the second derivative and is therefore not so evident.
- But the curvature of the quadratic spline changes abruptly at each knot, and the curve may not be pleasing to the eye.

Spline of Degree k

- The general definition of spline functions of arbitrary degree is as follows.

Definition 1

A function S is called a **spline of degree k** if:

- The domain of S is an interval $[a, b]$.
- $S, S', S'', \ldots, S^{(k-1)}$ are all continuous functions on $[a, b]$.
- There are points t_i (the knots of S) such that $a = t_0 < t_1 < \cdots < t_n = b$ and such that S is a polynomial of degree at most k on each subinterval $[t_i, t_{i+1}]$.

Higher-order splines

- If we want the approximating spline to have a continuous m^{th} derivative, a spline of degree at least m + 1 is selected.

- Higher-degree splines are used whenever more smoothness is needed in the approximating function.

- The choice of degree most frequently made for a spline function is **3**. The resulting splines are termed **cubic splines**.

Cubic Splines

- We join cubic polynomials together in such a way that the resulting spline function has two continuous derivatives everywhere.
- At each knot, three continuity conditions will be imposed.
- Since S, S′, and S′′ are continuous, the graph of the function will appear smooth to the eye.
- Discontinuities, of course, will occur in the third derivative but cannot be easily detected visually, which is one reason for choosing degree 3.

Cubic Splines (cont.)

- Experience has shown, moreover, that using splines of degree greater than 3 seldom yields any advantage.
- For technical reasons, odd-degree splines behave better than even-degree splines (when interpolating at the knots).
- Finally, a very elegant theorem shows that in a certain precise sense, the cubic interpolating spline function is the best interpolating function available.
- Thus, our emphasis on the cubic splines is well justified.

Natural Cubic Spline

- We turn next to interpolating a given table of function values by a cubic spline whose knots coincide with the values of the independent variable in the table.
- As earlier, we start with the table:

x	t_0	t_1	$\cdots$	t_n
y	y_0	y_1	$\cdots$	y_n

- The t_i's are the knots and are assumed to be arranged in ascending order.

- The function S that we wish to construct consists of n cubic polynomial pieces:

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

- In this formula, S_i denotes the cubic polynomial that will be used on the subinterval $[t_i, t_{i+1}]$.
- The interpolation conditions are

$$S(t_i) = y_i \quad (0 \le i \le n)$$

- The continuity conditions are imposed only at the **interior** knots $t_1, t_2, \ldots, t_{n-1}$.

- These conditions are written as

$$\lim_{x \to t_i^-} S^{(k)}(t_i) = \lim_{x \to t_i^+} S^{(k)}(t_i) \quad (k = 0, 1, 2)$$

- It turns out that two more conditions must be imposed to use all the degrees of freedom available.
- The choice that we make for these two extra conditions is

$$S''(t_0) = S''(t_n) = 0 \tag{2}$$

- The resulting spline function is then termed a **natural cubic spline**.

- Additional ways to close the system of equations for the spline coefficients are **periodic cubic splines** and **clamped cubic splines**.
- A clamped spline is a spline curve whose slope is fixed at both end points:
  $S'(t_0) = d_0$
  $S'(t_n) = d_n$
- A periodic cubic spline has
  $S(t_0) = S(t_n)$
  $S'(t_0) = S'(t_n)$
  $S''(t_0) = S''(t_n)$
- For all continuous differential functions, clamped and natural cubic splines yield the least oscillations about the function f that it interpolates.

- We now verify that the number of conditions imposed equals the number of coefficients available.
- There are $n + 1$ knots and hence n subintervals.
- On each of these subintervals, we shall have a different cubic polynomial.
- Since a cubic polynomial has four coefficients, a total of $4n$ coefficients are available.

- As for the conditions imposed, we have specified that within each interval the interpolating polynomial must go through two points, which gives $2n$ conditions.

- The continuity adds no additional conditions.

- The first and second derivatives must be continuous at the $n - 1$ interior points, for $2(n - 1)$ more conditions.

- The second derivatives must vanish at the two endpoints for a total of $2n + 2(n - 1) + 2 = 4n$ conditions.

Example 2

Derive the equations of the natural cubic interpolating spline for the following table:

x	-1	0	1
y	1	2	-1

- Our approach is to determine the parameters a, b, c, d, e, f, g, and h so that $S(x)$ is a natural cubic spline, where

$$S(x) = \begin{cases} S_0(s) = ax^3 + bx^2 + cx + d & x \in [-1, 0] \\ S_1(s) = ex^3 + fx^2 + gx + h & x \in [0, 1] \end{cases}$$

where the two cubic polynomials are $S_0(x)$ and $S_1(x)$.

x	-1	0	1
y	1	2	-1

- From these interpolation conditions, we have interpolation conditions
$S(-1) = S_0(-1) = -a + b - c + d = 1$
$S(0) = S_0(0) = d = 2$
$S(0) = S_1(0) = h = 2$
$S(1) = S_1(1) = e + f + g + h = -1$
- Taking the first derivatives, we obtain

$$S'(x) = \begin{cases} S_0'(x) = 3ax^2 + 2bx + c \\ S_1'(x) = 3ex^2 + 2fx + g \end{cases}$$

- From the continuity condition of S', we have $S_0'(0) = S_1'(0)$, and we set $c = g$.
- Next taking the second derivatives, we obtain

$$S''(x) = \begin{cases} S_0''(x) = 6ax + 2b \\ S_1''(s) = 6ex + 2f \end{cases}$$

x	-1	0	1
y	1	2	-1

- From the continuity condition of S'', we have $S_0''(0) = S_1''(0)$, and we let $b = f$.
- For S to be a natural cubic spline, we must have $S_0''(-1) = 0$, $S_1''(1) = 0$, and we obtain $3a = b$ and $3e = -f$.
- From all of these equations, we obtain
  $a = -1,\ b = -3,\ c = -1, d = 2,$
  $e = 1,\ f = -3,\ g = -1,\ h = 2$

Algorithm for Natural Cubic Spline

- From the previous example, it is evident that we need to develop a systematic procedure for determining the formula for a natural cubic spline, given a table of interpolation values.
- This is our objective in the material on the next several pages.
- Since S'' is continuous, the numbers

$$z_i \equiv S''(t_i) \quad (0 \leq i \leq n)$$

  are unambiguously defined.
- We do not yet know the values $z_1, z_2, \ldots, z_{n-1}$, but, of course, $z_0 = z_n = 0$ by (2).

- If the z_i's were known, we could construct S as now described.
- On the interval $[t_i, t_{i+1}]$, S'' is a linear polynomial that takes the values z_i and z_{i+1} at the endpoints.
- Thus,

$$S_i''(x) = \frac{z_{i+1}}{h_i}(x - t_i) + \frac{z_i}{h_i}(t_{i+1} - x) \tag{3}$$

  with $h_i = t_{i+1} - t_i$ for $0 \leq i \leq n - 1$.
- To verify that (3) is correct, notice that
  $S_i''(t_i) = z_i$
  $S_i''(t_{i+1}) = z_{i+1}$
  S_i'' is linear in x
- If this is integrated twice, we obtain S_i itself:

$$S_i(x) = \frac{z_{i+1}}{6h_i}(x - t_i)^3 + \frac{z_i}{6h_i}(t_{i+1} - x)^3 + cx + d$$

  where c and d are constants of integration.

- By adjusting the integration constants, we obtain a form for S_i that is easier to work with, namely,

$$S_i(x) = \frac{z_{i+1}}{6h_i}(x - t_i)^3 + \frac{z_i}{6h_i}(t_{i+1} - x)^3 + C_i(x - t_i) + D_i(t_{i+1} - x) \quad (4)$$

  where C_i and D_i are constants.
- If we differentiate (4) twice, we obtain (3).

- The interpolation conditions $S_i(t_i) = y_i$ and $S_i(t_{i+1}) = y_{i+1}$ can be imposed now to determine the appropriate values of C_i and D_i.
- The reader should do so and verify that the result is

$$\begin{aligned} S_i(x) = {} & \frac{z_{i+1}}{6h_i}(x - t_i)^3 + \frac{z_i}{6h_i}(t_{i+1} - x)^3 \\ & + \left(\frac{y_{i+1}}{h_i} - \frac{h_i}{6}z_{i+1}\right)(x - t_i) + \left(\frac{y_i}{h_i} - \frac{h_i}{6}z_i\right)(t_{i+1} - x) \end{aligned} \tag{5}$$

- When the values $z_0, z_1, \ldots, z_n$ have been determined, the spline function $S(x)$ is obtained from equations of this form for $S_0(x), S_1(x), \ldots, S_{n-1}(x)$.
- We now show how to determine the z_i's.
- One condition remains to be imposed—namely, the continuity of S'.
- At the interior knots t_i for $1 \leq i \leq n-1$, we must have $S'_{i-1}(t_i) = S'_i(t_i)$, as can be seen in the figure.

- We have, from (5),

$$S_i'(x) = \frac{z_{i+1}}{2h_i}(x - t_i)^2 - \frac{z_i}{2h_i}(t_{i+1} - x)^2 + \frac{y_{i+1}}{h_i} - \frac{h_i}{6}z_{i+1} - \frac{y_i}{h_i} + \frac{h_i}{6}z_i$$

- This gives

$$S_i'(t_i) = -\frac{h_i}{6}z_{i+1} - \frac{h_i}{3}z_i + b_i \tag{6}$$

where

$$b_i = \frac{1}{h_i}(y_{i+1} - y_i) \tag{7}$$

- Analogously, we have

$$S_{i-1}'(t_i) = \frac{h_{i-1}}{6}z_{i-1} + \frac{h_{i-1}}{3}z_i + b_{i-1}$$

- When these are set equal to each other, the resulting equation can be rearranged as

$$h_{i-1}z_{i-1} + 2(h_{i-1} + h_i)z_i + h_i z_{i+1} = 6(b_i - b_{i-1})$$

for $1 \leq i \leq n-1$.

- By letting

$$\begin{aligned} u_i &= 2(h_{i-1} + h_i) \\ v_i &= 6(b_i - b_{i-1}) \end{aligned} \tag{8}$$

we obtain a tridiagonal system of equations:

$$\begin{cases} z_0 = 0 \\ h_{i-1}z_{i-1} + u_i z_i + h_i z_{i+1} = v_i \qquad (1 \leq i \leq n-1) \\ z_n = 0 \end{cases} \tag{9}$$

to be solved for the z_i's.

- The simplicity of the first and last equations is a result of the natural cubic spline conditions $S''(t_0) = S''(t_n) = 0$.

- Now consider (9) in matrix form:

$$\begin{bmatrix} 1 & 0 & & & & \\ h_0 & u_1 & h_1 & & & \\ & h_1 & u_2 & h_2 & & \\ & & \ddots & \ddots & \ddots & \\ & & & h_{n-2} & u_{n-1} & h_{n-1} \\ & & & & 0 & 1 \end{bmatrix} \begin{bmatrix} z_0 \\ z_1 \\ z_2 \\ \vdots \\ z_{n-1} \\ z_n \end{bmatrix} = \begin{bmatrix} 0 \\ v_1 \\ v_2 \\ \vdots \\ v_{n-1} \\ 0 \end{bmatrix}$$

- On eliminating the first and last equations, we have

$$\begin{bmatrix} u_1 & h_1 & & & \\ h_1 & u_2 & h_2 & & \\ & \ddots & \ddots & \ddots & \\ & & h_{n-3} & u_{n-2} & h_{n-2} \\ & & & h_{n-2} & u_{n-1} \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \\ \vdots \\ z_{n-2} \\ z_{n-1} \end{bmatrix} = \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_{n-2} \\ v_{n-1} \end{bmatrix} \tag{10}$$

which is a symmetric tridiagonal system of order $n-1$.

One Question

- Can this algorithm fail because of divisions by zero?

$$\begin{bmatrix} u_1 & h_1 & & & \\ h_1 & u_2 & h_2 & & \\ & \ddots & \ddots & \ddots & \\ & & h_{n-3} & u_{n-2} & h_{n-2} \\ & & & h_{n-2} & u_{n-1} \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \\ \vdots \\ z_{n-2} \\ z_{n-1} \end{bmatrix} = \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_{n-2} \\ v_{n-1} \end{bmatrix} \tag{10}$$

which is a symmetric tridiagonal system of order $n-1$.

Example 3

Repeat Exp 1 by constructing the natural cubic spline through the points $(-1, 1), (0, 2)$, and $(1, -1)$.
Also, plot the results in order to visualize the spline curve.

- From the given values, we have
  $t_0 = -1, t_1 = 0, t_2 = 1, y_0 = 1, y_1 = 2, y_2 = -1$
- Consequently, we obtain
  $h_0 = t_1 - t_0 = 1,\ h_1 = t_2 - t_1 = 1,$
  $b_0 = (y_1 - y_0)/h_0 = 1,$
  $b_1 = (y_2 - y_1)/h_1 = -3,$
  $u_1 = 2(h_0 + h_1) = 4,$
  $v_1 = 6(b_1 - b_0) = -24.$
- Then the tridiagonal system (9) is

$$\begin{cases} z_0 = 0 \\ z_0 + 4z_1 + z_2 = -24 \\ z_2 = 0 \end{cases}$$

- Evidently, we obtain the solution
  $z_0 = 0, z_1 = -6,\ z_2 = 0$

x	-1	0	1
y	1	2	-1

$$S_i(x) = \frac{z_{i+1}}{6h_i}(x - t_i)^3 + \frac{z_i}{6h_i}(t_{i+1} - x)^3 + \left(\frac{y_{i+1}}{h_i} - \frac{h_i}{6}z_{i+1}\right)(x - t_i) + \left(\frac{y_i}{h_i} - \frac{h_i}{6}z_i\right)(t_{i+1} - x) \tag{5}$$

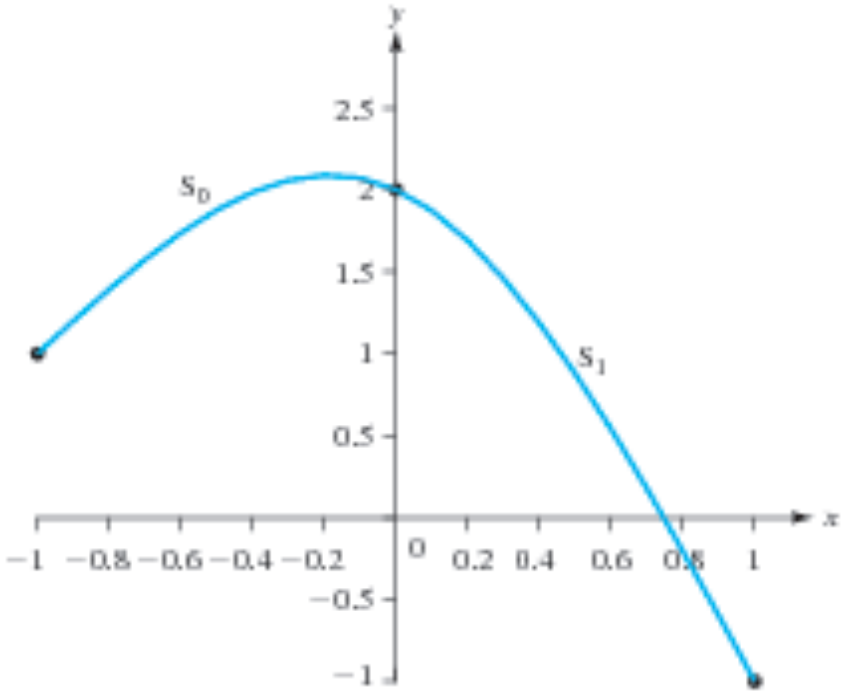
- From (5), we have

$$S(x) = \begin{cases} S_0(x) = -(x+1)^3 + 3(x+1) - x & x \in [-1, 0] \\ S_1(x) = -(1-x)^3 - x + 3(1-x) & x \in [0, 1] \end{cases}$$

or

$$S(x) = \begin{cases} S_0(x) = -x^3 - 3x^2 - x + 2 & x \in [-1, 0] \\ S_1(x) = x^3 - 3x^2 - x + 2 & x \in [0, 1] \end{cases}$$

- This agrees with the results from Exp 1.
- The resulting natural spline curve through the given points is shown in the next figure.



Natural cubic spline for Exp 1 and 2

Recaps

$$z_i \equiv S''(t_i) \quad (0 \le i \le n)$$

$$b_i = \frac{1}{h_i}(y_{i+1} - y_i) \quad h_i = t_{i+1} - t_i \text{ for } 0 \le i \le n-1.$$

$$S_i(x) = \frac{z_{i+1}}{6h_i}(x - t_i)^3 + \frac{z_i}{6h_i}(t_{i+1} - x)^3 + \left(\frac{y_{i+1}}{h_i} - \frac{h_i}{6}z_{i+1}\right)(x - t_i) + \left(\frac{y_i}{h_i} - \frac{h_i}{6}z_i\right)(t_{i+1} - x) \tag{5}$$

- By letting

$$u_i = 2(h_{i-1} + h_i)$$

$$v_i = 6(b_i - b_{i-1}) \tag{8}$$

we obtain a tridiagonal system of equations:

$$\begin{cases} z_0 = 0 \\ h_{i-1}z_{i-1} + u_i z_i + h_i z_{i+1} = v_i \qquad (1 \le i \le n-1) \\ z_n = 0 \end{cases} \tag{9}$$

to be solved for the z_i's.

- The simplicity of the first and last equations is a result of the natural cubic spline conditions $S''(t_0) = S''(t_n) = 0$.

- Now consider (9) in matrix form:

$$\begin{bmatrix} 1 & 0 & & & & \\ h_0 & u_1 & h_1 & & & \\ & h_1 & u_2 & h_2 & & \\ & & \ddots & \ddots & \ddots & \\ & & & h_{n-2} & u_{n-1} & h_{n-1} \\ & & & & 0 & 1 \end{bmatrix} \begin{bmatrix} z_0 \\ z_1 \\ z_2 \\ \vdots \\ z_{n-1} \\ z_n \end{bmatrix} = \begin{bmatrix} 0 \\ v_1 \\ v_2 \\ \vdots \\ v_{n-1} \\ 0 \end{bmatrix}$$

- On eliminating the first and last equations, we have

$$\begin{bmatrix} u_1 & h_1 & & & \\ h_1 & u_2 & h_2 & & \\ & \ddots & \ddots & \ddots & \\ & & h_{n-3} & u_{n-2} & h_{n-2} \\ & & & h_{n-2} & u_{n-1} \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \\ \vdots \\ z_{n-2} \\ z_{n-1} \end{bmatrix} = \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_{n-2} \\ v_{n-1} \end{bmatrix} \tag{10}$$

which is a symmetric tridiagonal system of order $n-1$.

- We could use procedure **Tri** to solve this system or design an algorithm specifically for it.
- In Gaussian elimination **without pivoting**, the forward elimination phase would modify the u_i's and v_i's as follows:

$$\begin{cases} u_i \leftarrow u_i - \dfrac{h_{i-1}^2}{u_{i-1}} \\ v_i \leftarrow v_i - \dfrac{h_{i-1}v_{i-1}}{u_{i-1}} \end{cases} \quad (i = 2, 3, \ldots, n-1)$$

- The back substitution phase yields

$$\begin{cases} z_{n-1} & \leftarrow & \dfrac{v_{n-1}}{u_{n-1}} \\ z_i & \leftarrow & \dfrac{v_i - h_i z_{i+1}}{u_i} \quad (i = n-2, n-3, \ldots, 1) \end{cases}$$

- Putting all this together leads to the following algorithm, designed especially for the tridiagonal (10).

Solving the Natural Cubic Spline Tridiagonal System Directly

Algorthm 1

Given the interpolation points (t_i, y_i) *for* $i = 0, 1, \ldots, n$*:*

1. *Compute for* $i = 0, 1, \ldots, n-1$*:*

   $$h_i = t_{i+1} - t_i; \quad b_i = (y_{i+1} - y_i)/h_i$$

2. *Set* $u_1 = 2(h_0 + h_1); \quad v_1 = 6(b_1 - b_0)$

   Compute inductively for $i = 2, 3, \ldots, n-1$*:*

   $$u_i = 2(h_i + h_{i-1}) - h_{i-1}^2/u_{i-1}$$
   $$v_i = 6(b_i - b_{i-1}) - h_{i-1}v_{i-1}/u_{i-1}$$

3. *Set* $z_n = 0; \quad z_0 = 0$

   Compute inductively for $i = n-1, n-2, \ldots, 1$*:*

   $$z_i = v_i - h_i z_{i+1}/u_i$$

$$S_i(x) = \frac{z_{i+1}}{6h_i}(x - t_i)^3 + \frac{z_i}{6h_i}(t_{i+1} - x)^3 + \left(\frac{y_{i+1}}{h_i} - \frac{h_i}{6}z_{i+1}\right)(x - t_i) + \left(\frac{y_i}{h_i} - \frac{h_i}{6}z_i\right)(t_{i+1} - x) \tag{5}$$

- (5) is not the best computational form for evaluating the cubic polynomial $S_i(x)$.
- We would prefer to have it in the form

$$S_i(x) = A_i + B_i(x - t_i) + C_i(x - t_i)^2 + D_i(x - t_i)^3 \tag{11}$$

because nested multiplication can then be utilized.

- Notice that (11) is the Taylor expansion of S_i about the point t_i.
- Hence

$$A_i = S_i(t_i), \quad B_i = S_i'(t_i), \quad C_i = \tfrac{1}{2}S_i''(t_i), \quad D_i = \tfrac{1}{6}S_i'''(t_i)$$

- Therefore
  $A_i = y_i$, $C_i = z_i/2$
- The coefficient of x^3 in (11) is D_i, whereas the coefficient of x^3 in (5) is $(z_{i+1} - z_i)/6h_i$.

- Therefore, we obtain

$$D_i = \frac{1}{6h_i}(z_{i+1} - z_i)$$

- Finally, (6) provides the value of $S_i'(t_i)$, which is

$$B_i = -\frac{h_i}{6}z_{i+1} - \frac{h_i}{3}z_i + \frac{1}{h_i}(y_{i+1} - y_i)$$

- Thus, the nested form of $S_i(x)$ is

$$S_i(x) = y_i + (x - t_i)\left(B_i + (x - t_i)\left(\frac{z_i}{2} + \frac{1}{6h_i}(x - t_i)(z_{i+1} - z_i)\right)\right) \quad (12)$$

Pseudocode for Natural Cubic Splines

- We now write routines for determining a natural cubic spline based on a table of values and for evaluating this function at a given value.
- First, we use Algorithm 1 for directly solving the tridiagonal (10).
- This procedure, called **Spline3_Coef**, takes $n+1$ table values (t_i, y_i) in arrays (t_i) and (y_i) and computes the z_i's, storing them in array (z_i).
- Intermediate (working) arrays (h_i), (b_i), (u_i), and (v_i) are needed.

procedure Spline3_Coef $(n, (t_i), (y_i), (z_i))$
integer i, n; **real array** $(t_i)_{0:n}, (y_i)_{0:n}, (z_i)_{0:n}$
for $i = 0$ **to** $n - 1$
  $h_i \leftarrow t_{i+1} - t_i$
  $b_i \leftarrow (y_{i+1} - y_i)/h_i$
$u_1 \leftarrow 2(h_0 + h_1)$
$v_1 \leftarrow 6(b_1 - b_0)$
for $i = 2$ **to** $n - 1$
  $u_i \leftarrow 2(h_i + h_{i-1}) - h_{i-1}^2/u_{i-1}$
  $v_i \leftarrow 6(b_i - b_{i-1}) - h_{i-1}v_{i-1}/u_{i-1}$
$z_n \leftarrow 0$
for $i = n - 1$ **to** 1 **step** -1
  $z_i \leftarrow (v_i - h_i z_{i+1})/u_i$
$z_0 \leftarrow 0$
end procedure Spline3_Coef

- Now a procedure called **Spline3_Eval** is written for evaluating (12), the natural cubic spline function $S(x)$, for x a given value.
- The procedure **Spline3_Eval** first determines the interval $[t_i, t_{i+1}]$ that contains x and then evaluates $S_i(x)$ using the nested form of this cubic polynomial:

real function **Spline3_Eval** $(n, (t_i), (y_i), (z_i), x)$
integer i; **real** h, tmp
real array $(t_i)_{0:n}, (y_i)_{0:n}, (z_i)_{0:n}$
for $i = n - 1$ **to** 0 **step** -1
  if $x - t_i \geq 0$ **then** exit loop
$h \leftarrow t_{i+1} - t_i$
$tmp \leftarrow (z_i/2) + (x - t_i)(z_{i+1} - z_i)/(6h)$
$tmp \leftarrow -(h/6)(z_{i+1} + 2z_i) + (y_{i+1} - y_i)/h + (x - t_i)(tmp)$
Spline3_Eval $\leftarrow y_i + (x - t_i)(tmp)$
end function **Spline3_Eval**

- The function **Spline3_Eval** can be used repeatedly with different values of x after one call to procedure **Spline3_Coef**.
- For example, this would be the procedure when plotting a natural cubic spline curve.
- Since procedure **Spline3_Coef** stores the solution of the tridiagonal system corresponding to a particular spline function in the array (z_i), the arguments $n, (t_i), (y_i)$, and (z_i) must not be altered between repeated uses of **Spline3_Eval**.

Space Curves

- In two dimensions, two cubic spline functions can be used together to form a **parametric representation** of a complicated curve that turns and twists.
- Select points on the curve and label them $t = 0, 1, \ldots, n$.
- For each value of t, read off the x- and y-coordinates of the point, thus producing a table:

t	0	1	$\cdots$	n
x	x_0	x_1	$\cdots$	x_n
y	y_0	y_1	$\cdots$	y_n

- Then fit $x = S(t)$ and $y = \overline{S}(t)$, where S and $\overline{S}$ are natural cubic spline interpolants.
- The two functions S and $\overline{S}$ give a parametric representation of the curve.

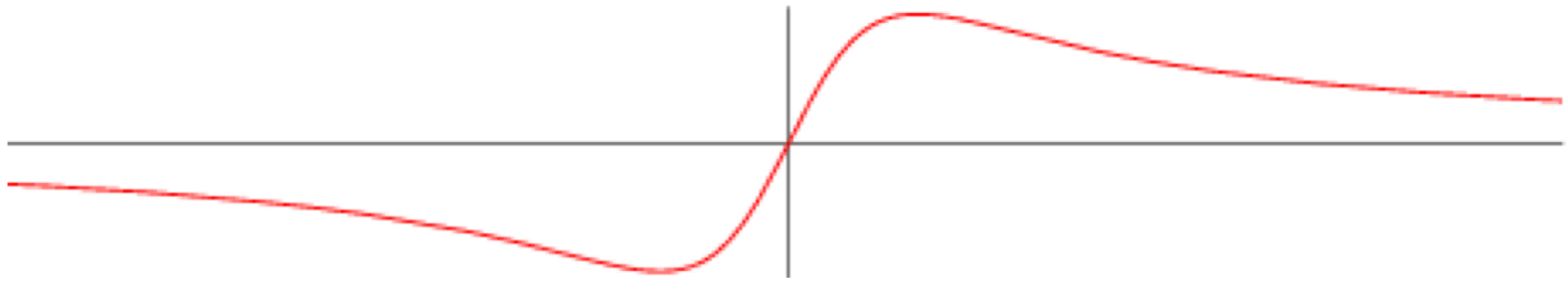
Serpentine Curve

A curve named and studied by Newton in 1701 and had been studied earlier by L'Hospital and Huygens in 1692.

Example 5

Select 13 points on the well-known **serpentine curve** given by

$$y = \frac{x}{1/4 + x^2}$$



Example 5

Select 13 points on the well-known **serpentine curve** given by

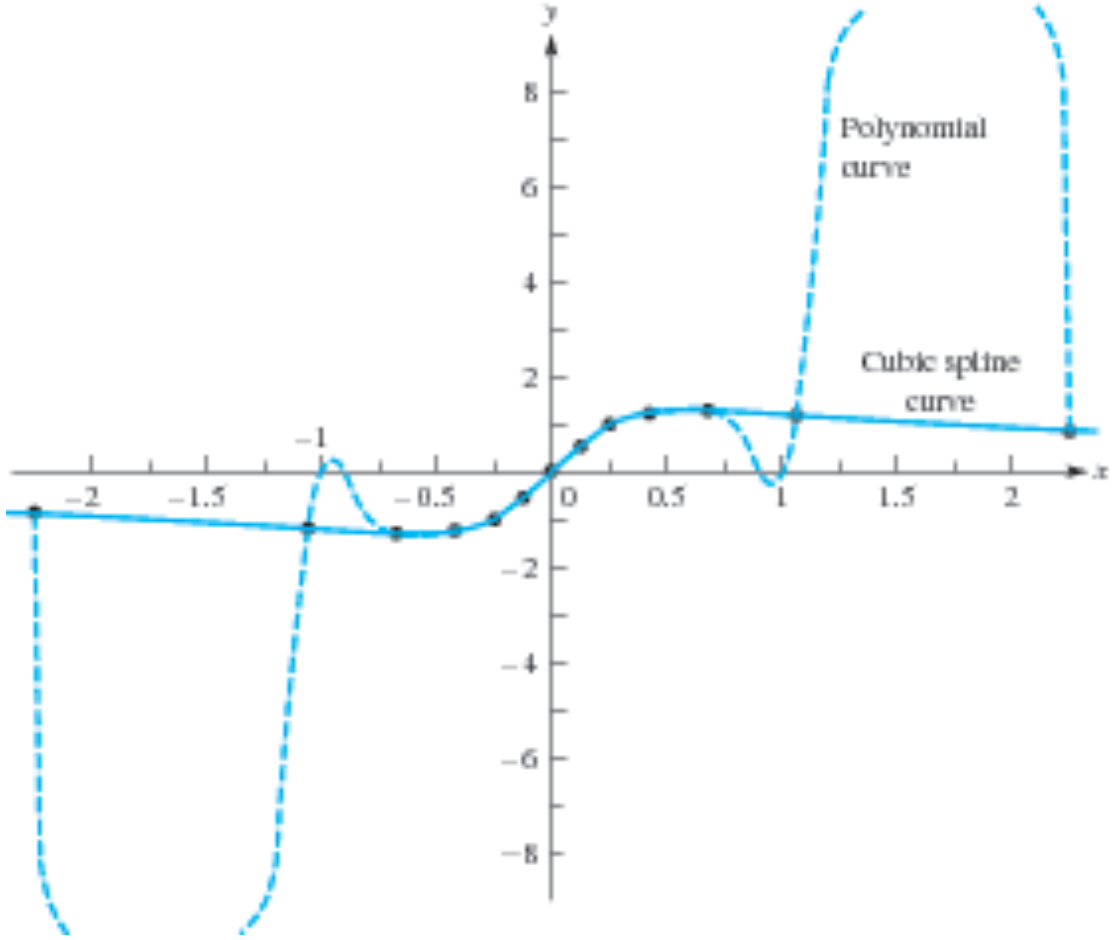
$$y = \frac{x}{1/4 + x^2}$$

So that the knots will not be equally spaced, write the curve in parametric form:

$$\begin{cases} x = \frac{1}{2}\tan\theta \\ y = \sin 2\theta \end{cases}$$

and take $\theta = i(\pi/14)$, where $i = -6, -5, \ldots, 5, 6$. Plot the natural cubic spline curve and the interpolation polynomial in order to compare them.

- This is example of curve fitting using both the polynomial interpolation routines **Coef** and **Eval** and the cubic spline routines **Spline3_Coef** and **Spline3_Eval**.
- The next figure shows the resulting cubic spline curve and the high-degree polynomial curve (dashed line) from an automatic plotter.
- The polynomial becomes extremely erratic after the fourth knot from the origin and oscillates wildly, whereas the spline is a near perfect fit.

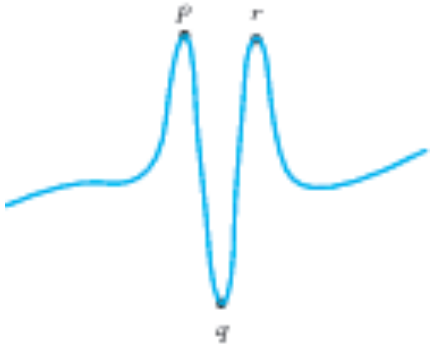


Serpentine curve

Smoothness Property

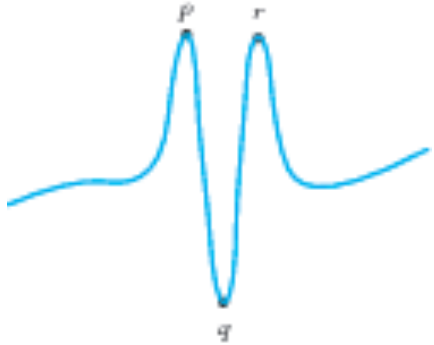
- **Why do spline functions serve the needs of data fitting better than ordinary polynomials?**
- To answer this, one should understand that interpolation by polynomials of high degree is often unsatisfactory because polynomials may exhibit wild **oscillations**.
- Polynomials are smooth in the technical sense of possessing continuous derivatives of all orders, whereas in this sense, spline functions are **not** smooth.

- Wild oscillations in a function can be attributed to its derivatives being very large.
- Consider the function whose graph is shown in the next figure.
- The slope of the chord that joins the points p and q is very large in magnitude.
- By the Mean-Value Theorem, the slope of that chord is the value of the derivative at some point between p and q.
- Thus, the derivative must attain large values.
- Indeed, somewhere on the curve between p and q, there is a point where $f'(x)$ is large and negative.



Wildly oscillating function

- Similarly, between q and r, there is a point where $f'(x)$ is large and positive.
- Hence, there is a point on the curve between p and r where $f''(x)$ is large.
- This reasoning can be continued to higher derivatives if there are more oscillations.
- This is the behavior that spline functions do **not** exhibit.
- In fact, the following result shows that from a certain point of view, natural cubic splines are the **best** functions to use for curve fitting.



Wildly oscillating function

Cubic Spline Smoothness Theorem

Theorem 1

If S is the natural cubic spline function that interpolates a twice-continuously differentiable function f at knots $a = t_0 < t_1 < \cdots < t_n = b$, then

$$\int_a^b [S''(x)]^2\,dx \leq \int_a^b [f''(x)]^2\,dx$$

- The interpretation of the integral inequality in the theorem is that the average value of $[S''(x)]^2$ on the interval $[a, b]$ is never larger than the average value of this expression with any twice-continuous function f that agrees with S at the knots.
- The quantity $[f''(x)]^2$ is closely related to the curvature of the function f.