Lecture 3
Interpolation

1 Introduction to Interpolation

Interpolation is a process of finding a function whose graph will pass through a given set of discrete points

$$\{(x_0, y_0), (x_1, y_1), (x_2, y_2), \dots (x_n, y_2)\} = \{(x_i, y_i)\}_{i=0}^{n}.$$

Example 1 *Let $x_0 = 0$, $x_1 = \frac{\pi}{4}$ and $x_2 = \frac{\pi}{2}$ and $y_i = \cos x_i$, $i = 0, 1, 2$. Thus, we have three data points:*

$$(0, 1), \quad \left(\frac{\pi}{4}, \frac{\sqrt{2}}{2}\right), \quad \left(\frac{\pi}{2}, 0\right).$$

We look for a quadratic polynomial $P_2(x) = a_2x^2 + a_1x + a_0$ such that its graph passes through these points. In other words, we want

$$\begin{aligned} P_2(x_0) &= y_0, \\ P_2(x_1) &= y_1, \\ P_2(x_2) &= y_2. \end{aligned}$$

Thus, quadratic polynomial P_2 must satisfy the following conditions

$$\begin{aligned} P_2(0) &= 1, \\ P_2\left(\frac{\pi}{4}\right) &= \frac{\sqrt{2}}{2}, \\ P_2\left(\frac{\pi}{2}\right) &= 0. \end{aligned}$$

The graph of polynomial P_2 is given in the figure below. In what follows we will derive explicit formulas for obtaining interpolation polynomials.

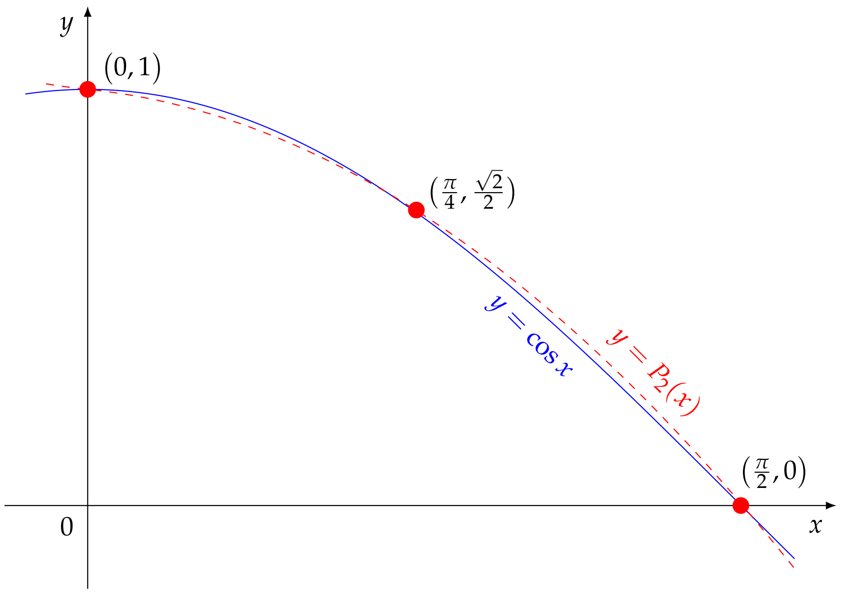


Figure 1: Interpolation example 1

2 Why do we need Interpolation?

There are two main purposes for interpolation.

The first purpose is to obtain a function (given analytically, i.e. by a formula), whose graph will pass through a given set of discrete data points. Suppose that through some measurements we gathered data y_i associated with parameters x_i and we want to find a function f with the property that its graph passes points (x_i, y_i), i.e. $f(x_i) = y_i$.

Clearly, there were many functions with this property, and therefore, we might restrict the class of functions that fit the data, for example, polynomials, piece-wise polynomials, trigonometric polynomials. Or, for example, data can come from the functions of the form

$$a_0 + a_1 e^x + a_2 e^{2x} + a_3 e^{3x} + \ldots + a_n e^{nx}.$$

The goal is then to find by interpolation the coefficients a_0, a_1, a_2, $a_3, \ldots, a_n$ that satisfy given data.

In the other case, a function may be given by a table for selected values of x and by interpolation we want to obtain the value of the function for x not in the table. For example, we have the following data table:

x	3.10	3.25	3.45	3.72	3.91	4.05
y	2.4235	2.6721	3.7953	4.9627	5.2173	6.0721

Table 1: Data table

We want to find out the value at $x = 3.3$ or at $x = 3.81$. Construct by interpolation a polynomial P and then compute $P(3.3)$ and $P(3.81)$ to find the needed values.

In modern computer graphics or engineering drawings it is often necessary to find a curve passing through given points (x_i, y_i) that is "pleasing to the eyes". How do we connect a set of points to make a smooth curve? Connecting them with straight line segments will often give a curve with many corners, whereas what was intended was a smooth curve. Later we will consider piece-wise interpolation exactly for this purpose with a particular case of spline interpolation.

The 2nd purpose of interpolation is to approximate functions with simpler ones, usually polynomials or piece-wise polynomials. For example, this can be done in order to differentiate or integrate easier a function.

As an example why this is important, consider evaluating the following integral

$$\int_0^1 \frac{1}{1+x^{10}}\,dx.$$

To do it analytically might be difficult. Instead, we could substitute the integrand function $f(x) = \frac{1}{1+x^{10}}$ with a polynomial approximation obtained by interpolation, since we know that integrating a polynomial is straightforward. To this end, consider an equally spaced partition of interval $[0, 1]$, say with 5 points:

$$x_0 = 0,\ x_1 = 0.25,\ x_2 = 0.5,\ x_3 = 0.75,\ x_4 = 1.$$

Compute $y_i = f(x_i),\ i = 0, 1, 2, 3, 4$. Compute the interpolation polynomial P passing these points ($P(x_i) = y_i$) and substitute this polynomial in the integral to compute

$$\int_0^1 \frac{1}{1+x^{10}}\,dx \approx \int_0^1 P(x)\,dx,$$

since integrating polynomials is straightforward.

As mentioned before, usually polynomials are the class of functions used in interpolation.

3 Linear Interpolation

Suppose that two data points (x_0, y_0) and (x_1, y_1) are given. It is well known that there is a unique straight line passing through these two points. In other words, there exists a linear function (a polynomial of degree 1):

$$P_1(x) = a_0 + a_1 x$$

such that

$$P_1(x_0) = y_0 \quad \text{and} \quad P_1(x_1) = y_1.$$

This polynomial P_1 can be written in various forms:

$$\begin{aligned} P_1(x) &= y_0 + \frac{y_1 - y_0}{x_1 - x_0}(x - x_0) \\ &= \frac{x - x_1}{x_0 - x_1} y_0 + \frac{x - x_0}{x_1 - x_0} y_1 \\ &= \frac{(x_1 - x)y_0 + (x - x_0)y_1}{x_1 - x_0}. \end{aligned}$$

Example 2 *Consider the table of values for function $y = \tan x$:*

x	1	1.1	1.2	1.3
$\tan x$	1.5574	1.9648	2.5722	3.6021

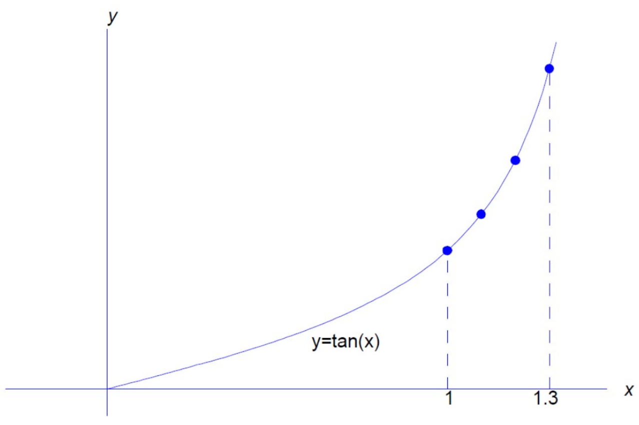


Figure 2: Data for interpolation in Example 2

We want to use interpolation to calculate $\tan 1.15$.

Use data points $(1.1,\ 1.9648)$ *and* $(1.2,\ 2.5722)$ *to write linear interpolant:*

$$\begin{aligned} P_1(x) &= y_0 + \frac{y_1 - y_0}{x_1 - x_0}(x - x_0) \\ &= 1.9648 + \frac{2.5722 - 1.9648}{1.2 - 1.1}(x - 1.1) \\ &= -4.7166 + 6.074x \end{aligned}$$

and approximate

$$\tan(1.15) \approx P_1(1.15) = 2.2685.$$

The true value is $\tan(1.15) = 2.2345$. *And in this case the error is* $2.2345 - 2.2685 = -0.034$. *Later we will derive formulas for the estimation of the error in interpolation, in order to know when we have sufficient accuracy in our interpolant.*

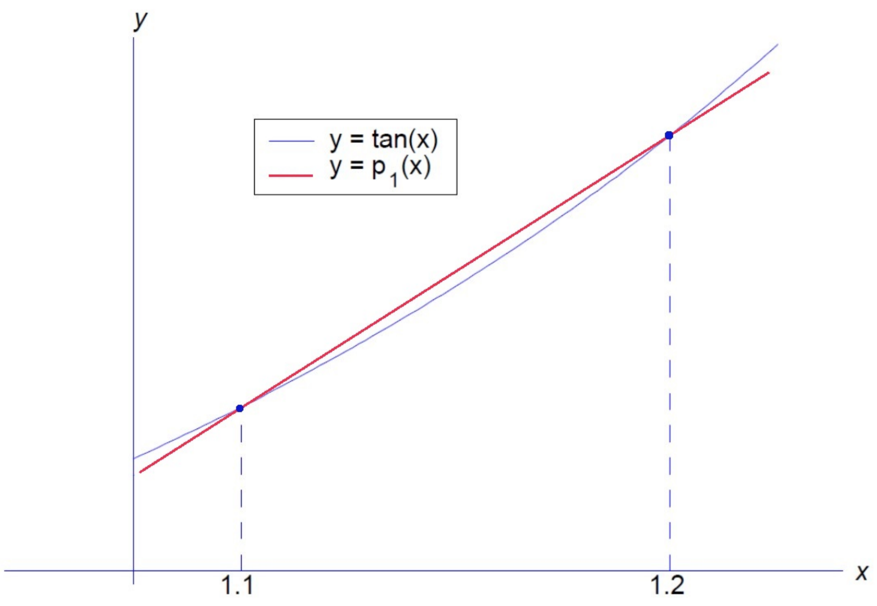


Figure 3: Linear interpolation in Example 2

4 Quadratic Interpolation

We want to find a polynomial

$$P_2(x) = a_0 + a_1 x + a_2 x^2,$$

which satisfies

$$P_2(x_i) = y_i, \quad i = 0, 1, 2$$

for given data points (x_0, y_0), (x_1, y_1) and (x_2, y_2). One formula for such a polynomial follows:

$$\boxed{P_2(x) = y_0 L_0(x) + y_1 L_1(x) + y_2 L_2(x)} \tag{1}$$

with

$$\begin{aligned} L_0(x) &= \frac{(x-x_1)(x-x_2)}{(x_0-x_1)(x_0-x_2)}, \\ L_1(x) &= \frac{(x-x_0)(x-x_2)}{(x_1-x_0)(x_1-x_2)}, \\ L_2(x) &= \frac{(x-x_0)(x-x_1)}{(x_2-x_0)(x_2-x_1)}. \end{aligned} \tag{2}$$

It can be easily verified that

$$P_2(x_0) = y_0, \quad P_2(x_1) = y_1, \quad P_2(x_2) = y_2.$$

The above formula (1)is called **Lagrange's form** of the quadratic interpolation polynomial.

4.1 Lagrange Basis Functions

The functions (2) are called **Lagrange basis functions** for quadratic interpolation. They have the properties

$$L_i(x_j) = \begin{cases} 0, & \text{if } i = j, \\ 1, & \text{if } i \neq j \end{cases}$$

for $i, j = 0, 1, 2$. Also, note that they all are polynomials of degree 2. Their graphs are presented below.

As a consequence of each $L_i(x)$ being polynomials of degree 2, we have that the interpolant

$$P_2(x) = y_0 L_0(x) + y_1 L_1(x) + y_2 L_2(x)$$

must have also degree $\leqslant 2$.

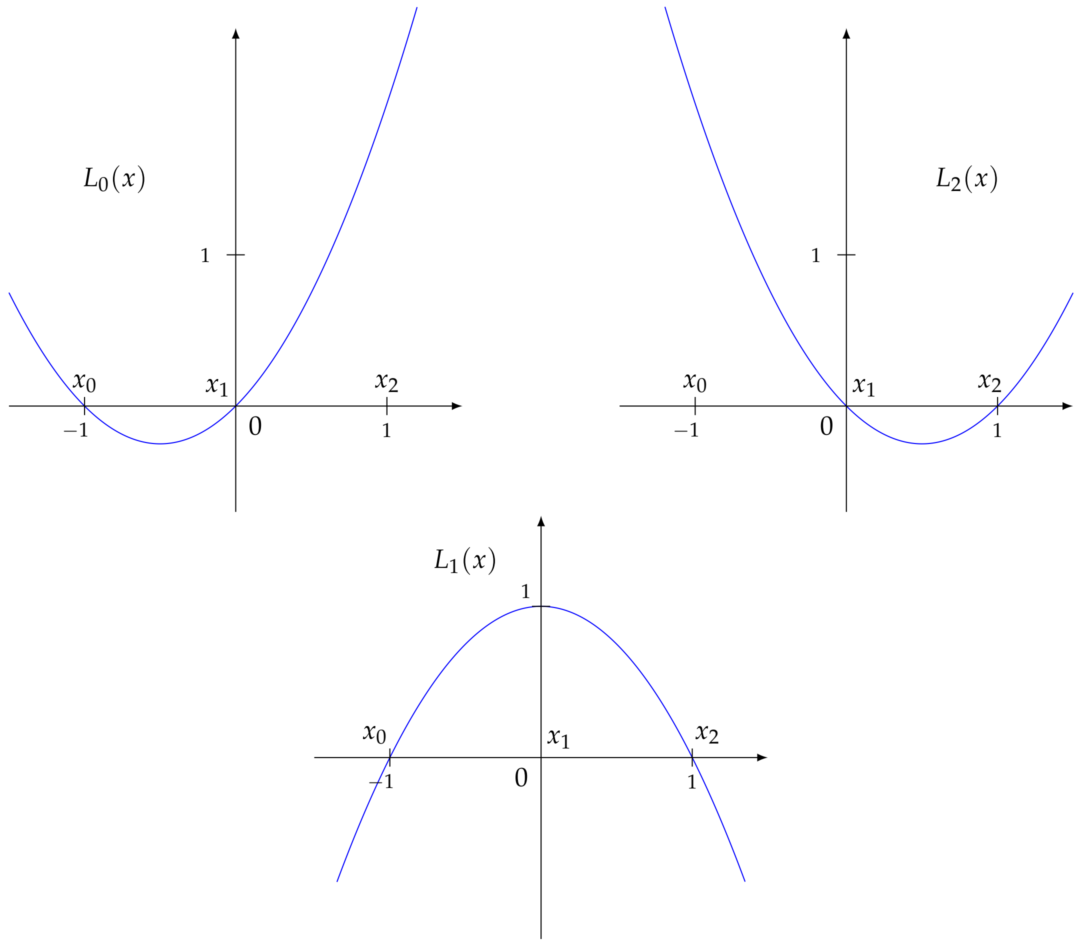


Figure 4: Graphs of Lagrange basis functions L_0,L_1 and L_2

4.2 Uniqueness

Another important question is whether interpolation polynomial $P_2(x)$ is unique or there exists another polynomial, call it $Q(x)$, for which $\deg(Q) \leqslant 2$ and

$$Q(x_i) = y_i, \quad i = 0, 1, 2.$$

Introduce

$$R(x) = P_2(x) - Q(x).$$

Since both P_2 and Q are quadratic polynomials, obviously we have $\deg(R) \leqslant 2$. Moreover,

$$R(x_i) = P_2(x_i) - Q(x_i) = y_i - y_i = 0$$

for all three node points x_0, x_1, and x_2. How many polynomials $R(x)$ are there of degree at most 2 and having three distinct zeros? The answer is that only the zero polynomial satisfies these properties, and therefore $R(x) = 0$ for all x, and consequently $Q(x) = P_2(x)$ for all x.

4.3 Special Cases

Consider the data points

$$(x_0, 1), (x_1, 1), (x_2, 1).$$

Clearly, these data points lie on the horizontal line $x = 1$. What is the polynomial $P_2(x)$ in this case?

In this case the polynomial interpolant is $P_2(x) \equiv 1$ meaning that $P_2(x)$ is the constant function 1. First of all, the constant function satisfies the property of being of degree $\leqslant 2$. Next, it clearly interpolates the given data. Therefore by the uniqueness of quadratic interpolation, $P_2(x)$ must be the constant function 1.

Consider now the data points

$$(x_0, mx_0),\ (x_1, mx_1),\ (x2, mx_2),$$

where m is some constant. Obviously, these data points lie on the straight line $y = mx$. What is $P_2(x)$ in this case? By an argument similar to that above, $P_2(x) = mx$.Thus, the degree of $P_2(x)$ can be less than 2.

5 Higher Degree Interpolation

We consider now the case of interpolation by polynomials of a general degree n. We want to find a polynomial $P_n(x)$ for which $\deg(P_n) \leqslant n$ and

$$P_n(x_i) = y_i, \quad i = 0, 1, 2, \dots, n$$

with given data points

$$(x_0, y_0),\ (x_1, y_1),\ \dots,\ (x_n, y_n).$$

The solution to this interpolation problem is given by **Lagrange's formula**:

$$\boxed{P_n(x) = y_0 L_0(x) + y_1 L_1(x) + y_2 L_2(x) + \dots + y_n L_n(x),}$$

where the **Lagrange basis functions** are given by

$$L_k = \frac{(x - x_0)(x - x_1)\dots(x - x_{k-1})(x - x_{k+1})\dots(x - x_n)}{(x_k - x_0)(x_k - x_1)\dots(x_k - x_{k-1})(x_k - x_{k+1})\dots(x_k - x_n)} \quad \text{for } k = 0, 1, 2 \dots, n.$$

Observe that Lagrange basis functions L_k for all $k = 0, 1, \dots, n$ have the property that

$$L_k(x_k) = 1, \quad \text{and } L_k(x_j) = 0, \text{ if } j \neq k.$$

In a manner analogous to the quadratic case, we can show that the above P_n is the only solution to the interpolation problem.

Example 3 *Recall the data from Example 2.*

x	1	1.1	1.2	1.3
$\tan x$	1.5574	1.9648	2.5722	3.6021

We used the nodes 1.1 and 1.2 to get the interpolation polynomial P_1 and its value at $x = 1.15$. In the same manner, using nodes 1, 1.1 and 1.2 we can write polynomial P_2 of degree 2 and compute $P_2(1.15)$, and finally consider polynomial P_3 associated with all four nodes and its value at the same point $x = 1.15$. The results together with the corresponding errors are presented in the table below:

n	$P_n(1.15)$	Error
1	2.2685	$-3.4e-02$
2	2.2435	$-9.0e-03$
3	2.2296	$4.9e-03$

Table 2: Example 3

Note that the approximation of $\tan 1.15$ *improves with increasing degree n, but not at a very rapid rate. In fact, the error becomes worse when n is increased further. Later we will see that interpolation of a much higher degree, say $n \geqslant 10$, is often poorly behaved when the node points $\{xi\}$ are evenly spaced.*

6 Newton's divided difference

Definition 1 *Let f be a real-valued function and let x_0 and x_1 be two distinct points. A **first order divide difference** of function f is defined by*

$$f[x_0, x_1] = \frac{f(x_1) - f(x_0)}{x_1 - x_0}.$$

By the Mean-Value Theorem from Calculus, for any diffrentiable function f

$$f(x_1) - f(x_0) = f'(\xi)(x_1 - x_0)$$

for some (generally unknown) point ξ between x_0 and x_1. Thus,

$$f[x_0, x_1] = f'(\xi)$$

and the first order divided difference is very much like first derivative, especially if points x_0 and x_1 are close. In fact,

$$f[x_0, x_1] \approx f'(\tfrac{x_0+x-1}{2})$$

is quite a good approximation of the derivative.

Definition 2 *Given three distinct points x_0, x_1, and x_2, define the **second order divided difference** of function f as*

$$f[x_0, x_1, x_2] = \frac{f[x_1, x_2] - f[x_0, x_1]}{x_2 - x_0}.$$

It can be shown that

$$f[x_0, x_1, x_2] = \tfrac{1}{2} f''(\xi)$$

for some ξ betwen x_0, x_1 and x_2. Also

$$f[] = \tfrac{1}{2} f''(x_1),$$

in case that point x_i are equally spaced i.e. $x_1 - x_0 = x_2 - x_1$.

Example 4 *Consider the table of data*

x	1	1.1	1.2	1.3	1.4
$\cos x$	0.54030	0.45360	0.36236	0.26750	0.16997

Table 3: Data for Example 4

Let x_0=1, $x_1 = 1.1$ and $x_2 = 1.2$. Then

$$\begin{aligned} f[x_0, x_1] &= \frac{0.45360 - 0.54030}{1.1 - 1.0} = -0.86700, \\ f[x_1, x_2] &= \frac{0.36236 - 0.45360}{1.2 - 1.1} = -0.91240, \\ f[x_0, x_1, x_2] &= \frac{-0.91240 - (-0.86700)}{1.2 - 1.0} = -0.22700 \end{aligned}$$

For comparison,

$$f'(\tfrac{x_0+x_1}{2}) = -\sin 1.15 = -0.86742,$$
$$\tfrac{1}{2} f''(x_1) = -\tfrac{1}{2}\cos 1.1 = -0.22680.$$

Definition 3 *Given $n+1$ distinct points $x_0, x_1, \ldots, x_n$ with $n \geqslant 2$, define **n-th order divided difference** of function f by*

$$f[x_0, x_1, \ldots, x_n] = \frac{f[x_1, x_2, \ldots, x_n] - f[x_0, x_1, \ldots, x_{n-1}]}{x_n - x_0}.$$

It is a recursive definition of a divided difference of order n using divided differences of order $n-1$. It can be shown that this divided difference approximates the n-th order derivative of a function f that is n-times continuously differentiable

$$f[x_0, x_1, \dots, x_n] = \tfrac{1}{n!} f^{(n)}(\xi)$$

for some ξ in the interval $[\min\{x_0, x_1, \dots, x_n\}, \max\{x_0, x_1, \dots, x_n\}]$.

Example 5 *Consider the data table from previous example. We will use notation*

$$D^k f(x_i) = f[x_i, x_{i+1}, \dots, x_{i+k}].$$

Let's compute the corresponding divided differences

i	x_i	$f(x_i)$	$Df(x_i)$	$D^2 f(x_i)$	$D^3 f(x_i)$	$D^4 f(x_i)$
0	1.0	0.54030	−0.8670	−0.2270	0.1533	0.0125
1	1.1	0.45360	−0.9124	−0.1810	0.1583	
2	1.2	0.36236	−0.9486	−0.1355		
3	1.3	0.26750	−0.9753			
4	1.4	0.16997				

Table 4: Divided differences. Example 5

6.1 Order of nodes and coincident nodes

Looking at

$$f[x_0, x_1] = \frac{f(x_1) - f(x_0)}{x_1 - x_0} = \frac{f(x_0) - f(x_1)}{x_0 - x_1} = f[x_1, x_0]$$

we can conclude that for the 1st order divided difference the order of nodes doesn't matter. Similarly, we can rewrite the 2nd order divided difference as

$$\begin{aligned} f[x_0, x_1, x_2] &= \frac{f[x_1, x_2] - f[x_0, x_1]}{x_2 - x_0} \\ &= \frac{f(x_0)}{(x_0 - x_1)(x_0 - x_2)} + \frac{f(x_1)}{(x_1 - x_0)(x_1 - x_2)} + \frac{f(x_2)}{(x_2 - x_0)(x_2 - x_1)} \end{aligned}$$

from which it follows that the order of nodes in the second order divided difference also does not matter. In other words, mathematically we can write that

$$f[x_0, x_1, x_2] = f[x_{i_0}, x_{i_1}, x_{i_2}]$$

for any permutation (i_0, i_1, i_2) of $(0, 1, 2)$.

In general, we can show that the value of $f[x_0, x_1, \dots, x_n]$ is independent of the order of the arguments $\{x_0, x_1, \dots, x_n\}$.

Next question concerns to what happens when some of the nodes $\{x_0, x_1, \dots, x_n\}$ are not distinct. First, start by investigating what happens when they all come together as a single point x_0.

For first order divided differences, we have

$$\lim_{x_1 \to x_0} f[x_0, x_1] = \lim_{x_1 \to x_0} \frac{f(x_1) - f(x_0)}{x_1 - x_0} = f'(x_0).$$

Thus, we can extend the definition of $f[x_0, x_1]$ to coincident nodes using

$$f[x_0, x_0] = f'(x_0).$$

In a similar fashion we can show that we might use definition

$$f[x_0, x_0, x_0] = \frac{1}{2} f''(x_0)$$

and for the general case

$$f[\underbrace{x_0, x_0, \ldots, x_0}_{n+1 \text{ times}}] = \frac{1}{n!} f^{(n)}(x_0).$$

What do we do when only some of the nodes are coincident? This too can be dealt with, although we do so here only by examples. Use recursion!

$$f[x_0, x_1, x_1] = \frac{f[x_1, x_1] - f[x_0, x_1]}{x_1 - x_0} = \frac{f'(x_1) - f[x_0, x_1]}{x_1 - x_0}.$$

The recursion formula can be used in general in this way to allow all possible combinations of possibly coincident nodes.

6.2 The Newton's divided difference formula for the interpolation polynomial

Let the data values for interpolation problem be generated by a function f:

$$y_i = f(x_i), \qquad i = 0, 1, 2, \ldots, n$$

Given $n+1$ data points

$$(x_0, y_0),\ (x_1, y_1),\ (x_2, y_2),\ \ldots,\ (x_n, y_n)$$

we are looking for a polynomial P_n of degree at most n, i.e. $\deg P_n \leqslant n$ such that

$$P_n(x_i) = y_i, \qquad i = 0, 1, 2, \ldots, n.$$

One possibility is to consider the Lagrange formula given in section 5. The other way, more preferable in practice is to use the divided differences

$$f[x_0, x_1],\ f[x_0, x_1, x_2],\ \ldots,\ f[x_0, x_1, x_2, \ldots, x_n]$$

. Thus, iteratively we compute interpolation polynomials $[P_1,\ P_2,\ \ldots, P_n$ as follows

$$\begin{aligned}
P_1(x) &= f(x_0) + f[x_0, x_1](x - x_0), \\
P_2(x) &= f(x_0) + f[x_0, x_1](x - x_0) + f[x_0, x_1, x_2](x - x_0)(x - x_1), \\
&= P_1(x) + f[x_0, x_1, x_2](x - x_0)(x - x_1), \\
P_3(x) &= P_2(x) + f[x_0, x_1, x_2, x_3](x - x_0)(x - x_1)(x - x_2).
\end{aligned}$$

For the general case, we have the recursive formula for interpolation polynomial P_n of f at points $x_0, x_1, \ldots, x_n$

$$\boxed{P_n(x) = P_{n-1}(x) + f[x_0, x_1, x_2, \ldots, x_n](x - x_0)(x - x_1)\ldots(x - x_{n-1})}$$

in which P_{n-1} interpolates f at points $x_0, x_1, \ldots, x_{n-1}$.

Example 6 *Recall the table from previous example with* $D^k f(x_i) = f[x_i, x_{i+1}, \ldots, x_{i+k}], \quad k = 1, 2, 3, 4.$ *Then*

i	x_i	$f(x_i)$	$Df(x_i)$	$D^2f(x_i)$	$D^3f(x_i)$	$D^4f(x_i)$
0	1.0	0.54030	−0.8670	−0.2270	0.1533	0.0125
1	1.1	0.45360	−0.9124	−0.1810	0.1583	
2	1.2	0.36236	−0.9486	−0.1355		
3	1.3	0.26750	−0.9753			
4	1.4	0.16997				

Table 5: Divided differences computed in Example 5

$$\begin{aligned}
P_1(x) &= 0.5403 - 0.8670(x - 1), \\
P_2(x) &= P_1(x) - 0.2270(x - 1)(x - 1.1), \\
P_3(x) &= P_2(x) + 0.1533(x - 1)(x - 1.1)(x - 1.2), \\
P_4(x) &= P_3(x) + 0.0125(x - 1)(x - 1.1)(x - 1.2)(x - 1.3).
\end{aligned}$$

Observe that using Newton's divided difference we not only compute P_4, but also we got polynomial P_1 that interpolates x_0 and x_1, polynomial P_2 that interpolates points x_0, x_1 and x_2, and so on.
Using this table and these formulas, we have the following table of interpolants for the value $x = 1.05$.

The true value is $\cos(1.05) = 0.49757105$. *Let's take a look at the errors for all for interpolation polynomials*

n	$P_n(1.05)$	Error
1	0.49695	$6.20e-04$
2	0.49752	$5.00e-05$
3	0.49758	$-1.00e-05$
4	0.49757	0.0

Table 6: Example 6

In order to evaluate the divided difference interpolation polynomial introduce

$$\begin{aligned} d_1 &= f[x_0, x_1], \\ d_2 &= f[x_0, x_1, x_2], \\ d_3 &= f[x_0, x_1, x_2, x_3], \\ &\vdots \\ d_n &= f[x_0, x_1, x_2, x_3, \ldots, x_n]. \end{aligned}$$

Then the formula

$$\begin{aligned} P_n(x) = f(x_0) &+ f[x_0, x_1](x - x_0) \\ &+ f[x_0, x_1, x_2](x - x_0)(x - x_1) \\ &+ f[x_0, x_1, x_2, x_3](x - x_0)(x - x_1)(x - x_2) \\ &+ \cdots \\ &+ f[x_0, x_1, x_2, \ldots, x_n](x - x_0)(x - x_1)(x - x_2) \cdots (x - x_n) \end{aligned}$$

can be written as

$$\begin{aligned} P_n(x) = f(x_0) + (x - x_0)(d_1 + (x - x_1)(d_2 + (x - x_2)(d_3 + \cdots \\ + (x - x_{n-2})(d_n - 1 + (x - x_{n-1})d_n))) \ldots). \end{aligned}$$

Thus we have a nested polynomial evaluation, and this is quite efficient in computational cost.

Also, in terms of practical use, the Newton's divided difference formula is preferable, since it is easier to compute (we have a recursive procedure), and if more data are added for interpolation we do not need to start the computations all over again as in the case of Lagrange formula.

7 Error in interpolation

7.1 Error in linear interpolation

Let $P_1(x)$ denote the linear polynomial interpolating function f at points x_0 and x_1, with f a given function (e.g. $f(x) = \cos x$). What is the error $f(x) - P_1(x)$?

Let f be twice continuously differentable function defined on an interval $[a, b]$ which contains the points $\{x_0, x_1\}$. Then for $a \leqslant x \leqslant b$,

$$f(x) - P_1(x) = \frac{(x - x_0)(x - x_1)}{2} f''(\xi)$$

for some (generally unknown) point ξ between the minimum and maximum of points x_0, x_1 and x. If x_1 and x are close to x_0, then

$$f(x) - P_1(x) \approx \frac{(x - x_0)(x - x_1)}{2} f''(x_0).$$

Thus the error acts like a quadratic polynomial, with zeros at x_0 and x_1.

7.2 Error for general case

Theorem 1 *Let $[a,b]$ be a given interval on which function f is $n+1$-times continuously differentiable and assume the points $x_0, x_1, \ldots, x_n$, and x are in $[a,b]$. Then the error in interpolating function f by polynomial P_n is given by*

$$\boxed{f(x) - P_n(x) = \frac{(x-x_0)(x-x_1)\cdots(x-x_n)}{(n+1)!} f^{(n+1)}(\xi)}$$

with ξ some point between the minimum and maximum of the points in $\{x_0, x_1, \ldots, x_n\}$.

Define function

$$\Psi_n(x) = (x-x_0)(x-x_1)\cdots(x-x_n),$$

which is a polynomial of degree $n+1$ with roots $\{x_0, x_1, \ldots, x_n\}$. The error formula becomes

$$f(x) - P_n(x) = \frac{\Psi_n}{(n+1)!} \cdot f^{(n+1)!}(\xi)$$

As an example, consider the quadratic interpolation case with evenly spaced points

$$x_0 = -h, \quad x_1 = 0, \quad x_2 = h$$

and let $x \in [-h,h]$. Then the error formula from Theorem 1 becomes

$$\boxed{f(x) - P_2(x) = \frac{(x+h)x(x-h)}{3!} f^{(3)}(\xi) = \frac{\Psi_2(x)}{3!} f^{(3)}(\xi),}$$

where

$$\Psi_2(x) = (x-h)x(x+h)$$

and its graph is presented below

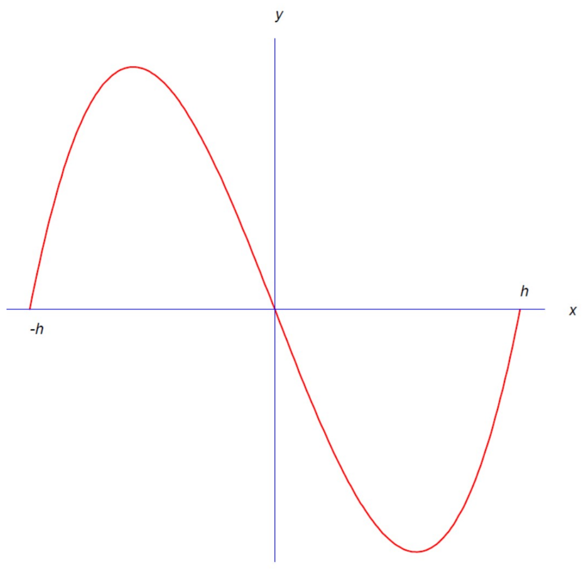


Figure 5: Function Ψ_2

In the error formula, we can bound function $|\Psi_2|$ with its maximum on interval $[-h,h]$. Also, since we do not know ξ, we could as well replace $f^{(3)}(\xi)$ with a maximum of $|f^{(3)}(x)|$ on interval $[-h,h]$. This yields

$$|f(x) - P_2(x)| \leqslant \frac{|\Psi_2(x)|}{6} \max_{-h\leqslant x\leqslant h} |f^{(3)}(x)| \leqslant \frac{1}{6} \cdot \max_{-h\leqslant x\leqslant h} |\Psi_2(x)| \cdot \max_{-h\leqslant x\leqslant h} |f^{(3)}(x)|.$$

Next let's compute

$$\max_{-h\leqslant x\leqslant h} |\Psi_2(x)|.$$

Using Calculus we can see that maximum is achieved at $x = \frac{h}{\sqrt{3}}$, and it is

$$\max_{-h\leqslant x\leqslant h} |\Psi_2(x)| = \frac{2h^3}{3\sqrt{3}}.$$

Therefore, for $-h \leqslant x \leqslant h$, the error is bounded by

$$|f(x) - P_2(x)| \leqslant \frac{h^3}{9\sqrt{3}} \max_{-h\leqslant x\leqslant h} |f^{(3)}(x)| = M\frac{h^3}{9\sqrt{3}},$$

where value of M depends on function f.

When bounding the error for the case of general n

$$|f(x) - P_n(x)| = \frac{|\Psi_n(x)|}{(n+1)!} \cdot |f^{(n+1)}(\xi)|$$

we replace $f^{(n+1)}(\xi)$ with its maximum, say M, over the interval containing $\{x, x_0, \dots, x_n\}$ (as we did previously in the linear and quadratic cases):

$$|f(x) - P_n(x)| \leqslant \frac{|\Psi_n(x)|}{(n+1)!} M.$$

It is obvious that behavior (or distribution) and size of the error is determined by the behavior and size of the function

$$\frac{\Psi_n(x)}{(n+1)!} = \frac{(x-x_0)(x-x_1)\cdots(x-x_n)}{(n+1)!}$$

over the interval determined by the minimum and maximum of the points in $\{x, x_0, \dots, x_n\}$. For evenly spaced node points on $[0,1]$, with $x_0 = 0$ and $x_n = 1$, we give below the graphs for $n = 2, 3, 4, \dots, 9$.

What can we learn from the given graphs? The interpolation nodes are determined by using

$$h = \frac{1}{n},\ x_0 = 0,\ x_1 = h,\ x_2 = 2h,\ \dots, x_= nh = 1$$

and for this case,

$$\Psi_n(x) = x(x-h)(x-2h)\cdots(x-1).$$

From the table below we can observe that the maximum

$$Q_n = \max_{x_0\leqslant x\leqslant x_n} \frac{|\Psi_n(x)|}{(n+1)!}$$

becomes smaller with increasing n.

n	Q_n	n	Q_n
1	$1.25e-01$	6	$4.76e-07$
2	$2.41e-02$	7	$2.20e-08$
3	$2.06e-03$	8	$9.11e-10$
4	$1.48e-04$	9	$3.39e-11$
5	$9.01e-06$	10	$1.15e-12$

From the graphs, there is enormous variation in the size of $\Psi_n(x)$ as x varies over interval $[0,1]$, and thus there is also enormous variation in the error as x so varies. For example, in the $n = 9$ case,

$$\max_{x_0\leqslant x\leqslant x_1} |\frac{\Psi_n(x)|}{(n+1)!} = 3.39 \cdot 10^{-11},$$

$$\max_{x_4\leqslant x\leqslant x_5} \frac{|\Psi_n(x)|}{(n+1)!} = 6.89 \cdot 10^{-13},$$

and the ratio of these two errors is approximately 49. Thus the interpolation error is likely to be around 49 times larger when $x_0 \leqslant x \leqslant x_1$ as compared to the case when $x_4 \leqslant x \leqslant x_5$. When doing table interpolation, the point x at which you are interpolating should be centrally located with respect to the interpolation nodes $\{x_0, x_1, \dots, x_n\}$ being used to define the interpolation, if possible. Otherwise we should expect a much larger error, especially close to the endpoints of the interpolation interval.

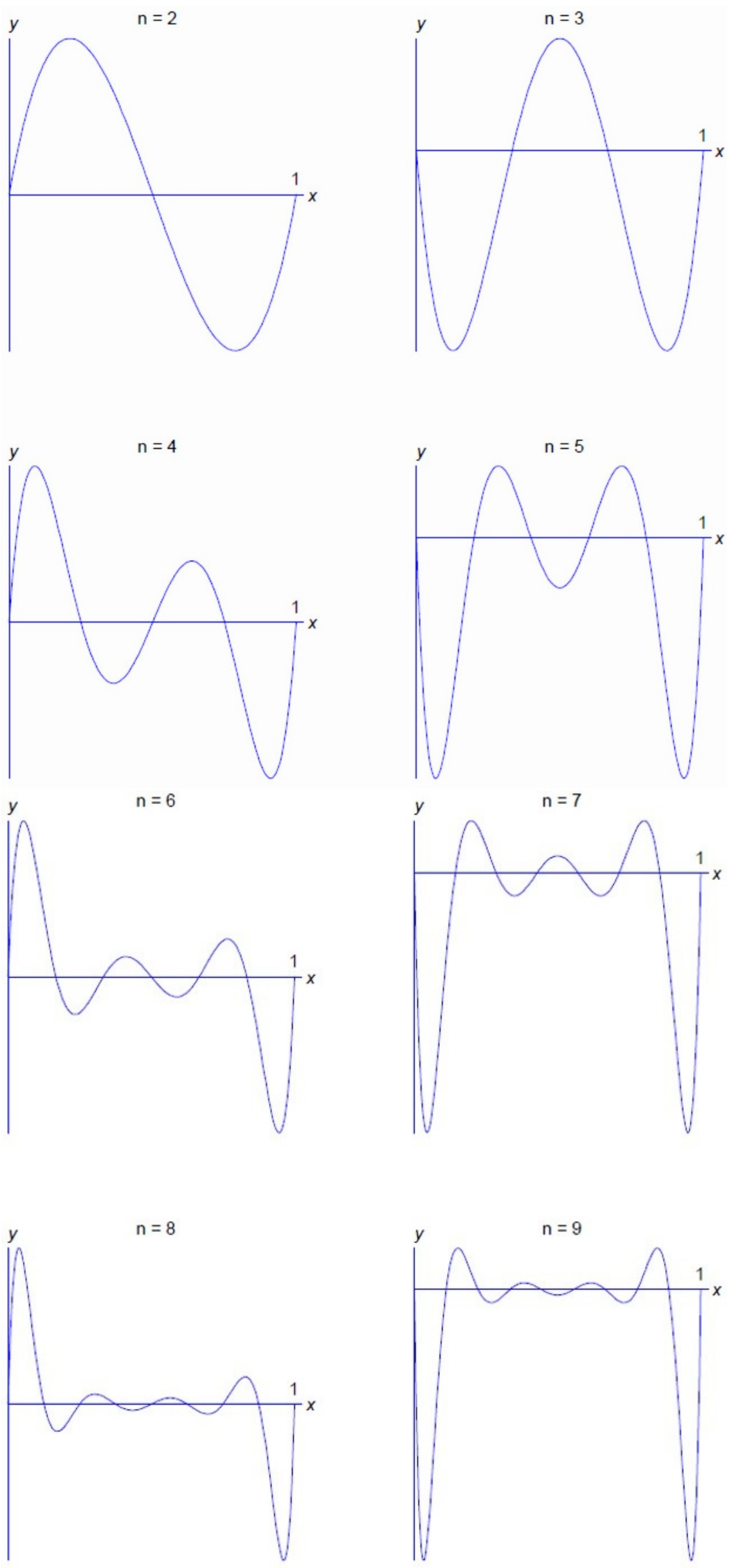


Figure 6: Graphs of functions $\Psi_n, \quad n = 2, 3, 4, \ldots, 9$

8 Approximation problem (Runge's example)

Consider now the problem of using an interpolation polynomial to approximate a given function f on a given interval $[a, b]$. In particular, take $n + 1$ interpolation nodes

$$a \leqslant x_0 < x_1 < \cdots < x_{n-1} < x_n \leqslant b$$

and produce the interpolation polynomial $P_n(x)$ (of degree at most n) that interpolates function f at the given node points. We would like to have

$$\max_{a \leqslant x \leqslant b} |f(x) - P_n(x)| \to 0 \quad \text{as } n \to \infty$$

Does it happen?

Recall the error bound for interpolation problem:

$$\max_{a \leqslant x \leqslant b} |f(x) - P_n(x)| \leqslant \max_{a \leqslant x \leqslant b} \frac{|\Psi_n(x)|}{(n+1)!} \cdot \max_{a \leqslant x \leqslant b} |f^{(n+1)}(\xi)|. \tag{E}$$

We begin with an example using evenly spaced node points. Use evenly spaced node points:

$$h = \frac{b-a}{n}, \quad x_i = a + ih \text{ for } i = 0, 1, \ldots, n$$

For some functions, such as $f(x) = e^x$, the maximum error goes to zero quite rapidly. But the size of the derivative term $f^{(n+1)}(x)$ in error estimate (E) above can badly hurt or destroy the convergence of other cases. A classical example is the so-called **Runge's example** presented below.

Example 7 *Consider function*

$$f(x) = \frac{1}{1+x^2} \quad \textit{on interval } [-5, 5]$$

and let $P_n(x)$ be its interpolation polynomial for the case $n = 10$ (in other words, on 11 data points evenly spaced on $[-5, 5]$).

In figure below, we show the graphs of f and P_{10}) on $[-5, 5]$. Observe that interpolation polynomial exhibit large variations, even bigger than the maximum of the function itself. It can be proven that for this function, the maximum error on interval $[-5, 5]$ does not converge to zero. Thus the use of evenly spaced nodes is not necessarily a good approach to approximating a function f by interpolation, even if function is smooth enough. Note that function f in this example is infinitely many times differentiable.

Recall the general error bound (E). There is nothing we really do with the derivative term for f, but we can examine the way of defining the nodes $\{x_0, x_1, \ldots, x_n\}$ within the interval $[a, b]$. We ask how these nodes can be chosen so that the maximum of $|\Psi_n(x)|$ over $[a, b]$ is made as small as possible.

This problem has quite an elegant solution, and it will be considered later. The node points $\{x_0, x_1, \ldots, x_n\}$ turn out to be the zeros of a particular polynomial $T_{n+1}(x)$ of degree $n + 1$, called a **Chebyshev polynomial**. These zeros can be computed explicitly, and with them

$$\max_{a \leqslant x \leqslant b} \frac{|\Psi_n(x)|}{(n+1)!} = \left(\frac{b-a}{2}\right)^{n+1} 2^{-n}$$

This turns out to be smaller than for evenly spaced cases; and although this polynomial interpolation does not work for all functions f, it works for all differentiable functions and more.

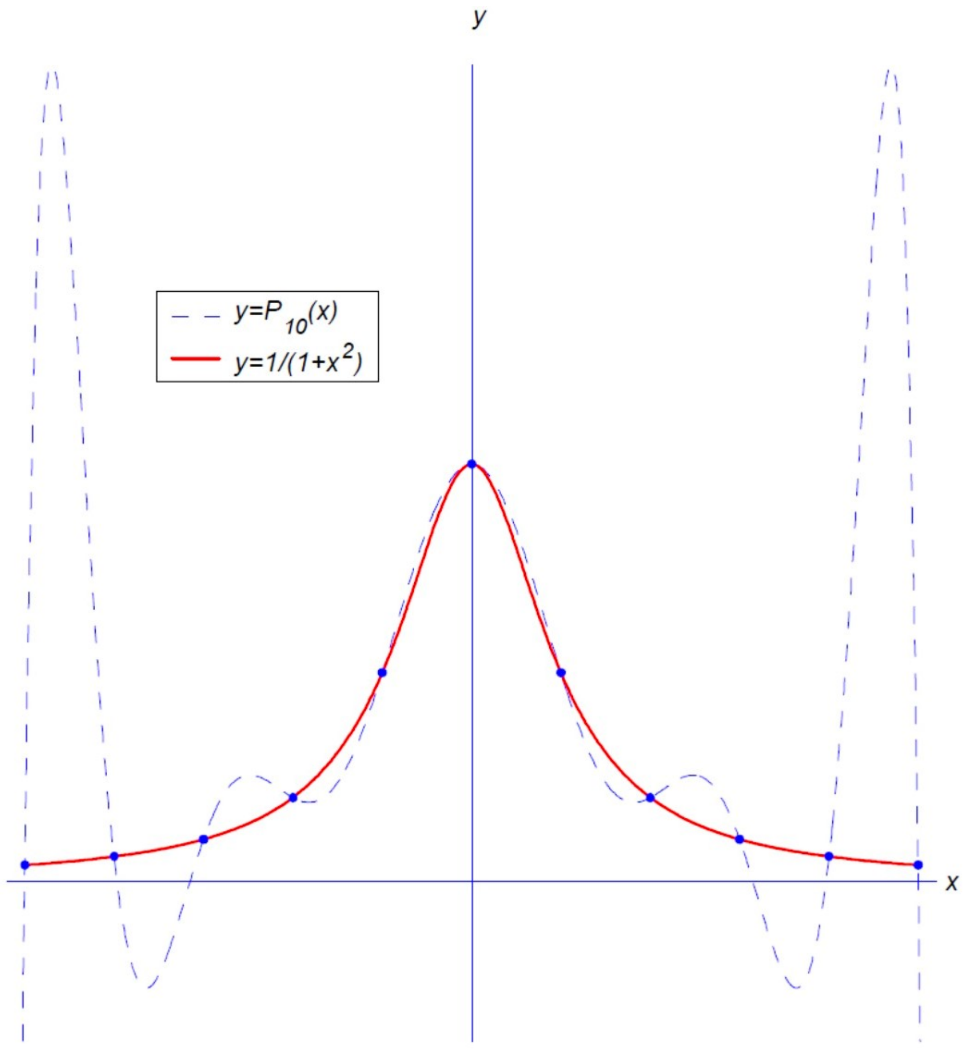


Figure 7: Runge's example for $n = 10$

9 Piece-wise polynomial interpolation

Recall the Runge's example of higher degree polynomial interpolation of the function $f(x) = \frac{1}{1+x^2}$ on interal $[-5, 5]$. The interpolants $P_n(x)$ oscillated a great deal, whereas the function f was non-oscillatory. To obtain interpolants that are better behaved, we look at other forms of interpolating functions.

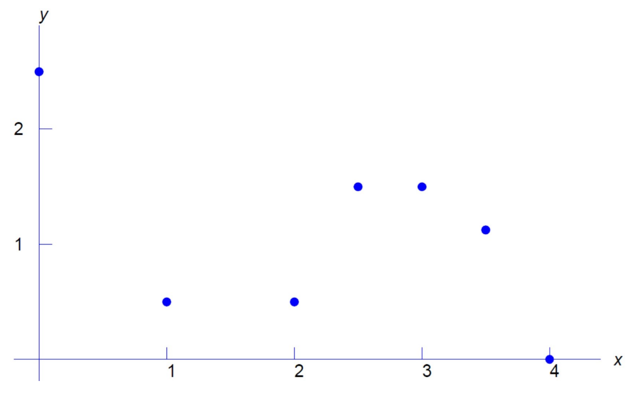
Example 8 *Consider the data from the table below*

x	0	1	2	2.5	3	3.5	4
y	2.5	0.5	0.5	1.5	1.5	1.125	0

Table 7: Data table for example 8

One possibility is to construct a polynomial of degree 6 (since we have 7 data points) shown in figure 8. As we have seen in previous sections such a polynomial might have oscillatory behavior. Especially near the endpoints. What are methods of interpolating this data, other than using a degree 6 polynomial?

Sometimes we are rather interested in the "shape" than in straight polynomial interpolation with one polynomial. One possibility is to connect the data points with straight lines which will be called piece-wise linear



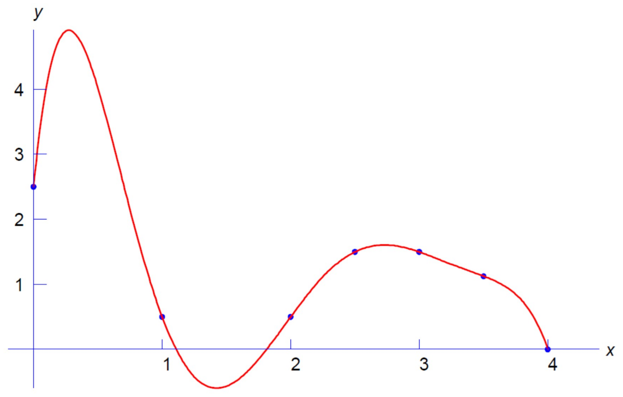
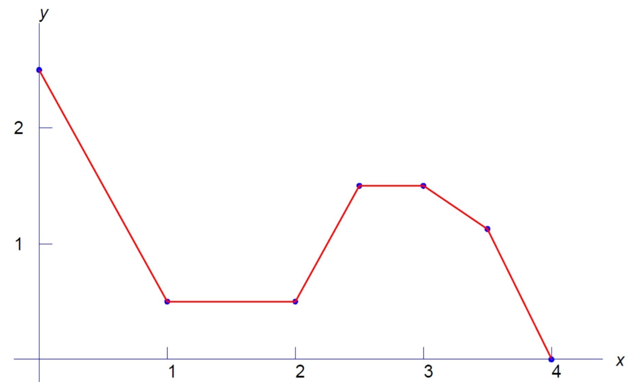


Figure 8: Data for example 8 (left) and the graph of P_6 interpolating polynomial (right)

interpolation. This piece-wise polynomial will characterize better the "shape" of data. But the graph of such a polynomial will not be *smooth*, meaning that we are having "corner" points where the interpolant is not differentiable. Next reasonable option will be to consider quadratic piece-wise polynomial interpolant. Shown in figure 9 below there are the graphs of piece-wise linear and a piece-wise quadratic interpolating functions.



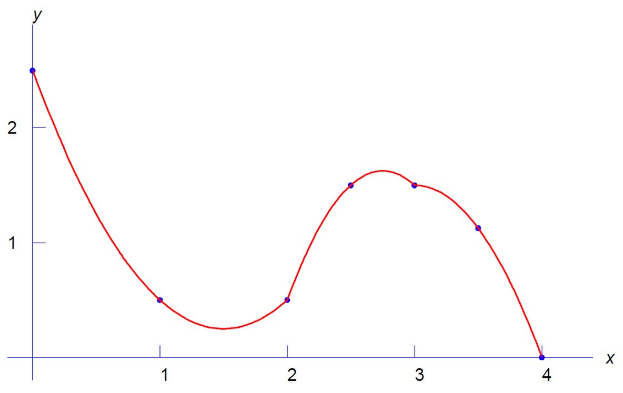


Figure 9: Piece-wise linear (left) and piece-wise quadratic (right) interpolants

Since we only have the data to consider, we would generally want to use an interpolant that had somewhat the shape of that of the piece-wise linear interpolant.

Consider being given a set of data points (x_1, y_1), (x_2, y_2), $\ldots$, (x_n, y_n), with

$$x_1 < x_2 < \cdots < x_n.$$

Then the simplest way to connect the points (x_j, y_j) is by straight line segments. This is called a **piece-wise linear interpolant** of the data $\{(x_j, y_j)\}_{j=1}^{n}$. This graph has "corners", and often we expect the interpolant to have a smooth graph.

To obtain a somewhat smoother graph, consider using piece-wise quadratic interpolation. Begin by constructing the quadratic polynomial that interpolates the first three points:

$$\{(x_1, y_1),\ (x_2, y_2),\ (x_3, y_3)\}.$$

Then construct the quadratic polynomial that interpolates the next three points

$$\{(x_3, y_3),\ (x_4, y_4),\ (x_5, y_5)\}.$$

Continue this process of constructing quadratic interpolants on the sub-intervals

$$[x_1, x_3],\ [x_3, x_5],\ [x_5, x_7], \ldots$$

If the number of sub-intervals is even (and therefore n is odd), then this process comes out fine, with the last interval being $[x_{n-2}, x_n]$. This was illustrated in figure 9 for the preceding data. If, however, n is even, then the approximation on the last interval must be handled by some modification of this procedure.

With piece-wise quadratic interpolants, however, there are "corners" on the graph of the interpolating function. With our preceding example, they are at $x_3 = 2$ and $x_5 = 3$. How do we avoid this?

Piece-wise polynomial interpolants are used in many applications. We will consider them later, to obtain numerical integration formulas.

10 Spline interpolation

Consider a set of data points (x_1, y_1), (x_2, y_2), $\ldots$, (x_n, y_n) be given in an interval $[a, b]$, with

$$a \leqslant x_1 < x_2 < \cdots < x_n \leqslant b.$$

Often we use $[a, b] = [x_1, x_n]$. Consider finding functions s for which the following properties hold:

$$\begin{array}{ll} (1) & s(x_i) = y_i, \quad i = 1, 2, \ldots, n \\ (2) & s(x),\ s'(x),\ s''(x) \text{ are continuous on } [x_1, x_n]. \end{array}$$

Then among such functions s satisfying these properties, find the one which minimizes the integral

$$\int_{x_1}^{x_n} \left|s''(x)\right|^2 dx$$

The idea of minimizing the integral is to obtain an interpolating function for which the first derivative does not change rapidly. It turns out there is a unique solution to this problem, and it is called a **natural cubic spline function**.

10.1 Spline functions

Definition 4 *Given a set of node points* $\{(x_i, y_i)\}_{i=1}^{n}$ *satisfying*

$$a \leqslant x_1 < x_2 < \cdots < x_n \leqslant b,$$

a cubic interpolating spline function *s on $[a, b]$ with "knots" $\{x_i\}$ has the following properties:*

(1) *On each sub-interval* $[a, x_1]$, $[x_1, x_2]$, $\ldots$, $[x_{n-1}, x_n]$, $[x_n, b]$, $s(x)$ *is a cubic polynomial;*
(2) $s(x)$, $s'(x)$, $s''(x)$ *are continuous on* $[a, b]$*;*
(3) $s(x_i) = y_i, \quad i = 1, 2, \ldots, n$

Return to the earlier problem of choosing an interpolating function s(x) to minimize the integral

$$\int_{x_1}^{x_n} \left|s''(x)\right|^2 dx.$$

There is a unique solution to this problem. The solution s is a cubic interpolating spline function, and moreover, it satisfies

$$s''(x_1) = s''(x_n) = 0.$$

Spline functions satisfying these *boundary conditions* are called **"natural" cubic spline functions**, and the solution to our minimization problem is a "natural cubic interpolating spline function".

We will consider next a a method to construct this function from the interpolation data. Motivation for these boundary conditions can be given by looking at the physics of bending thin beams of flexible materials to pass thru the given data. To the left of x_1 and to the right of x_n, the beam is straight and therefore the second derivatives are zero at the transition points x_1 and x_n.

For simplification purposes, let's consider four data points

$$(x_1, y_1),\ (x_2, y_2),\ (x_3, y_3),\ (x_4, y_4)$$

with $x_1 < x_2 < x_3 < x_4$. Suppose that s is a cubic interpolating spline function satisfying properties from definition 4. Then on each of the sub-intervals

$$[x_1, x_2],\ [x_2, x_3],\ [x_3, x_4]$$

s is a cubic polynomial. Since s is a cubic polynomial on each of the three sub-intervals, it follows that its second derivative s'' is a linear polynomial. Denote

$$M_i = s''(x_i), \quad i = 1, 2, 3, 4.$$

Then on the first sub-interval $[x_1, x_2]$, since s'' is linear on it, and $s''(x_1) = M_1$, and $s''(x_2) = M_2$ we have

$$s''(x) = \frac{(x_2 - x)M_1 + (x - x_1)M2}{x_2 - x_1}.$$

By integrating twice the above expression we recover function s:

$$s(x) = \frac{(x_2 - x)^3 M_1 + (x - x_1)^3 M_2}{6(x_2 - x_1)} + C_1 x + C_2.$$

Integration constants C_1 and C_2 are determined form conditions

$$s(x_1) = y_1 \text{ and } s(x_2) = y_2.$$

In the end we get that on sub-interval $[x_1, x_2]$, we have

$$s(x) = \frac{(x_2 - x)^3 M_1 + (x - x_1)^3 M_2}{6(x_2 - x_1)} + \frac{(x_2 - x)y_1 + (x - x_1)y_2}{x_2 - x_1} - \frac{x_2 - x_1}{6}\Big((x_2 - x)M_1 + (x - x_1)M_2\Big).$$

It can be checked that the above formula satisfies the interpolation conditions $s(x_1) = y_1$ and $s(x_2) = y_2$.

Repeat the same procedure on sub-intervals $[x_2, x_3]$ and $[x_3, x_4]$ to get similar formulas.

For $x_2 \leqslant x \leqslant x_3$, we have:

$$s(x) = \frac{(x_3 - x)^3 M_2 + (x - x_2)^3 M_3}{6(x_3 - x_2)} + \frac{(x_3 - x)y_2 + (x - x_3)y_3}{x_3 - x_2} - \frac{x_3 - x_2}{6}\Big((x_3 - x)M_2 + (x - x_2)M_3\Big).$$

For $x_3 \leqslant x \leqslant x_4$, we have:

$$s(x) = \frac{(x_4 - x)^3 M_3 + (x - x_3)^3 M_4}{6(x_4 - x_3)} + \frac{(x_4 - x)y_3 + (x - x_3)y_4}{x_4 - x_3} - \frac{x_4 - x_3}{6}\Big((x_4 - x)M_3 + (x - x_3)M_4\Big).$$

We still do not know the values of the second derivatives $\{M1, M2, M3, M4\}$. The above formulas guarantee that s and s'' are continuous for $x_1 \leqslant x \leqslant x_4$. For example, the formula on sub-interval $[x_1, x_2]$ yields

$$s(x_2) = y_2, \qquad s''(x_2) = M_2.$$

The formula on $[x_2, x_3]$ also yields

$$s(x_2) = y_2, \qquad s''(x_2) = M_2,$$

meaning that at node point x_2 s and s'' are continuous. All that is lacking is to make s' continuous at x_2 and x_3. Thus we require

$$\lim_{x \to x_2 + 0} s'(x) = \lim_{x \to x_2 - 0} s'(x) \quad \text{and} \quad \lim_{x \to x_3 + 0} s'(x) = \lim_{x \to x_3 - 0} s'(x).$$

To simplify even more, assume that our node points are evenly spaced with partition step h:

$$x_1,\ x_2 = x_1 + h,\ x_3 = x_1 + 2h,\ x_4 = x_1 + 3h.$$

Earlier derived formulas for spline functions simplify to

$$s(x) = \frac{(x_2-x)^3M_1 + (x-x_1)^3M_2}{6h} + \frac{(x_2-x)y_1 + (x-x_1)y_2}{h} - \frac{h}{6}\Big((x_2-x)M_1 + (x-x_1)M_2\Big)$$

for $x_1 \leqslant x \leqslant x_2$ with similar formulas for $[x_2, x_3]$ and $[x_3, x_4]$.

Skipping all algebraic manipulations, the conditions

$$\lim_{x\to x_2+0} s'(x) = \lim_{x\to x_2-0} s'(x) \quad \text{and} \quad \lim_{x\to x_3+0} s'(x) = \lim_{x\to x_3-0} s'(x)$$

will imply the following pair of equations

$$\begin{aligned} \frac{h}{6}M_1 + \frac{2h}{3}M_2 + \frac{h}{6}M_3 &= \frac{y_3-y_2}{h} - \frac{y_2-y_1}{h} \\ \frac{h}{6}M_2 + \frac{2h}{3}M_3 + \frac{h}{6}M_4 &= \frac{y_4-y_3}{h} - \frac{y_3-y_2}{h}. \end{aligned}$$

Since x_i and y_i are known, this gives us two equations in four unknowns M_1, M_2, M_3, M_4. The earlier boundary conditions on s'' gives us immediately

$$M_1 = 0 \quad \text{and} \quad M_4 = 0.$$

Therefore we can solve the linear system of two equations for two unknowns M_2 and M_3. Once M_i are known, substitute them in the formulas for $s(x)$ to get the spline function.

Example 9 *Consider the interpolation data:*

x	1	2	3	4
y	1	$\frac{1}{2}$	$\frac{1}{3}$	$\frac{1}{4}$

Table 8: Data table for example 9

Obviously the data points are lying on the graph of function $f(x) = \frac{1}{x}$. In this case $h = 1$, and linear system becomes

$$\begin{aligned} \tfrac{2}{3}M_2 + \tfrac{1}{6}M_3 &= y_3 - 2y_+ y_1 = \tfrac{1}{3} \\ \tfrac{1}{6}M_2 + \tfrac{2}{3}M_3 &= y_4 - 2y_3 + y_2 = \tfrac{1}{12}. \end{aligned}$$

This system has the solution

$$M_2 = \tfrac{1}{2} \quad \textit{and} \quad M_3 = 0.$$

Together with $M_1 = 0$ and $M_4 = 0$ this leads to the following spline function on each sub-interval:

On $[1,2]$ we get

$$s(x) = \tfrac{1}{12}(x-1)^3 - \tfrac{7}{12}(x-1) + 1.$$

Similarly, for $2 \leqslant x \leqslant 3$ we have

$$s(x) = -\tfrac{1}{12}(x-2)^3 + \tfrac{1}{4}(x-2)^2 - \tfrac{1}{3}(x-2) + \tfrac{1}{2}$$

and for $3 \leqslant x \leqslant 4$ we have

$$s(x) = -\tfrac{1}{12}(x-4) + \tfrac{1}{4}$$

Coming back to our first example at the beginning of this section, below are the graphs of 6 degree interpolation polynomial P_6 together with a natural cubic spline interpolant s

x	0	1	2	2.5	3	3.5	4
y	2.5	0.5	0.5	1.5	1.5	1.125	0

Note that cubic spline better preserves the "shape" of the data.

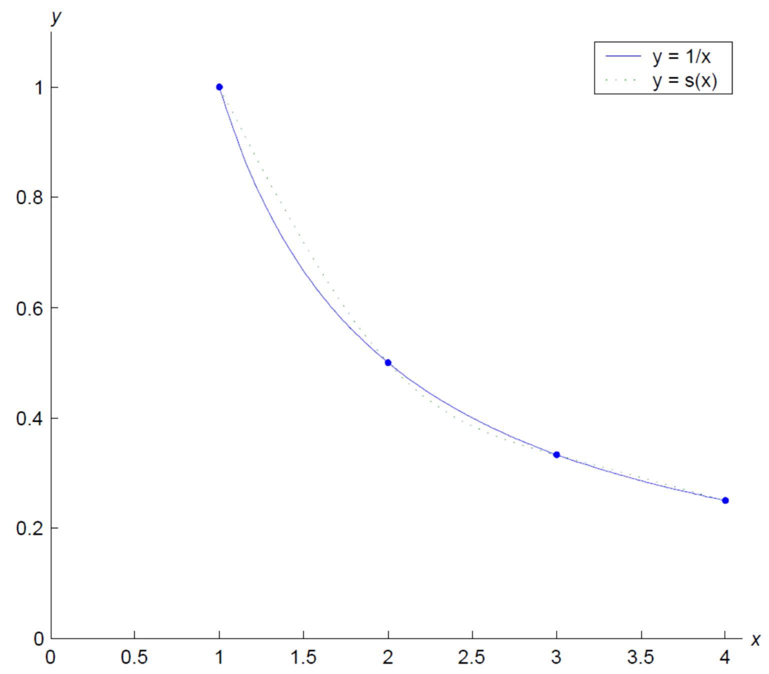


Figure 10: Natural cubic interpolating spline for example 9

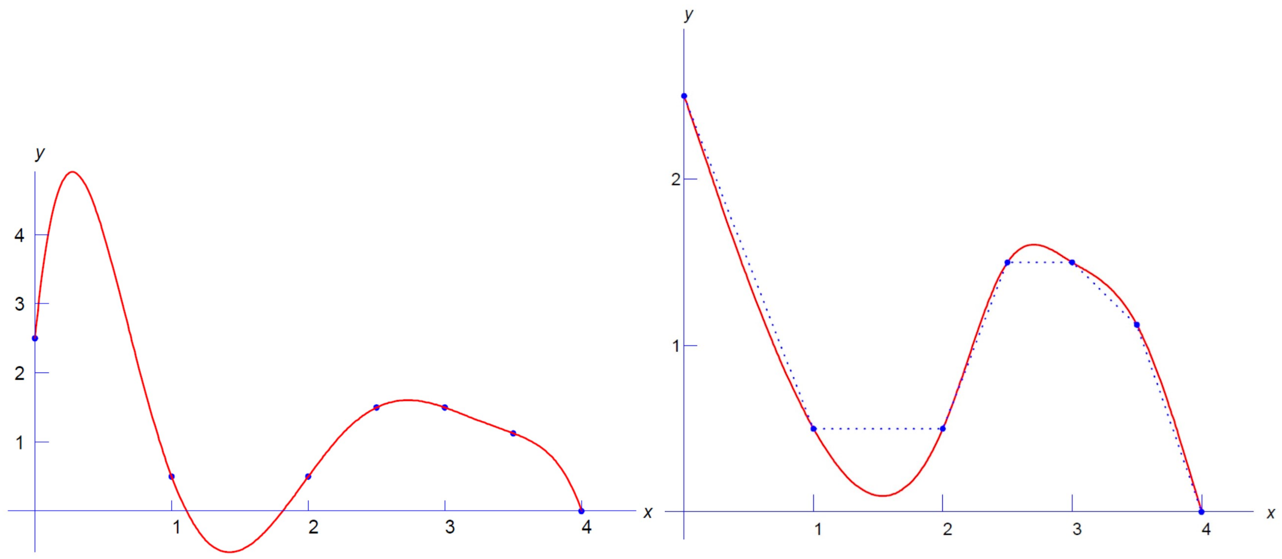


Figure 11: Polynomial P_6 (left) and natural cubic spline s (right) interpolating same data

Recall the equations needed to find unknowns M_i:

$$\frac{h}{6}M_1 + \frac{2h}{3}M_2 + \frac{h}{6}M_3 = \frac{y_3 - y_2}{h} - \frac{y_2 - y_1}{h},$$
$$\frac{h}{6}M_2 + \frac{2h}{3}M_3 + \frac{h}{6}M_4 = \frac{y_4 - y_3}{h} - \frac{y_3 - y_2}{h}.$$

Besides the natural boundary conditions $M_1 = 0 = M_4$, sometimes other boundary conditions are imposed on spline function s to help in determining the values of M_2 and M_3. For example, the data in our numerical example were generated from the function $f(x) = \frac{1}{x}$. With it, $f''(x) = \frac{2}{x^3}$, and thus we could use

$$M_1 = f'(1) = 2, \quad M_4 = f'(4) = \tfrac{1}{32}.$$

With this we are led to a new formula for spline function s, one that approximates function $f(x) = \frac{1}{x}$ more closely. This and other boundary will be introduced in the general case.

10.2 General case for spline problem

Consider the spline interpolation problem with n nodes

$$(x_1, y_1),\ (x_2, y_2), \ldots, (x_n, y_n)$$

and assume the node points $\{x_i\}$ are evenly spaced,

$$x_j = x_1 + (j-1)h, \qquad j = 1, 2, \ldots, n.$$

We have that the interpolating spline s on $x_j \leqslant x \leqslant x_{j+1}$ for $j = 1, 2, \ldots, n$ is given by

$$s(x) = \frac{(x_{j+1}-x)^3 M_j + (x-x_j)^3 M_{j+1}}{6h} + \frac{(x_{j+1}-x)y_j + (x-x_j)y_{j+1}}{h} - \frac{h}{6}\Big((x_{j+1}-x)M_j + (x-x_j)M_{j+1}\Big).$$

To enforce continuity of s' at the interior node points $x_2, x_3, \ldots, x_{n-1}$, the second derivatives $\{M_j\}$ must satisfy the linear equations

$$\frac{h}{6}M_{j-1} + \frac{2h}{3}M_j + \frac{h}{6}M_{j+1} = \frac{y_{j-1} - 2y_j + y_{j+1}}{h}$$

for $j = 2, 3, \ldots, n-1$. Writing them out,

$$\begin{aligned}
\frac{h}{6}M_1 + \frac{2h}{3}M_2 + \frac{h}{6}M_3 &= \frac{y_1 - 2y_2 + y_3}{h},\\
\frac{h}{6}M_2 + \frac{2h}{3}M_3 + \frac{h}{6}M_4 &= \frac{y_2 - 2y_3 + y_4}{h},\\
\frac{h}{6}M_3 + \frac{2h}{3}M_4 + \frac{h}{6}M_5 &= \frac{y_3 - 2y_4 + y_5}{h},\\
&\vdots\\
\frac{h}{6}M_{n-2} + \frac{2h}{3}M_{n-1} + \frac{h}{6}M_n &= \frac{y_{n-2} - 2y_{n-1} + y_n}{h}.
\end{aligned}$$

This is a system of $n-2$ linear equations in the n unknowns $\{M_1, \ldots, M_n\}$. Two more conditions must be imposed on function s in order to have the number of equations equal the number of unknowns, namely n. With the added boundary conditions, this form of linear system can be solved very efficiently.

10.3 Boundary conditions

"Natural" boundary conditions

$$s''(x_1) = 0, \quad \text{and} \quad s''(x_n) = 0.$$

Spline functions satisfying these conditions are called "natural cubic splines". They arise out the minimization problem stated earlier. But generally they are not considered as good as some other cubic interpolating splines.

"Clamped" boundary conditions. We add the conditions

$$s'(x_1) = \alpha \quad \text{and]} \quad s'(x_n) = \beta$$

with α and β two given slopes for the endpoints of $s(x)$ on $[x_1, x_n]$. This has many quite good properties when compared with the natural cubic interpolating spline, but it does require knowing the derivatives at the endpoints.

"Not a knot" boundary conditions. This is more complicated to explain, but it is the version of cubic spline interpolation that is implemented in Matlab.

As before, let the interpolation nodes be

$$(x_1, y_1),\ (x_2, y_2),\ (x_3, y_3), \ldots, (x_n, y_n)$$

We separate these points into two categories. For constructing the interpolating cubic spline function, we use the points

$$(x_1, y_1),\ (x_3, y_3), \ldots, (x_{n-2}, y_{n-2}),\ (x_n, y_n).$$

Thus deleting two of the points: (x_2, y_2) and (x_{n-1}, y_{n-1}). We now have $n-2$ points, and the interpolating spline function s can be determined on the intervals

$$[x_1, x_3],\ [x_3, x_4], \ldots, [x_{n-3}, x_{n-2}],\ [x_{n-2}, x_n].$$

This leads to $n-4$ equations in the $n-2$ unknowns $M_1, M_3, \ldots, M_{n-2}, M_n$. The two additional boundary conditions are

$$s(x_2) = y2 \quad \text{and} \quad s(x_{n-1}) = y_{n-1}.$$

These two conditons translate into two additional equations, and we obtain a system of $n-2$ linear simultaneous equations in the $n-2$ unknowns $M_1, M_3, \ldots, M_{n-2}, M_n$.

10.4 Matlab spline library

Given data points

$$(x_1, y_1),\ (x_2, y_2),\ (x_3, y_3), \ldots, (x_n, y_n)$$

type arrays containing the x and y coordinates:

$$\begin{aligned} &x = [x_1\, x_2\, \ldots\, x_n];\\ &y = [y_1\, y_2\, \ldots\, y_n];\\ &plot(x,\ y,\ 'o'); \end{aligned}$$

The last statement will draw a plot of the data points, marking them with the letter 'oh'. To find the interpolating cubic spline function and evaluate it at the points of another array xx, say

$$\begin{aligned} &h = (x_n - x_1)/(10 * n);\\ &xx = x_1 : h : x_n;\\ &yy = spline(x, y, xx);\\ &plot(x, y,' o', xx, yy); \end{aligned}$$

The last statement will plot the data points, as before, and it will plot the interpolating spline $s(x)$ as a continuous curve.

10.5 Error in cubic spline interpolation

Let an interval $[a, b]$ be given, and then define

$$h = \frac{b-a}{n-1}, \quad x_j = a + (i-1)h, \quad i = 1, 2, \ldots, n.$$

Suppose we want to approximate a given function f on the interval $[a, b]$ using cubic spline interpolation. Define

$$y_i = f(x_i), \quad i = 1, 2, \ldots, n.$$

Let s_n denote the cubic spline interpolating this data and satisfying the "not a knot" boundary conditions. Then it can be shown that for a suitable constant C,

$$E_n \equiv \max_{a \leqslant x \leqslant b} |f(x) - s_n(x)| \leqslant Ch^4.$$

The corresponding bound for natural cubic spline interpolation contains only a term of h^2 rather than h^4, so it does not converge to zero as rapidly.

Example 10 *Consider function $f(x) = \tan x$ on interval $[0, 5]$. The following table gives values of the maximum error E_n for various values of n. The values of h are being successively halved. Also the values of previus error divided by new error:*

$$R_n = \frac{1}{2} n / E_n$$

are provided. It can be observed that with halving h the error decreases by a factor between 11 and 22.

n	E_n	R_n
7	$7.09e-03$	
13	$3.24e-04$	21.9
25	$3.06e-05$	10.6
49	$1.48e-06$	20.7
97	$9.04e-08$	16.4