Most functions cannot be evaluated exactly:

$$\sqrt{x}, e^x, \ln x, \text{ trigonometric functions}$$

since by using a computer we are limited to the use of elementary arithmetic operations

$$+, -, \times, \div$$

With these operations we can only evaluate polynomials and rational functions (polynomial divided by polynomials).

# Interpolation

Given points

$$x_0, x_1, \ldots, x_n$$

and corresponding values

$$y_0, y_1, \ldots, y_n$$

find a function $f(x)$ such that

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

The interpolation function $f$ is usually taken from a restricted class of functions: **polynomials**.

## Interpolation of functions

$f(x)$

$$x_0, x_1, \ldots, x_n$$
$$f(x_0), f(x_1), \ldots, f(x_n)$$

Find a polynomial (or other special function) such that

$$p(x_i) = f(x_i), \qquad i = 0, \ldots, n.$$

What is the error $f(x) = p(x)$?

## Linear interpolation

Given two sets of points $(x_0, y_0)$ and $(x_1, y_1)$ with $x_0 \neq x_1$, draw a line through them, i.e., the graph of the linear polynomial

| $x_0$ | $x_1$ |
|---|---|
| $y_0$ | $y_1$ |

$$\ell(x) = \frac{x - x_1}{x_0 - x_1} y_0 + \frac{x - x_0}{x_1 - x_0} y_1$$

$$\ell(x) = \frac{(x_1 - x) y_0 + (x - x_0) y_1}{x_1 - x_0} \tag{5.1}$$

We say that $\ell(x)$ interpolates the value $y_i$ at the point $x_i$, $i = 0, 1$, or
$\ell(x_i) = y_i, \quad i = 0, 1$

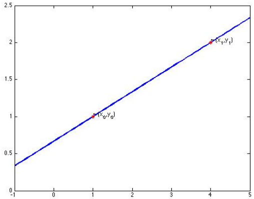


Figure: Linear interpolation

## Example

Let the data points be $(1,1)$ and (4,2). The polynomial $P_1(x)$ is given by

$$P_1(x) = \frac{(4-x)\cdot 1 + (x-1)\cdot 2}{3} \tag{5.2}$$

The graph $y = P_1(x)$ and $y = \sqrt{x}$, from which the data points were taken.

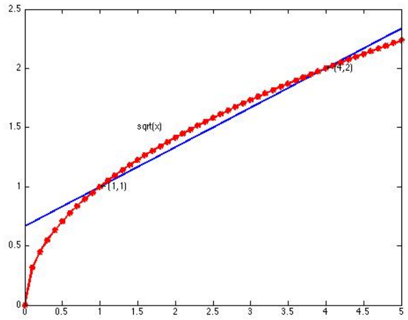


Figure: $y = \sqrt{x}$ and its linear interpolating polynomial (5.2)

## Example

Obtain an estimate of $e^{0.826}$ using the function values

$$e^{0.82} \doteq 2.270500, \qquad e^{0.83} \doteq 2.293319$$

Denote $x_0 = 0.82$, $x_1 = 0.83$. The interpolating polynomial $P_1(x)$ interpolating $e^x$ at $x_0$ and $x_1$ is

$$P_1(x) = \frac{(0.83 - x) \cdot 2.270500 + (x - 0.82) \cdot 2.293319}{0.01} \tag{5.3}$$

and

$$P_1(0.826) = 2.2841914$$

while the true value s

$$e^{0.826} \doteq 2.2841638$$

to eight significant digits.

Assume three data points $(x_0, y_0), (x_1, y_1), (x_2, y_2)$, with $x_0, x_1, x_2$ distinct. We construct the quadratic polynomial passing through these points using **Lagrange's folmula**

$$P_2(x) = y_0 L_0(x) + y_1 L_1(x) + y_2 L_2(x) \tag{5.4}$$

with **Lagrange interpolation basis functions** for quadratic interpolating polynomial

$$\begin{aligned} L_0(x) &= \frac{(x-x_1)(x-x_2)}{(x_0-x_1)(x_0-x_2)} \\ L_1(x) &= \frac{(x-x_0)(x-x_2)}{(x_1-x_0)(x_1-x_2)} \\ L_2(x) &= \frac{(x-x_0)(x-x_1)}{(x_2-x_0)(x_2-x_1)} \end{aligned} \tag{5.5}$$

Each $L_i(x)$ has degree 2 $\Rightarrow P_2(x)$ has degree $\leq 2$. Moreover

$$\begin{aligned} L_i(x_j) &= 0, \quad j \neq i \\ L_i(x_i) &= 1 \end{aligned} \quad \text{for } 0 \leq i, j \leq 2 \text{ i.e., } L_i(x_j) = \delta_{i,j} = \begin{cases} 1, & i = j \\ 0, & i \neq j \end{cases}$$

the **Kronecker delta function**.

$P_2(x)$ interpolates the data

$$P_2(x) = y_i, \qquad \text{i=0,1,2}$$

## Example

Construct $P_2(x)$ for the data points $(0,-1),(1,-1),(2,7)$. Then

$$P_2(x)=\frac{(x-1)(x-2)}{2}\cdot(-1)+\frac{x(x-2)}{-1}\cdot(-1)+\frac{x(x-1)}{2}\cdot 7 \tag{5.6}$$

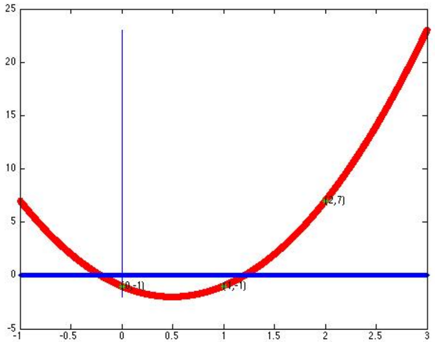


Figure: The quadratic interpolating polynomial (5.6)

With linear interpolation: obvious that there is only one straight line passing through two given data points.
With **three** data points: only **one quadratic** interpolating polynomial whose graph passes through the points.
Indeed: assume $\exists Q_2(x)$, $\deg(Q_2) \le 2$ passing through $(x_i, y_i), i = 0, 1, 2$, then it is equal to $P_2(x)$. The polynomial

$$R(x) = P_2(x) - Q_2(x)$$

has $\deg(R) \le 2$ and

$$R(x_i) = P_2(x_i) - Q_2(x_i) = y_i - y_i = 0, \qquad \text{for } i = 0, 1, 2$$

So $R(x)$ is a polynomial of degree $\le 2$ with three roots $\Rightarrow R(x) \equiv 0$

**Example**

Calculate a quadratic interpolate to $e^{0.826}$ from the function values

$$e^{0.82} \doteq 2.27050 \quad e^{0.83} \doteq 2.293319 \quad e^{0.84} \doteq 2.231637$$

With $x_0 = e^{0.82}, x_1 = e^{0.83}, x_2 = e^{0.84}$, we have

$$P_2(0.826) \doteq 2.2841639$$

to eight digits, while the true answer $e^{0.826} \doteq 2.2841638$ and $P_1(0.826) \doteq 2.2841914$.

## Lagrange's Formula

Given $n+1$ data points $(x_0, y_0), (x_1, y_1), \ldots, (x_n, y_n)$ with all $x_i$'s distinct, $\exists$ unique $P_n$, deg$(P_n) \leq n$ such that

$$P_n(x_i) = y_i,\ i = 0, \ldots, n$$

given by **Lagrange's Formula**

$$P_n(x) = \sum_{i=0}^{n} y_i L_i(x) = y_0 L_0(x) + y_1 L_1(x) + \cdots y_n L_n(x) \tag{5.7}$$

where $L_i(x) = \prod_{j=0, j\neq i}^{n} \frac{x - x_j}{x_i - x_j} = \frac{(x-x_0)\cdots(x-x_{i-1})(x-x_{i+1})\cdots(x-x_n)}{(x_i-x_0)\cdots(x_i-x_{i-1})(x_i-x_i)\cdots(x_i-x_n)}$,

$$L_i(x_j) = \delta_{ij}$$

Remark; The **Lagrange's formula** (5.7) is suited for theoretical uses, but is impractical for computing the value of an interpolating polynomial: knowing $P_2(x)$ does not lead to a less expensive way to compute $P_3(x)$.
But for this we need some preliminaries, and we start with a *discrete version of the derivative of a function* $f(x)$.

**Definition (First-order divided difference)**

Let $x_0 \neq x_1$, we define the **first-order divided difference** of $f(x)$ by

$$f[x_0, x_1] = \frac{f(x_1) - f(x_2)}{x_1 - x_2} \tag{5.8}$$

If $f(x)$ is differentiable on an interval containing $x_0$ and $x_1$, then the *mean value theorem* gives

$$f[x_0, x_1] = f'(c), \text{ for } c \text{ between } x_0 \text{ and } x_1.$$

Also if $x_0, x_1$ are close together, then

$$f[x_0, x_1] \approx f'\left(\frac{x_0 + x_1}{2}\right)$$

usually a very good approximation.

**Example**

Let $f(x) = \cos(x), x_0 = 0.2, x_1 = 0.3.$

Then

$$f[x_0, x_1] = \frac{\cos(0.3) - \cos(0.2)}{0.3 - 0.2} \doteq -0.2473009 \tag{5.9}$$

while

$$f'\left(\frac{x_0 + x_1}{2}\right) = -\sin(0.25) \doteq -0.2474040$$

so $f[x_0, x_1]$ is a very good approximation of this derivative.

## Higher-order divided differences are defined recursively

- Let $x_0, x_1, x_2 \in \mathbb{R}$ distinct. **The second-order divided difference**

$$f[x_0, x_1, x_2] = \frac{f[x_1, x_2] - f[x_0, x_1]}{x_2 - x_0} \tag{5.10}$$

- Let $x_0, x_1, x_2, x_3 \in \mathbb{R}$ distinct. **The third-order divided difference**

$$f[x_0, x_1, x_2, x_3] = \frac{f[x_1, x_2, x_3] - f[x_0, x_1, x_2]}{x_3 - x_0} \tag{5.11}$$

- In general, let $x_0, x_1, \ldots, x_n \in \mathbb{R}$, $n+1$ distinct numbers. **The divided difference of order** $n$

$$f[x_0, \ldots, x_n] = \frac{f[x_1, \ldots, x_n] - f[x_0, \ldots, x_{n-1}]}{x_n - x_0} \tag{5.12}$$

or the **Newton divided difference**.

**Theorem**

*Let $n \geq 1, f \in C^n[\alpha, \beta]$ and $x_0, x_1, \ldots, x_n$ $n+1$ distinct numbers in $[\alpha, \beta]$. Then*

$$\boldsymbol{f[x_0, x_1, \ldots, x_n] = \frac{1}{n!} f^{(n)}(c)} \tag{5.13}$$

*for some unknown point $c$ between the maximum and the minimum of $x_0, \ldots, x_n$.*

**Example**

Let $f(x) = \cos(x), x_0 = 0.2, x_1 = 0.3, x_2 = 0.4$.

The $f[x_0, x_1]$ is given by (5.9), and

$$f[x_1, x_2] = \frac{\cos(0.4) - \cos(0.3)}{0.4 - 0.3} \doteq -0.3427550$$

hence from (5.11)

$$f[x_0, x_1, x_2] \doteq \frac{-0.3427550 - (-0.2473009)}{0.4 - 0.2} = -0.4772705 \tag{5.14}$$

For $n = 2$, (5.13) becomes

$$f[x_0, x_1, x_2] = \frac{1}{2} f''(c) = -\frac{1}{2}\cos(c) \approx -\frac{1}{2}\cos(0.3) \doteq -0.4776682$$

which is nearly equal to the result in (5.14).

The divided differences (5.12) have special properties that help simplify work with them.
(1) Let $(i_0, i_1, \ldots, i_n)$ be a permutation (rearrangement) of the integers $(0, 1, \ldots, n)$. It can be shown that

$$f[x_{i_0}, x_{i_1}, \ldots, x_{i_n}] = f[x_0, x_1, \ldots, x_n] \tag{5.15}$$

The original definition (5.12) seems to imply that the order of $x_0, x_1, \ldots, x_n$ is important, but (5.15) asserts that **it is not true**.
For $n = 1$

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

For $n = 2$ we can expand (5.11) to get

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

(2) The definitions (5.8), (5.11) and (5.12) extend to the case where some or all of the $x_i$ coincide, provided that $f(x)$ is sufficiently differentiable.

For example, define

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

For an arbitrary $n \geq 1$, let all $x_i \to x_0$; this leads to the definition

$$f[x_0, \ldots, x_0] = \frac{1}{n!} f^{(n)}(x_0) \tag{5.16}$$

For the cases where only some of nodes coincide: using (5.15), (5.16) we can extend the definition of the divided difference.
For example

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

## MATLAB program: evaluating divided differences

Given a set of values $f(x_0), \ldots, f(x_n)$ we need to calculate the set of divided differences

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

We can use the MATLAB function $\mathrm{divdif}$ using the function call
**divdif_y = divdif(x_nodes, y_values)**
Note that MATLAB does not allow zero subscripts, hence $x\_nodes$ and $y\_values$ have to be redefined as vectors containing $n+1$ components:

$$\begin{aligned} x\_nodes &= [x_0, x_1, \ldots, x_n] \\ x\_nodes(i) &= x_{i-1}, \quad i = 1, \ldots, n+1 \\ y\_values &= [f(x_0), f(x_1), \ldots, f(x_n)] \\ y\_values(i) &= f(x_{i-1}), \quad i = 1, \ldots, n+1 \end{aligned}$$

## MATLAB program: evaluating divided differences

```
function divdif_y = divdif(x_nodes,y_values) %
% This is a function
% divdif_y = divdif(x_nodes,y_values)
% It calculates the divided differences of the function
% values given in the vector y_values, which are the values of
% some function f(x) at the nodes given in x_nodes.  On exit,
% divdif_y(i) = f[x_1,...,x_i], i=1,...,m
% with m the length of x_nodes.  The input values x_nodes and
% y_values are not changed by this program.
%
divdif_y = y_values;
m = length(x_nodes);
for i=2:m
for j=m:-1:i
divdif_y(j) = (divdif_y(j)-divdif_y(j-1)) ...
        /(x_nodes(j)-x_nodes(j-i+1));
end
end
```

This program is illustrated in this table.

| i | $x_i$ | $\cos(x_i)$ | $D_i$ |
|---|---|---|---|
| 0 | 0.0 | 1.000000 | 0.1000000E+1 |
| 1 | 0.1 | 0.980067 | -0.9966711E-1 |
| 2 | 0.4 | 0.921061 | -0.4884020E+0 |
| 3 | 0.6 | 0.825336 | 0.4900763E-1 |
| 4 | 0.8 | 0.696707 | 0.3812246E-1 |
| 5 | 1.0 | 0.540302 | -0.3962047E-2 |
| 6 | 1.2 | 0.36358 | -0.1134890E-2 |

*Table:* Values and divided differences for $\cos(x)$

# Newton interpolation formula

Interpolation of $f$ at $x_0, x_1, \ldots, x_n$
Idea: use $P_{n-1}(x)$ in the definition of $P_n(x)$:

$$P_n(x) = P_{n-1}(x) + C(x)$$

$$\left.\begin{array}{ll} C(x) \in \mathcal{P}_n & \text{correction term} \\ C(x_i) = 0 & i = 0, \ldots, n-1 \end{array}\right\} \Longrightarrow$$

$$\Longrightarrow C(x) = a(x - x_0)(x - x_1) \ldots (x - x_{n-1})$$

$$P_n(x) = ax^n + \ldots$$

**Definition:** The divided difference of $f(x)$ at points $x_0, \dots, x_n$

is denoted by $f[x_0, x_1, \dots, x_n]$ and is defined to be the coefficient of $x^n$ in $P_n(x; f)$.

$$C(x) = f[x_0, x_1, \dots, x_n](x - x_0)\dots(x - x_{n-1})$$

$$p_n(x; f) = p_{n-1}(x; f) + f[x_0, x_1, \dots, x_n](x - x_0)\dots(x - x_{n-1}) \quad (5.17)$$

$$p_1(x; f) = f(x_0) + f[x_0, x_1](x - x_0) \quad (5.18)$$

$$p_2(x; f) = f(x_0) + f[x_0, x_1](x - x_0) + f[x_0, x_1, x_2](x - x_0)(x - x_1) \quad (5.19)$$

$$\vdots$$

$$p_n(x; f) = f(x_0) + f[x_0, x_1](x - x_0) + \dots + f[x_0, \dots, x_n](x - x_0)\dots(x - x_{n-1}) \quad (5.20)$$

$\Longrightarrow$ Newton interpolation formula

- For (5.18), consider $p_1(x_0), p_1(x_1)$. Easily, $p_1(x_0) = f(x_0)$, and

$$p_1(x_1) = f(x_0) + (x_1 - x_0)\left[\frac{f(x_1) - f(x_0)}{x_1 - x_0}\right] = f(x_0) + [f(x_1) - f(x_0)] = f(x_1)$$

So: $\deg(p_1) \leq 1$ and satisfies the interpolation conditions. Then the uniqueness of polynomial interpolation $\Rightarrow$ (5.18) is the linear interpolation polynomial to $f(x)$ at $x_0, x_1$.

- For (5.19), note that

$$p_2(x) = p_1(x) + (x - x_0)(x - x_1)f[x_0, x_1, x_2]$$

It satisfies: $\deg(P_2) \leq 2$ and

$$\begin{aligned}
p_2(x_i) &= p_1(x_i) + 0 = f(x_i), \quad i = 0, 1 \\
p_2(x_2) &= f(x_0) + (x_2 - x_0)f[x_0.x_1] + (x_2 - x_0)(x_2 - x_1)f[x_0, x_1, x_2] \\
&= f(x_0) + (x_2 - x_0)f[x_0.x_1] + (x_2 - x_1)\{f[x_1, x_2] - f[x_0, x_1]\} \\
&= f(x_0) + (x_1 - x_0)f[x_0.x_1] + (x_2 - x_1)f[x_1, x_2] \\
&= f(x_0) + \{f(x_1) - f(x_0)\} + \{f(x_2) - f(x_1)\} = f(x_2)
\end{aligned}$$

By the uniqueness of polynomial interpolation, this is the quadratic interpolating polynomial to $f(x)$ at $\{x_0, x_1, x_2\}$.

## Example

Find $p(x) \in P_2$ such that $p(-1) = 0, p(0) = 1, p(1) = 4$.

$$p(x) = p(x_0) + p[x_0, x_1](x - x_0) + p[x_0, x_1, x_2](x - x_0)(x - x_1)$$

| $x_i$ | $p(x_i)$ | $p[x_i, x_{i+1}]$ | $p[x_i, x_{i+1}, x_{i+1}]$ |
|---|---|---|---|
| $-1$ | $0$ | 1 | $1$ |
| $0$ | $1$ | 3 | |
| $1$ | $4$ | | |

$$p(x) = 0 + 1(x + 1) + 1(x + 1)(x - 0) = x^2 + 2x + 1$$

## Example

Let $f(x) = \cos(x)$. The previous table contains a set of nodes $x_i$, the values $f(x_i)$ and the divided differences computed with $\mathrm{divdif}$

$$D_i = f[x_0, \ldots, x_i] \quad i \geq 0$$

| | $p_n(0.1)$ | $p_n(0.3)$ | $p_n(0.5)$ |
|---|---|---|---|
| 1 | 0.9900333 | 0.9700999 | 0.9501664 |
| 2 | 0.9949173 | 0.9554478 | 0.8769061 |
| 3 | 0.9950643 | 0.9553008 | 0.8776413 |
| 4 | 0.9950071 | 0.9553351 | 0.8775841 |
| 5 | 0.9950030 | 0.9553369 | 0.8775823 |
| 6 | 0.9950041 | 0.9553365 | 0.8775825 |
| True | 0.9950042 | 0.9553365 | 0.8775826 |

*Table:* Interpolation to $\cos(x)$ using (5.20)

This table contains the values of $p_n(x)$ for various values of $n$, computed with $\mathrm{interp}$, and the true values of $f(x)$.

- In general, the interpolation node points $x_i$
  need not to be evenly spaced,
  nor be arranged in any particular order
  to use the divided difference interpolation formula (5.20)
- To evaluate (5.20) efficiently we can use a nested multiplication algorithm

$$\begin{aligned} P_n(x) &= D_0 + (x-x_0)D_1 + (x-x_0)(x-x_1)D_2 \\ &\quad + \ldots + (x-x_0)\cdots(x-x_{n-1})D_n \\ &\text{with } D_0 = f(x_0), \quad D_i = f[x_0,\ldots,x_i] \text{ for } i \geq 1 \\ P_n(x) &= D_0 + (x-x_0)[D_1 + (x-x_1)[D_2 + \cdots \\ &\quad + (x-x_{n-2})[D_{n-1} + (x-x_{n-1})D_n]\cdots]] \end{aligned} \tag{5.21}$$

For example

$$P_3(x) = D_0 + (x-x_0)D_1 + (x-x_1)[D_2 + (x-x_2)D_3]$$

(5.21) uses only $n$ multiplications to evaluate $P_n(x)$ and is more convenient for a fixed degree $n$. To compute a sequence of interpolation polynomials of increasing degree is more efficient to se the original form (5.20).

# MATLAB: evaluating Newton divided difference for polynomial interpolation

```
function p_eval = interp(x_nodes,divdif_y,x_eval) %
% This is a function
% p_eval = interp(x_nodes,divdif_y,x_eval)
% It calculates the Newton divided difference form of
% the interpolation polynomial of degree m-1, where the
% nodes are given in x_nodes, m is the length of x_nodes,
% and the divided differences are given in divdif_y.  The
% points at which the interpolation is to be carried out
% are given in x_eval; and on exit, p_eval contains the
% corresponding values of the interpolation polynomial.
%
m = length(x_nodes);
p_eval = divdif_y(m)*ones(size(x_eval));
for i=m-1:-1:1
p_eval = divdif_y(i) + (x_eval - x_nodes(i)).*p_eval;
end
```

## Formula for error $E(t) = f(t) - p_n(t; f)$

$$P_n(x) = \sum_{j=0}^{n} f(x_j) L_j(x)$$

### Theorem

Let $n \geq 0$, $f \in C^{n+1}[a, b]$, $x_0, x_1, \ldots, x_n$ distinct points in $[a, b]$. Then

$$f(x) - P_n(x) = (x - x_0) \ldots (x - x_n) f[x_0, x_1, \ldots, x_n, x] \tag{5.22}$$

$$= \frac{(x - x_0)(x - x_1) \cdots (x - x_n)}{(n+1)!} f^{(n+1)}(c_x) \tag{5.23}$$

for $x \in [a, b]$, $c_x$ unknown between the minimum and maximum of $x_0, x_1, \ldots, x_n$.

## Proof of Theorem

Fix $t \in \mathbb{R}$. Consider

$$p_{n+1}(x; f) - \text{interpolates } f \text{ at } x_0, \ldots, x_n, t$$
$$p_n(x; f) - \text{interpolates } f \text{ at } x_0, \ldots, x_n$$

From Newton

$$p_{n+1}(x; f) = p_n(x; f) + f[x_0, \ldots, x_n, t](x - x_0) \ldots (x - x_n)$$

Let $x = t$.

$$f(t) := p_{n+1}(t, f) = p_n(t; f) + f[x_0, \ldots, x_n, t](t - x_0) \ldots (t - x)$$
$$E(t) := f[x_0, x_1, \ldots, x_n, t](t - x_0) \ldots (t - x_n) \qquad \blacksquare$$

## Examples

- $f(x) = x^n$, $f[x_0, \ldots, x_n] = \frac{n!}{n!} = 1$ or $p_n(x; f) = 1 \cdot x^n \Rightarrow f[x_0, \ldots, x_n] = 1$
- $f(x) \in P_{n-1}, \qquad f[x_0, x_1, \ldots, x_n] = 0$
- $f(x) = x^{n+1}, \qquad f[x_0, \ldots, x_n] = x_0 + x_1 + \ldots + x_n$

$$R(x) = x^{n+1} - p_n(x; f) \in P^{n+1}$$

$$R(x_i) = 0, \qquad i = 0, \ldots, n$$

$$R(x_i) = (x - x_0) \ldots (x - x_n) = x^{n+1} - \underbrace{(x_0 + \ldots + x_n)x^n + \ldots}_{P_n}$$

$$\Longrightarrow f[x_0, \ldots, x_n] = x_0 + x_1 + \ldots + x_n.$$

If $f \in P_m$,

$$f[x_0, \ldots, x_m, x] = \begin{cases} \text{polynomial degree } m - n - 1 & \text{if } n \leq m - 1 \\ 0 & \text{if } n > m - 1 \end{cases}$$

### Example

Take $f(x) = e^x, x \in [0,1]$ and consider the error in linear interpolation to $f(x)$ using $x_0, x_1$ satisfying $0 \le x_0 \le x_1 \le 1$.

From (5.23)

$$e^x - P_1(x) = \frac{(x-x_0)(x-x_1)}{2} e^{c_x}$$

for some $c_x$ between the max and min of $x_0, x_1$ and $x$. Assuming $x_0 < x < x_1$, the error is negative and **approximately** a quadratic polynomial

$$e^x - P_1(x) = -\frac{(x_1-x)(x-x_0)}{2} e^{c_x}$$

Since $x_0 \le c_x \le x_1$

$$\frac{(x_1-x)(x-x_0)}{2} e^{x_0} \le |e^x - P_1(x)| = \frac{(x_1-x)(x-x_0)}{2} e^{x_1}$$

For a bound independent of $x$

$$\max_{x_0 \le x \le x_1} \frac{(x_1 - x)(x - x_0)}{2} = \frac{h^2}{8}, \quad h = x_1 - x_0$$

and $e^{x_1} \le e$ on $[0, 1]$

$$|e^x - P_1(x)| \le \frac{h^2}{8} e, \quad 0 \le x_0 \le x \le x_1 \le 1$$

independent of $x, x_0, x_1$.
Recall that we estimated $e^{0.826} \doteq 2.2841914$ by $e^{0.82}$ and $e^{0.83}$, i.e., $h = 0.01$.

$$|e^x - P_1(x)| \le \frac{h^2}{8} e = |e^x - P_1(x)| \le \frac{0.01^2}{8} 2.72 = 0.0000340,$$

The actual error being $-0.0000276$, it satisfies this bound.

### Example

Again let $f(x) = e^x$ on $[0, 1]$, but consider the quadratic interpolation.

$$e^x - P_2(x) = \frac{(x - x_0)(x - x_1)(x - x_2)}{6} e^{c_x}$$

for some $c_x$ between the min and max of $x_0, x_1, x_2$ and $x$.
Assuming evenly spaced points, $h = x_1 - x_0 = x_2 - x_1$, and $0 \le x_0 < x < x_2 \le 1$, we have as before

$$|e^x - P_2(x)| \le \left| \frac{(x - x_0)(x - x_1)(x - x_2)}{6} \right| e^1$$

while

$$\max_{x_0 \le x \le x_2} \frac{(x - x_0)(x - x_1)(x - x_2)}{6} = \frac{h^3}{9\sqrt{3}} \qquad (5.24)$$

hence

$$|e^x - P_2(x)| \le \frac{h^3 e}{9\sqrt{3}} \approx 0.174h^3$$

For $h = 0.01, 0 \leq x \leq 1$

$$|e^x - P_2(x)| \leq 1.74 \times 10^{-7}$$

Let
$$w_2(x) = \frac{(x+h)x(x-h)}{6} = \frac{x^3 - xh}{6}$$
a translation along $x$-axis of polynomial in (5.24). Then $x = \pm\frac{h}{\sqrt{3}}$ satisfies
$$0 = w_2'(x) = \frac{3x^2 - h^2}{6} \quad \text{and gives} \left| w_2\left(\pm\frac{h}{\sqrt{3}}\right)\right| = \frac{h^3}{9\sqrt{3}}$$

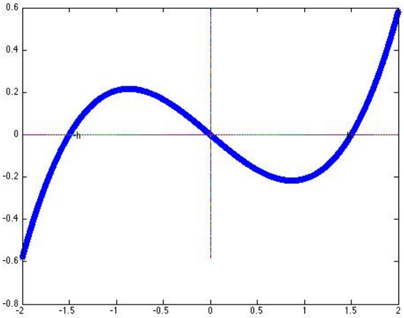


Figure: $y = w_2(x)$

When we consider the error formula (5.23) or (5.22), the polynomial

$$\psi_n(x) = (x - x_0) \cdots (x - x_n)$$

is crucial in determining the behaviour of the error. Let assume that $x_0, \ldots, x_n$ are evenly spaced and $x_0 \le x \le x_n$.

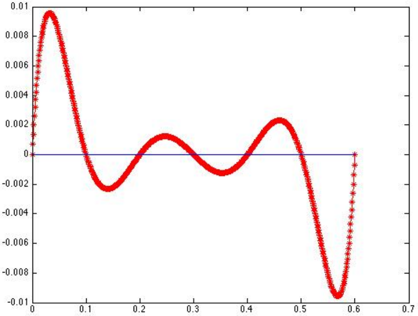


Figure: $y = \Psi_6(x)$

Remark that the interpolation error

- is relatively larger in $[x_0, x_1]$ and $[x_5, x_6]$, and
- is likely to be smaller near the middle of the node points

$\Rightarrow$ In practical interpolation problems, high-degree polynomial interpolation with evenly spaced nodes is seldom used.
But when the set of nodes is suitable chosed, high-degree polynomials can be very useful in obtaining polynomial approximations to functions.

**Example**

Let $f(x) = \cos(x), h = 0.2, n = 8$ and then interpolate at $x = 0.9$.

Case (i) $x_0 = 0.8, x_8 = 2.4 \Rightarrow x = 0.9 \in [x_0, x_1]$. By direct calculation of $P_8(0.9)$,

$$\cos(0.9) - P_8(0.9) \doteq -5.51 \times 10^{-9}$$

Case (ii) $x_0 = 0.2, x_8 = 1.8 \Rightarrow x = 0.9 \in [x_3, x_4]$, where $x_4$ is the midpoint.

$$\cos(0.9) - P_8(0.9) \doteq 2.26 \times 10^{-10},$$

a factor of 24 smaller than the first case.

## Example

Le $f(x) = \frac{1}{1+x^2}, x \in [-5,5], n > 0$ an even integer, $h = \frac{10}{n}$

$$x_j = -5 + jh, \qquad j = 0, 1, 2, \ldots, n$$

It can be shown that for many points $x$ in $[-5,5]$, the sequence of $\{P_n(x)\}$ does not converge to $f(x)$ as $n \to \infty$.

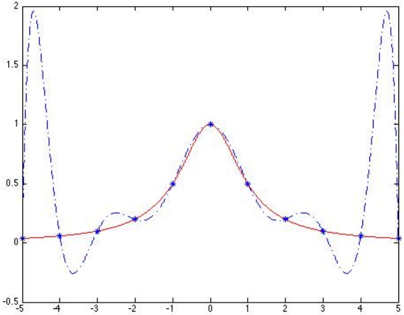


Figure: The interpolation to $\frac{1}{1+x^2}$

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

The simplest method of interpolating data in a table: connecting the node points by straight lines: the curve $y = \ell(x)$ is not very smooth.

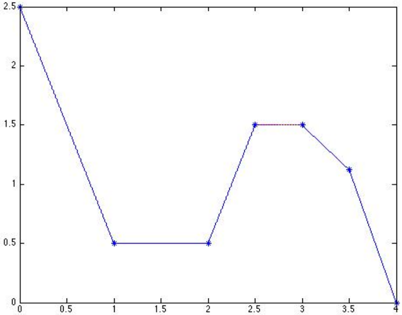


Figure: $y = \ell(x)$: **piecewise linear interpolation**

We want to construct a **smooth** curve that interpolates the given data point that follows the shape of $y = \ell(x)$.

Next choice: use polynomial interpolation. With seven data point, we consider the interpolating polynomial $P_6(x)$ of degree 6.

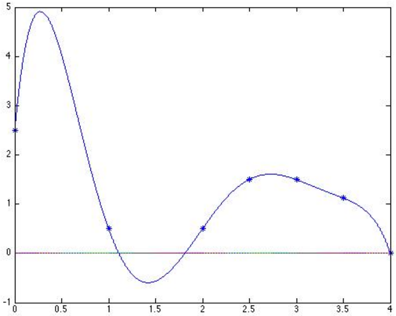


Figure: $y = P_6(x)$: **polynomial interpolation**

Smooth, but **quite different** from $y = \ell(x)$!!!

A third choice: connect the data in the table using a succession of quadratic interpolating polynomials: on each $[0,2],[2,3],[3,4]$.

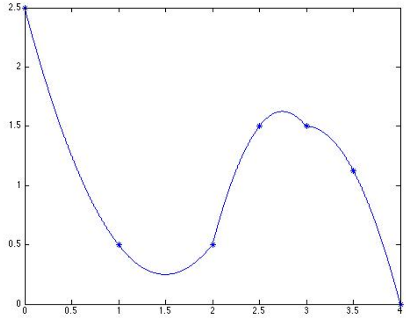


Figure: $y = q(x)$: **piecewise quadratic interpolation**

Is smoother than $y = \ell(x)$, follows it more closely than $y = P_6(x)$, but at $x = 2$ and $x = 3$ the graph has corners, i.e., $q'(x)$ is discontinuous.

## Cubic spline interpolation

Suppose $n$ data points $(x_i, y_i), i = 1, \ldots, n$ are given and

$$a = x_1 < \ldots < x_n = b$$

We seek a function $S(x)$ defined on $[a, b]$ that interpolates the data

$$S(x_i) = y_i, \quad i = 1, \ldots, n$$

Find $S(x) \in C^2[a, b]$, a natural cubic spline such that

$S(x)$is a polynomial of degree $\leq 3$on each subinterval $[x_{j-1}, x_j], j = 2, 3, \ldots, n$
$S(x_i) = y_i, \qquad i = 1, \ldots, n,$
$S''(a) = S''(b).$

On $[x_{i-1}, x_i]$: 4 degrees of freedom (cubic spline) $\Rightarrow 4(n-1)$ DOF.

$$S(x_i) = y_i: \qquad n \text{ constraints}$$

$$\left.\begin{array}{ll} S'(x_i - 0) = S'(x_i + 0) & \text{continuity } 2 = 1, \ldots, n-1 \\ S''(x_i - 0) = S''(x_i + 0) & i = 2, \ldots, n-1 \\ S(x_i - 0) = S(x_i + 0) & i = 2, \ldots, n-1 \end{array}\right\} 3n - 6$$

$n + (3n - 6) = 4n - 6$ constraints

Two more constraints needed to be added, Boundary Conditions:

$$S'' = 0 \qquad \text{at } a = x_1, x_n = b,$$

for a natural cubic spline.
(Linear system, with symmetric, positive definite, diagonally dominant matrix.)

Introduce the variables $M_1, \ldots, M_n$ with

$$M_i \equiv S''(x_i), \quad i = 1, 2, \ldots, n$$

Since $S(x)$ is cubic on each $[x_{j-1}, x_j] \Rightarrow S''(x)$ is linear, hence determined by its values at two points:

$$S''(x_{j-1}) = M_{j-1}, \quad S''(x_j) = M_j$$

$$\Rightarrow S''(x) = \frac{(x_j - x)M_{j-1} + (x - x_{j-1})M_j}{x_j - x_{j-1}}, \quad x_{j-1} \le x \le x_j \tag{5.25}$$

Form the second antiderivative of $S''(x)$ on $[x_{j-1}, x_j]$ and apply the interpolating conditions

$$S(x_{j-1}) = y_{j-1}, \qquad S(x_j) = y_j$$

we get

$$S(x) = \frac{(x_j - x)^3 M_{j-1} + (x - x_{j-1})^3 M_j}{6(x_j - x_{j-1})} + \frac{(x_j - x)y_{j-1} + (x - x_{j-1})y_j}{6(x_j - x_{j-1})}$$
$$- \frac{1}{6}(x_j - x_{j-1}) \left[(x_j - x)M_{j-1} + (x - x_{j-1})M_j\right] \tag{5.26}$$

for $x \in [x_{j-1}, x_j], j = 1, \ldots, n$.

Formula (5.26) implies that $S(x)$ is continuous over all $[a, b]$ and satisfies the interpolating conditions $S(x_i) = y_i$.
Similarly, formula (5.25) for $S''(x)$ implies that is continuous on $[a, b]$.
To ensure the continuity of $S'(x)$ over $[a, b]$: we require $S''(x)$ on $[x_{j-1}, x_j]$ and $[x_j, x_{j+1}]$ have to give the same value at their common point$x_j, j = 2, 3, \ldots, n-1$, leading to the **tridiagonal linear system**

$$\frac{x_j - x_{j-1}}{6} M_{j-1} + \frac{x_{j+1} - x_{j-1}}{3} M_j + \frac{x_{j+1} - x_j}{6} M_{j+1} \tag{5.27}$$
$$= \frac{y_{j+1} - y_j}{x_{j+1} - x_j} - \frac{y_j - y_{j-1}}{x_j - x_{j-1}}, \quad j = 2, 3, \ldots, n-1$$

These $n-2$ equations together with the assumption $S''(a) = S''(b) = 0$:

$$M_1 = M_n = 0 \tag{5.28}$$

leads to the values $M_1, \ldots, M_n$, hence the function $S(x)$.

## Example

Calculate the natural cubic spline interpolating the data

$$\left\{(1,1),(2,\frac{1}{2}),(3,\frac{1}{3}),(4,\frac{1}{4})\right\}$$

$n=4$, and all $x_{j-1}-x_j=1$. The system (5.27) becomes

$$\begin{aligned} \tfrac{1}{6}M_1+\ \tfrac{2}{3}M_2+\tfrac{1}{6}M_3 \qquad\qquad &= \tfrac{1}{3} \\ \tfrac{1}{6}M_2+\tfrac{2}{3}M_3+\tfrac{1}{6}M_4 &= \tfrac{1}{12} \end{aligned}$$

and with (5.28) this yields

$$M_2=\frac{1}{2}, \quad M_3=0$$

which by (5.26) gives

$$S(x)=\begin{cases} \frac{1}{12}x^3-\frac{1}{4}x^2-\frac{1}{3}x+\frac{3}{2}, & 1\le x\le 2 \\ -\frac{1}{12}x^3+\frac{3}{4}x^2-\frac{7}{3}x+\frac{17}{6}, & 2\le x\le 3 \\ -\frac{1}{12}x+\frac{7}{12}, & 3\le x\le 4 \end{cases}$$

Are $S'(x)$ and $S''(x)$ continuous?

## Example

Calculate the natural cubic spline interpolating the data

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

$n = 7$, system (5.27) has 5 equations

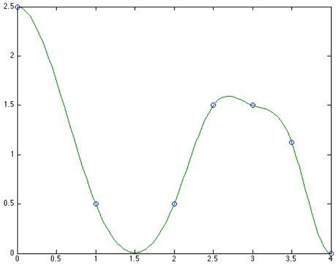


Figure: Natural cubic spline interpolation $y = S(x)$

Compared to the graph of linear and quadratic interpolation $y = \ell(x)$ and $y = q(x)$, the cubic spline $S(x)$ no longer contains corners.

So far we only interpolated data points, wanting a smooth curve.

When we seek a spline to interpolate a known function, we are interested also in the accuracy.

Let $f(x)$ be given on $[a, b]$, that we want to interpolate on evenly spaced values of $x$. For $n > 1$, let

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

and $S_n(x)$ be the natural cubic spline interpolating $f(x)$ at $x_1, \ldots, x_n$. It can be shown that

$$\max_{a \le x \le b} |f(x) - S_n(x)| \le ch^2 \tag{5.29}$$

where $c$ depends on $f''(a), f''(b)$ and $\max_{a \le x \le b} |f^{(4)}(x)|$. The reason why $S_n(x)$ doesn't converge more rapidly (have an error bound with a higher power of $h$) is that $f''(a) \neq 0 \neq f''(b)$, while by definition $S_n''(a) = S_n''(b) = 0$. For functions $f(x)$ with $f''(a) = f''(b) = 0$, the RHS of (5.29) can be replaced with $ch^4$.

To improve on $S_n(x)$, we look for other interpolating functions $S(x)$ that interpolate $f(x)$ on

$$a = x_1 < x_2 < \ldots < x_n = b$$

Recall the definition of natural cubic spline:

1. $S(x)$ cubic on each subinterval $[x_{j-1}, x_j]$;
2. $S(x), S'(x)$ and $S''(x)$ are continuous on $[a, b]$;
3. $S''(x_1) = S''(x_n) = 0$.

We say that $S(x)$ is a cubic spline on $[a, b]$ if

1. $S(x)$ cubic on each subinterval $[x_{j-1}, x_j]$;
2. $S(x), S'(x)$ and $S''(x)$ are continuous on $[a, b]$;

With the interpolating conditions

$$S(x_i) = y_i, \qquad i = 1, \ldots, n$$

the representation formula (5.26) and the tridiagonal system (5.27) are still valid.

This system has $n-2$ equations and $n$ unknowns: $M_1, \ldots, M_n$. By replacing the end conditions (5.28) $S''(x_1) = S''(x_n) = 0$, we can obtain other interpolating cubic splines.

If the data $(x_i, y_i)$ is obtained by evaluating a function $f(x)$

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

then we choose endpoints (boundary conditions) for $S(x)$ that would yield a better approximation to $f(x)$. We require

$$S'(x_1) = f'(x_1), \qquad S'(x_n) = f'(x_n)$$

or

$$S''(x_1) = f''(x_1), \qquad S''(x_n) = f''(x_n)$$

When combined with(5.26-5.27), either of these conditions leads to a unique interpolating spline $S(x)$, dependent on which of these conditions is used. In both cases, the RHS of (5.29) can be replaced by $ch^4$, where $c$ depends on $\max_{x\in[a,b]} \left|f^{(4)}(x)\right|$.

If the derivatives of $f(x)$ are not known, then extra interpolating conditions can be used to ensure that the error bond of (5.29) is proportional to $h^4$. In particular, suppose that

$$x_1 < z_1 < x_2, \quad x_{n-1} < z_2 < x_n$$

and $f(z_1), f(z_2)$ are known. Then use the formula for $S(x)$ in (5.26) and

$$s(z_1) = f(z_1), \quad s(z_2) = f(z_2) \tag{5.30}$$

This adds two new equations to the system (5.27), one for $M_1$ and $M_2$, and another equation for $M_{n-1}, M_n$. This form is preferable to the interpolating natural cubic spline, and is almost equally easy to produce. This is the default form of spline interpolation that is implemented in $\mathrm{MATLAB}$. The form of spline formed in this way is said to satisfy the **not-a-knot interpolation boundary conditions**.

Interpolating cubic spline functions are a popular way to represent data analytically because

1. are relatively smooth: $C^2$,
2. do not have the rapid oscillation that sometime occurs with the high-degree polynomial interpolation,
3. are relatively easy to work with on a computer.

They do not replace polynomials, but are a very useful extension of them.

The standard package MATLAB contains the function
rm spline.
The standard calling sequence is

$$y = spline(x\_nodes, y\_nodes, x)$$

which produces the cubic spline function $S(x)$ whose graph passes through the points $\{(\xi_i, \eta_i) : i = 1, \dots, n\}$ with

$$(\xi_i, \eta_i) = (x\_nodes(i), y\_nodes\_(i))$$

and $n$ the length of $x\_nodes$ (and $y\_nodes$). The *not-a-knot interpolation conditions* of (5.30) are used. The point $(\xi_2, \eta_2)$ is the point $(z_1, f(z_1))$ of $(5.30)$ and $(\xi_{n-1}, \eta_{n-1})$ is the point $(z_2, f(z_2))$.

**Example**

Approximate the function $f(x) = e^x$ on the interval $[a, b] = [0, 1]$. For $n > 0$, define $h = 1/n$ and interpolating nodes

$$x_1 = 0, x_2 = h, x_3 = 2h, \ldots, x_{n+1} = nh = 1$$

Using spline, we produce the cubic interpolating spline interpolant $S_{n,1}$ to $f(x)$. With the not-a-knot interpolation conditions, the nodes $x_2$ and $x_n$ are the points $z_1$ and $z_2$ in (5.30). For a general smooth function $f(x)$, it turns out that te magnitude of the error $f(x) - S_{n,1}(x)$ is largest around the endpoints of the interval of approximation.

## Example

Two interpolating nodes are inserted, the midpoints of the subintervals $[0, h]$ and $[1-h, 1]$:

$$x_1 = 0, x_2 = \frac{1}{2}h, x_3 = h, x_4 = 2h, \ldots, x_{n+1} = (n-1)h, x_{n+2} = 1 - \frac{1}{2}h, x_{n+3} = 1$$

Using spline results in a cubic spline function $S_{n,2}(x)$; with the not-a-knot interpolation conditions conditions, the nodes $x_2$ and $x_{n+2}$ are the points $z_1$ and $z_2$ of (5.30). Generally, $S_{n,2}$ is a more accurate approximation than is $S_{n,1}(x)$.

The cubic polynomials produced for $S_{n,2}(x)$ by spline for the intervals $[x_1, x_2]$ and $[x_2, x_3]$ are the same, thus we can use the polynomial for $[0, \frac{1}{2}h]$ for the entire interval $[0, h]$. Similarly for $[1-h, h]$.

| $n$ | $E_n^{(1)}$ | Ratio | $E_n^{(2)}$ | Ratio |
|---|---|---|---|---|
| 5 | 1.01E-4 | | 1.11E-5 | |
| 10 | 6.92E-6 | 14.6 | 7.88E-7 | 14.1 |
| 20 | 4.56E-7 | 15.2 | 5.26E-8 | 15.0 |
| 40 | 2.92E-8 | 15.6 | 3.39E-9 | 15.5 |

*Table:* Cubic spline approximation to $f(x) = e^x$