**Example 100.** Under which conditions is

$$S(x) = \begin{cases} S_1(x), & \text{if } x \in [0,a], \\ S_2(x), & \text{if } x \in [a,b], \end{cases}$$

is a cubic spline? A natural cubic spline?

**Solution.** $S(x)$ is a cubic spline if $S_1(x)$ and $S_2(x)$ are cubic polynomials such that

$$S_1(a) = S_2(a), \quad S_1'(a) = S_2'(a), \quad S_1''(a) = S_2''(a).$$

$S(x)$ is a natural cubic spline if, in addition, $S_1''(0) = 0$ and $S_2''(b) = 0$.

**Comment.** Together with the three conditions coming from prescribing the values $S(0)$, $S(a)$ and $S(b)$, these are $8$ conditions in order for $S(x)$ to be a natural cubic spline. $8$ equations are just the right number to uniquely determine the underlying $2 \cdot 4 = 8$ unknowns.

**Example 101.** The following function $S(x)$ is a cubic spline.

$$S(x) = -1 - \frac{2}{9}(a-5)x - \frac{1}{3}(2a-1)x^2 + \frac{1}{36}x^3 \begin{cases} (-10a-13), & \text{if } x \in [-2,0], \\ 8(4a+7), & \text{if } x \in [0,1]. \end{cases}$$

(a) Spell out the conditions we need to check to see that this is a cubic spline.

(b) What are the underlying data points?

(c) Is there a choice of $a$ such that $S(x)$ is a natural cubic spline?

**Solution.**

(a) Write $S_1(x)$ for $S(x)$ on $[-2, 0]$ and $S_2(x)$ for $S(x)$ on $[0, 1]$. Then, as in the previous example, the conditions for $S(x)$ to be a cubic spline are

$$S_1(0) = S_2(0), \quad S_1'(0) = S_2'(0), \quad S_1''(0) = S_2''(0).$$

These conditions are visibly satisfied since the formulas for $S_1(x)$ and $S_2(x)$ agree up to a multiple of $x^3$.

(b) The knots of the spline are $-2, 0, 1$. We compute $S(-2) = 1$, $S(0) = -1$, $S(1) = 2$.
Hence the data points are $(-2, 1)$, $(0, -1)$, $(1, 2)$.

(c) In order for $S(x)$ to be a natural spline, we need $S''(-2) = 0$ as well as $S''(1) = 0$. Using

$$S''(x) = -\frac{2}{3}(2a-1) + \frac{1}{6}x \begin{cases} (-10a-13), & \text{if } x \in [-2,0], \\ 8(4a+7), & \text{if } x \in [0,1], \end{cases}$$

we have $S''(-2) = -\frac{2}{3}(2a-1) - \frac{1}{3}(-10a-13) = 2a+5$ and $S''(1) = -\frac{2}{3}(2a-1) + \frac{4}{3}(4a+7) = 4a+10$.
Both of these are $0$ if and only if $a = -\frac{5}{2}$. Therefore, $S(x)$ is a natural cubic spline if $a = -\frac{5}{2}$.

**Comment.** Can you explain why do the two segments of the spline only differ in the cubic term?
[*Hint*: Note that $0$ is a knot and look again at the first part.]

**Example 102.** Python Let us construct cubic splines using Python with `scipy`.

```
>>> from scipy import linspace, interpolate
```

**Comment.** Many basic functions like `linspace` are provided by both `numpy` and `scipy`.

We start by defining the data points that we wish to interpolate.

```
>>> xpoints = [1, 2, 4, 5, 7]
```

```
>>> ypoints = [2, 1, 4, 3, 2]
```

We can then construct the cubic spline with natural boundary conditions as follows.

```
>>> spline = interpolate.CubicSpline(xpoints, ypoints, bc_type='natural')
```

**Comment.** Other standard choices for the boundary conditions include `'not-a-knot'` (the default) as well as `'clamped'` and `'periodic'` (this one requires the first and last point to have the same $y$-coordinates).
https://docs.scipy.org/doc/scipy/reference/generated/scipy.interpolate.CubicSpline.html

The resulting natural cubic spline is piecewise defined by a collection of cubic polynomials. We can plot it as we did in Example 78 (this time we also include a legend).

```
>>> import matplotlib.pyplot as plt
```

```
>>> xplot = linspace(1, 7, 100)
```

```
>>> plt.plot(xplot, spline(xplot), '-', label='spline (natural)')
```

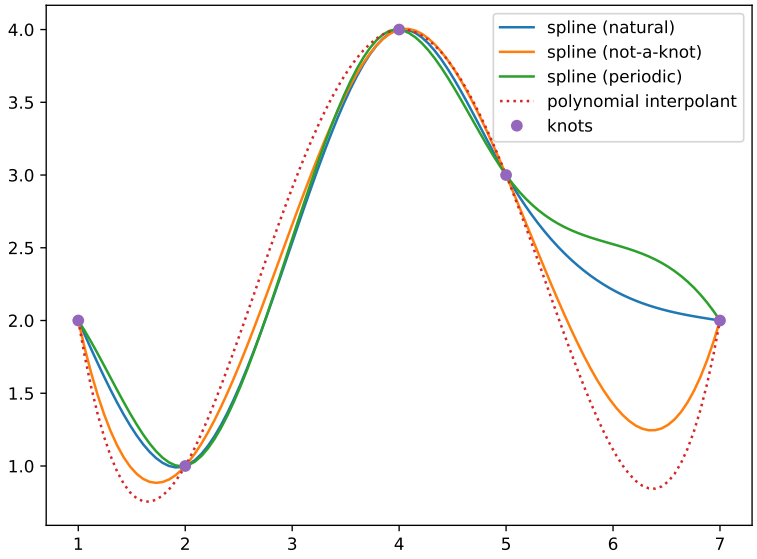
```
>>> plt.plot(xpoints, ypoints, 'o', label='knots')
```

```
>>> plt.legend()
```

```
>>> plt.show()
```

The resulting plot is a simpler version of the following one where we also included two other cubic splines as well as the polynomial interpolant:

**Homework.** Can you reproduce this plot?




Can you identify (some of) the splines without the labels? Try other knots and plot the splines!

**For instance.** The periodic spline is easily identified here because of the matching derivatives at the endpoints.
The natural spline is the one that is most like a clothesline pinned to the knots.
The not-a-knot spline is closer to polynomial interpolation.

If desired, we can access the piecewise polynomials as follows:

```
>>> spline.c

   [[ 6.37096774e-01 -6.49193548e-01  8.54838710e-01 -9.67741935e-02]
    [ 2.22044605e-16  1.91129032e+00 -1.98387097e+00  5.80645161e-01]
    [-1.63709677e+00  2.74193548e-01  1.29032258e-01 -1.27419355e+00]
    [ 2.00000000e+00  1.00000000e+00  4.00000000e+00  3.00000000e+00]]

>>>
```

For instance, the first column refers to $2 - 1.637(x-1) + 0.637(x-1)^3$ (the cubic used on $[1,2]$, the first interval) while the fourth column encodes $3 - 1.274(x-5) + 0.581(x-5)^2 - 0.097(x-5)^3$ (the cubic used on $[5,7]$, the last interval).

**Comment.** The exact cubics are $2 - \frac{203}{124}(x-1) + \frac{79}{124}(x-1)^3$ and $3 - \frac{79}{62}(x-5) + \frac{18}{31}(x-5)^2 - \frac{3}{31}(x-5)^3$.

Note how, for the first one, $S_1(x)$, we can immediately see that $S_1''(1) = 0$. Because we created a natural cubic spline, we also have $S_4''(7) = 0$. (Check it from the above exact formula!)

**Example 103.** In the case of four nodes/knots, how is the polynomial interpolant related to the cubic splines?

**Solution.** Note that the polynomial interpolant for four nodes is a cubic polynomial.

On the other hand, each cubic spline consists of three cubic polynomials $S_1, S_2, S_3$. In the case of the not-a-knot cubic spline, we have $S_1 = S_2$ as well as $S_3 = S_2$, which implies that all three are equal so that the not-a-knot cubic spline is a single cubic polynomial (interpolating the four given points).

Therefore, the polynomial interpolant must equal the not-a-knot cubic spline in this case.

# Working with functions: differentiation + integration

## Numerical differentiation

We know from Calculus that $f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$.

To numerically approximate $f'(x)$ we could use $f'(x) \approx \frac{1}{h}[f(x+h) - f(x)]$ for small $h$.

In this section, we analyze this and other ways of numerically differentiating a function.

**Application.** These approximations are crucial for developing tools to numerically solve (partial) differential equations by discretizing them.

**Review.** We can express **Taylor's theorem** (Theorem 48) in the following manner:

$$f(x+h) = \underbrace{f(x) + f'(x)h + \frac{1}{2}f''(x)h^2 + \ldots + \frac{1}{n!}f^{(n)}(x)h^n}_{\text{Taylor polynomial}} + \underbrace{\frac{1}{(n+1)!}f^{(n+1)}(\xi)h^{n+1}}_{\text{error}}$$

This form is particularly convenient for the kind of error analysis that we are doing here.

**Important notation.** When the exact form of the error is not so important, we simply write $O(h^{n+1})$.

**Definition 104.** We write $e(h) = O(h^n)$ if there is a constant $C$ such that $|e(h)| \leqslant Ch^n$ for all small enough $h$.

For our purposes, $e(h)$ is usually an error term and this notation allows us to talk about that error without being more precise than necessary.

If $e(h)$ is the error, then we often say that an approximation is of order $n$ if $e(h) = O(h^n)$.

**Caution.** This notion of order is different from the order of convergence that we discussed in the context of fixed-point iteration and Newton's method.

**Example 105.** Determine the order of the approximation $f'(x) \approx \frac{1}{h}[f(x+h) - f(x)]$.

**Comment.** This approximation of the derivative is called a **(first) forward difference** for $f'(x)$.

Likewise, $f'(x) \approx \frac{1}{h}[f(x) - f(x-h)]$ is a **(first) backward difference** for $f'(x)$.

**Solution.** By Taylor's theorem, $f(x+h) = f(x) + hf'(x) + \frac{h^2}{2}f''(x) + \frac{h^3}{6}f'''(x) + O(h^4)$. It follows that

$$\frac{1}{h}[f(x+h) - f(x)] = f'(x) + \boxed{\frac{h}{2}f''(x) + O(h^2)} = f'(x) + \boxed{O(h)}.$$

The $\boxed{\text{error}}$ is of order $1$.

**Comment.** The presence of the term $\frac{h}{2}f''(x)$ tells us that the order is exactly $1$ unless $f''(x) = 0$ (that is, the order cannot generally be improved to $\delta$ for some $\delta < 1$).