Introduction:

Objects manufactured by industry have curved lines. CAD software offers a catalog of simple shapes (segments of lines, arcs of circles, conics, etc.) but they are not enough. The designer needs a richer family of curves.

Numerical control machines could only machine simple shapes precisely. A second category of objects, on the other hand, would have a priori imprecise form, determined experimentally. Aircraft propellers, boat hulls and car bodies were drawn freehand, without their shapes being able to be described by a mathematical formula.

The problem that we are often confronted with is the following: how to represent, from a computational point of view, "free-form" curves or surfaces, i.e. curves or continuous surfaces, with a generally regular appearance, whose shape is not governed, a priori, by any particular equation?

Mode of representation:

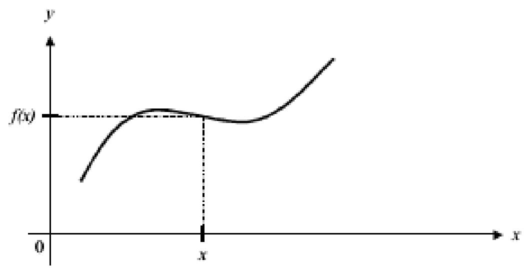
Here we summarize the main types of equations associated with the notion of curve or surface. The letters x, y and z, as well as the illustrations, suggest the use of a Cartesian coordinate system, but there is nothing, of course, to prevent the use of other coordinate systems.

Equations implicites :

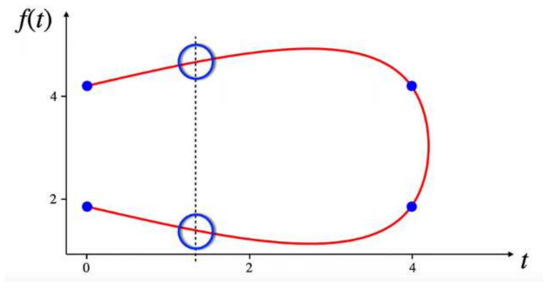
1. 2D curves: $f{:}\mathbb{R}^2 \to \mathbb{R} \quad f(x,y)=0$
2. 3D Curves: $f{:}\mathbb{R}^3 \to \mathbb{R}^2 \quad f(x,y)=\begin{bmatrix}0\\0\end{bmatrix}$
3. Surface : $f{:}\mathbb{R}^3 \to \mathbb{R} \quad f(x,y,z)=0$

Real functions of one or two variables:

4. 2D curves:

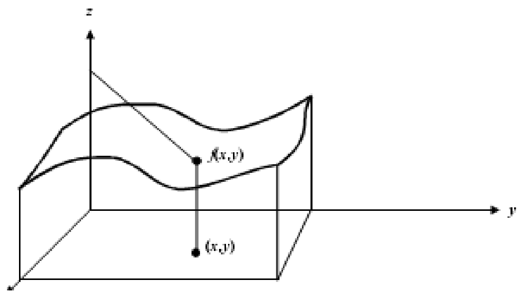


$f{:}\mathbb{R} \to \mathbb{R} \quad y=f(x)$



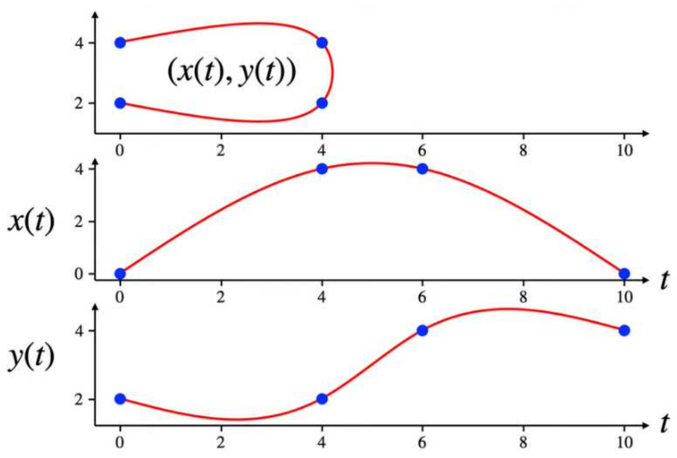
- Surface :

$f: \mathbb{R}^2 \rightarrow \mathbb{R} \qquad z = f(x,y)$



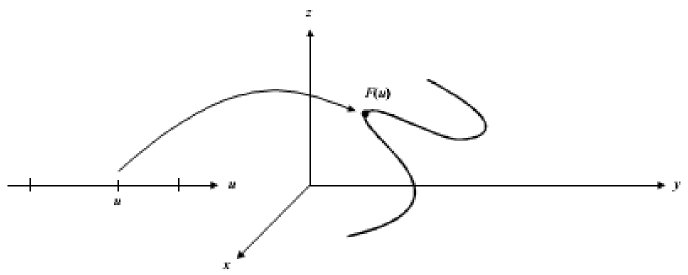
Parameterized curves and surfaces:

- 2D curves:



$F: I \subset \mathbb{R} \rightarrow \mathbb{R}^2 \qquad (x,y) = f(u)$

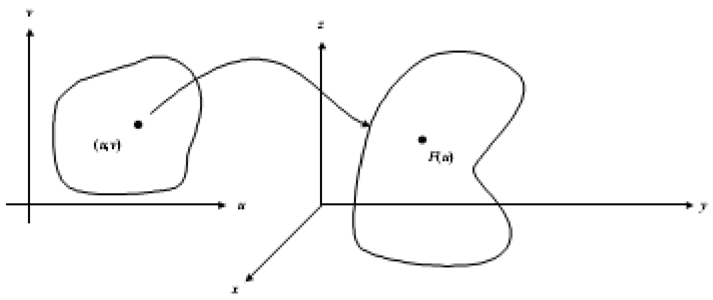
- 3D Curves:



$F: I \subset R \rightarrow R^3 \quad (x,y,z)=f(u)$

- Surface :

$F: I \subset R^2 \rightarrow R^3 \quad (x,y,z)=f(u,v)$



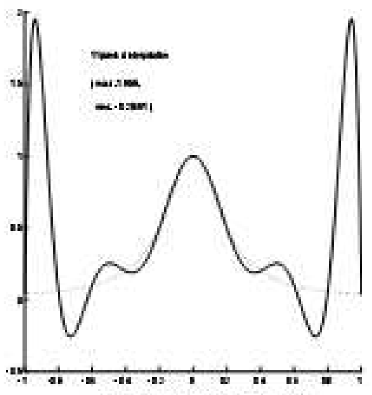
Comparison Summary:

Property	Parametric	Explicit	Implicit
Form	Coordinates as functions of parameters	One coordinate as a function of others	Relationship between coordinates without explicit function
Complexity	Handles complex shapes easily	Simple for basic shapes	Handles complex shapes but harder to visualize
Computational Cost	Depends on the parameterization	Simple to evaluate	Can be computationally expensive
Multi-valued	Can represent multi-valued functions	Cannot represent multi-valued functions	Can represent multi-valued functions
Applications	Animation, modeling, CAD, graphics	Simple geometry, basic plotting	Physics simulations, geometry, volume rendering

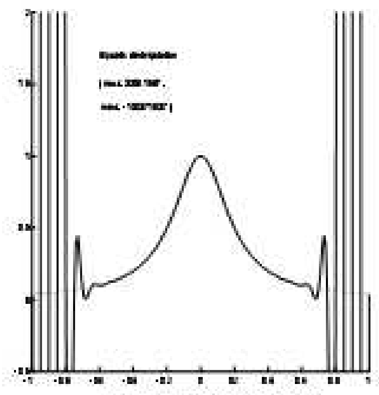
Each representation has its strengths, and the best choice depends on the problem you're solving—parametric is great for design and animation, explicit is simple for plotting, and implicit is powerful for handling complex shapes in simulations.

Piecewise polynomials approach:

The use of polynomials of high degree is tricky and sometimes leads to large oscillations. Polynomials of high degree are then not very adequate. The regularity of a function can be measured through its derivatives.



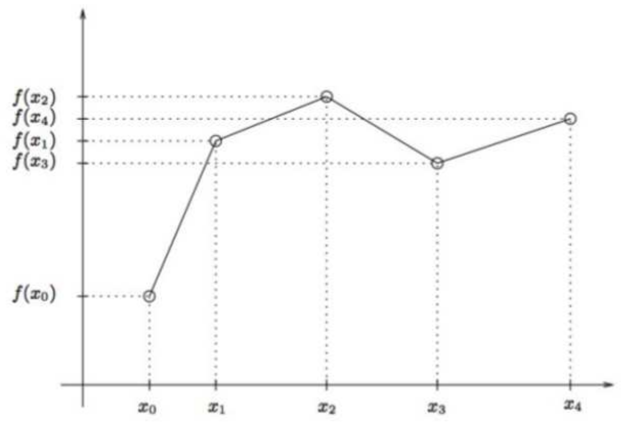
11 points d'interpolation



41 points d'interpolation

The goal of this approach is to keep the polynomial form, but to make sure that the degree of the polynomial is not increased. The idea for this is to make a function that is a succession of polynomials of low degree (degree 3 in the general case) such that the connections between polynomials are as regular as possible.

Let's first illustrate this idea with a simple example called "linear piecewise interpolation". The term linear splines is also used.

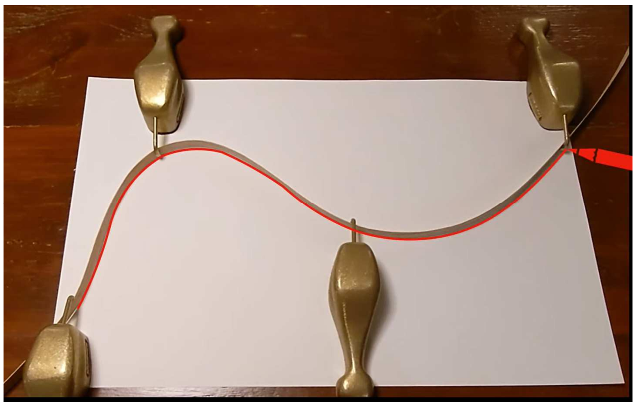


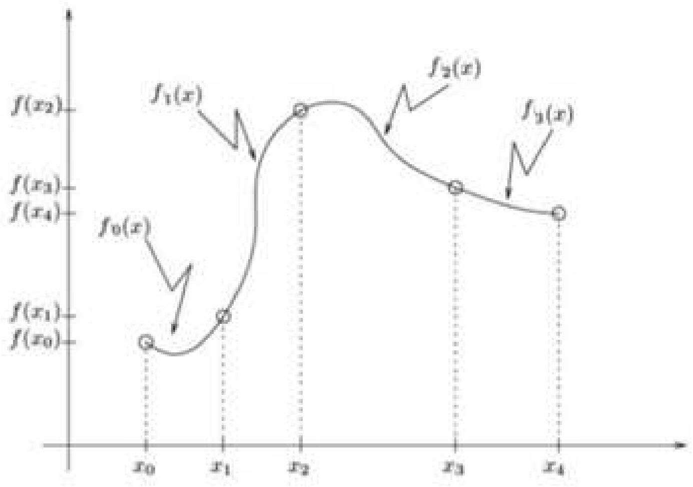
It is difficult to imagine how such a curve could be used to design a car body or an airplane wing. It is therefore necessary to be more careful at the junction of the different curve segments. The linear spline is continuous but is not differentiable and we will now show that we can do much better.

Cubic splines represent a very interesting compromise between the regularity of the curve obtained and the degree of the polynomials used.

Cubic spline interpolation :

We consider a set of points (x_0,y_0), (x_1,y_1), (x_{i-1},y_{i-1}), (x_i,y_i), (x_{i+1},y_{i+1}),… (x_{i-1},y_{i-1}). Pour réaliser An interpolation between these points one with cubic spline, a polynomial of the third degree is used between each two points.





The equation to the right of a point (xi,yi) is fi with a y value of fi(xi) at the point xi. Similarly, the equation to the left of a point (xi,yi) is fi+1 with a y-value of fi+1(xi) at the point xi.

The cubic spline function fi is constructed on the basis of the following criteria:

- Curves are polynomials of the third degree:

$$f_i(x) = a_i + b_i x + c_i x^2 + d_i x^3 \qquad - (1)$$

- the curves pass through all the points:

$$f_i(x_i) = y_i\ ,\ \ i = 0 \dots n \qquad (2)$$

$$f_i(x_i) = f_{i+1}(x_{i+1})\ ,\ \ i = 1 \dots n-1 \qquad (2')$$

- The tangent or first derivative is the same for both functions on both sides of a point.

$f'_i(x_i) = f'_{i+1}(x_{i+1})\ ,\ \ i = 1 \dots n-1$ (3)

- The second derivative is the same in both sides of a point:
- $f''_i(x_i) = f''_{i+1}(x_{i+1})\ ,\ \ i = 1 \dots n-1$ (4)

At the end we will have 4n unknowns with 4n-2 equation. The two missing equations are determined on the basis of the boundary conditions for the starting point f1(x0) and the ending point fn(xn). Historically, one of these conditions has been used.

Natural cubic spline: the second derivatives for the spline are considered null at the ends:

$$f_1''(x_0) = f_n''(x_n) = 0 \qquad \text{- (5a)}$$

Parabolic spline: the second derivatives at the ends are equal to those of the adjacent points. The result is that the curve becomes parabolic at these ends.

$$\begin{aligned} f_1''(x_0) &= f_1''(x_1) \\ f_n''(x_n) &= f_n''(x_{n-1}) \end{aligned} \qquad \text{- (5b)}$$

Not-a-knot Cubic Spline: The curve degrades to a single cubic curve on the last segments of both ends. To do this, the second derivatives equal to:

$$\begin{aligned} f_1''(x_0) &= 2f_1''(x_1) - f_2''(x_2) \\ f_n''(x_n) &= 2f_n''(x_{n-1}) - f_{n-1}''(x_{n-2}) \end{aligned} \qquad \text{- (5c)}$$

Usually, equations 2 to 5 are combined to obtain a system of tridiagonal equations n+1 by n+1, the solution of which gives us the parameters for each segment of the cubic spline.

Calculation of an interpolation cubic spline:

In each interval [*xi, xi+1*] (of length *h= xi+1- xi*) we can use a polynomial of the form:

$$p_i(x) = f_i + f_i'(x - x_i) + \frac{f_i''}{2!}(x - x_i)^2 + \frac{f_i'''}{3!}(x - x_i)^3 \quad \text{pour } i = 0, 1, 2, \cdots, n-1 \qquad \textbf{(6)}$$

This form is the Taylor development around the xi point
In express f_i, and as a function of as well as the conditions imposed by equations 2 to 4 we can arrive at the following expression:$f_i' f_i''' f_i''$

$$\frac{h_i}{(h_i+h_{i+1})}f''_i + 2f''_{i+1} + \frac{h_{i+1}}{(h_i+h_{i+1})}f''_{i+2} \quad = \quad 6f[x_i, x_{i+1}, x_{i+2}] \qquad \textbf{(7)}$$

$$\text{pour} \quad i = 0, 1, 2, \cdots, n-2$$

With:

- $h_i = x_{i+1} - x_i$
- $f[x_i, x_{i+1}] = \dfrac{f(x_{i+1}) - f(x_i)}{h_i}$
- $f[x_i, x_{i+1}, x_{i+2}] = \dfrac{f[x_{i+1}, x_{i+2}] - f[x_i, x_{i+1}]}{(h_i + h_{i+1})}$

And imposing one of the conditions at the extremities from equations 5a, 5b and 5c expressed as a function of f''_i as for a natural spline for example: $f''_0 = f''_3 = 0$

We can arrive at the system of tridiagonal equations as a function of f''_i the following:

$$\begin{pmatrix} \frac{h_0+h_1}{3} & \frac{h_1}{6} & 0 & & & \\ \frac{h_1}{6} & \frac{h_1+h_2}{3} & \frac{h_2}{6} & & 0 & \\ 0 & \frac{h_2}{6} & \frac{h_2+h_3}{3} & \frac{h_3}{6} & & \\ & & \ddots & \ddots & \ddots & \\ & 0 & & \frac{h_{n-3}}{6} & \frac{h_{n-3}+h_{n-2}}{3} & \frac{h_{n-2}}{6} \\ & & & & \frac{h_{n-2}}{6} & \frac{h_{n-2}+h_{n-1}}{3} \end{pmatrix} \begin{pmatrix} \sigma''_1 \\ \sigma''_2 \\ . \\ . \\ . \\ \sigma''_{n-1} \end{pmatrix} = \begin{pmatrix} \frac{\sigma_2-\sigma_1}{h_1} - \frac{\sigma_1-\sigma_0}{h_0} \\ \frac{\sigma_3-\sigma_2}{h_2} - \frac{\sigma_2-\sigma_1}{h_1} \\ . \\ . \\ . \\ \frac{\sigma_n-\sigma_{n-1}}{h_{n-1}} - \frac{\sigma_{n-1}-\sigma_{n-2}}{h_{n-2}} \end{pmatrix} \qquad \textbf{(8)}$$

($\sigma''_i = f''_i$)

Special case where the nodes are equidistant:

When the data is equidistant (*hi=h*) the system matrix is simplified:

$$\frac{1}{2}f''_i + 2f''_{i+1} + \frac{1}{2}f''_{i+2} \quad = \quad 6f[x_i, x_{i+1}, x_{i+2}] \qquad \textbf{(9)}$$

$$\text{pour} \quad i = 0, 1, 2, \cdots, n-2$$

In matrix form:

$$\begin{pmatrix} 4 & 1 & 0 & & & & \\ 1 & 4 & 1 & & & 0 & \\ 0 & 1 & 4 & 1 & & & \\ & & \ddots & \ddots & \ddots & & \\ & 0 & & & 1 & 4 & 1 \\ & & & & & 1 & 4 \end{pmatrix} \begin{pmatrix} \sigma''_1 \\ \sigma''_2 \\ . \\ . \\ . \\ \sigma''_{n-1} \end{pmatrix} = \frac{6}{h^2} \begin{pmatrix} \sigma_2 - 2\sigma_1 + \sigma_0 \\ \sigma_3 - 2\sigma_2 + \sigma_1 \\ . \\ . \\ . \\ \sigma_n - 2\sigma_{n-1} + \sigma_{n-2} \end{pmatrix} \qquad \textbf{(10)}$$

($\sigma''_i = f''_i$)

In summary, To perform interpolation using cubic splines, we must first calculate the second derivatives f''_i by solving the system of equation completed by the conditions at the ends.

Subsequently, we must determine the interval in which the interpolation point x is located and calculate the polynomial in this interval using formula 6 in which we replace:

$$\begin{cases} f_i &= f(x_i) \\ f_i' &= f[x_i, x_{i+1}] - \dfrac{h_i f_i''}{3} - \dfrac{h_i f_{i+1}''}{6} \\ f_i''' &= \dfrac{f_{i+1}'' - f_i''}{h_i} \end{cases} \qquad (11)$$

Exemple :

Considering the following 4 points: (1, 1), (2 , 4), (4 , 9), (5 , 11). All the information necessary to calculate the cubic spline can be found in the following table.

Données pour le calcul de la spline

i	x_i	$f(x_i)$	$f[x_i, x_{i+1}]$	$f[x_i, x_{i+1}, x_{i+2}]$	h_i
0	1	1			1
			3		
1	2	4		$-\frac{1}{6}$	2
			$\frac{5}{2}$		
2	4	9		$-\frac{1}{6}$	1
			2		
3	5	11			

The first equation (i = 0) of the system becomes:

$$\left(\frac{1}{3}\right) f_0'' + 2f_1'' + \left(\frac{2}{3}\right) f_2'' = 6\left(-\frac{1}{6}\right)$$

and the second equation (i = 1) is written:

$$\left(\frac{2}{3}\right) f_1'' + 2f_2'' + \left(\frac{1}{3}\right) f_3'' = 6\left(-\frac{1}{6}\right)$$

To obtain the natural spline ( ) , v$f_0'' = f_3'' = 0$/stem:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ \frac{1}{3} & 2 & \frac{2}{3} & 0 \\ 0 & \frac{2}{3} & 2 & \frac{1}{3} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f_0'' \\ f_1'' \\ f_2'' \\ f_3'' \end{bmatrix} = \begin{bmatrix} 0 \\ -1 \\ -1 \\ 0 \end{bmatrix}$$

whose solution is:

$$f_0'' = 0,\ f_1'' = -3/8,\ f_2'' = -3/8 \text{ et } f_3'' = 0.$$

To obtain the equation of the spline in the first interval, we must use the relations 6 and 11. We obtain:

$$\begin{aligned} f_0 &= 1 \\ f'_0 &= f[x_0, x_1] - \frac{h_0 f''_0}{3} - \frac{h_0 f''_1}{6} = 3 - \frac{(1)(0)}{3} - \frac{(1)(-3/8)}{6} = \frac{49}{16} \\ f'''_0 &= \frac{f''_1 - f''_0}{h_0} = -\frac{3}{8} \end{aligned}$$

et on a :

$$p_0(x) = 1 + \frac{49}{16}(x-1) + \frac{0}{2}(x-1)^2 - \frac{3}{48}(x-1)^3$$

This polynomial is defined only in the interval [1, 2]. For example, it can be evaluated in *x* = 1.5 to obtain 2.523 4375.

Avantages et inconvénients :

The spline curves are very harmonious. There is indeed no break in the radius of curvature. For this reason, cubic spline interpolation can be used in CAD.
Interpolation splines also have disadvantages:

- It is not possible to control a spline locally because the modification of the coordinates of a point influences the entire curve
- Can the spline also become oscillating if the derivatives of the function to be interpolated become too large (>> 1).
- The spline depends on the choice of coordinate system, so it does not have a geometric invariance property.

For points (0,0),(1,0.5), (2,2) and (3,1.5), find the interpolating cubic spline

satisfying and .

Solution:

▲

We can easily see that for so and

Also, since this is the type I boundary condition problem, we can calculate that

and

Therefore, plug into the system of equations, we have

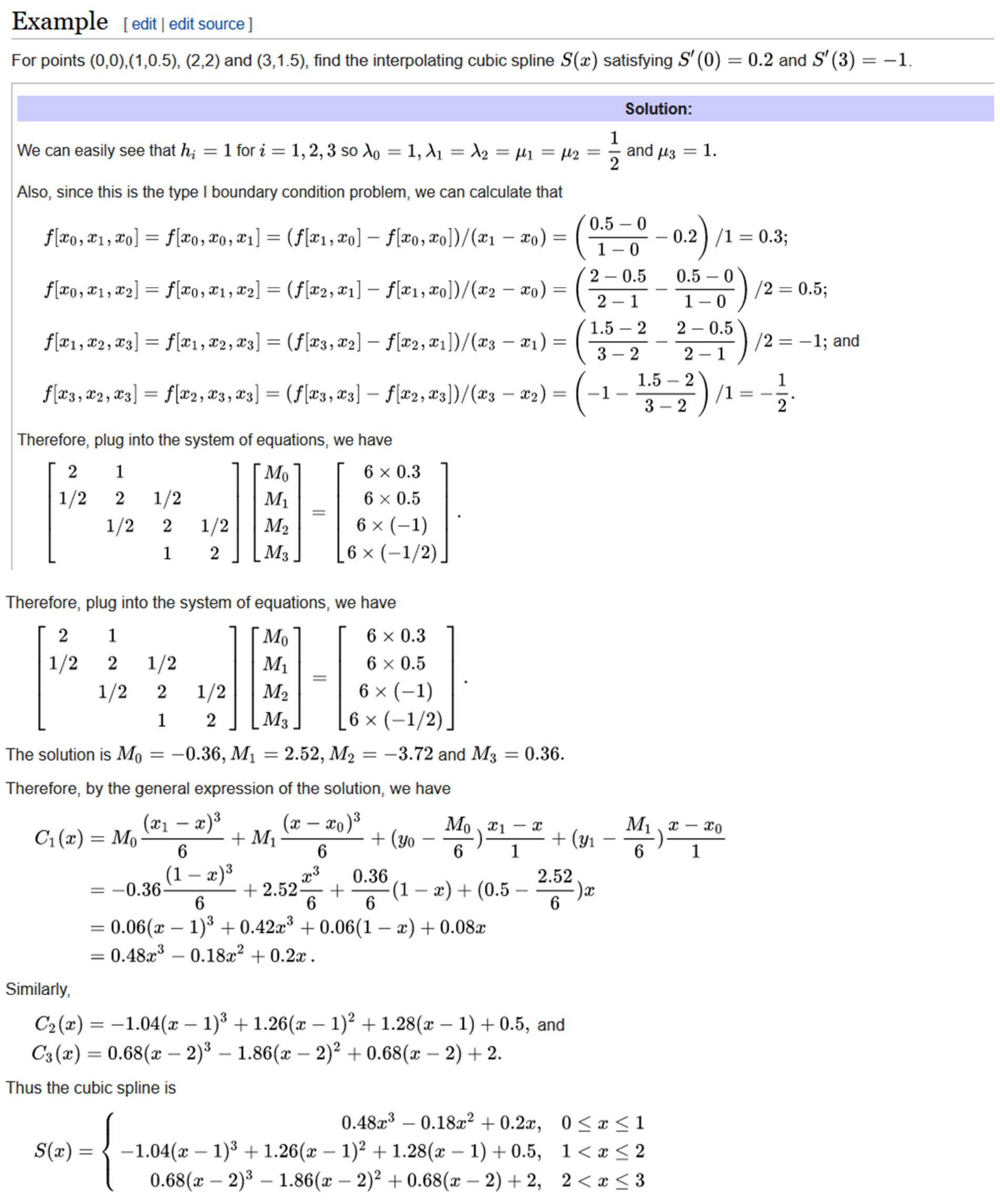
The solution is and

Therefore, by the general expression of the solution, we have

Similarly,

and

Thus the cubic spline is



Example [edit | edit source]

For points (0,0),(1,0.5), (2,2) and (3,1.5), find the interpolating cubic spline $S(x)$ satisfying $S'(0) = 0.2$ and $S'(3) = -1$.

Solution:

We can easily see that $h_i = 1$ for $i = 1, 2, 3$ so $\lambda_0 = 1, \lambda_1 = \lambda_2 = \mu_1 = \mu_2 = \frac{1}{2}$ and $\mu_3 = 1$.

Also, since this is the type I boundary condition problem, we can calculate that

$$f[x_0, x_1, x_0] = f[x_0, x_0, x_1] = (f[x_1, x_0] - f[x_0, x_0])/(x_1 - x_0) = \left(\frac{0.5-0}{1-0} - 0.2\right)/1 = 0.3;$$

$$f[x_0, x_1, x_2] = f[x_0, x_1, x_2] = (f[x_2, x_1] - f[x_1, x_0])/(x_2 - x_0) = \left(\frac{2-0.5}{2-1} - \frac{0.5-0}{1-0}\right)/2 = 0.5;$$

$$f[x_1, x_2, x_3] = f[x_1, x_2, x_3] = (f[x_3, x_2] - f[x_2, x_1])/(x_3 - x_1) = \left(\frac{1.5-2}{3-2} - \frac{2-0.5}{2-1}\right)/2 = -1; \text{ and}$$

$$f[x_3, x_2, x_3] = f[x_2, x_3, x_3] = (f[x_3, x_3] - f[x_2, x_3])/(x_3 - x_2) = \left(-1 - \frac{1.5-2}{3-2}\right)/1 = -\frac{1}{2}.$$

Therefore, plug into the system of equations, we have

$$\begin{bmatrix} 2 & 1 & & \\ 1/2 & 2 & 1/2 & \\ & 1/2 & 2 & 1/2 \\ & & 1 & 2 \end{bmatrix} \begin{bmatrix} M_0 \\ M_1 \\ M_2 \\ M_3 \end{bmatrix} = \begin{bmatrix} 6 \times 0.3 \\ 6 \times 0.5 \\ 6 \times (-1) \\ 6 \times (-1/2) \end{bmatrix}.$$

Therefore, plug into the system of equations, we have

$$\begin{bmatrix} 2 & 1 & & \\ 1/2 & 2 & 1/2 & \\ & 1/2 & 2 & 1/2 \\ & & 1 & 2 \end{bmatrix} \begin{bmatrix} M_0 \\ M_1 \\ M_2 \\ M_3 \end{bmatrix} = \begin{bmatrix} 6 \times 0.3 \\ 6 \times 0.5 \\ 6 \times (-1) \\ 6 \times (-1/2) \end{bmatrix}.$$

The solution is $M_0 = -0.36, M_1 = 2.52, M_2 = -3.72$ and $M_3 = 0.36$.

Therefore, by the general expression of the solution, we have

$$\begin{aligned} C_1(x) &= M_0\frac{(x_1-x)^3}{6} + M_1\frac{(x-x_0)^3}{6} + (y_0 - \frac{M_0}{6})\frac{x_1-x}{1} + (y_1 - \frac{M_1}{6})\frac{x-x_0}{1} \\ &= -0.36\frac{(1-x)^3}{6} + 2.52\frac{x^3}{6} + \frac{0.36}{6}(1-x) + (0.5 - \frac{2.52}{6})x \\ &= 0.06(x-1)^3 + 0.42x^3 + 0.06(1-x) + 0.08x \\ &= 0.48x^3 - 0.18x^2 + 0.2x\,. \end{aligned}$$

Similarly,

$$\begin{aligned} C_2(x) &= -1.04(x-1)^3 + 1.26(x-1)^2 + 1.28(x-1) + 0.5, \text{ and} \\ C_3(x) &= 0.68(x-2)^3 - 1.86(x-2)^2 + 0.68(x-2) + 2. \end{aligned}$$

Thus the cubic spline is

$$S(x) = \begin{cases} 0.48x^3 - 0.18x^2 + 0.2x, & 0 \le x \le 1 \\ -1.04(x-1)^3 + 1.26(x-1)^2 + 1.28(x-1) + 0.5, & 1 < x \le 2 \\ 0.68(x-2)^3 - 1.86(x-2)^2 + 0.68(x-2) + 2, & 2 < x \le 3 \end{cases}$$

Bezier Curves:

Because of the disadvantages mentioned above, polynomial interpolation by spline is not always very suitable for certain CAD applications, more specifically in the automotive industry. In the 1960s, numerical control machines appeared, so it was necessary to describe shapes (such as body curves) with mathematical equations.

The first solution was to linearly interpolate a large number of points. This method has many disadvantages:

1. For the machine, there are a lot of parameters.
2. It is impossible to enlarge (but also to translate, distort, . . .) a part of a part without adding additional stitches.
3. Placing points is not intuitive for designers.
4. It is very tedious to change the curve

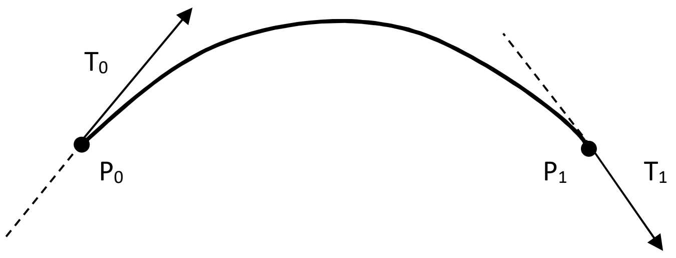
Another process was therefore necessary to express a curve with few parameters and for them to be natural.

The revolutionary idea of Bézier curves is the use of control points and not interpolation points. This means that the curve does not pass through the given points but approaches them. Bézier curves are therefore not interpolations but approximations. There are several advantages to this:

1. The curve is stable, it is easy to distort the curve without unexpected results.
2. It is easy to modify the curve, you only need to modify the control points which are few in number.
3. The placement of the control points is relatively obvious.
4. It is easier to have a natural, fluid, unabrupt curve with control points than with interpolation points.
5. And all this while keeping the advantages of the digital model (enlargement, deformation, etc.).

Cubic Hermite curves:

They are polynomial curves of degree 3 defined by the position and tangent to the starting and ending points.



$$\mathrm{C(x)} = a_0 + a_1 \cdot x + a_2 \cdot x^2 + a_3 \cdot x^3$$

In parametric form:

$$\mathrm{C}\,(t) = \begin{Bmatrix} x(t) = a_0 + a_1 \cdot t + a_2 \cdot t^2 + a_3 \cdot t^3 \\ y(t) = b_0 + b_1 \cdot t + b_2 \cdot t^2 + b_3 \cdot t^3 \\ z(t) = c_0 + c_1 \cdot t + c_2 \cdot t^2 + c_3 \cdot t^3 \end{Bmatrix} \qquad t \in [0,1]$$

$$\mathrm{C'}\,(t) = \begin{Bmatrix} x(t) = a_1 + 2a_2t + 3a_3t^2 \\ y(t) = b_1 + 2b_2t + 3b_3t^2 \\ z(t) = c_1 + 2c_2t + 3c_3t^2 \end{Bmatrix}$$

Constraints to be respected:

1. The curve passes through the point P1: C(0)=P1
2. The curve passes through the point P2: C(1)=P2
3. The tangent at the point P1 equal to T1: C'(0)=T1
4. The tangent to the point P2 equals T2: C'(1)=T2

Applying these constraints for the x-coordinate we arrive at the following system of equations:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 2 & 3 \end{bmatrix} \cdot \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} p_{0x} \\ p_{1x} \\ T_{0x} \\ T_{1x} \end{bmatrix}$$

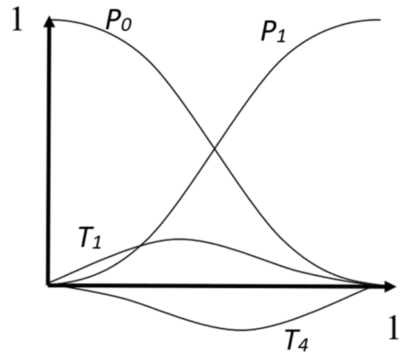
$$\begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -3 & 3 & -2 & -1 \\ 2 & -2 & 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} p_{0x} \\ p_{1x} \\ T_{0x} \\ T_{1x} \end{bmatrix}$$

This gives:

$$\mathrm{x(t)} = (1-3t^2+2t^3)p_{0x} + (3t^2-2t^3)p_{1x} + (t-2t^2+t^3)T_{0x} + (-t^2+t^3)T_{1x}$$

By performing the same resonance for the y and z coordinates, we obtain:

$$\mathrm{C(t)} = (1-3t^2+2t^3)p_0 + (3t^2-2t^3)p_1 + (t-2t^2+t^3)T_0 + (-t^2+t^3)T_1$$

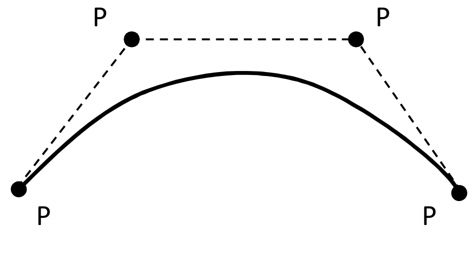


Cubic Bézier curve:

These are polynomial curves of degree 3 defined by the positions of four points.

In this case, the tangents to the first and last point will be deduced by the respective positions of the first two and the last two points.

$$T_0 = 3(p_1 - p_0), T_1 = 3(p_3 - p_2)$$



The result is the following formulation:

$$C(t) = (1-t)^3 p_0 + 3t(1-t)^2 p_1 + 3t^2(1-t) p_2 + t^3 p_3$$

Sous forme matricielle :

$$C(t) = \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} p_0 \\ p_1 \\ p_2 \\ p_3 \end{bmatrix}$$

Which can also be put in the form:

$$\mathrm{C(t)} = B_0(t) p_0 + B_1(t) p_1 + B_2(t) p_2 + B_3(t) p_3$$

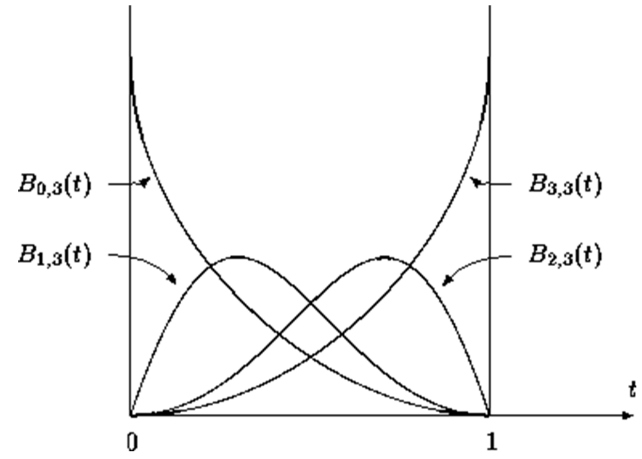
Avec :

$$B_0(t) = (1 - t^3)$$

$$B_1(t) = 3t(1-t)^2$$

$$B_2(t) = 3t^2(1-t)$$

$$B_3(t) = t^3$$



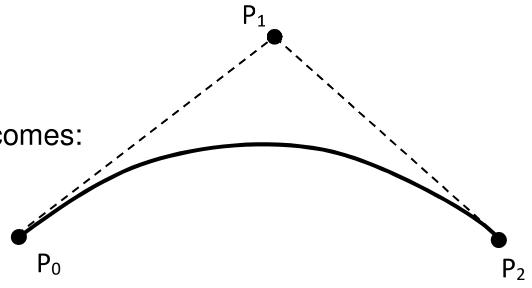
And cubic Bézier curve can also be expressed in compact form as follows:

$$C(t) = \sum_{i=0}^{3} P_i B_{i,3}(t)$$

The Bi,3(t) functions are the cubic Bernstein functions.

The general form of these functions at a degree n is given by the following expression:

$$B_{i,n}(t) = \left(\frac{n!}{i!\,(n-i)!} \right) t^i (1-t)^{n-i}$$



The general expression of a Bézier curve of degree n becomes:

$$C(t) = \sum_{i=0}^{n} P_i B_{i,n}(t)$$

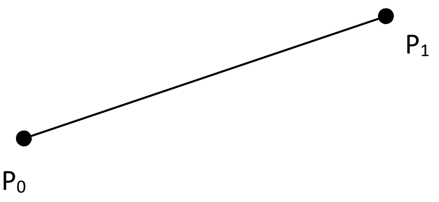
For a quadratic Bézier curve (degree 2) :

$$C(t) = \sum_{i=0}^{2} P_i B_{i,2}(t)$$

$$C(t) = (1-t)^2 p_0 + 2t(1-t) p_1 + t^2 p_2$$

A first-degree Bézier curve:

$$\mathrm{C(t)} = (1-t) p_0 + t p_1$$



Determination of a point on a Bézier curve:

To determine the coordinates (x, y, z) for a given value of the parameter t on a Bézier curve of degree *n*, it is always possible to use the general formulation, but it is rarely used in practice because it can lead to approximation errors during its programming.

In practice, a method based on **De Casteljau's algorithm** is usually used, named after the mathematician Paul De Casteljau, who had also been working at Citroën on the definition of free forms since 1959, but this work was kept secret until 1975. This algorithm is based on the following idea: if we take a Bézier curve, of degree 3 for example, it is a combination of two quadratic curves

$$\mathrm{C(t)} = (1-t^3) p_0 + 3t(1-t)^2 p_1 + 3t^2(1-t) p_2 + t^3 p_3$$

$$C(t) = (1-t)\underbrace{((1-t)^2 p_0 + 2t(1-t) p_1 + t^2 p_2)}_{quadratiqu\ e} + t\underbrace{((1-t)^2 p_1 + 2t(1-t) p_2 + t^2 p_3)}_{quadratiqu\ e}$$

This idea can be generalized for a curve of degree *n* which can give in this case two curves of degree n-1 as follows:

$$C^n(p_0,\ldots,p_n) = (1-t)C^{n-1}(p_0,\ldots,p_{n-1}) + tC^{n-1}(p_1,\ldots,p_n)$$

This expression can be rewritten according to the checkpoints to obtain the following formulation:

$$P_i^k(t_0) = (1-t_0)P_i^{k-1}(t_0) + t_0 P_{i+1}^{k-1}(t_0), \quad \begin{cases} k = 1,\ldots,n \\ i = 0,\ldots,n-k \end{cases}$$

We can verify with this formulation that:

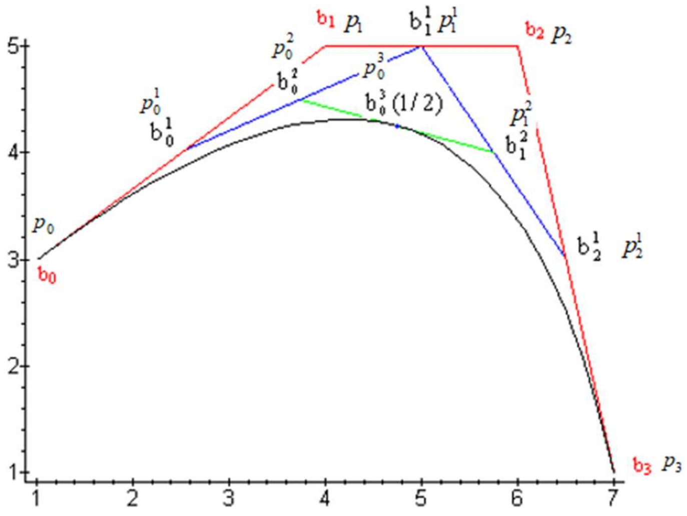
$$C(t_0) = P_0^n(t_0)$$

<u>Prenons un exemple : dans le cas où n=3</u>

We take a plan with a reference point $(O,\vec{i},\vec{j})$ lets : P_0 (1,3), P_1 (4,5), P_2 (6,5), P_3 (7,1), $t \in [0.1]$

The intermediate coefficients can be written as a triangular table of points, called the De Casteljau scheme :

$$\begin{array}{llll} p_0 & & & \\ p_1 & p_0^1 & & \\ p_2 & p_1^1 & p_0^2 & \\ p_3 & p_2^1 & p_1^2 & p_0^3 \end{array}$$



Disadvantages of Bézier curves:

Bézier curves have a lot of advantages for practical use. Nevertheless, they have some disadvantages:

1. Because the curve does not pass through the control points, it can be difficult to control the curve, although, as we have seen, the use of control points often makes the design easier. Thus, Bézier curves would not be very efficient in drawing a trend line on measurements. For these types of problems, we turn more to cubic interpolation and cubic spline interpolation.
2. Another disadvantage is the lack of local control, the shift of one point causes the whole curve to move. In the automotive industry, for example, it is annoying that the whole part changes shape when we only want to vary a part of the part. In terms of calculations, the whole room will have to be calculated again.
3. The third disadvantage is the degree of the curves. Indeed, for a complex shape, we have to use a lot of control points so the degree of the curve is high.

B-Splines:

It is a question of building a curve that senses all the advantages of Bézier curves but without its disadvantages. Thus, the curve must approximate the control points, be simple to manipulate, have the same properties as Bézier curves, etc. The degree of the curve should not be proportional to the number of checkpoints but. Changing a point should not affect the entire curve.

B-Splines were developed at Boeing in the 70s and 80s. The main idea of B-Splines is to replace Bernstein polynomials with functions. Then, we'll sum these functions with the control points to get the curve. A B-Spline does not only depend on control points but also on a knot vector.

Courbes B-Splines :

A B-Spline curve is defined by:

$$C(t) = \sum_{i=0}^{n} P_i N_{i,k}(t)$$

With:

1. Pi (i=1,...,n-1) are the control points.
2. Ni,k the basic functions of B-Spline

For k=1:

$$N_{i,1}(t) = \begin{cases} 1 & \text{if } t \in [t_i, t_{i+1}) \\ 0 & \text{otherwise.} \end{cases} \qquad (1)$$

Pour k>1 :

$$N_{i,k}(t) = \left(\frac{t - t_i}{t_{i+k-1} - t_i}\right) N_{i,k-1}(t) + \left(\frac{t_{i+k} - t}{t_{i+k} - t_{i+1}}\right) N_{i+1,k-1}(t) \quad , t \in [k-1, n+1) \qquad (2)$$

Remarks:

1. Curve defined by k pieces of degree k-1, k is the order of the segments of the curve and therefore the degree of the B-Spline does not depend on the number of control points.
2. Each piece of curve is defined on an interval [ti; ti+1].
3. Each Ni,k has a support [ti , ti+k]

   Therefore:

   1. N0,k (t) a un support [t0 , tk]
   2. Nn,k(t) a un support [tn , tn+k]
4. Therefore: there must be (n + k + 1) nodes, the nodal vector is then [t0, t1,..., tn+k]
5. The change: Let be a Pi and Ni checkpoint$_{;\,k}$(t) the associated basic function. So, if we move Pi we only change the part of the curve for which t € [ti; ti+k]. So the modification is local while it is global for the Bézier curve.
6. Each basic function Ni,k(t) covers k intervals.
7. There is at most k Ni; k(t) non-zero over an interval.
8. So each piece of curve is defined by k control points:
   1. the 1st piece : (P0, P1,…, Pk$_{-1}$)
   2. the last piece: (Pn-k+1,..., Pn)
9. The set of nodes that *ti* (i=0,..., n+k) is an increasing sequence of real numbers. The vectoof knots can be:

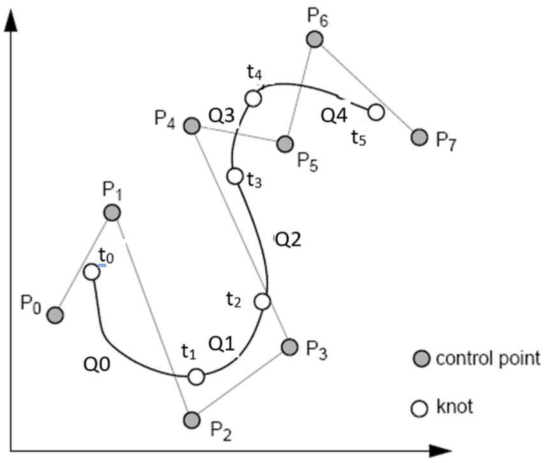
   Uniform: where the elements of the vector are spaced with a constant pitch.

   Examples: {0 1 2 3 4 5 6 7}, {-0.2 -0.1 0.2 0.1}, {0 0.25 0.5 0.75 1}

   Non-Uniform: where the spacing interval between the elements of the vector is different.

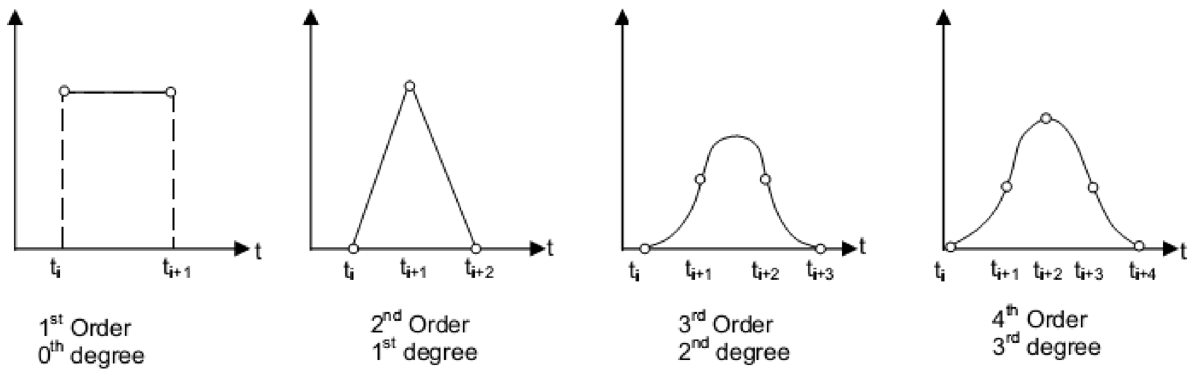
   Examples: {0 0 0 1 1 2 2 2}, {0 0.25 0.75 0.8 1}, {0 1 1 1 2 3 4 5}, If then the coefficient we have, we set this quotient equal to 0.$\left(\frac{\cdots}{0}\right)$

$$t_i = t_{i+1} \qquad \frac{t - t_i}{t_{i+k} - t_i}$$



Graph of Ni,k functions:

For order 1, the basic functions will have constant values on their supports. The function of order 2 is composed of two line segments that can be expressed by polynomials of degree 1. For order 3, which is most often used, the functions N are composed of three segments of degree 2, two hyperbolas, and a parabola in the middle.



Support or range of basic functions:

We have seen on graphs that a function Ni,k(t) cancels outside the interval [ti, ti+k[. We can recursively demonstrate that for example Ni,3(t) is obtained from Ni,1(t), Ni$_{+1,1}$(t), Ni$_{+2,1}$(t), Ni$_{+3,1}$(t). and we have defined that: Ni,1(t) = $1_{[ti,\ ti+1]}$. So Ni,3(t) is indeed zero outside the interval [ti, ti+3[. In the same way we can unmount Ni,k(t) cancels outside of the interval [ti, ti+k[.

Uniform cubic B-splines:

The ti parameters are uniformly distributed over all control points, namely:

$$t_i = t_{i-1} + 1.$$

So *t* varies by 1 for a curve segment and we make the change of variable *t=t-ti* . The position of a point P is then defined by:

$$P(t) = \frac{1}{6}[(1-t)^3 P_{i-3} + (3t^3 - 6t^2 + 4)P_{i-2} + (-3t^3 + 3t^2 + 3t + 1)P_{i-1} + t^3 P_i],$$

Pour : $0 \leq t < 1.$

Or in matrix form:

$$P(t) = (t^3, t^2, t, 1) \cdot \frac{1}{6} \begin{pmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 0 & 3 & 0 \\ 1 & 4 & 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} P_{i-3} \\ P_{i-2} \\ P_{i-1} \\ P_i \end{pmatrix}$$

$$P(t) = T^t \cdot M_{bspline} \cdot P.$$

Pros and cons of B-spline:

B-Splines solve the problems we have seen for Bézier curves, it is possible to control B-Splines locally and adding points does not increase the degree of the curve. In addition, it is possible to interpolate the checkpoints even if it is not very conclusive.
The main disadvantage of B-Splines is their complexity. Indeed, it is not easy to calculate the basic functions. Control points are no longer the only parameters of curves, there is also the knot vector. It is difficult to manage points and nodes at the same time. That's why we only vary the checkpoints.

Rational polynomial curves:

Segments of rational curves are ratios of polynomials:

$$x(t) = X(t)/W(t),\ y(t) = Y(t)/W(t),\ z(t) = Z(t)/W(t).$$

Where $X(t), Y(t), Z(t), W(t)$ are polynomial curves whose control points are defined in homogeneous coordinates. All the definitions previously introduced apply here by adding a third (2D) or fourth (3D) coordinate to the points treated. We can therefore calculate rational Bézier or non-uniform rational B-splines (NURBS). The interest of rational curves is twofold:

1. Rational curves are invariant by rotation, change of scale, translation, and perspective projection. This means that a transformation can be applied to checkpoints only.
2. A second interest is that rational (quadratic) splines allow conics to be accurately traced without the need for many control points.

NURBS Curves:

The only differences between NURBS and B-Splines are the type of node vector chosen and the wi weight vector.
The formulation of a NURBS curve is as follows:

$$C(t) = \sum_{i=0}^{n} \frac{N_{i,k} w_i}{\sum_{j=0}^{n} N_{j,k} w_j} P_i = \frac{\sum_{i=1}^{n} N_{i,k} w_i P_i}{\sum_{i=1}^{n} N_{i,k} w_i}$$

Which can be put in the form:

$$C(t) = \sum_{i=0}^{n} R_{i,k}(t) P_i$$

With

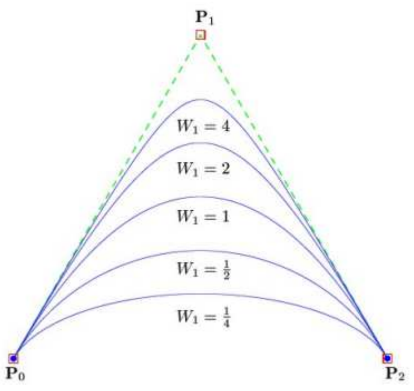
$$R_{i,k}(t) = \frac{\sum_{i=1}^{n} N_{i,k} w_i P_i}{\sum_{i=1}^{n} N_{i,k} w_i}$$

The knot vector for NURBS is of the non-uniform type, i.e. the spacing between two consecutive knot values is not constant. As an example :

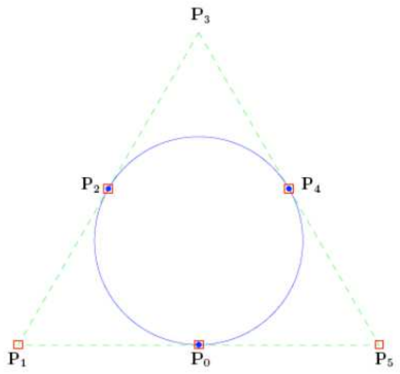
$$\begin{aligned} \mathbf{t} &= [t_1\ t_2\ t_3\ \dots\ t_{\text{n+d-1}}\ t_{\text{n+d}}\ t_{\text{n+d+1}}] \\ &= [1\ \ 1.5\ \ 5\ \ 8\ \ 9\ \ 11\ \ \dots\ \ n+d-1\ \ \ n+d\ \ \ n+d+1] \end{aligned}$$

However, the definition of spline functions remains the same, with the only difference being that we cannot express the resulting spline functions in polynomial form and thus, in matrix form.

For the wi weight vector, each value in the vector is associated with a Pi control point. The weight value affects the effect that the checkpoint has on the shape of the patch (segment) to which it is associated.
This effect can be compared to a magnet: if wi > 1, the patch to which the associated Pi checkpoint belongs moves closer to the checkpoint. If wi< 0, the patch to which the associated Pi checkpoint belongs moves away from the checkpoint. Finally, if wi = 1, no effect is applied to the patch to which the associated Pi checkpoint belongs.



$\mathbf{T} = (0, 0, 0, 1, 1, 1)$
$\mathbf{W} = (1, W_1, 1)$
$\mathbf{P} = \{(0, 0); (1, \sqrt{3}); (2, 0)\}$



$\mathbf{T} = (0, 0, 0, 1, 1, 2, 2, 3, 3, 3)$
$\mathbf{W} = (1, \frac{1}{2}, 1, \frac{1}{2}, 1, \frac{1}{2}, 1)$
$\mathbf{P} = \{(1, 0); (0, 0); (\frac{1}{2}, \frac{\sqrt{3}}{2}); (1, \sqrt{3}); (\frac{3}{2}, \frac{\sqrt{3}}{2}); (2, 0); (1, 0)\}$