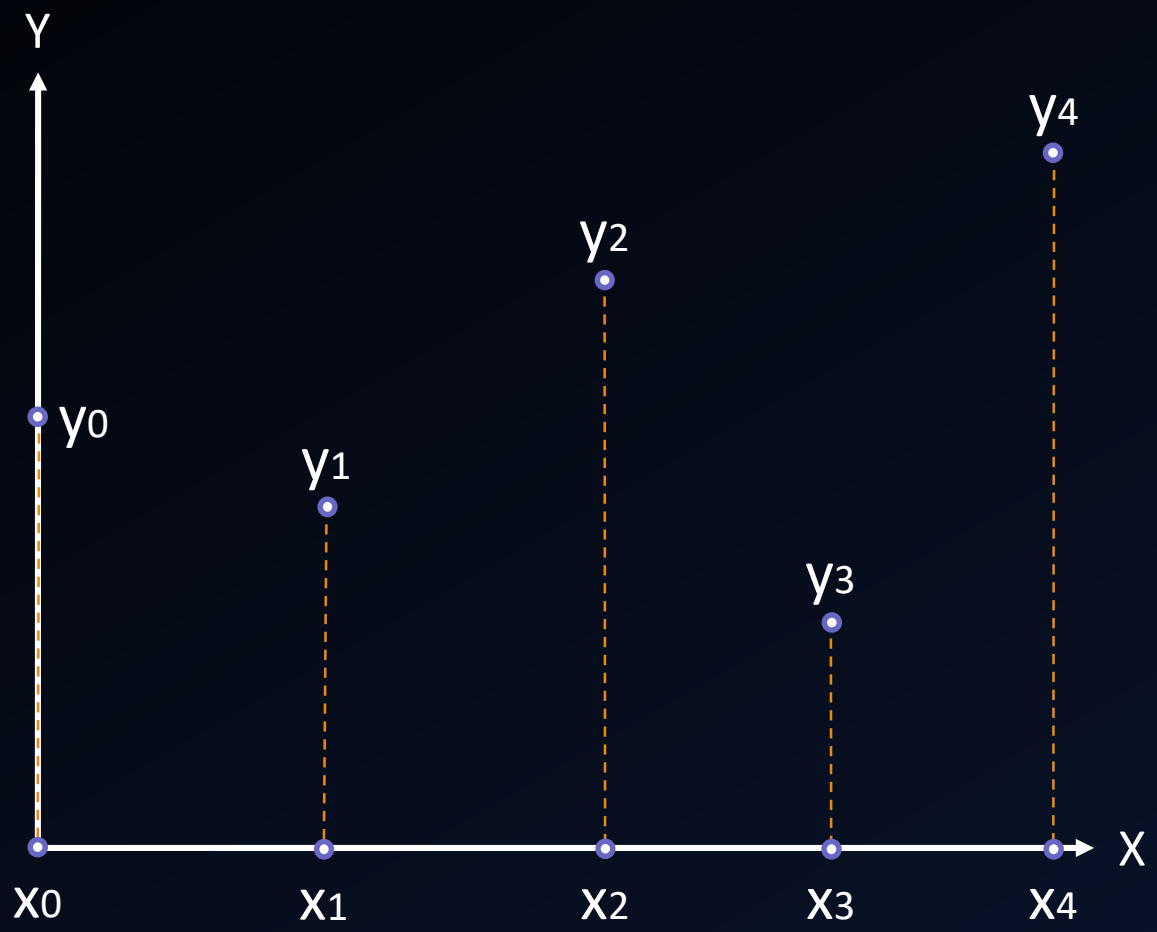
# CURVE FITTING

- Interpolation
  - Linear
  - Polynomial
  - Spline
- Approximation
  - Bezier Curves
  - B-splines

Author: prof. Yevhenii Borodavka

# Interpolation

In the mathematical field of numerical analysis, interpolation is a type of estimation, a method of constructing (finding) new data points based on the range of a discrete set of known data points.

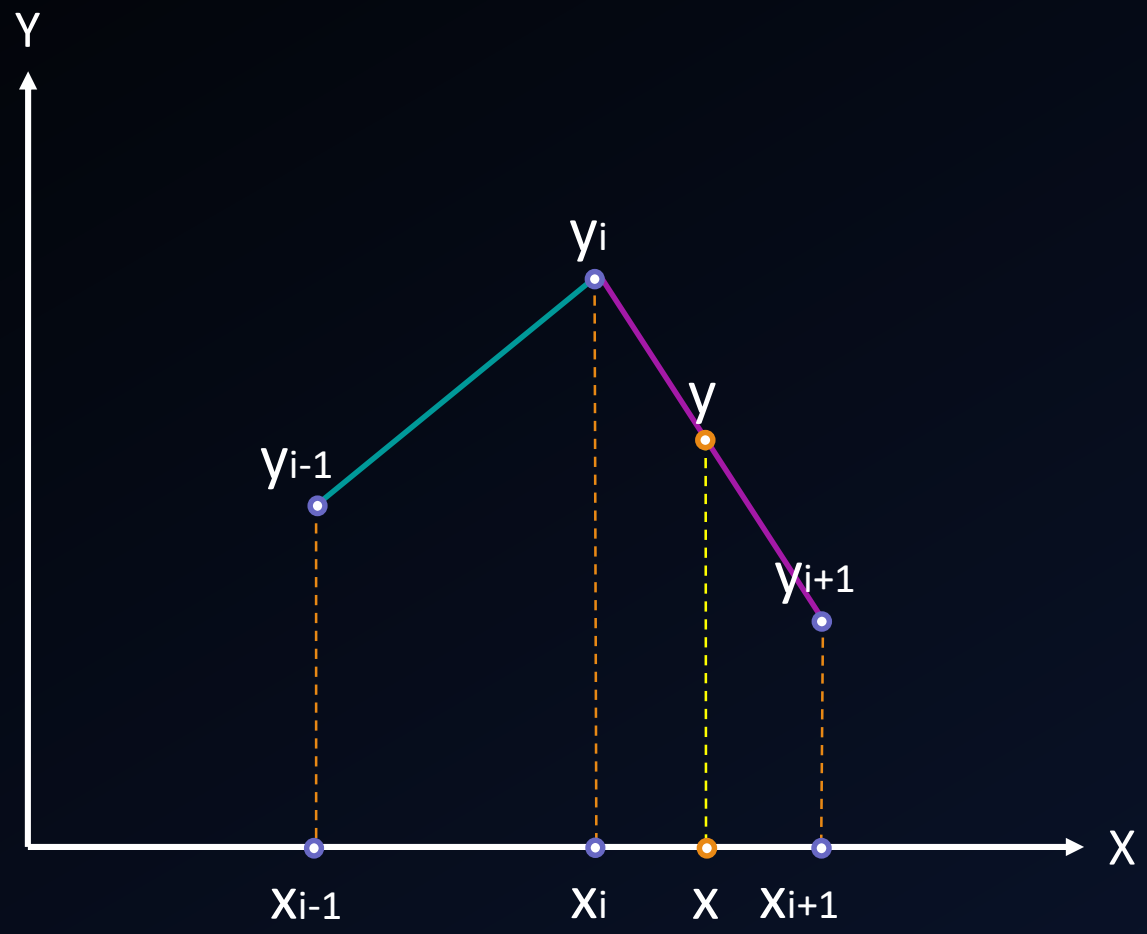



In engineering and science, one often has a number of data points, obtained by sampling or experimentation, which represent the values of a function for a limited number of values of the independent variable. It is often required to interpolate; that is, estimate the value of that function for an intermediate value of the independent variable.

$x_i < x_{i+1}$

# Interpolation : Linear

In mathematics, linear interpolation is a method of curve fitting using linear polynomials to construct new data points within the range of a discrete set of known data points.




Linear interpolation is performed on each interval between two points of the entire set. The resulting interpolation of the curve is the union of individual particles.

$$\frac{y - y_i}{y_{i+1} - y_i} = \frac{x - x_i}{x_{i+1} - x_i}$$

$$y = y_i + (x - x_i) \cdot \frac{(y_{i+1} - y_i)}{(x_{i+1} - x_i)}$$

# Interpolation : Polynomial

In numerical analysis, polynomial interpolation is the interpolation of a given bivariate data set by the polynomial of the lowest possible degree that passes through the points of the dataset. There is always a unique such polynomial, commonly given by two explicit formulas, the Lagrange polynomials and Newton polynomials (divided-differences method).

The Lagrange polynomial formula:

$$L(x) = \sum_{j=0}^{n} y_j l_j(x) \qquad l_j(x) = \prod_{i=0, j\neq i}^{n} \frac{(x - x_i)}{(x_i - x_j)}$$

The Newton polynomial formula:

$$L_n(x) = f(x_0) + (x - x_0)f(x_0, x_1) + (x - x_0)(x - x_1)f(x_0, x_1, x_2) + \cdots$$
$$+(x - x_0)(x - x_1) \dots (x - x_{n-1})f(x_0, x_1, \dots, x_n)$$

$$f(x_0, x_1, \dots, x_n) = \sum_{i=0}^{n} \frac{f(x_i)}{(x_i - x_0) \dots (x_i - x_{i-1})(x_i - x_{i+1}) \dots (x_i - x_n)}$$

# Interpolation : Polynomial

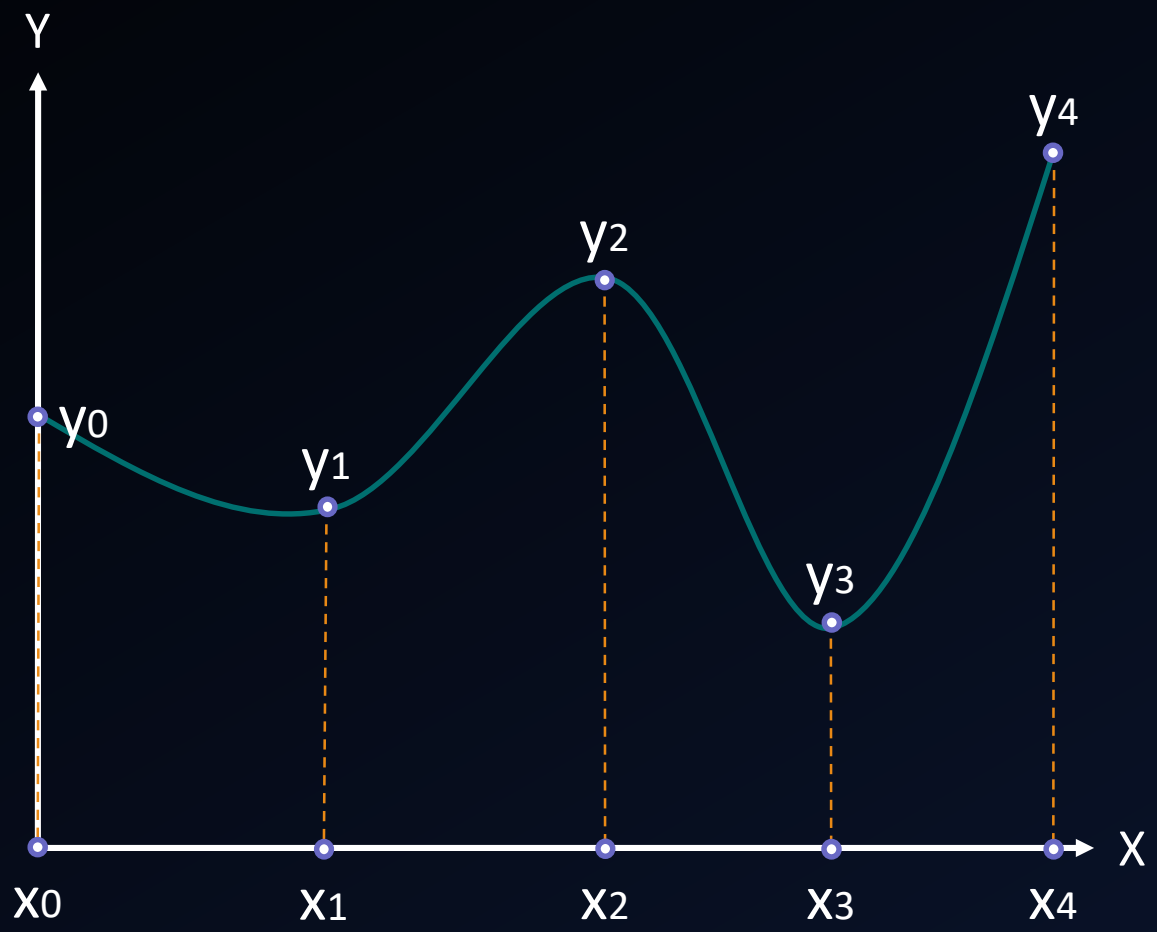
Lagrange is sometimes said to require less work and is sometimes recommended for problems in which it is known, in advance, from previous experience, how many terms are needed for sufficient accuracy.

The divided difference methods have the advantage that more data points can be added, for improved accuracy. The terms based on the previous data points can continue to be used. With the ordinary Lagrange formula, to do the problem with more data points would require re-doing the whole problem.

As can be seen from the definition of the divided differences new data points can be added to the data set to create a new interpolation polynomial without recalculating the old coefficients. And when a data point changes we usually do not have to recalculate all coefficients. Furthermore, if the $x_i$ are distributed equidistantly the calculation of the divided differences becomes significantly easier. Therefore, the divided-difference formulas are usually preferred over the Lagrange form for practical purposes.

# Interpolation : Spline

In the mathematical field of numerical analysis, spline interpolation is a form of interpolation where the interpolant is a special type of piecewise polynomial called a spline.

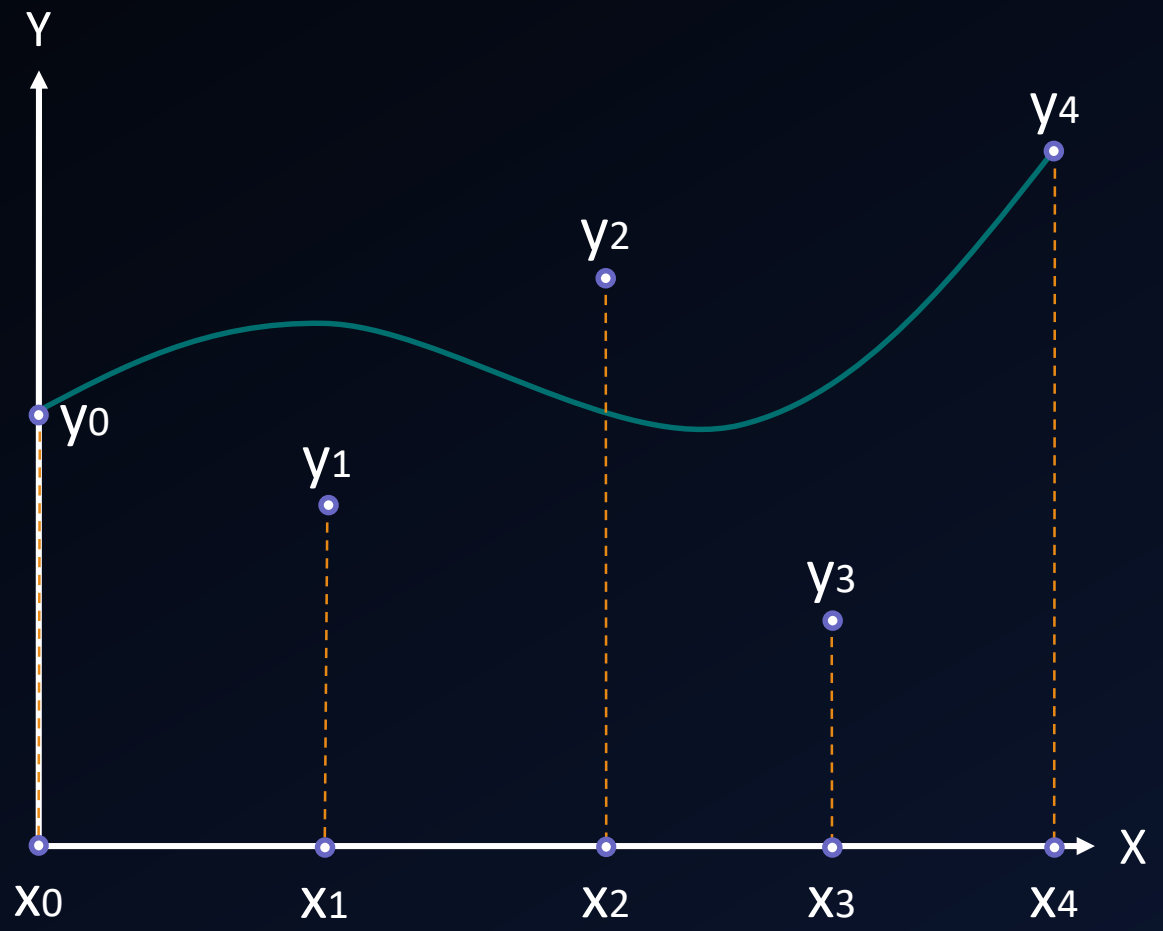



Originally, *spline* was a term for elastic rulers that were bent to pass through a number of predefined points, or *knots*. These were used to make technical drawings for shipbuilding and construction by hand, as illustrated in the figure.

The spline can only be achieved if polynomials of degree 3 or higher are used. The classical approach is to use polynomials of exactly degree 3 — cubic splines.

# Approximation

In opposite to interpolation methods of curve fitting there are methods that try to fit curves approximately to knots. The most known methods are the Bezier curve and B-splines.



# Approximation : Bezier curves

The Bézier curve is named after French engineer Pierre Bézier (1910–1999), who used it in the 1960s to design curves for Renault cars' bodywork. The mathematical basis for Bézier curves — the Bernstein polynomials — was established in 1912, but the polynomials were not applied to graphics until some 50 years later when mathematician Paul de Casteljau in 1959 developed de Casteljau's algorithm, a numerically stable method for evaluating the curves.

A set of ”control points” defines a Bézier curve *points* $\mathbf{P}_0$ through $\mathbf{P}_n$, where $n$ is called the order of the curve ($n$ = 1 for linear, 2 for quadratic, 3 for cubic, etc.). The first and last control points are always the curve’s endpoints; however, the intermediate control points generally do not lie on the curve.

Given distinct points $\mathbf{P}_0$ and $\mathbf{P}_1$, a linear Bézier curve is simply a line between those two points. The curve is given by

$$B(t) = P_0 + t(P_1 - P_0) = (1 - t)P_0 + tP_1, 0 \leq t \leq 1$$

# Approximation : Bezier curves

A quadratic Bézier curve is the path traced by the function **B**($t$), given points $\mathbf{P}_0$, $\mathbf{P}_1$, and $\mathbf{P}_2$

$$B(t) = (1-t)[(1-t)P_0 + tP_1] + t[(1-t)P_1 + tP_2], 0 \leq t \leq 1$$

which can be interpreted as the linear interpolant of corresponding points on the linear Bézier curves from $\mathbf{P}_0$ to $\mathbf{P}_1$ and from $\mathbf{P}_1$ to $\mathbf{P}_2$ respectively. Rearranging the preceding equation yields:

$$B(t) = (1-t)^2 P_0 + 2(1-t)tP_1 + t^2 P_2, 0 \leq t \leq 1$$

Four points $\mathbf{P}_0$, $\mathbf{P}_1$, $\mathbf{P}_2$, and $\mathbf{P}_3$ in the plane or in higher-dimensional space define a cubic Bézier curve. The curve starts at $\mathbf{P}_0$ going toward $\mathbf{P}_1$ and arrives at $\mathbf{P}_3$ coming from the direction of $\mathbf{P}_2$. Usually, it will not pass through $\mathbf{P}_1$ or $\mathbf{P}_2$; these points are only there to provide directional information.

$$B(t) = (1-t)^3 P_0 + 3(1-t)^2 tP_1 + 3(1-t)t^2 P_2 + t^3 P_3, 0 \leq t \leq 1$$

# Approximation : Bezier curves

Bézier curves can be defined for any degree *n*. The parametric formula of the Bézier curves is:

$$B(t) = \sum_{i=0}^{n} P_i b_i^n(t)\,, 0 \le t \le 1$$

where Pi are called *control points* for the Bézier curve
$b_i^n(t)$ the polynomials are known as Bernstein basis polynomials of degree *n:*

$$B_i^n(t) = C_i^n t^i (1-t)^{n-i},$$

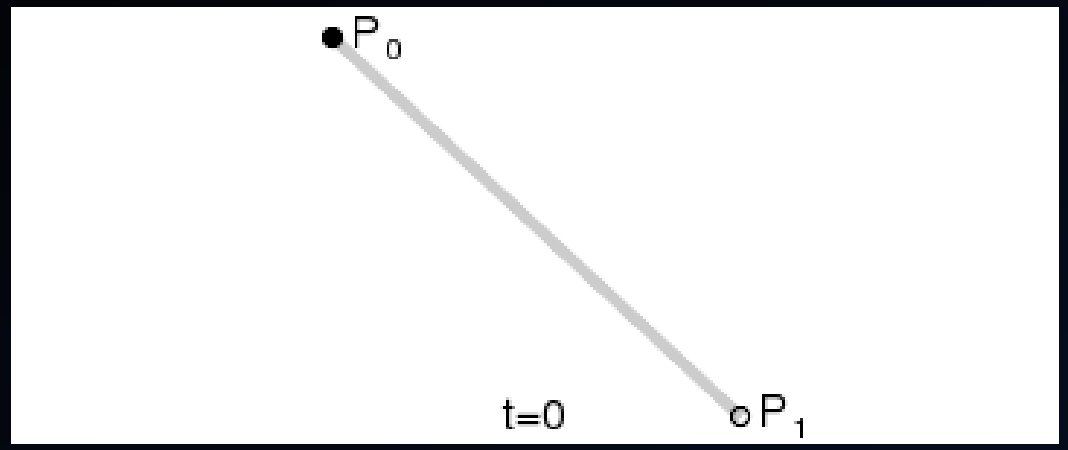
$C_i^n$ is the binomial coefficient:

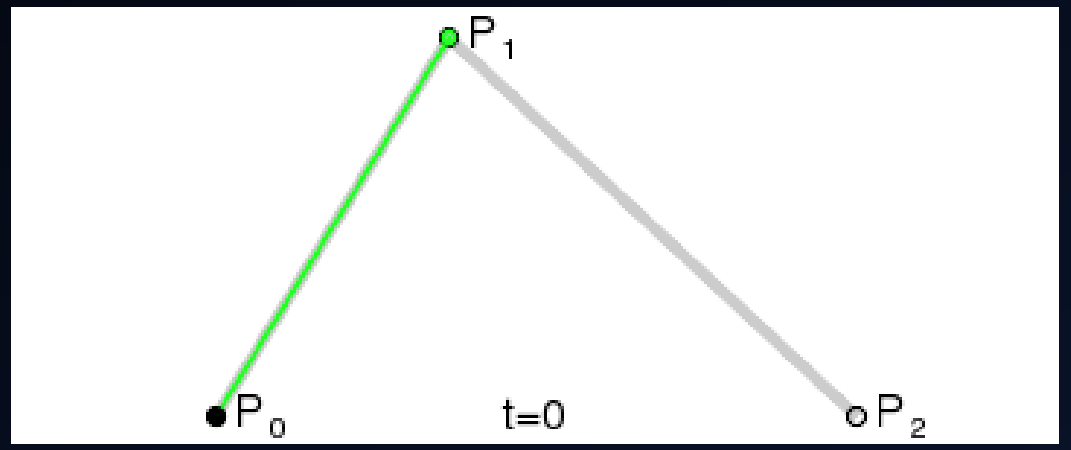
$$C_i^n(t) = \frac{n!}{i!\,(n-i)!}$$

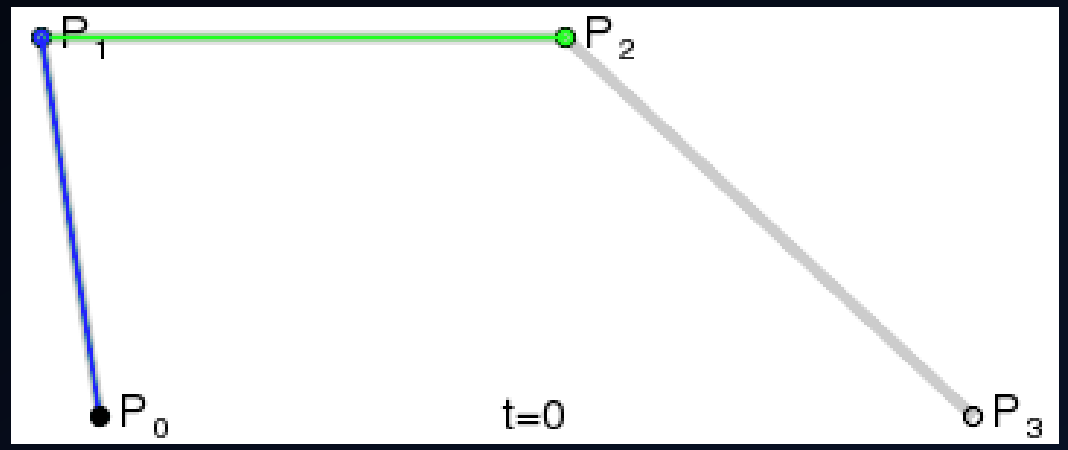
There are also the recursive definition, explicit definition, and polynomial form of the Bézier curves.

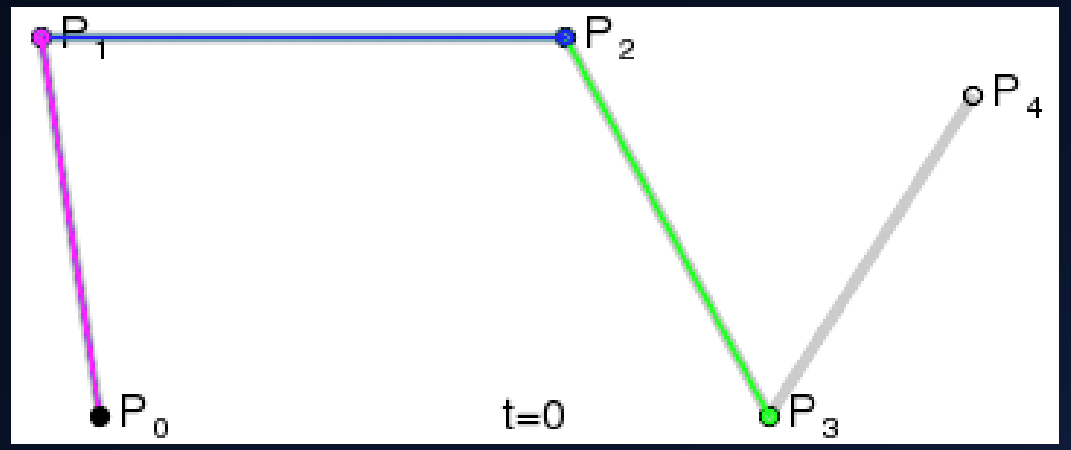
# Approximation : Bezier curves

The examples of the Bézier curves of degree 1, 2, 3, and 4.



# Approximation : B-splines

In the mathematical subfield of numerical analysis, a B-spline or basis spline is a spline function that has minimal support with respect to a given degree, smoothness, and domain partition. Any spline function of a given degree can be expressed as a linear combination of B-splines of that degree. Cardinal B-splines have knots that are equidistant from each other. B-splines can be used for curve-fitting and numerical differentiation of experimental data.

The concept of the **B-spline** curve came to resolve the disadvantages having by the **Bezier curve,** as we all know that both curves are parametric in nature. In the Bezier curve we face a problem, when we change any of the control points respective locations the whole curve shape changes. But here in the B-spline curve, only a specific segment of the curve shape gets changed or affected by the changing of the corresponding location of the control points.

In the **B-spline curve**, the control points impart local control over the curve shape rather than global control like the **Bezier curve.**

# Approximation : B-splines

Let **B(t)** denote the curve as a function of parameter **t**, then B-spline is:

$$B(t) = \sum_{i=1}^{n+1} P_i \, N_i^k(t), \qquad t_{min} \le t \le t_{max}, \qquad 2 \le k \le n+1,$$

where $P_i$ is **n+1** vertices of the polygon,
$N_i^k(t)$ — normalized basis function for the B-spline.
For the **i-th** normalized basis function order of **k** (**k-1** degree) defined by the Cox-de Boor recursion formula:

$$N_i^1(t) = \begin{cases} 1, & x_i \le t \le x_{i+1} \\ 0, & otherwise \end{cases},$$

$$N_i^k(t) = \frac{(t - x_i) N_i^{k-1}(t)}{x_{i+k-1} - x_i} + \frac{(x_{i+k} - t) N_{i+1}^{k-1}(t)}{x_{i+k} - x_{i+1}}.$$

# Approximation : B-splines

The $\mathbf{x_i}$ values are elements of the knot vector, which follow the condition $\mathbf{x_i \leq x_{i+1}}$.
There are three types of knot vectors:

- Uniform (Periodic)
- Open-Uniform
- Non-Uniform

The size of the knot vector is **n+k+1**.

The **uniform** knot vector has constant steps between adjacent elements:

[0 1 2 3 4] or [-0.2 -0.1 0 0.1 0.2].

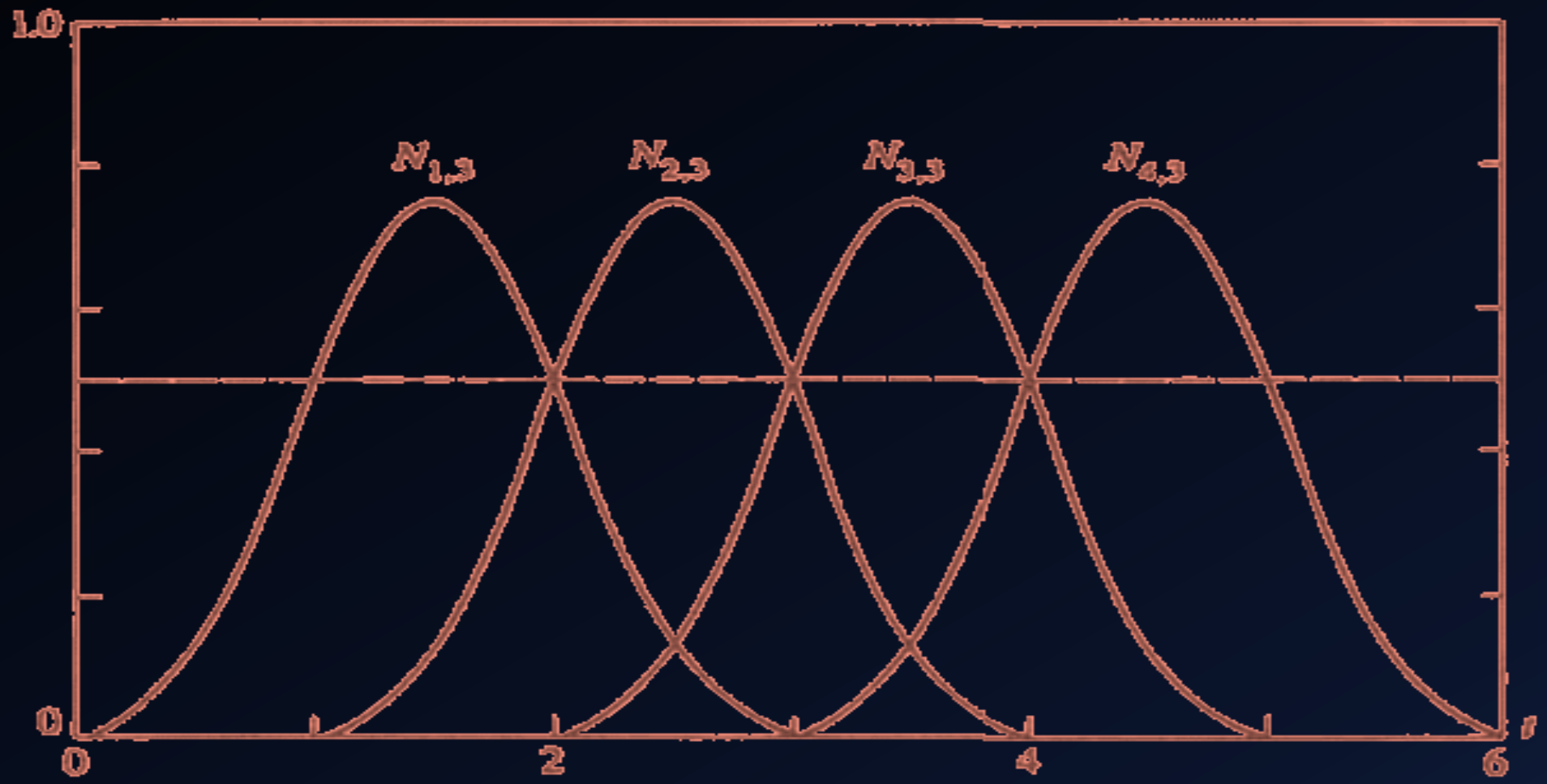
For given order **k** the uniform knot vectors generate periodic uniform basis functions:

$$N_i^k(t) = N_{i-1}^k(t-1) = N_{i+1}^k(t+1).$$

That is, each basis function is a parallel displacement of another function.

# Approximation : B-splines

Below are the basis functions of the periodic **uniform** B-spline with **n**+1=4, **k**=3, and [X]=[0 1 2 3 4 5 6 ].



# Approximation : B-splines

The **open-uniform** knot vector has identical values at the ends is equal to the order **k** of the basis function of the B-spline. Internal values have constant steps between adjacent elements:

$$\begin{aligned}
&\mathrm{k} = 2\ [0 \quad 0 \quad 1 \quad 2 \quad 3 \quad 4 \quad 4],\\
&\mathrm{k} = 3\ [0 \quad 0 \quad 0 \quad 1 \quad 2 \quad 3 \quad 3 \quad 3],\\
&\mathrm{k} = 4\ [0 \quad 0 \quad 0 \quad 0 \quad 1 \quad 2 \quad 2 \quad 2 \quad 2].
\end{aligned}$$

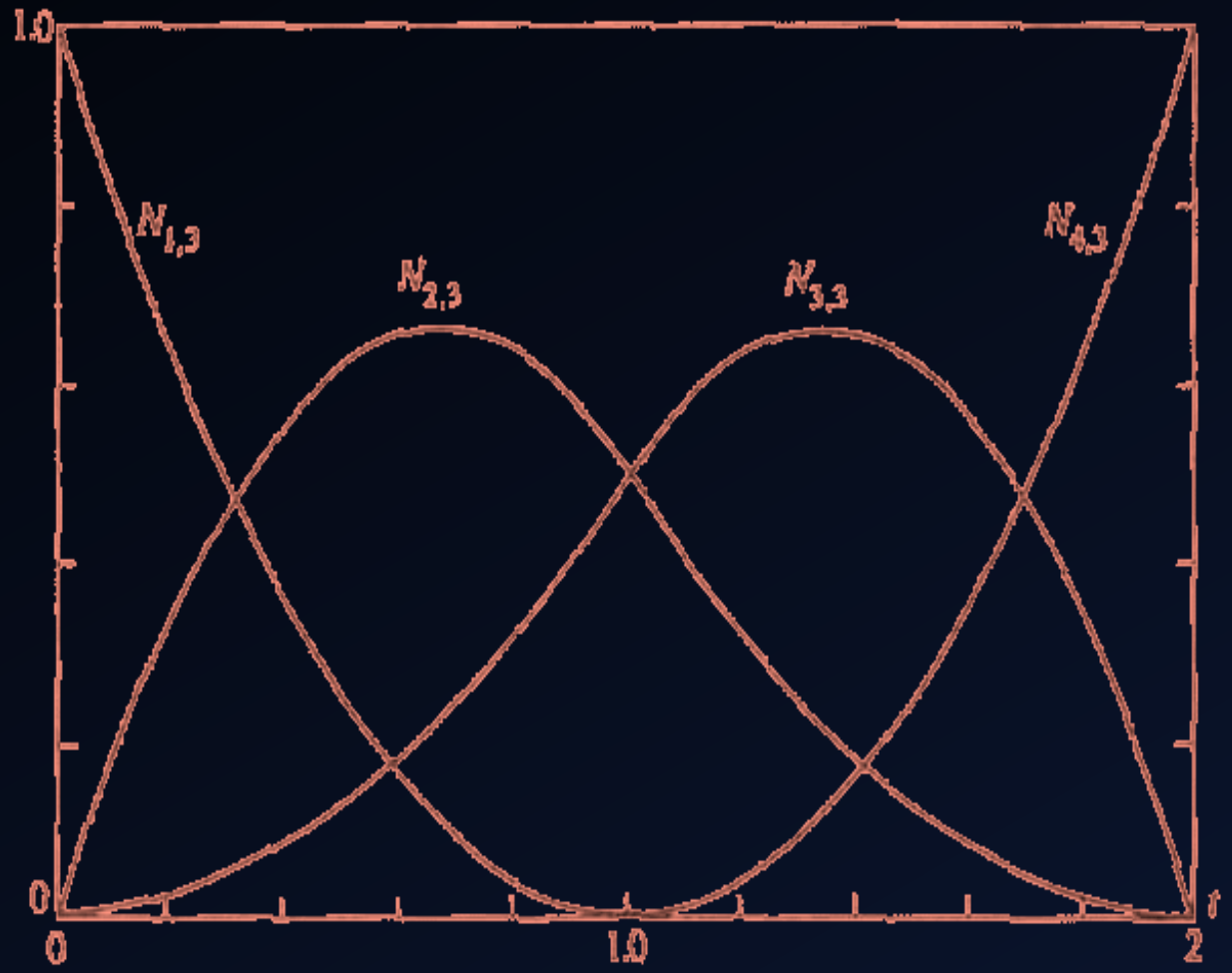
The formal definition of the **open-uniform** knot vector is:

$$\begin{aligned}
&x_i = 0; && 1 \le i \le k;\\
&x_i = i - k; && k + 1 \le i \le n + 1;\\
&x_i = n - k + 2; && n + 2 \le i \le n + k + 1.
\end{aligned}$$

The resulting basis functions behave approximately the same as Bezier curves. In fact, if the number of polygon vertices is equal to the order of the B-spline basis and an open-uniform knot vector is used, the B-spline basis reduces to the Bernstein basis. Hence, the B-spline is a Bezier curve. In this case, the node vector is simply **k** zeros followed by **k** ones.

# Approximation : B-splines

Below are the basis functions of the **open-uniform** B-spline with **n**+1=4, **k**=3, and [X]=[0 0 0 1 2 2 2 ].

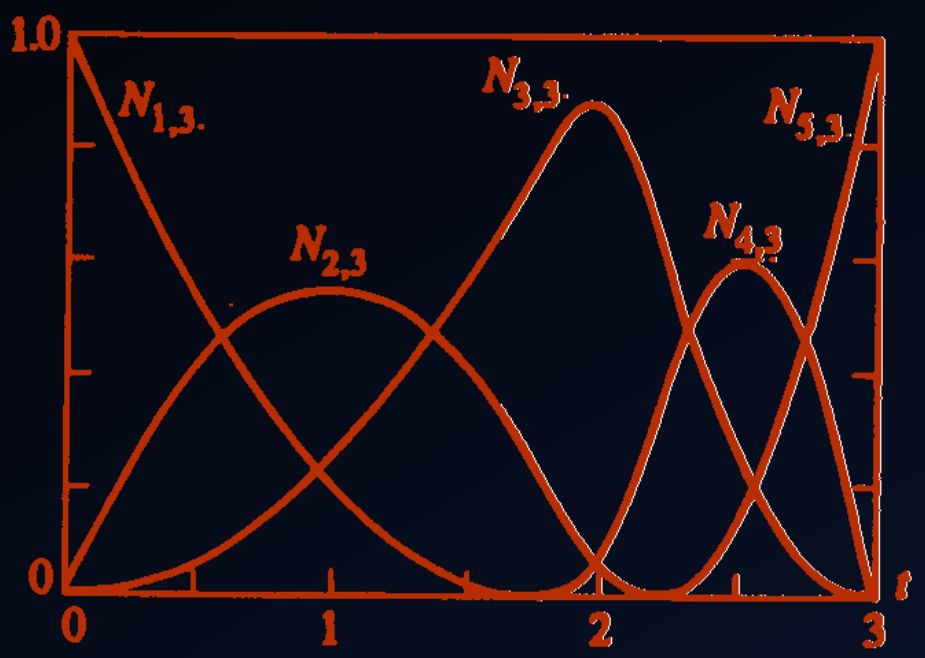


# Approximation : B-splines

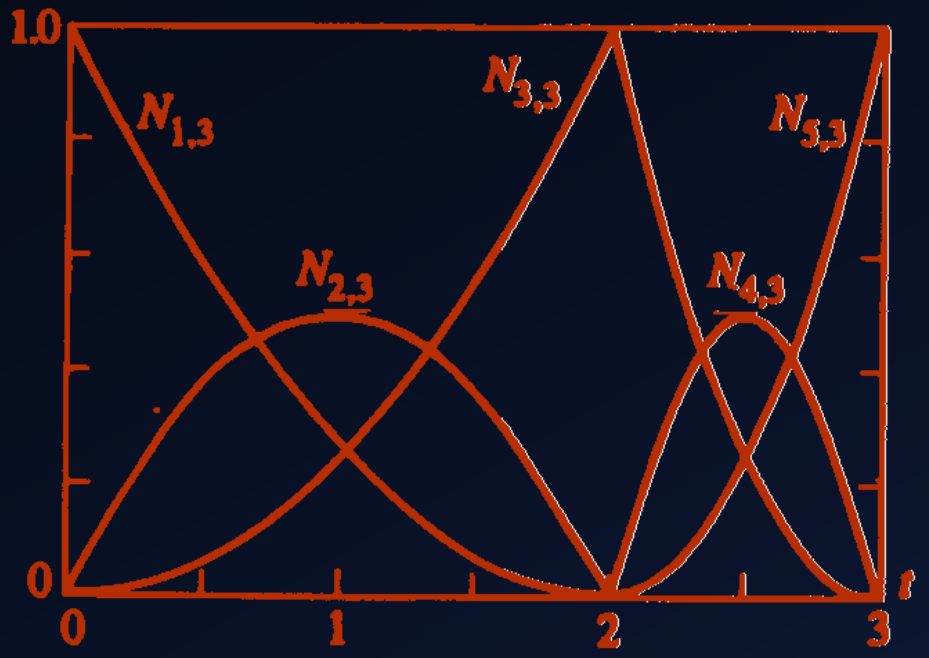
The **non-uniform** knot vectors have different steps between values and/or overlaps. Vectors can be periodic or open, for example:

[0 0 0 1 1 2 2 2] or [0 1 2 2 3 4] or [0 0.28 0.5 0.72 1]

Some examples of the basis functions of the **non-uniform** B-spline with **n**+1=5 and **k**=3 are below.




[X]=[0 0 0 1.8 2.2 3 3 3]

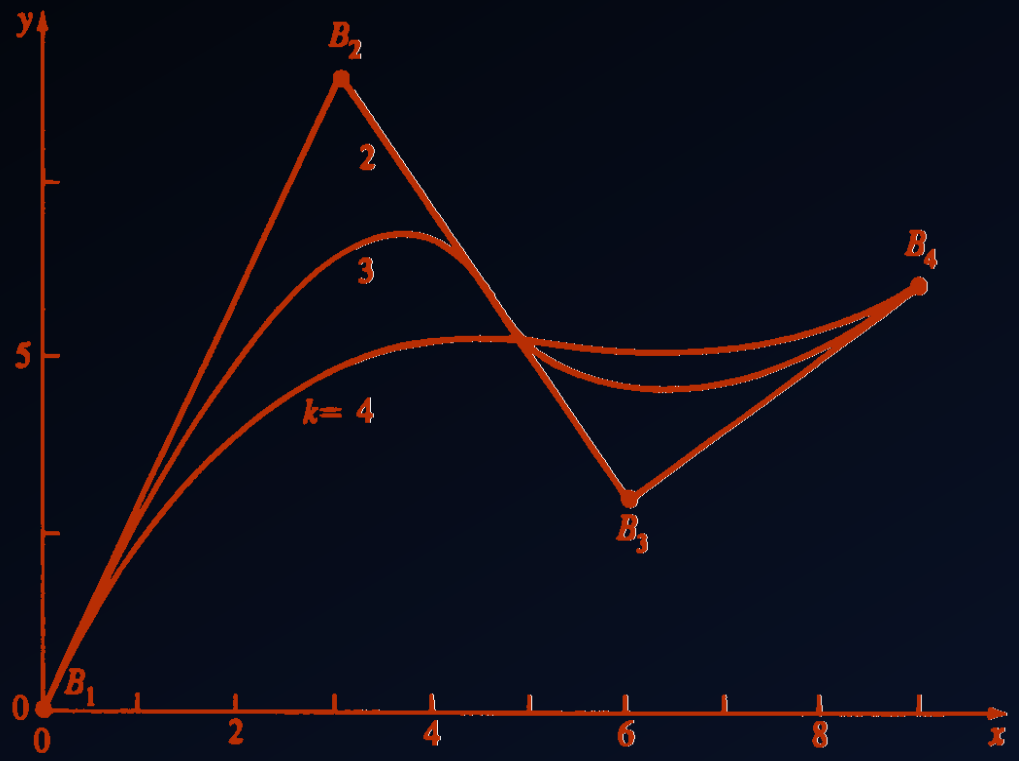



[X]=[0 0 0 2 2 3 3 3]

# Approximation : B-splines

The flexibility of the B-spline allows you to influence the shape of the curve in various ways:

- changing the type of knot vector: periodic uniform, open uniform, and non-uniform;
- changing the order of **k** basis functions;
- changing the number and location of vertices of the defining polygon;
- using repeated vertices;
- using repeated values in knot vectors.




The figure shows three open B-splines of different orders given by one set of four vertices. A fourth-order curve is a Bezier. The third-order curve consists of two parabolic segments that connect in the center of the second segment with $C^1$ continuity. The curve of the second order coincides with the defining polygon. It consists of three linear segments connecting at the second and third vertices with continuity $C^0$.

# Thank you!