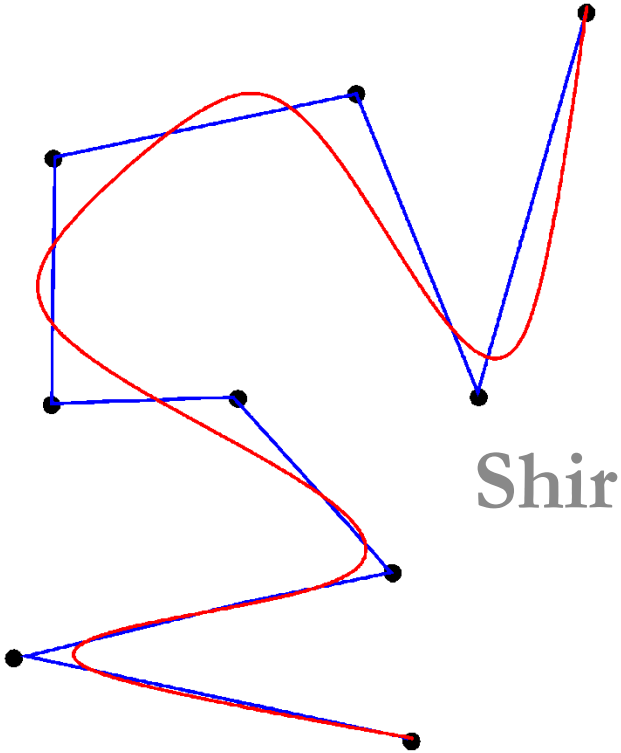
Curves and Surfaces



Shireen Elhabian and Aly A. Farag

University of Louisville

February 2010

“A smile is a curve that sets everything straight…”

Phyllis Diller
(American comedienne and actress, born 1917)

Outline

- Introduction
- Affine transformations
- Curves
  - What is a curve?
  - Affine invariance
  - Convex Hull
  - Lagrange Interpolation
  - Bezier Curves
- Surfaces

Introduction

We will discuss some of the existing curves associated with their interpolation techniques. We will start with few of the important properties which we desire for curves e.g. continuity and affine invariance. We will introduce transformations for manipulating curves and then proceed to discuss some of the most popular curves used in computer graphics. But let's first introduce the notion of parametric curves which will be later extended to define parametric surfaces.

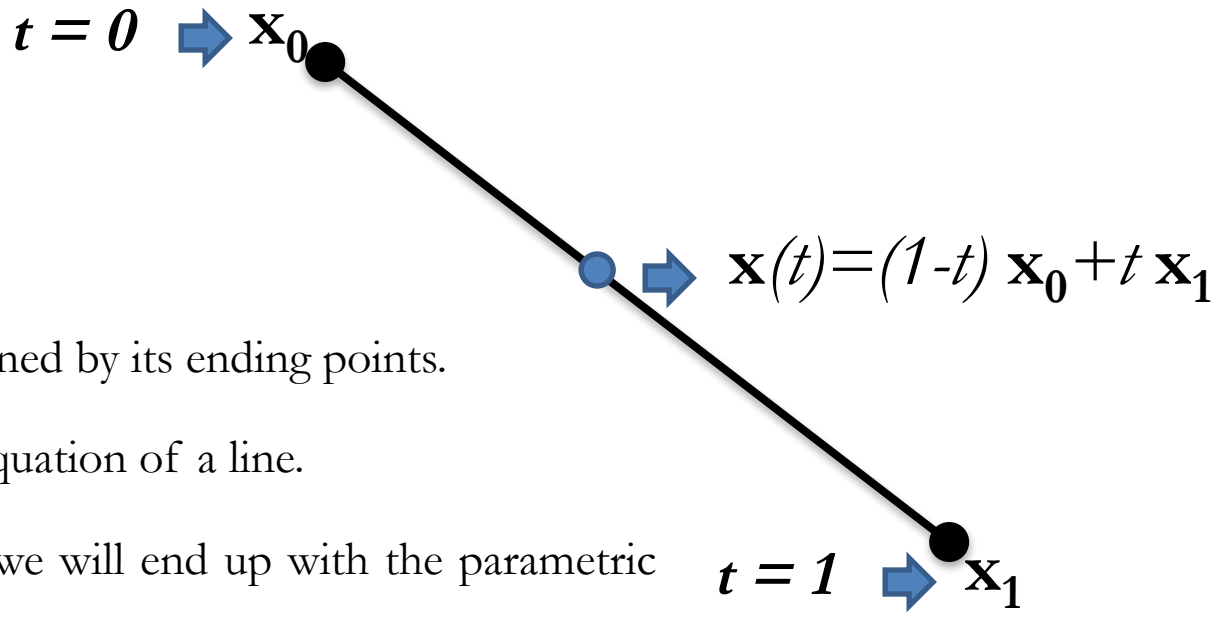
Points in space

- A **point**, which can be defined as follows:

Definition 1: *Real Euclidean d-space is given by* $\mathbb{R}^d = \{\mathbf{x} = (x_1, x_2, \dots, x_d) | \, x_i \in \mathbb{R}\}$ *where* $\mathbf{x}$ *denotes a point with d-coordinates.*

Example: $\mathbb{R}^1 = \mathbb{R}$ is the real line while $\mathbb{R}^2 = \{\mathbf{x} = (x, y)\}$ is the standard Euclidean plane and $\mathbb{R}^3 = \{\mathbf{x} = (x, y, z)\}$ is the three dimensional space.

Line Segment

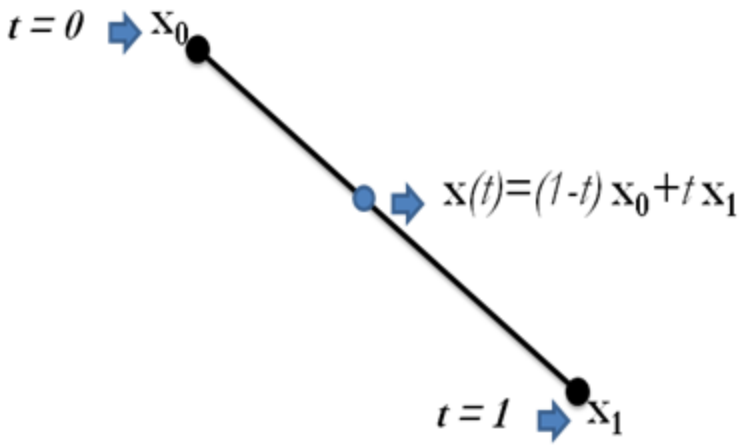


- A line segment can be defined by its ending points.
- Many ways to define the equation of a line.
- Think about it as a road, we will end up with the parametric form of the line.

Think of it as if you started at $\mathbf{x_0}$ then walk t of the distance between $\mathbf{x_0}$ and $\mathbf{x_1}$, hence your current position will be defined as follows;

$$\mathbf{x}(t) = \mathbf{x_0} + t(\mathbf{x_1} - \mathbf{x_0}) = \mathbf{x_0} + t\mathbf{x_1} - t\mathbf{x_0} = (1-t)\mathbf{x_0} + t\mathbf{x_1} \qquad (1)$$

Hence, what we have done now is representing any point on the line connecting $\mathbf{x_0}$ and $\mathbf{x_1}$ by a *weighted average* of the two ending points, this average is parameterized by one parameter t.



We can re-write (1) as:

$$\mathbf{x}(t) = f_0(t)\mathbf{x_0} + f_1(t)\mathbf{x_1} \qquad (2)$$

Where $f_0(t) = 1 - t$ and $f_1(t) = t$.

Since we are representing a line, the degree of $f_k(t)$ for $k = 0,1$ is one (i.e. linear functions), hence we only need two points to represent a line segment.

Eq(2) can be thought of as representing a line segment with *control points* $\mathbf{x_0}$ and $\mathbf{x_1}$ being interpolated with *basis functions* $f_k(t)$. This equation is usually referred to as the *parametric equation of a line*, let's now generalize this notion to curves and later to surfaces, where curves are represented by non-linear basis functions and hence we need more control points.

Affine Transformations

What is a transform?

Definition 2: *A transform/warp on* $\mathbb{R}^d$ *is any mapping* $W: \mathbb{R}^d \to \mathbb{R}^d$. *That is, each point* $\mathbf{x} \in \mathbb{R}^d$ *is mapped to exactly one point* $W(\mathbf{x})$ *also in* $\mathbb{R}^d$.

Definition 3: *Let* $W: \mathbb{R}^d \to \mathbb{R}^d$ *be a transform.* W *is said to be a* **linear** *transform/warp if and only if:*

(a) For all $\alpha \in \mathbb{R}$ *and all* $\mathbf{x} \in \mathbb{R}^d$ *we have* $W(\alpha\mathbf{x}) = \alpha W(\mathbf{x})$.
(b) For all $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d$ *we have* $W(\mathbf{x} + \mathbf{y}) = W(\mathbf{x}) + W(\mathbf{y})$.

This implies that $W(\mathbf{0}) = \mathbf{0}$ since $W(\mathbf{0}.\mathbf{x}) = \mathbf{0}.W(\mathbf{x}) = \mathbf{0}.$ An example of a linear transform/warp is the identity transform given by $W(\mathbf{x}) = \mathbf{x}$.

Recall …

Definition : *If* $x, y, z, w \in \mathbb{R}$ and $w \neq 0$, *then* $(x, y, z, w)^{\mathrm{T}}$ *is a homogeneous coordinate representation of the point* $\left(\frac{x}{w}, \frac{y}{w}, \frac{z}{w}\right)^{\mathrm{T}}$.

Definition 4: *Let* $W: \mathbb{R}^d \to \mathbb{R}^d$ *be a transform.* W *is said to be a* **translation** *if there exists* $\mathbf{t} \in \mathbb{R}^d$ *so that for all* $\mathbf{x} \in \mathbb{R}^d$ *we have* $W(\mathbf{x}) = \mathbf{x} + \mathbf{t}$. *A translation moves all vectors or points by a fixed distance in a fixed direction.*

Definition 5: *An* **affine** *transform is a transform that can be written as* $W(\mathbf{x}) = T(L(\mathbf{x}))$ *where* $L(.)$ *is a linear transform and* $T(.)$ *is a translation. This can also be written as* $W(\mathbf{x}) = L(\mathbf{x}) + \mathbf{t}$ *or* $W = T_{\mathbf{t}}L$.

Any linear transform in $\mathbb{R}^3$ can be represented by a 3x3 matrix of the following form;

$$L = \begin{pmatrix} \ell_{11} & \ell_{12} & \ell_{13} \\ \ell_{21} & \ell_{22} & \ell_{23} \\ \ell_{31} & \ell_{32} & \ell_{33} \end{pmatrix}$$

Scaling

In order to change the size of an object in $\mathbb{R}^3$, if we assume that the object is centered at the origin, then scaling is given by;

$$S = \begin{pmatrix} s & 0 & 0 \\ 0 & s & 0 \\ 0 & 0 & s \end{pmatrix}$$

To scale a point we apply the matrix S to the point $\mathbf{x} \in \mathbb{R}^3$ where $\mathbf{x} = \{x, y, z\}$ to get

$$\mathbf{x}' = S\mathbf{x} = \begin{pmatrix} s & 0 & 0 \\ 0 & s & 0 \\ 0 & 0 & s \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} sx \\ sy \\ sz \end{pmatrix}$$

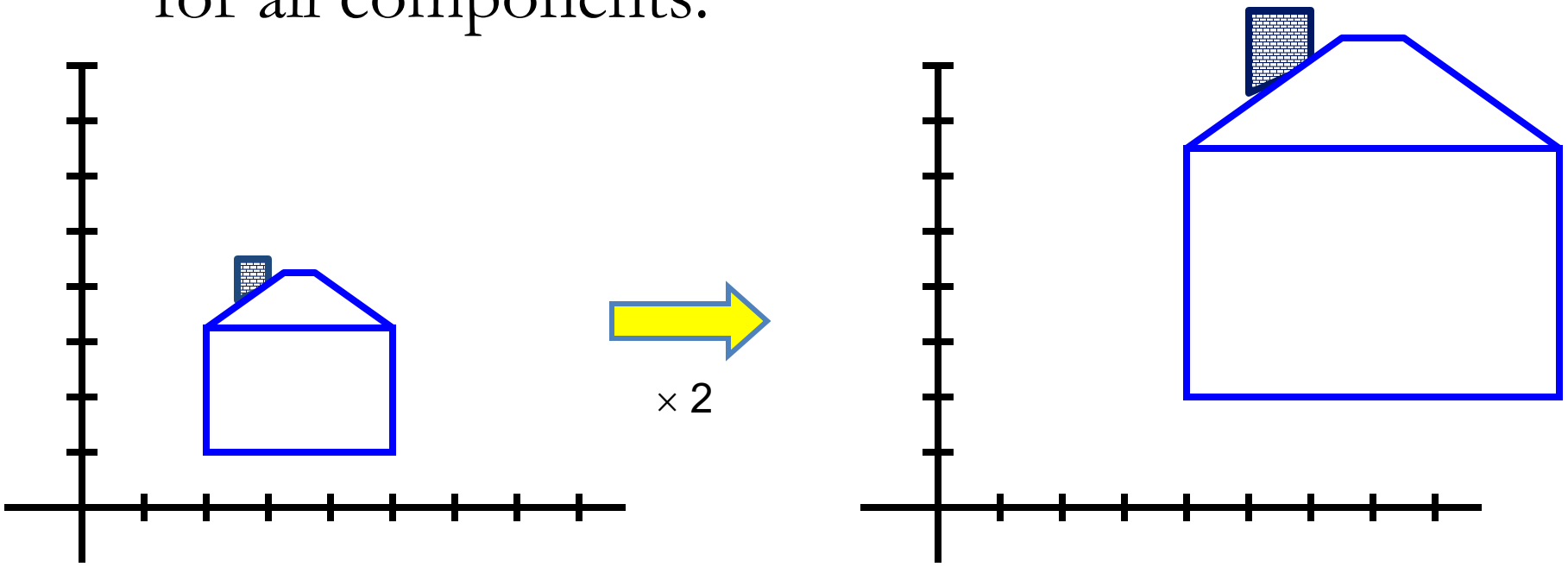
This is called *uniform scaling* since we change the size of the object in all directions with the same amount, however, it is not necessary to scale evenly in all directions, in this case we can define the *non-uniform scaling* matrix as follows;

$$S = \begin{pmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & s_z \end{pmatrix}$$

Where x, y, z connotes the three dimensions in a 3D Euclidean space.

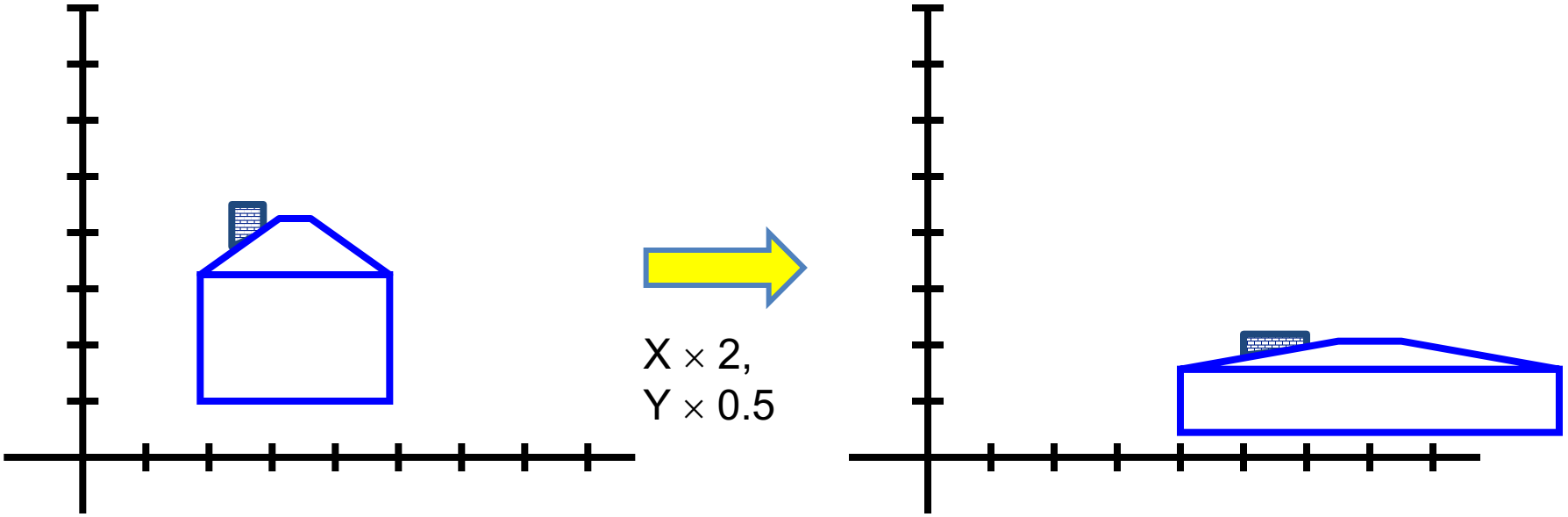
Scaling in 2D

- Scaling a coordinate means multiplying each of its components by a scalar
- Uniform scaling means this scalar is the same for all components:



Scaling in 2D

- Non-uniform scaling: different scalars per component:



- How can we represent this in matrix form?

Scaling in 2D

- Scaling operation:

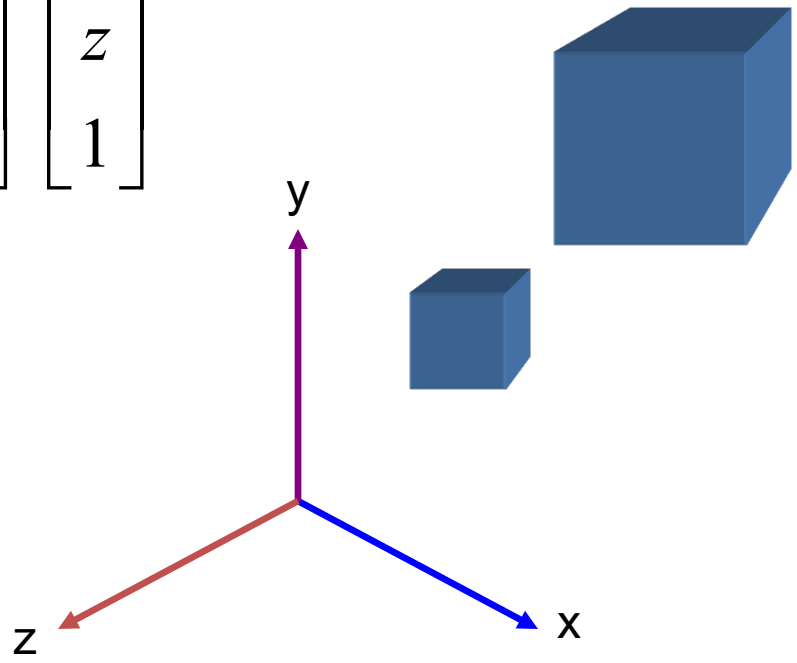
$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} ax \\ by \end{bmatrix}$$

- Or, in matrix form:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \underbrace{\begin{bmatrix} a & 0 \\ 0 & b \end{bmatrix}}_{\text{scaling matrix}} \begin{bmatrix} x \\ y \end{bmatrix}$$

Scaling in 3D

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$



Translation

Translations in $\mathbb{R}^3$ cannot be written as 3x3 matrices, however to allows a unified representation for affine transforms, we can use the homogeneous coordinate system which allows us to represent an affine transform as a matrix. A translation by $\mathbf{t} = (t_x, t_y, t_z)^{\mathrm{T}}$ is given by a 4x4 matrix defined as:

$$T_{\mathbf{t}} = \begin{pmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Linear transforms can also be represented as transforms on homogeneous coordinate systems as follows;

$$L = \begin{pmatrix} \ell_{11} & \ell_{12} & \ell_{13} & 0 \\ \ell_{21} & \ell_{22} & \ell_{23} & 0 \\ \ell_{31} & \ell_{32} & \ell_{33} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

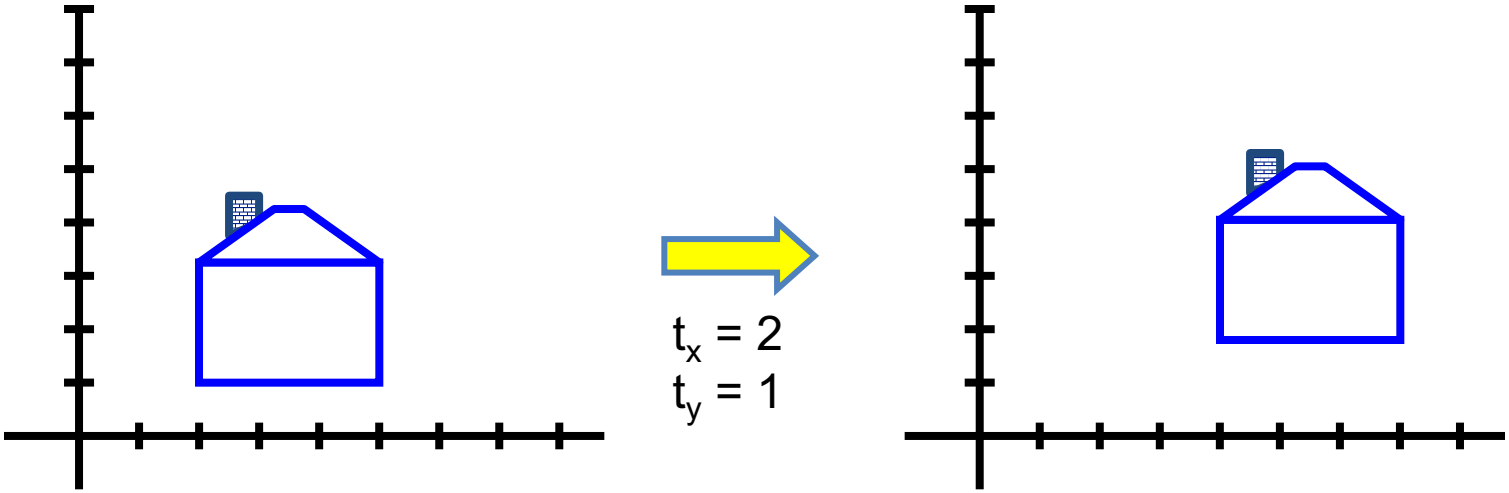
Affine transforms can be represented as combination of a linear transform and a translation, thus can be represented by the matrix product $W = T_{\mathbf{t}} L$ in the homogeneous coordinate system.

In the same manner, translation matrices can be defined in the 2D space as $S = \begin{pmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{pmatrix}$

Translation in 2D

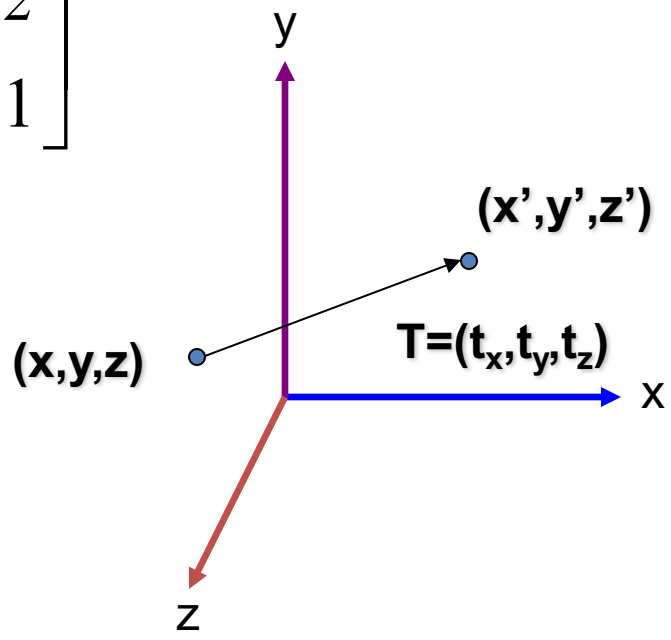
$$x' = x + t_x$$

$$y' = y + t_y$$



Translation in 3D

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$



Rotation

In $\mathbb{R}^3$, the rotation matrix about the z-axis can be written as;

$$R_{z,\alpha} = \begin{pmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

In the same manner, the rotation matrices about x and y axes are defined as follows;

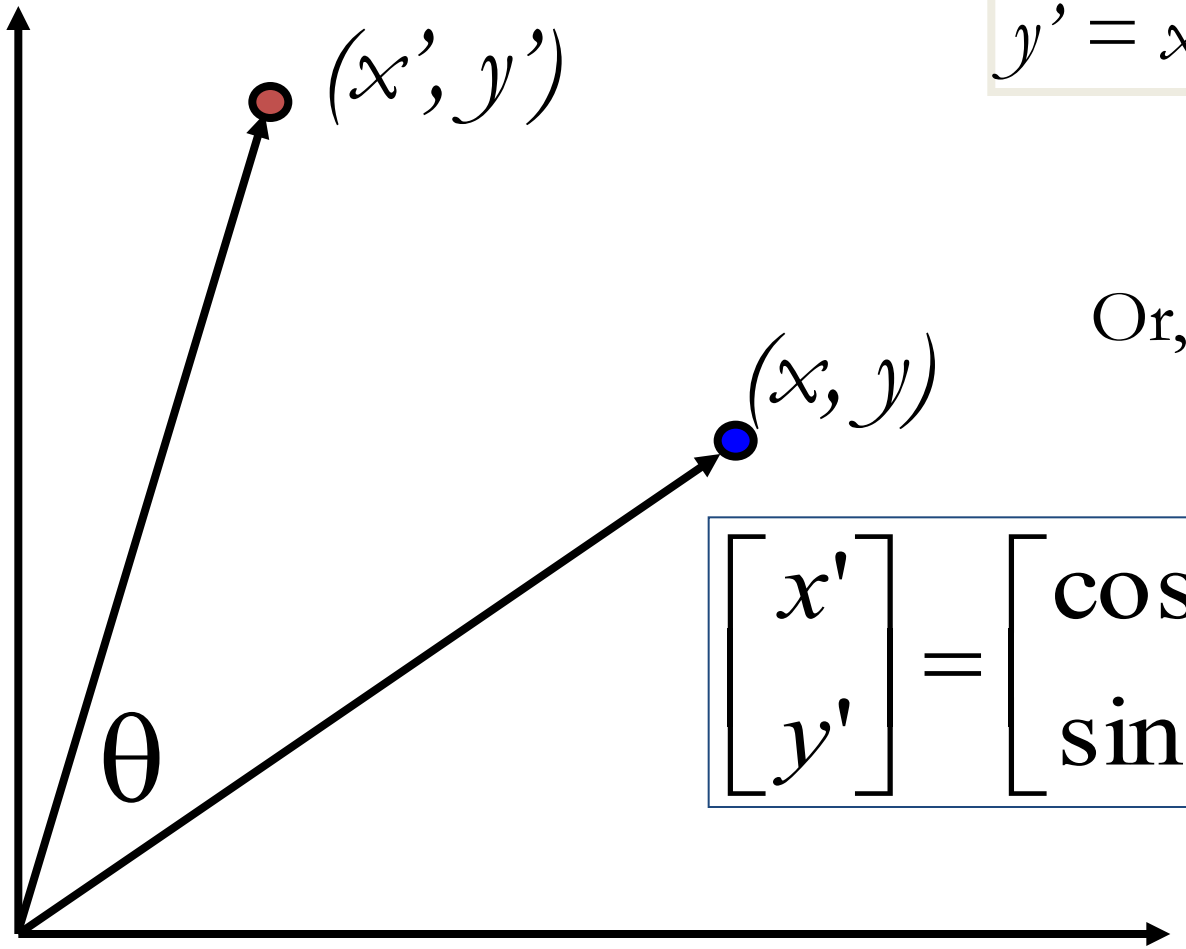
$$R_{x,\alpha} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{pmatrix}$$

And

$$R_{y,\alpha} = \begin{pmatrix} \cos\alpha & 0 & -\sin\alpha \\ 0 & 1 & 0 \\ \sin\alpha & 0 & \cos\alpha \end{pmatrix}$$

Rotation in 2D

$$x' = x\cos(\theta) - y\sin(\theta)$$
$$y' = x\sin(\theta) + y\cos(\theta)$$



Or, in matrix form

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Rotation in 3D

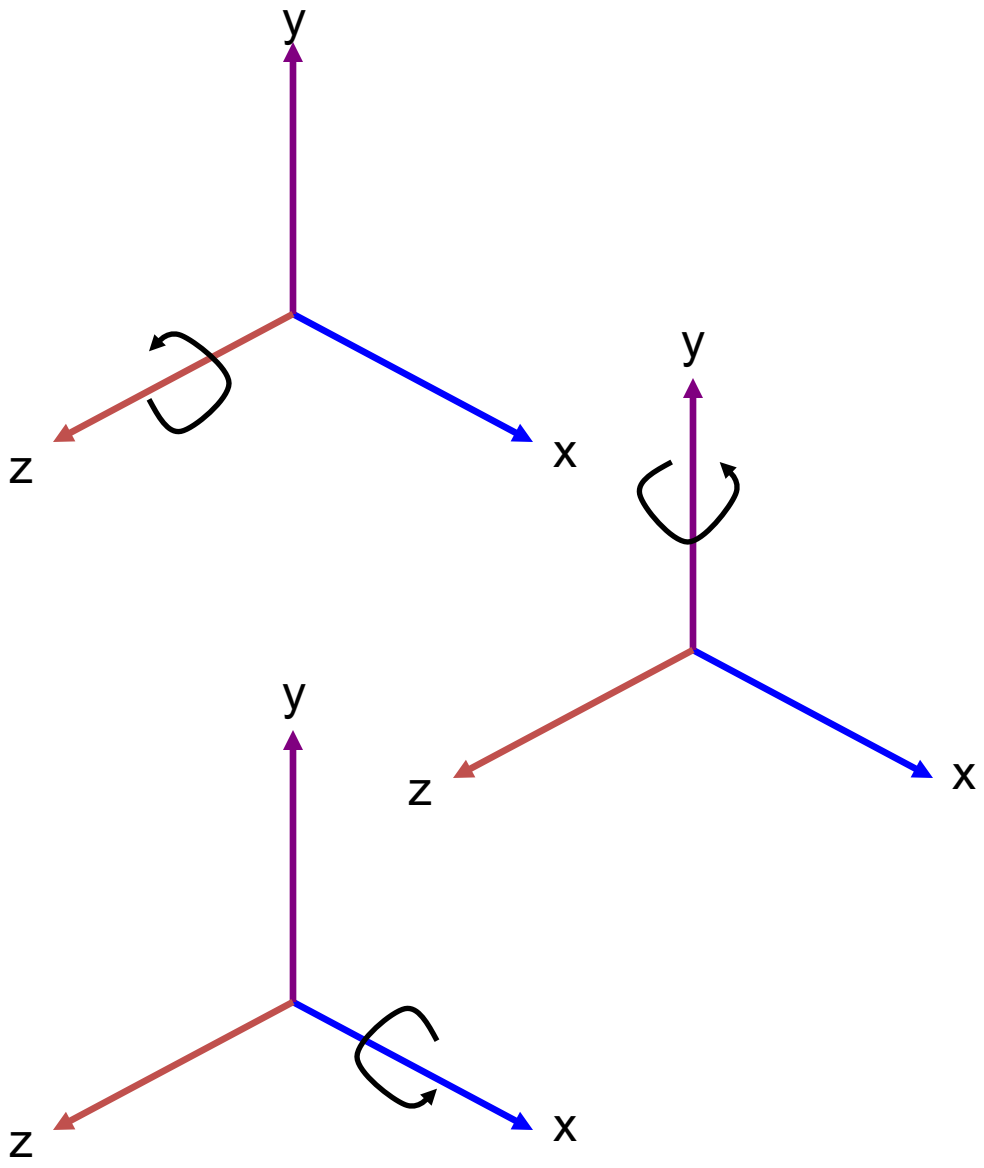
- How should we specify rotations?
- In 2D, it was always counterclockwise in the xy-plane.
- In 3D, we have more choices
  - xz-plane, yz-plane, an arbitrary plane.
- We could specify these in terms of the vector perpendicular to the plane of rotation.
  - z axis, y-axis, x-axis, arbitrary axis

Rotation in 3D – Euler Angles

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\gamma & -\sin\gamma & 0 & 0 \\ \sin\gamma & \cos\gamma & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\beta & 0 & \sin\beta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\beta & 0 & \cos\beta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha & 0 \\ 0 & \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$



Curves

What are they?

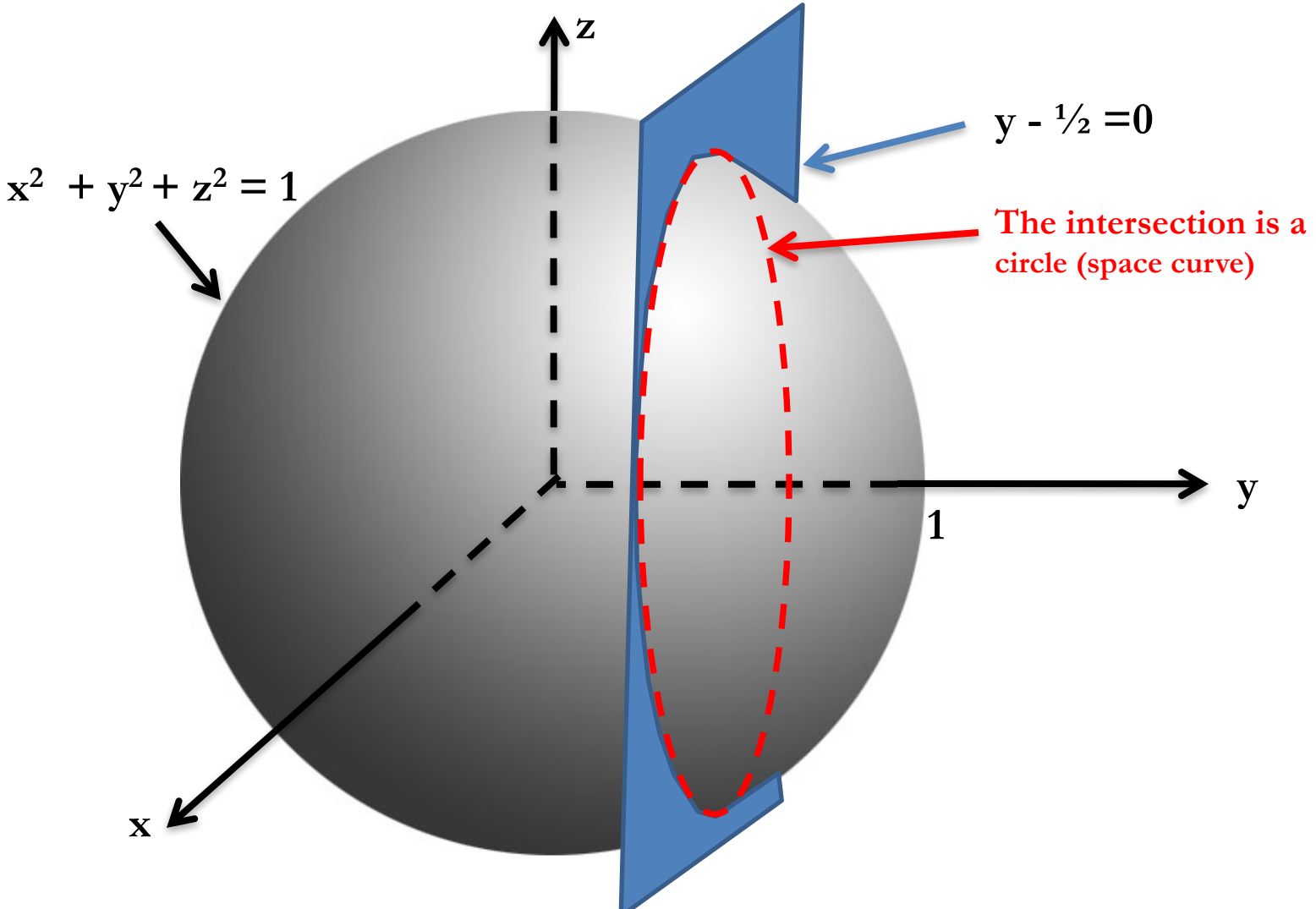
Definition 6: *Curves in the standard 2-dimensional Euclidean space* $\mathbb{R}^2$ *are called* **plane curves** *which can be described either in an explicit form,* $y = f(x)$*, i.e. as a function graph, or implicitly as a set of points which specify an equation* $f(x, y) = 0$*, this equation can be represented in a parametric form as* $\mathbf{x}(t) = (x(t), y(t))^{\mathrm{T}}$*, however we need to place restrictions on* f *such that the solution of* $f(x, y) = 0$ *do not fill the entire plane.*

Definition 7: Algebraic curves *are curves defined by* f *such that* f *is a polynomial function in two variables. Equations of the first degree define straight lines, while equations of the second degree define ellipses, parabolas or hyperbolas.*

Definition 8: *Curves in the standard 3-dimensional Euclidean space* $\mathbb{R}^3$ *are called* **space curves** *which can be defined as intersections of two surfaces defined implicitly by* $f(x, y, z) = 0$ *and* $g(x, y, z) = 0$.

Example – Space Curves

Example: the intersection of two surfaces $x^2 + y^2 + z^2 = 1$ (i.e. sphere) and $y - \frac{1}{2} = 0$ (i.e. plane) is a circle.



Parametric Curves

Definition 9: Parametric curves *are curves defined in the standard 3-dimensional Euclidean space* $\mathbb{R}^3$ *in terms of some parameter, say* t. *The curve can then be written as:* $\mathbf{x}(t) = (x(t), y(t), z(t))^{\mathrm{T}}$ *where* $x(t), y(t)$ *and* $z(t)$ *are continuous functions defined on some interval* $t \in [a, b]$ *where each value of* t *defines a point on the curve. The curve is defined to be the set of all such points.*

Note that parametric curves can also be defined in the standard 2-dimensional Euclidean space $\mathbb{R}^2$, hence it can be written as $\mathbf{x}(t) = (x(t), y(t))^{\mathrm{T}}$.

In most cases, it is assumed that $\mathbf{x}(t)$ can be differentiated at least twice, i.e. first and second derivates are defined for $t \in [a, b]$.

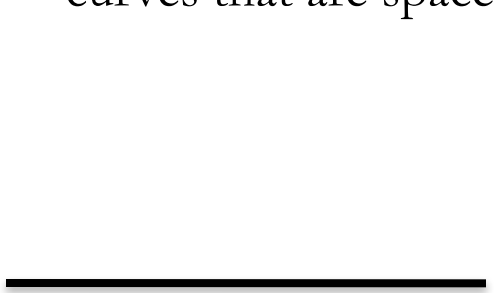
Definition 10: *Let* $\{\mathbf{x_1}, \mathbf{x_2}, \dots, \mathbf{x_n}\}$ *be a set of points in the d-dimensional Euclidean space, a* **curve** *can be defined in terms of these points as:*

$$\mathcal{C}: \quad \mathbf{x}(t) = f_1(t)\mathbf{x_1} + f_2(t)\mathbf{x_2} + \cdots + f_n(t)\mathbf{x_n} = \sum_{k=1}^{n} f_k(t)\mathbf{x_k}$$

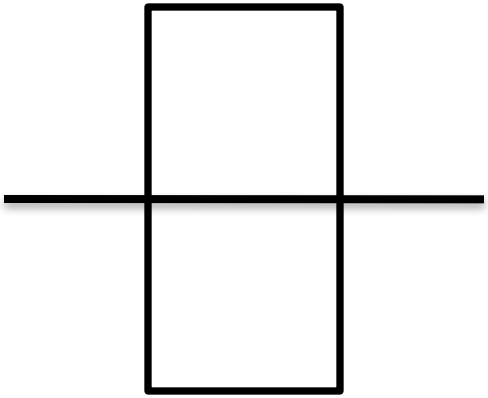
Where $f_k(t)$ *are continuous functions defined on the interval* $t \in [0,1]$. *The points* $\mathbf{x_1}, \mathbf{x_2}, \dots, \mathbf{x_n}$ *are called* **control point** *and* $f_k(t)$ *are called the* **basis functions** *of the curve* $\mathcal{C}$.

Parametric Curves

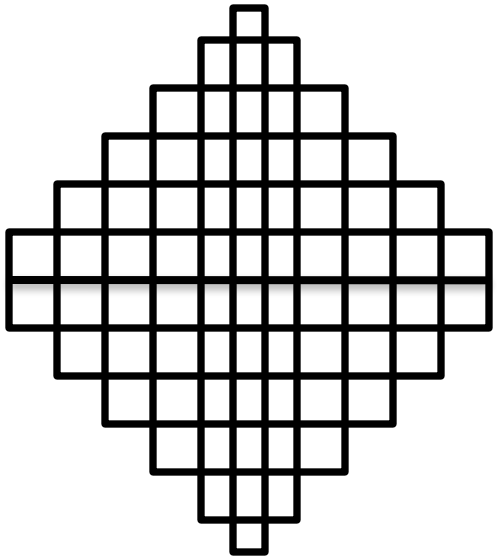
- Some restrictions should be placed on the continuous functions defining the curve in order not to fill the space, however there are still some parametric curves that are space filling, e.g. the **Peano curve**.



Peano curve - 0



Peano curve - 1



Peano curve - 2

In 1880 the Italian logician Giuseppe Peano (1858-1932) constructed the Peano curve, a base motif fractal which uses a line segment as base. The motif is dividing the line segment in three parts, and making a square up and down the middle part. This leads to a filled square, so the curve is a space-filling curve.

Affine Invariance

Given a curve defined by $\sum_{k=1}^{n} f_k(t)\mathbf{x_k}$ where $t \in [0,1]$, we would like to investigate what happens when an affine transform is applied to the curve.

Definition 11: *A curve is said to be* **affine invariant** *if the affine transform/warp* $W(.)$ *applied to the points generated by the curve, i.e.* $\mathbf{x}(t) = \sum_{k=1}^{n} f_k(t)\mathbf{x_k}$ *, produces precisely the same curve as transforming the control points of the curve, i.e* $\mathbf{x_k}$*, and then calculating the curve, that is:*

$$W\left(\sum_{k=1}^{n} f_k(t)\mathbf{x_k}\right) = \sum_{k=1}^{n} f_k(t)W(\mathbf{x_k})$$

This will be satisfied if the basis functions $f_k(t)$ *of the curve satisfy the property* $\sum_{k=1}^{n} f_k(t) = 1$ *for* $t \in [0,1]$.

Theorem 1: *Let* $\mathcal{C}$ *be a curve defined by* $\sum_{k=1}^{n} f_k(t)\mathbf{x_k}$ *where* $t \in [0,1]$. *If the basis functions* $f_k(t)$ *are a partition of unity that is* $\sum_{k=1}^{n} f_k(t) = 1$ *for* $t \in [0,1]$, *then* $\mathcal{C}$ *is affine invariant, i.e. for any affine transform* $W = TL$ *where* L *is a linear transform and* T *is a translation by* $\mathbf{u}$*, we have:*

$$W\left(\sum_{k=1}^{n} f_k(t)\mathbf{x_k}\right) = \sum_{k=1}^{n} f_k(t)W(\mathbf{x_k})$$

Proof left as homework ☹

Convex Hull

- Another useful property is that if a curve lies within its convex hull which can be defined as follows:

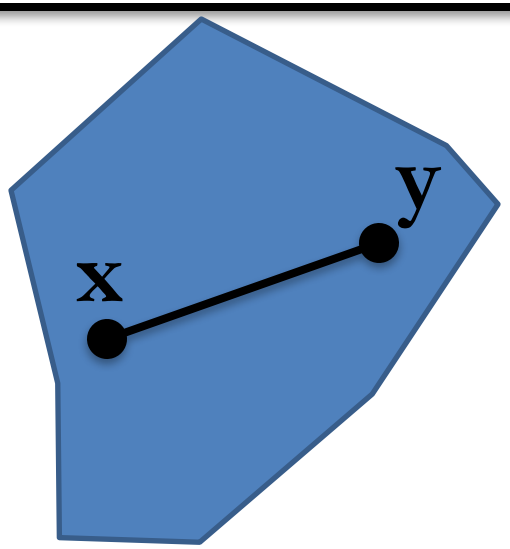
Definition 12: *Let* $\{\mathbf{x_1}, \mathbf{x_2}, \dots, \mathbf{x_n}\}$ *be a set of points in the d-dimensional Euclidean space* $\mathbb{R}^d$ *and* $a_1, a_2, \dots, a_n$ *be real numbers, then:*

(1) $\sum_{i=1}^{n} a_i \mathbf{x}_i = a_1\mathbf{x}_1 + a_2\mathbf{x}_2 + \dots + a_n\mathbf{x}_n$ *is called a* **linear combination** *of* $\mathbf{x_1}, \mathbf{x_2}, \dots, \mathbf{x_n}$.

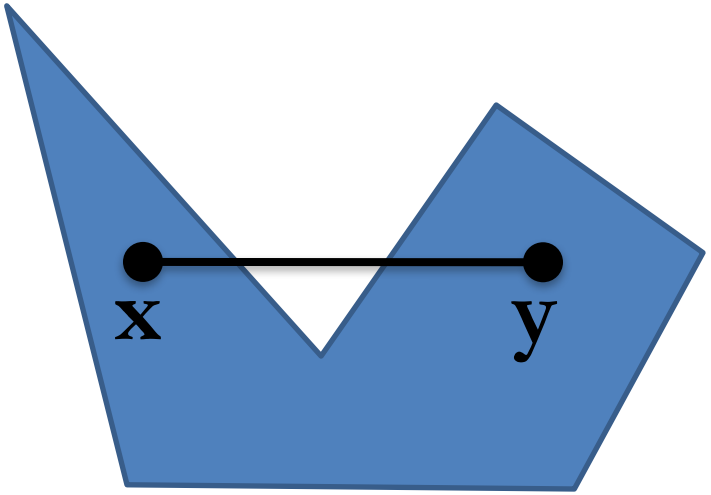
(2) If $\sum_{i=1}^{n} a_i = 1$, *then* $\sum_{i=1}^{n} a_i \mathbf{x}_i = a_1\mathbf{x}_1 + a_2\mathbf{x}_2 + \dots + a_n\mathbf{x}_n$ *is called an* **affine combination** *of* $\mathbf{x_1}, \mathbf{x_2}, \dots, \mathbf{x_n}$.

(3) If $\sum_{i=1}^{n} a_i = 1$ *and* $a_i \geq 0$, *then* $\sum_{i=1}^{n} a_i \mathbf{x}_i = a_1\mathbf{x}_1 + a_2\mathbf{x}_2 + \dots + a_n\mathbf{x}_n$ *is called a* **weighted average** *of* $\mathbf{x_1}, \mathbf{x_2}, \dots, \mathbf{x_n}$.

Definition 13: *Let* A *be a set of points in* $\mathbb{R}^d$. *The set* A *is* **convex** *if and only if for any two points* $\mathbf{x}, \mathbf{y} \in A$, *the line segment joining* $\mathbf{x}$ *and* $\mathbf{y}$ *is entirely in* A.



Convex set



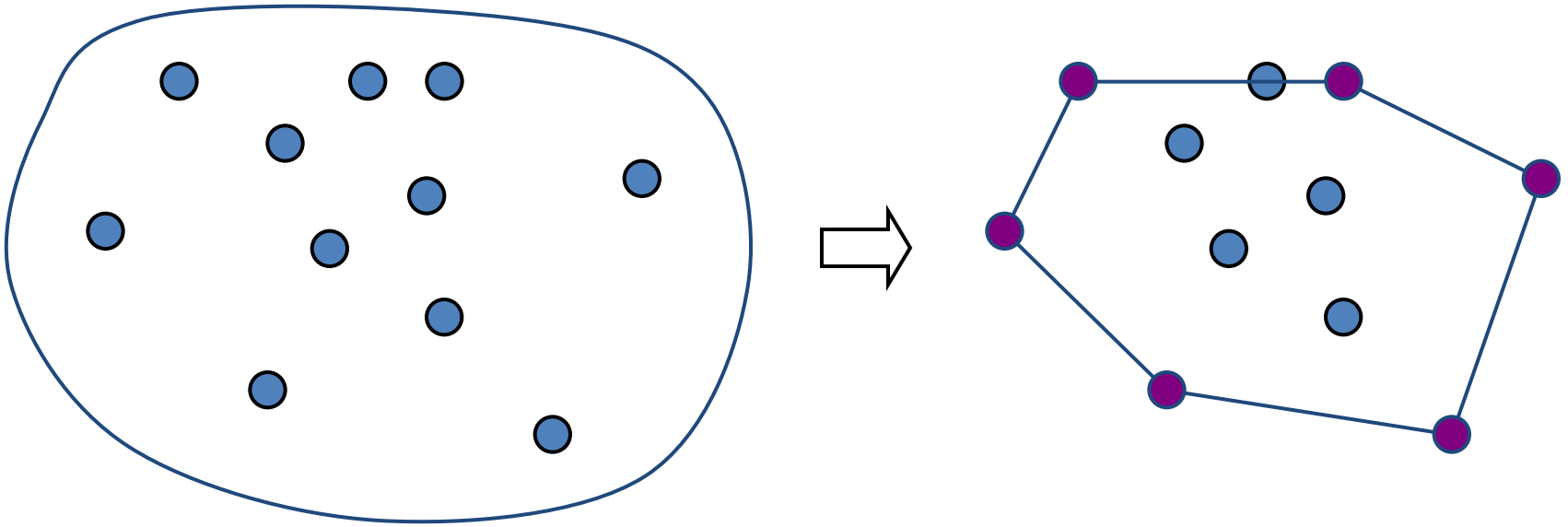
Non-convex set

Definition 14: *The* **convex hull** *of* A *is the smallest convex set containing* A*, hence the convex hull of the set* A *is the set of points that are weighted averages of points in that set, thus;*

$$chull(A) = \left\{ \mathbf{x} : \mathbf{x} = \sum_{i=1}^{n} a_i \mathbf{x}_i , \mathbf{x}_i \in A, \sum_{i=1}^{n} a_i = 1 , a_i \geq 0 \right\}$$

Where $A = \{\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_n}\}$.

Given a set of pins on a pin board and a rubber band around them. How does the rubber band look when it snaps tight? We represent the convex hull as the sequence of points on the convex hull polygon, in counter-clockwise order.



Interpolation

- In numerical analysis, *interpolation* is a method of constructing new points within the range of a discrete set of known points.
- From the engineering point of view, we often has a number of points, obtained by sampling or experimentation, and we wish to construct a function which closely fits these points, this is called curve fitting where interpolation is a special case, in which the function must go exactly through the known points.
- Another way to define interpolation is from its linguistic meaning, *inter* means between and *pole* means points or nodes, hence any method of calculating a new point between two or more known points is called interpolation.

n-degree Interpolation

When we perform *n*-degree interpolation, we need *n+1* known points, hence we will be given a set of control points, $\mathbf{x_0}, \mathbf{x_1}, \mathbf{x_2}, \dots, \mathbf{x_n}$ (indexing now starts from 0 rather than 1 to have *n+1* points).

According to Definition 10, a curve can be defined as or generated by $\sum_{k=1}^{n} f_k(t)\mathbf{x_k}$ where $f_k(t)$ are the basis functions of the curve, hence different basis functions will lead to different curves.

Curve generation can be thought of as an interpolation process where the control points are interpolated to generate points whose locus is the curve.

The simplest form of interpolation is Lagrange interpolation which is defined as follows:

Lagrange Interpolation

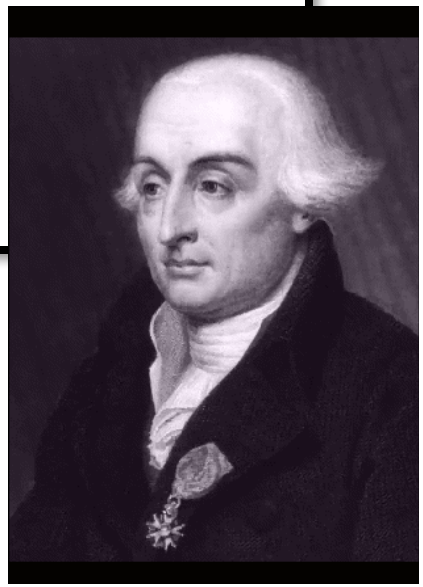
Definition 15: *Let* $\mathbf{x_0}, \mathbf{x_1}, \mathbf{x_2}, \dots, \mathbf{x_n}$ *be a set of control points,* **Lagrange interpolation** *of these points is given by :*

$$\mathbf{x}(t) = \sum_{k=0}^{n} L_k^n(t)\mathbf{x_k}$$

With

$$L_k^n(t) = \prod_{\substack{j=0 \\ j \neq k}}^{n} \frac{t - t_j}{t_k - t_j}$$

Where t_j *are the parameter values at which the point* $\mathbf{x_k}$ *should be interpolated and* $t \in [t_0, t_n]$



Joseph-Louis Lagrange, Italian-born mathematician and astronomer, who lived most of his life in Prussia and France, making significant contributions to all fields of analysis, to number theory, and to classical and celestial mechanics.

Example: Uniform Lagrange Interpolation

The coordinates for interpolation are given by the following controlling points, parameter values at which the control points should be interpolated are chosen to be $t_j = j$, this is called uniform Lagrange interpolation:

$$\mathbf{x_0} = (1,1)^{\mathrm{T}} \qquad t_0 = 0$$

$$\mathbf{x_1} = (2,3)^{\mathrm{T}} \qquad t_1 = 1$$

$$\mathbf{x_2} = (4,-1)^{\mathrm{T}} \qquad t_2 = 2$$

$$\mathbf{x_3} = (4.6,1.5)^{\mathrm{T}} \qquad t_3 = 3$$

Now, let's define the Lagrange basis functions with $n = 3$ (degree of interpolation) and $k = 0,1,2,3$

$$L_0^3(t) = \prod_{\substack{j=0 \\ j\neq 0}}^{3} \frac{t - t_j}{t_0 - t_j} = \left(\frac{t - t_1}{t_0 - t_1}\right)\left(\frac{t - t_2}{t_0 - t_2}\right)\left(\frac{t - t_3}{t_0 - t_3}\right) = \frac{(t-1)(t-2)(t-3)}{(0-1)(0-2)(0-3)}$$

$$= -\frac{1}{6}(t-1)(t-2)(t-3) = \boxed{-\frac{1}{6}t^3 + t^2 - \frac{11}{6}t + 1}$$

$$L_1^3(t) = \prod_{\substack{j=0 \\ j\neq 1}}^{3} \frac{t - t_j}{t_1 - t_j} = \left(\frac{t - t_0}{t_1 - t_0}\right)\left(\frac{t - t_2}{t_1 - t_2}\right)\left(\frac{t - t_3}{t_1 - t_3}\right) = \frac{(t-0)(t-2)(t-3)}{(1-0)(1-2)(1-3)}$$

$$= \frac{1}{2}t(t-2)(t-3) = \boxed{\frac{1}{2}t^3 - \frac{5}{2}t^2 + 3t}$$

$$L_2^3(t) = \prod_{\substack{j=0 \\ j\neq 2}}^{3} \frac{t - t_j}{t_2 - t_j} = \left(\frac{t - t_0}{t_2 - t_0}\right)\left(\frac{t - t_1}{t_2 - t_1}\right)\left(\frac{t - t_3}{t_2 - t_3}\right) = \frac{(t-0)(t-1)(t-3)}{(2-0)(2-1)(2-3)}$$

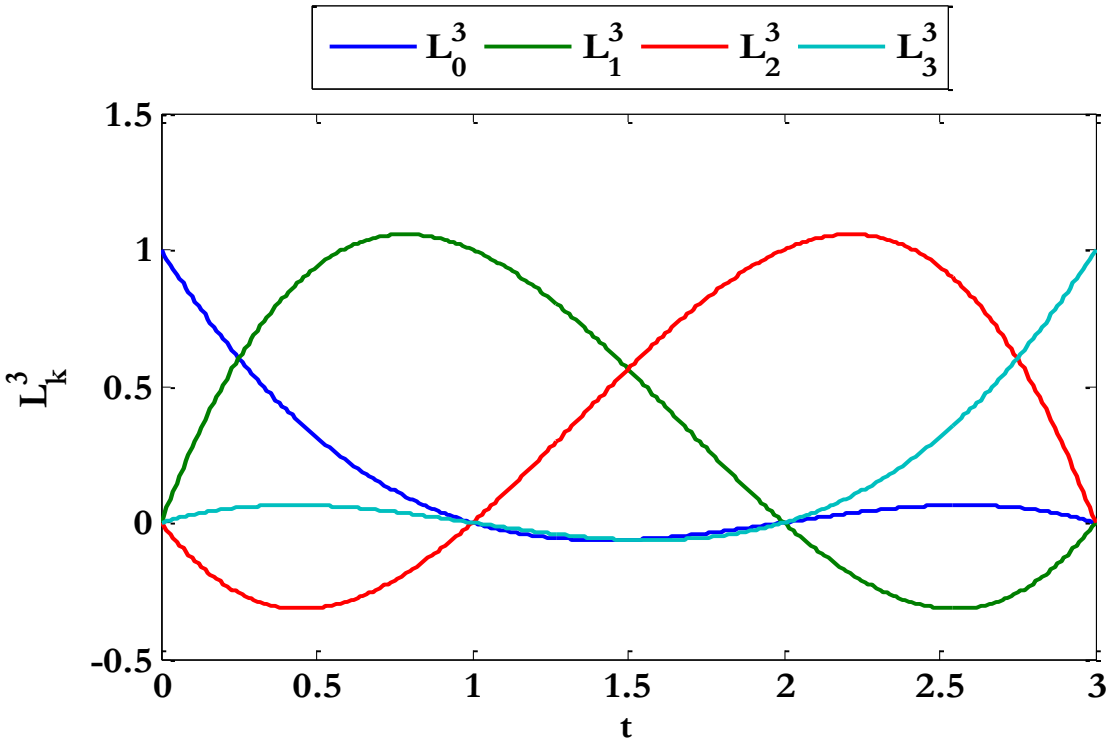
$$= -\frac{1}{2}t(t-1)(t-3) = \boxed{-\frac{1}{2}t^3 + 2t^2 - \frac{3}{2}t}$$

$$L_3^3(t) = \prod_{\substack{j=0 \\ j\neq 3}}^{3} \frac{t - t_j}{t_3 - t_j} = \left(\frac{t - t_0}{t_3 - t_0}\right)\left(\frac{t - t_1}{t_3 - t_1}\right)\left(\frac{t - t_2}{t_3 - t_2}\right) = \frac{(t-0)(t-1)(t-2)}{(3-0)(3-1)(3-2)}$$

$$= \frac{1}{6}t(t-1)(t-2) = \boxed{\frac{1}{6}t^3 - \frac{1}{2}t^2 + \frac{1}{3}t}$$

$$L_0^3(t) = -\frac{1}{6}t^3 + t^2 - \frac{11}{6}t + 1 \qquad L_2^3(t) = -\frac{1}{2}t^3 + 2t^2 - \frac{3}{2}t$$

$$L_1^3(t) = \frac{1}{2}t^3 - \frac{5}{2}t^2 + 3t \qquad L_3^3(t) = \frac{1}{6}t^3 - \frac{1}{2}t^2 + \frac{1}{3}t$$



Basis functions for uniform Lagrange interpolation of degree 3.

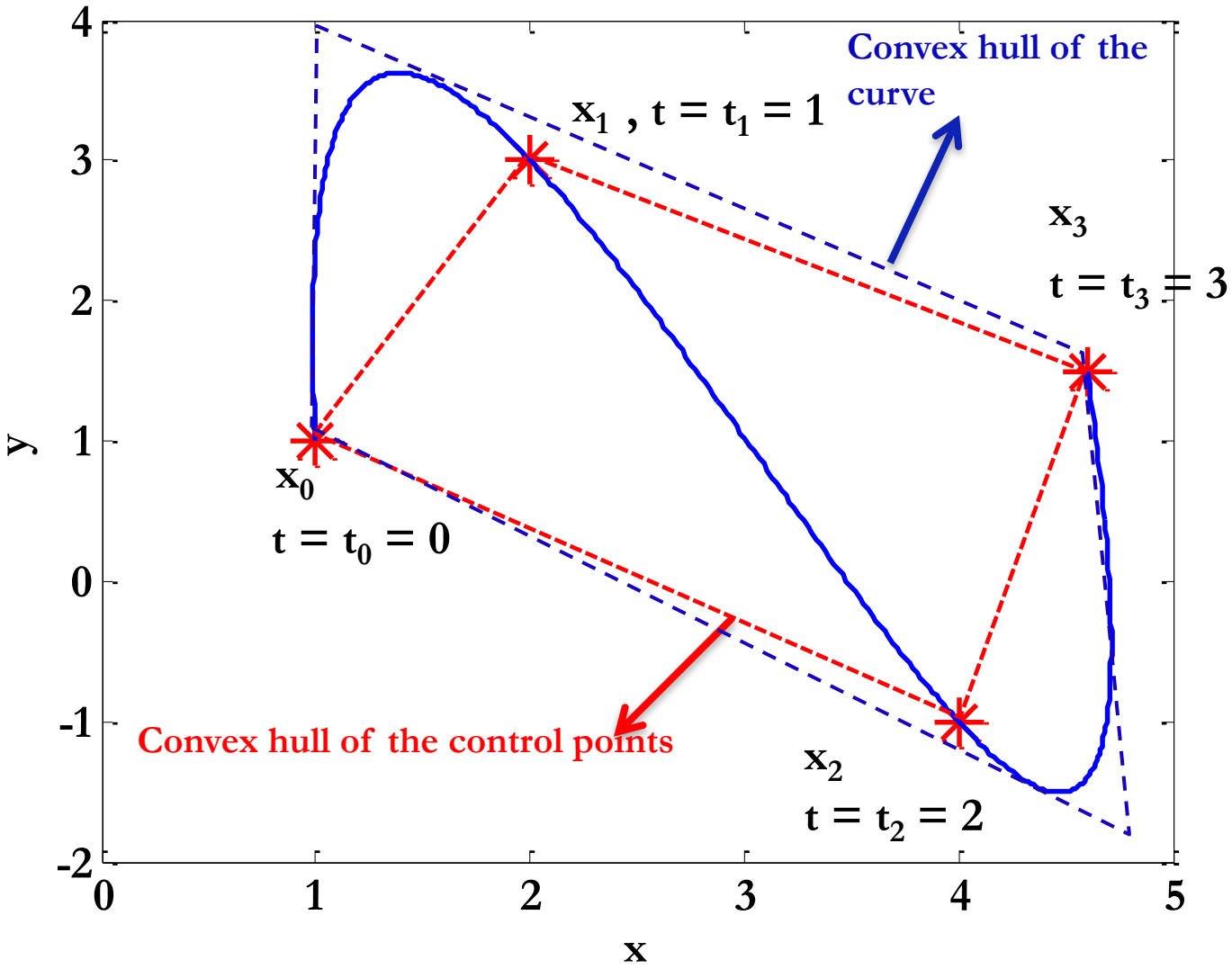
Let's check for the affine invariance, that is if $\sum_{k=0}^{n} L_k^n(t) = 1$

$$L_0^3(t) + L_1^3(t) + L_2^3(t) + L_3^3(t)$$
$$= \left(-\frac{1}{6}t^3 + t^2 - \frac{11}{6}t + 1\right) + \left(\frac{1}{2}t^3 - \frac{5}{2}t^2 + 3t\right) + \left(-\frac{1}{2}t^3 + 2t^2 - \frac{3}{2}t\right)$$
$$+ \left(\frac{1}{6}t^3 - \frac{1}{2}t^2 + \frac{1}{3}t\right) = 1$$

Hence the functions $L_k^n(t)$ are a partition of unity, thus curves produced by Lagrange interpolation are affine invariant.

The curve will be defined by;

$$\mathcal{C}: \quad \mathbf{x}(t) = L_0^3(t)\mathbf{x_0} + L_1^3(t)\mathbf{x_1} + L_2^3(t)\mathbf{x_2} + L_3^3(t)\mathbf{x_3}$$
$$= \left(-\frac{1}{6}t^3 + t^2 - \frac{11}{6}t + 1\right)\mathbf{x_0} + \left(\frac{1}{2}t^3 - \frac{5}{2}t^2 + 3t\right)\mathbf{x_1}$$
$$+ \left(-\frac{1}{2}t^3 + 2t^2 - \frac{3}{2}t\right)\mathbf{x_2} + \left(\frac{1}{6}t^3 - \frac{1}{2}t^2 + \frac{1}{3}t\right)\mathbf{x_3}$$



Uniform Lagrange interpolation of four points, control points are shown in read, the convex hull of the control points is shown in dashed-red line while the convex hull of the generated curve (shown in dashed-blue) is much larger than the convex hull of the control points, hence Lagrange curve does not satisfy the convex hull property.

In Matlab…

```
%% our variables
% parameter
syms t;
% controlling points (four points)
syms x0 x1 x2 x3;
```

```
%% basis functions (cubic)
L30 = expand((-1/6)*(t-1)*(t-2)*(t-3));
L31 = expand((1/2)*(t-0)*(t-2)*(t-3));
L32 = expand((-1/2)*(t-0)*(t-1)*(t-3));
L33 = expand((1/6)*(t-0)*(t-1)*(t-2));
sumL = L30 + L31 + L32 + L33;
```

```
%% the equation of the curve
x = L30*x0 + L31*x1 + L32*x2 + L33*x3;
```

```
%% the actual values of the controlling points
x0 = [1     1]';
x1 = [2     3]';
x2 = [4    -1]';
x3 = [4.6 1.5]';
control_pts = [x0 x1 x2 x3];
```

```
%% generating the curve
X = [];
i = 0;
for t = 0 : 0.01 : 3
    % evaluating the current point
    cur_x = eval(x);
    i = i + 1;
    X(:,i) = cur_x;

    % evaluating the individual basis functions
    basisL30(i)= eval(L30);
    basisL31(i)= eval(L31);
    basisL32(i)= eval(L32);
    basisL33(i)= eval(L33);
end
```

Next …

- However Lagrange interpolation does not satisfy the convex hull property.
- Typically, we would like the curve to be more confined, i.e. the area of the convex hull of the curve should not be much greater than the area of the convex hull of the control points.
- Next we will discuss some of the more popular curves and how they may be used to interpolate points.

Bézier Curves

Bézier curves are one of the most popular representations for curves.

Pierre Étienne Bézier (September 1, 1910 – November 25, 1999) was a French engineer and patentor (but not the inventor) of the Bézier curves and Bézier surfaces that are now used in most computer-aided design and computer graphics systems.

Definition 16: *Let* $\mathbf{x_0}, \mathbf{x_1}, \mathbf{x_2}, \dots, \mathbf{x_n}$ *be a set of control points, a* **Bézier curve** *of degree n is given by:*

$$\mathbf{x}(t) = \sum_{k=0}^{n} B_k^n(t)\mathbf{x_k} \qquad t \in [0,1]$$

where the basis functions $B_k^n(t)$ *are the Bernstein polynomials defined by;*

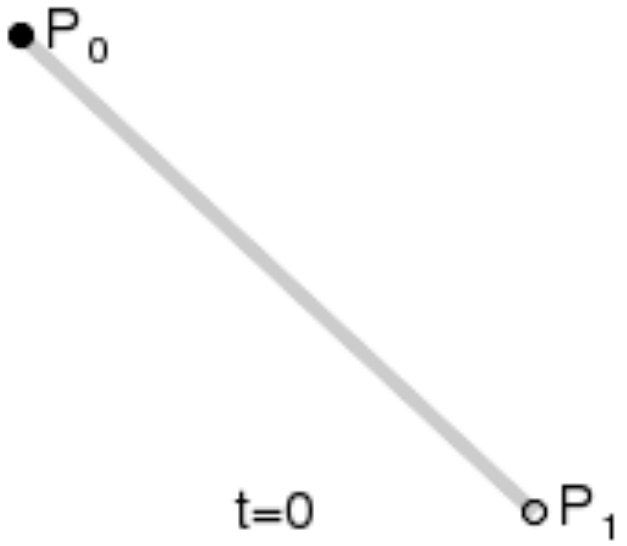
$$B_k^n(t) = \binom{n}{k} t^k (1-t)^{n-k}$$

where $\binom{n}{k} = \frac{n!}{k!(n-k)!}$

Bézier curves interpolate the end points $\mathbf{x_0}$ and $\mathbf{x_n}$, that is it connects the end points in a fashion directed by in-between control points, which do not lie on the curve, this is called endpoint interpolation property.

Definition 17: *Given two control points* $\mathbf{x_0}, \mathbf{x_1}$ *, a* **Linear Bézier curve** *is simply a straight line between those two points, the curve is given by:*

$$\mathbf{x}(t) = (1 - t)\mathbf{x_0} + t\mathbf{x_1} \quad , \quad t \in [0,1]$$

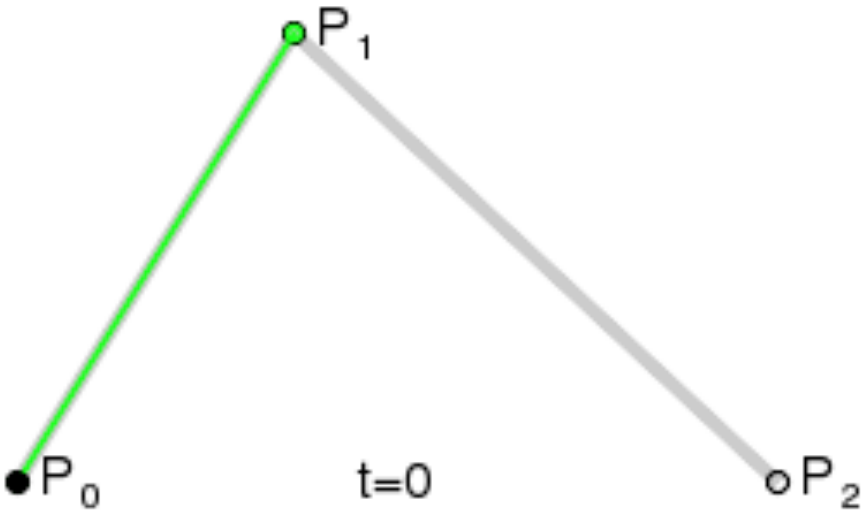


The *t* in the function for a linear Bézier curve can be thought of as describing how far x(*t*) is from P_0 to P_1. For example when *t=0.25*, x(*t*) is one quarter of the way from point P_0 to P_1. As *t* varies from 0 to 1, x(*t*) describes a curved line from P_0 to P_1.

Definition 18: *A* **quadratic Bézier curve** *is the path traced by the function* $\mathbf{x}(t)$ *given three control points* $\mathbf{x_0}, \mathbf{x_1}, \mathbf{x_2}$*:*

$$\mathbf{x}(t) = (1-t)^2\mathbf{x_0} + 2t(1-t)\mathbf{x_1} + t^2\mathbf{x_2} \quad , \quad t \in [0,1]$$

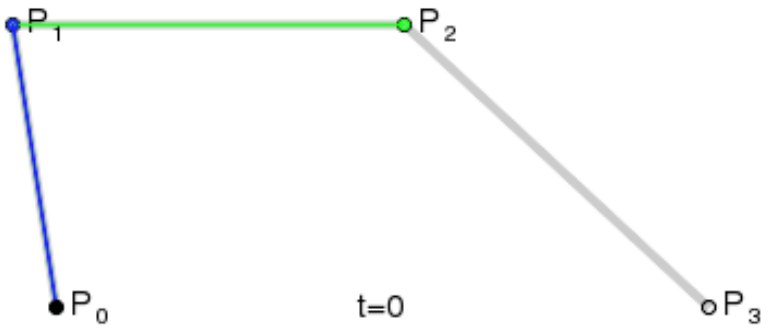
A quadratic Bézier curve is also a parabolic segment.

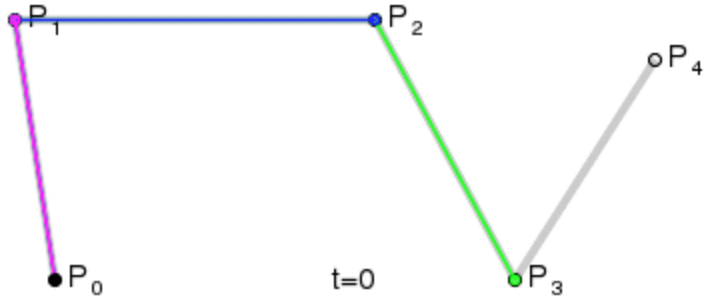


For quadratic Bézier curves one can construct intermediate points Q_0 and Q_1 such that as *t* varies from 0 to 1: Point Q_0 varies from P_0 to P_1 and describes a linear Bézier curve. Point Q_1 varies from P_1 to P_2 and describes a linear Bézier curve. Point x(*t*) varies from Q_0 to Q_1 and describes a quadratic Bézier curve.

Definition 19: *Four control points* $\mathbf{x_0}, \mathbf{x_1}, \mathbf{x_2}, \mathbf{x_3}$ *in the plane or in three-dimensional space define a* **cubic Bézier curve**, *the curve starts at* $\mathbf{x_0}$ *going toward* $\mathbf{x_1}$ *and arrives at* $\mathbf{x_3}$ *coming from the direction of* $\mathbf{x_2}$, *usually it will not pass through* $\mathbf{x_1}$ *or* $\mathbf{x_2}$, *these points are only there to provide directional information, the parametric form of the curve is:*

$$\mathbf{x}(t) = (1 - 3t + 3t^2 - t^3)\mathbf{x_0} + (3t - 6t^2 + 3t^3)\mathbf{x_1} + (3t^2 - 3t^3)\mathbf{x_2} + t^3\mathbf{x_3} \quad , \quad t \in [0,1]$$





For higher-order curves one needs correspondingly more intermediate points. For cubic curves one can construct intermediate points Q0, Q1 & Q2 that describe linear Bézier curves, and points R0 & R1 that describe quadratic Bézier curves. For fourth-order curves one can construct intermediate points Q0, Q1, Q2 & Q3 that describe linear Bézier curves, points R0, R1 & R2 that describe quadratic Bézier curves, and points S0 & S1 that describe cubic Bézier curves.

Definition 20: *Let* $\mathbf{x_0}, \mathbf{x_1}, \mathbf{x_2}, \dots, \mathbf{x_n}$ *be a set of control points, a* **Bézier curve** *of degree n given by:*

$$\mathbf{x}(t) = \sum_{k=0}^{n} B_k^n(t)\mathbf{x_k} \qquad t \in [0,1]$$

can be expressed recursively as follows: Let $\mathbf{x}_{\mathbf{x_0 x_1 \dots x_n}}(t)$ *denote the Bézier curve denoted by the control points* $\mathbf{x_0}, \mathbf{x_1}, \mathbf{x_2}, \dots, \mathbf{x_n}$*, then,*

$$\mathbf{x}(t) = \mathbf{x}_{\mathbf{x_0 x_1 \dots x_n}}(t) = (1-t)\mathbf{x}_{\mathbf{x_0 x_1 \dots x_{n-1}}}(t) + t\mathbf{x}_{\mathbf{x_1 x_2 \dots x_n}}(t)$$

Hence, the Bézier curve of degree n is a linear interpolation between two Bézier curves of degree n-1.

Definition 21: *Let* $\mathbf{x_0}, \mathbf{x_1}, \mathbf{x_2}, \dots, \mathbf{x_n}$ *be a set of control points of the Bézier curve, the polygon formed by connecting the Bézier points with lines, starting with* $\mathbf{x_0}$ *and finishing with* $\mathbf{x_n}$ *, is called the* **Bézier polygon**. *The convex hull of the Bézier polygon contains the Bézier curve.*

Bernstein Polynomials Properties

Theorem 2: *The Bernstein polynomials are a partition of unity, i.e.* $\sum_{k=0}^{n} B_k^n(t) = 1$, *hence Bézier curves are affine invariant.*

Proof left as homework ☹

Theorem 3: *The Bernstein polynomials* $B_k^n(t)$ *are defined such that* $0 \leq B_k^n(t) \leq 1$ *for* $t \in [0,1]$. *A point of the Bézier curve* $\mathbf{x}(t) = \sum_{k=0}^{n} B_k^n(t)\mathbf{x_k}$ *is thus a weighted average of the points* $\mathbf{x_0}, \mathbf{x_1}, \mathbf{x_2}, \dots, \mathbf{x_n}$. *The convex hull of the curve* $\mathbf{x}(t)$ *is the set of all weighted averages of* $\mathbf{x_0}, \mathbf{x_1}, \mathbf{x_2}, \dots, \mathbf{x_n}$. *The Bézier curve thus lies in the convex hull of the points* $\mathbf{x_0}, \mathbf{x_1}, \mathbf{x_2}, \dots, \mathbf{x_n}$, *where the convex hull is defined by a polygon created from these points.*

Example

- The most popular Bézier curves are Bézier curves of degree 3. Let the control points be given as follows:

$$\mathbf{x_0} = (1,1)^T$$

$$\mathbf{x_1} = (2,3)^T$$

$$\mathbf{x_2} = (4,-1)^T$$

$$\mathbf{x_3} = (4.6,1.5)^T$$

Now, let's define the Bernstein polynomials, i.e. Bézier basis functions with $n = 3$ and $k = 0,1,2,3$

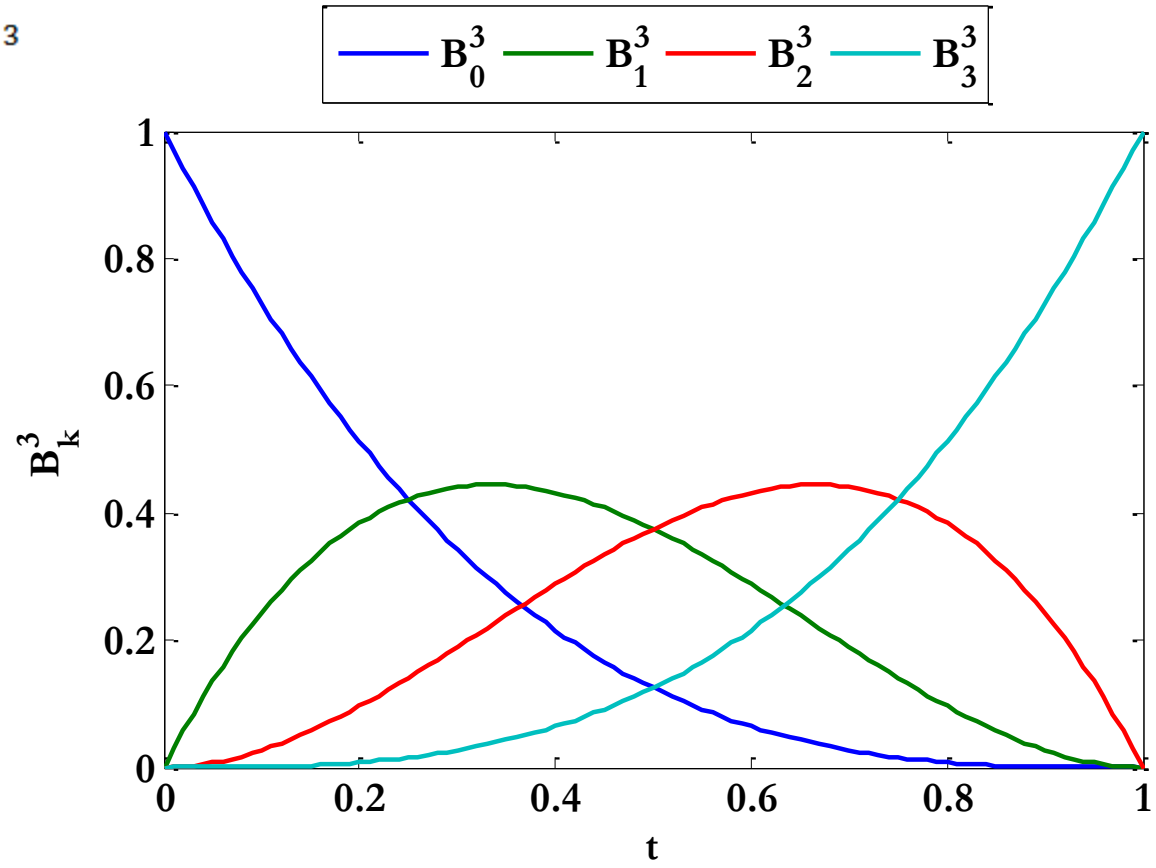
$$B_0^3(t) = \binom{3}{0} t^0(1-t)^{3-0} = \frac{3!}{0!\,(3-0)!}(1-t)^3 = (1-t)^3 = \boxed{1 - 3t + 3t^2 - t^3}$$

$$B_1^3(t) = \binom{3}{1} t^1(1-t)^{3-1} = \frac{3!}{1!\,(3-1)!}t(1-t)^2 = 3t(1-t)^2 = \boxed{3t - 6t^2 + 3t^3}$$

$$B_2^3(t) = \binom{3}{2} t^2(1-t)^{3-2} = \frac{3!}{2!\,(3-2)!}t^2(1-t) = 3t^2(1-t) = \boxed{3t^2 - 3t^3}$$

$$B_3^3(t) = \binom{3}{3} t^3(1-t)^{3-3} = \frac{3!}{3!\,(3-3)!}t^3 = \boxed{t^3}$$

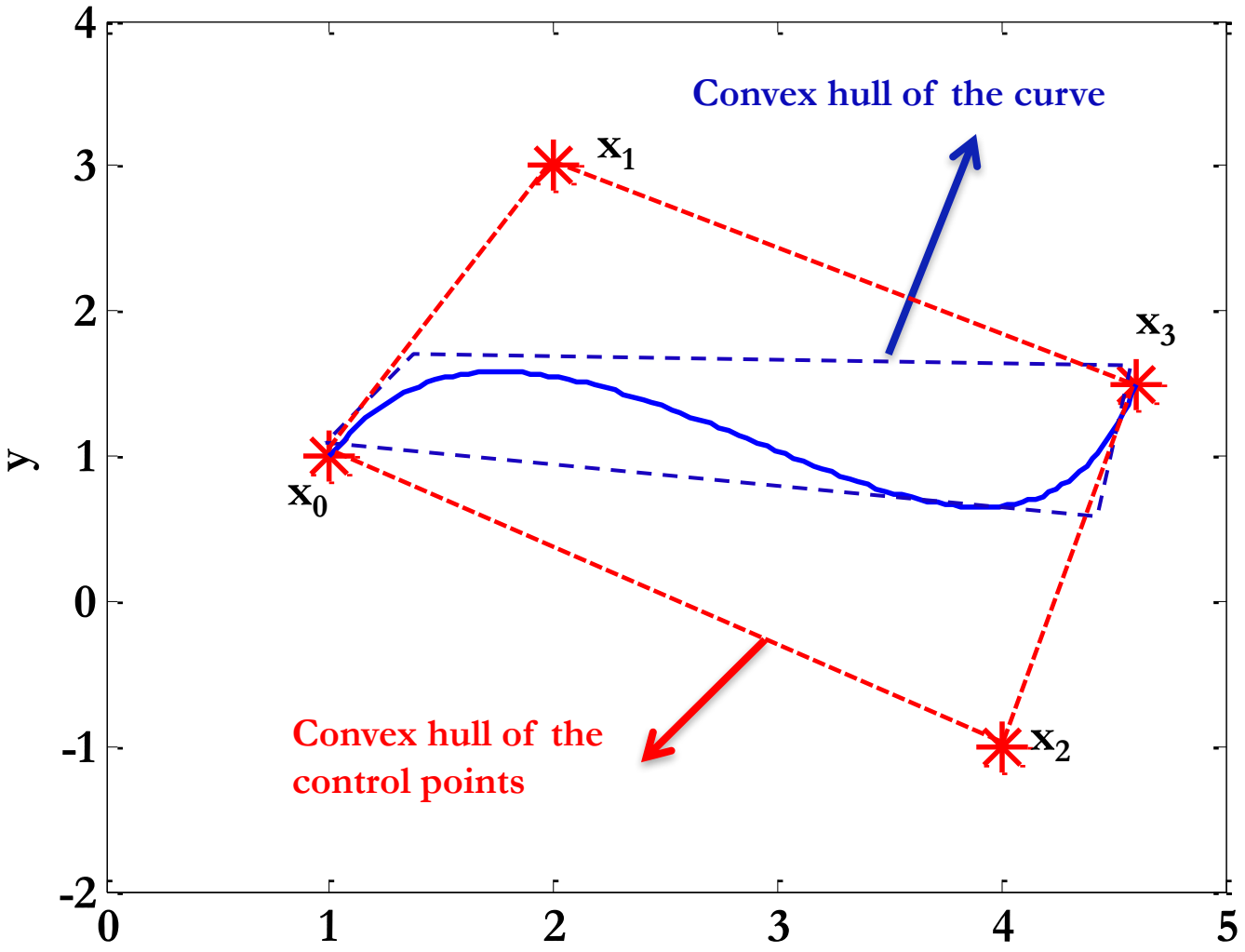
$$B_0^3(t) = 1 - 3t + 3t^2 - t^3$$
$$B_1^3(t) = 3t - 6t^2 + 3t^3$$
$$B_2^3(t) = 3t^2 - 3t^3$$
$$B_3^3(t) = t^3$$



Basis functions for Bézier curves of degree 3.

- The curve will be defined by;

$$\mathcal{C}: \quad \mathbf{x}(t) = B_0^3(t)\mathbf{x_0} + B_1^3(t)\mathbf{x_1} + B_2^3(t)\mathbf{x_2} + B_3^3(t)\mathbf{x_3}$$
$$= (1 - 3t + 3t^2 - t^3)\mathbf{x_0} + (3t - 6t^2 + 3t^3)\mathbf{x_1} + (3t^2 - 3t^3)\mathbf{x_2} + t^3\mathbf{x_3}$$



Bézier curve of four points and its convex hull , control points are shown in read, the convex hull of the control points is shown in dashed-red line while the convex hull of the generated curve (shown in dashed-blue) is inside the convex hull of the control points, hence Bézier curve satisfy the convex hull property.

In Matlab…

```
%% our variables
% parameter
syms t;
% controlling points (four points)
syms x0 x1 x2 x3;
```

```
%% basis functions (cubic)
B30 = expand((1-t)^3);
B31 = expand(3*t*((1-t)^2));
B32 = expand(3*(t^2)*(1-t));
B33 = t^3;
sumB = B30 + B31 + B32 + B33;
```

```
%% the equation of the curve
x = B30*x0 + B31*x1 + B32*x2 + B33*x3;
```

```
%% the actual values of the controlling points
x0 = [1     1]';
x1 = [2     3]';
x2 = [4    -1]';
x3 = [4.6 1.5]';
control_pts = [x0 x1 x2 x3];
```

```
%% generating the curve
X = [];
i = 0;
for t = 0 : 0.01 : 1
    % evaluating the current point
    cur_x = eval(x);
    i = i + 1;
    X(:,i) = cur_x;

    % evaluating the individual basis functions
    basisB30(i)= eval(B30);
    basisB31(i)= eval(B31);
    basisB32(i)= eval(B32);
    basisB33(i)= eval(B33);
end
```

Derivatives of Bézier Curves

Theorem 4: *The derivative of a Bézier curve is also a Bézier curve, furthermore, we have:*

$$\mathbf{x}'(0) = n(\mathbf{x_1} - \mathbf{x_0}) \text{ and } \mathbf{x}'(1) = n(\mathbf{x_n} - \mathbf{x_{n-1}})$$

Thus we have immediate form of the derivative at the end points. These are the tangents to the curve at the end points. This will be useful in defining curves that are piecewise continuous.

Homework ☹

Differentiate Bézier curve and prove Theorem 4

Piecewise Continuous Bézier Curves

- We can construct Bézier curves of arbitrary degree, however it becomes more difficult to control the curves since the Bézier curve is only guaranteed to interpolate end points. Instead we can create several Bézier segments that are piecewise continuous.

Definition 22: Piecewise continuous curves *are defined to be continuous curves, where there is also continuity at the points where they join.*

- There are different kinds of continuity which can be considered.

Definition 23: *Let* $k \geq 0$*, a function* $\mathbf{f}$ *is* C^k *continuous if* $\mathbf{f}$ *has the* k^{th} *derivative defined and continuous everywhere in the domain of* $\mathbf{f}$*.* C^0 *continuity is simply the usual definition of continuity.* $\mathbf{f}$ *is* C^∞ *continuous if* $\mathbf{f}$ *is* C^k *continuous for all* $k \geq 0$*.*

Definition 24: *Let* $\mathbf{f}$ *be a continuous function, let* $t = t(\mathbf{u})$ *be a continuous and strictly increasing function, let* $\boldsymbol{g}(\boldsymbol{u}) = \boldsymbol{f}(t(\mathbf{u}))$*, if* $\boldsymbol{g}(\boldsymbol{u})$ *has a continuous, non-zero first derivative everywhere in its domain, then* $\mathbf{f}$ *is said to be* G^1 *continuous.*

We can obtain C^1 continuity between two Bézier curves $\mathbf{q}(t)$ and $\mathbf{r}(t)$ defined as:

$$\mathbf{q}(t) = \sum_{k=0}^{n} B_k^n(t)\mathbf{q_k} \qquad \text{and} \qquad \mathbf{r}(t) = \sum_{k=0}^{n} B_k^n(t)\mathbf{r_k}$$

Assuming that $\mathbf{q}(t)$ will occur before $\mathbf{r}(t)$, by requiring $\mathbf{q}'(1) = \mathbf{r}'(0)$, using Theorem 4, we will have:

$$\mathbf{q}'(1) = n(\mathbf{q_n} - \mathbf{q_{n-1}}) \qquad \text{and} \qquad \mathbf{r}'(0) = n(\mathbf{r_1} - \mathbf{r_0})$$

thus we have

$$\mathbf{q_n} - \mathbf{q_{n-1}} = \mathbf{r_1} - \mathbf{r_0}$$

For G^1 continuity, we require that:

$$\mathbf{q_n} - \mathbf{q_{n-1}} = s(\mathbf{r_1} - \mathbf{r_0}) \qquad , s > 0$$

This can be adapted to Bézier curves of different degrees.

Theorem 5: *A piecewise smooth curve* C^1 *from* m *Bézier curves,* $\mathbf{q}^i(t), i = 1,2,\dots,m$ *of degree* n *by requiring that:*

$$\mathbf{q}_{\mathbf{n}}^{i} - \mathbf{q}_{\mathbf{n-1}}^{i} = \mathbf{q}_{\mathbf{1}}^{i+1} - \mathbf{q}_{\mathbf{0}}^{i+1} \quad ; i = 1,2,\dots,m$$

The continuous curve can then be defined by:

$$\mathbf{Q}(t) = \begin{cases} \mathbf{q}^1(t) & t \in [0, \frac{1}{m}) \\ \mathbf{q}^2(t) & t \in [\frac{1}{m}, \frac{2}{m}) \\ \vdots & \\ \mathbf{q}^m(t) & t \in [\frac{m-1}{m}, 1] \end{cases}$$

where the control points of $\mathbf{Q}(t)$ *are simply the control points of* $\mathbf{q}^i(t), i = 1,2,\dots,m$ *with the restriction mentioned above.*

Recall…

Definition 16: *Let* $\mathbf{x_0}, \mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_n}$ *be a set of control points, a* **Bézier curve** *of degree n is given by:*

$$\mathbf{x}(t) = \sum_{k=0}^{n} B_k^n(t)\mathbf{x_k} \qquad t \in [0,1]$$

where the basis functions $B_k^n(t)$ *are the Bernstein polynomials defined by;*

$$B_k^n(t) = \binom{n}{k} t^k (1-t)^{n-k}$$

where $\binom{n}{k} = \frac{n!}{k!(n-k)!}$

Definition 16 gives the equation of a Bézier curve which starts at $t = 0$ and ends at $t = 1$. It is useful, especially when fitting together a string of Bézier curves, to allows an arbitrary parameter interval.

Definition 24: *Let* $\mathbf{x_0}, \mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_n}$ *be a set of control points, a* **Bézier curve** *of degree n defined over an interval* $t \in [t_0, t_1]$ *is given by:*

$$\mathbf{x}(t) = \sum_{k=0}^{n} B_k^n(t)\mathbf{x_k} \quad , \quad t \in [t_0, t_1]$$

where the basis functions $B_k^n(t)$ *are the Bernstein polynomials defined over* $t \in [t_0, t_1]$ *given by;*

$$B_k^n(t) = \binom{n}{k} \left(\frac{t - t_0}{t_1 - t_0}\right)^k \left(\frac{t_1 - t}{t_1 - t_0}\right)^{n-k}$$

where $\binom{n}{k} = \frac{n!}{k!(n-k)!}$

Example

Let the control points of $\mathbf{q}^1$ be given as follows

$$\mathbf{q}_0^1 = (0,1)^T$$

$$\mathbf{q}_1^1 = (0.5,1)^T$$

$$\mathbf{q}_2^1 = (1,0.5)^T$$

$$\mathbf{q}_3^1 = (1,0)^T$$

Let the control points of $\mathbf{q}^2$ be given as follows

$$\mathbf{q}_0^2 = (1,0)^T$$

$$\mathbf{q}_1^2 = (1,-0.5)^T$$

$$\mathbf{q}_2^2 = (2,0)^T$$

$$\mathbf{q}_3^2 = (2,0.5)^T$$

Note that $\mathbf{q}_3^1 - \mathbf{q}_2^1 = \mathbf{q}_1^2 - \mathbf{q}_0^2$

The curve will be defined as follows:

$$\mathbf{Q}(t) = \begin{cases} \mathbf{q}^1(t) & t \in [0, \frac{1}{2}) \\ \mathbf{q}^2(t) & t \in [\frac{1}{2}, 1] \end{cases}$$

where;

$$\mathbf{q}^i(t) = \sum_{k=0}^{3} B_{k,i}^3(t)\mathbf{q}_\mathbf{k}^i = B_{0,i}^3(t)\mathbf{q}_\mathbf{0}^i + B_{1,i}^3(t)\mathbf{q}_\mathbf{1}^i + B_{2,i}^3(t)\mathbf{q}_\mathbf{2}^i + B_{3,i}^3(t)\mathbf{q}_\mathbf{3}^i$$

With:

$$B_{k,i}^n(t) = \binom{n}{k}\left(\frac{t - t_{i-1}}{t_i - t_{i-1}}\right)^k \left(\frac{t_i - t}{t_i - t_{i-1}}\right)^{n-k}$$

Let's find the basis functions for the first Bézier curve, where $i = 1, n = 3$ and $t \in [0, \frac{1}{2})$, hence $t_{i-1} = t_0 = 0$ and $t_i = t_1 = \frac{1}{2}$,

$$B_{0,1}^3(t) = \binom{3}{0}\left(\frac{t-0}{\frac{1}{2}-0}\right)^0\left(\frac{\frac{1}{2}-t}{\frac{1}{2}-0}\right)^{3-0} = \frac{3!}{0!\,(3-0)!}(1-2t)^3 = (1-2t)^3$$
$$= \boxed{1 - 6t + 12t^2 - 8t^3}$$

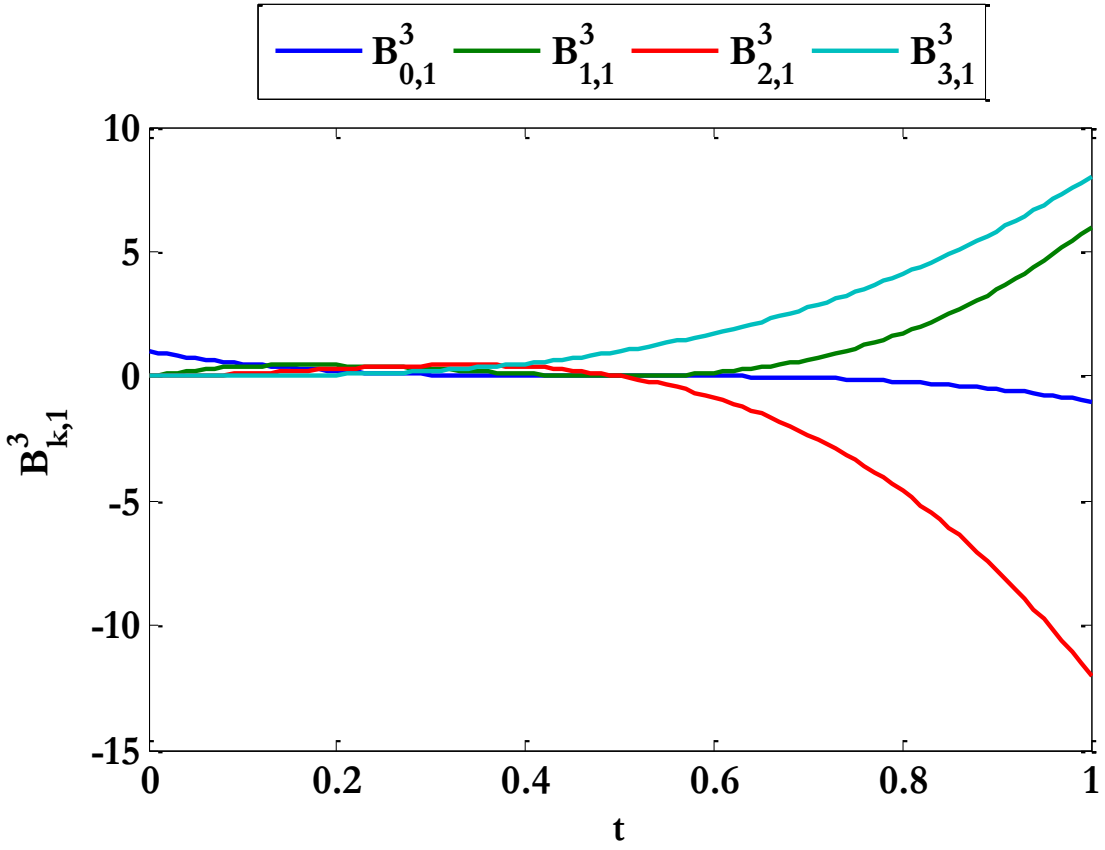
$$B_{1,1}^3(t) = \binom{3}{1}\left(\frac{t-0}{\frac{1}{2}-0}\right)^1\left(\frac{\frac{1}{2}-t}{\frac{1}{2}-0}\right)^{3-1} = \frac{3!}{1!\,(3-1)!}(2t)(1-2t)^2 = 3(2t)(1-2t)^2$$
$$= \boxed{6t - 24t^2 + 24t^3}$$

$$B_{2,1}^3(t) = \binom{3}{2}\left(\frac{t-0}{\frac{1}{2}-0}\right)^2\left(\frac{\frac{1}{2}-t}{\frac{1}{2}-0}\right)^{3-2} = \frac{3!}{2!\,(3-2)!}(2t)^2(1-2t) = 3(2t)^2(1-2t)$$
$$= \boxed{12t^2 - 24t^3}$$

$$B_{3,1}^3(t) = \binom{3}{3}\left(\frac{t-0}{\frac{1}{2}-0}\right)^3\left(\frac{\frac{1}{2}-t}{\frac{1}{2}-0}\right)^{3-3} = \frac{3!}{3!\,(3-3)!}(2t)^3 = (2t)^3 = \boxed{8t^3}$$

Thus,

$$\mathbf{q}^1(t) = B_{0,1}^3(t)\mathbf{q}_\mathbf{0}^1 + B_{1,1}^3(t)\mathbf{q}_\mathbf{1}^1 + B_{2,1}^3(t)\mathbf{q}_\mathbf{2}^1 + B_{3,1}^3(t)\mathbf{q}_\mathbf{3}^1$$
$$= (1 - 6t + 12t^2 - 8t^3)\mathbf{q}_\mathbf{0}^1 + (6t - 24t^2 + 24t^3)\mathbf{q}_\mathbf{1}^1 + (12t^2 - 24t^3)\mathbf{q}_\mathbf{2}^1$$
$$+ (8t^3)\mathbf{q}_\mathbf{3}^1$$



$$B_{0,1}^3(t) = 1 - 6t + 12t^2 - 8t^3$$
$$B_{1,1}^3(t) = 6t - 24t^2 + 24t^3$$
$$B_{2,1}^3(t) = 12t^2 - 24t^3$$
$$B_{3,1}^3(t) = 8t^3$$

Now, let's find the basis functions for the second Bézier curve, where $i = 2, n = 3$ and $t \in [\frac{1}{2}, 1]$, hence $t_{i-1} = t_1 = \frac{1}{2}$ and $t_i = t_2 = 1$,

$$B_{0,2}^3(t) = \binom{3}{0}\left(\frac{t-\frac{1}{2}}{1-\frac{1}{2}}\right)^0\left(\frac{1-t}{1-\frac{1}{2}}\right)^{3-0} = \frac{3!}{0!\,(3-0)!}(2-2t)^3 = (2-2t)^3$$
$$= \boxed{8 - 24t + 24t^2 - 8t^3}$$

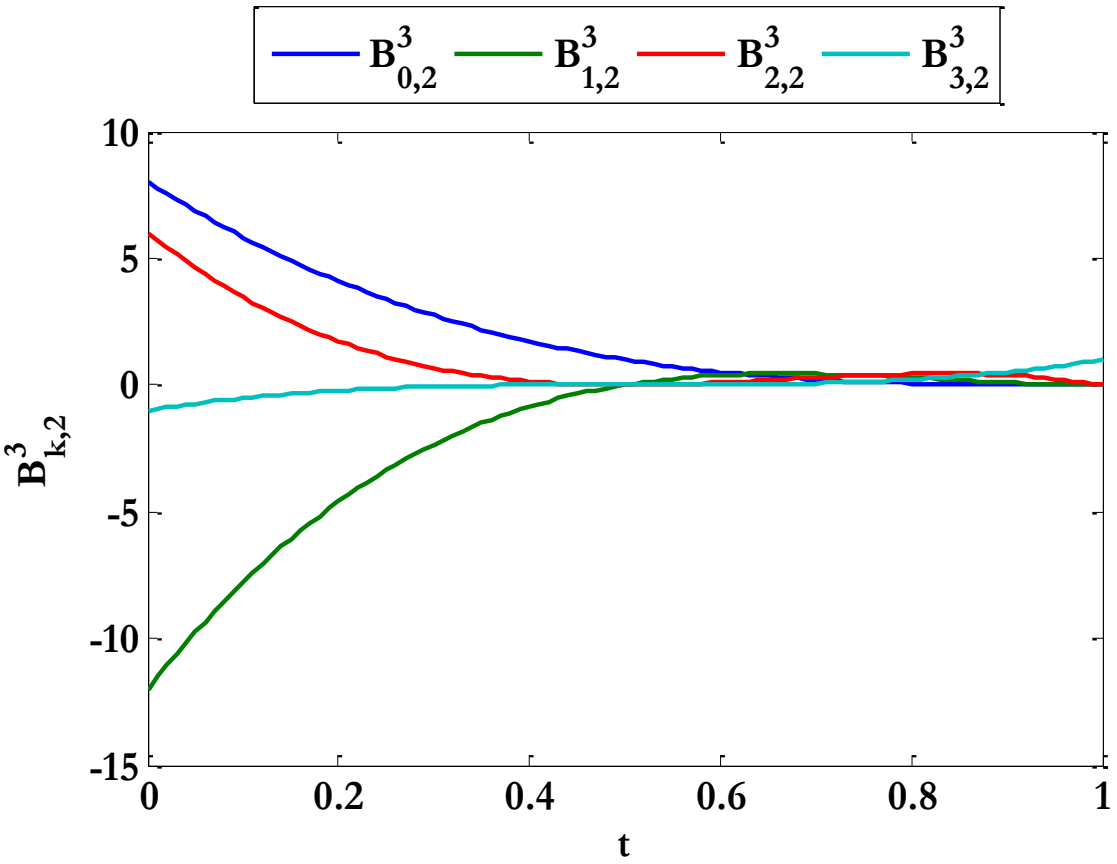
$$B_{1,2}^3(t) = \binom{3}{1}\left(\frac{t-\frac{1}{2}}{1-\frac{1}{2}}\right)^1\left(\frac{1-t}{1-\frac{1}{2}}\right)^{3-1} = \frac{3!}{1!\,(3-1)!}(2t-1)(2-2t)^2 = 3(2t-1)(2-2t)^2$$
$$= \boxed{-12 + 48t - 60t^2 + 24t^3}$$

$$B_{2,2}^3(t) = \binom{3}{2}\left(\frac{t-\frac{1}{2}}{1-\frac{1}{2}}\right)^2\left(\frac{1-t}{1-\frac{1}{2}}\right)^{3-2} = \frac{3!}{2!\,(3-2)!}(2t-1)^2(2-2t) = 3(2t-1)^2(2-2t)$$
$$= \boxed{6 - 30t + 48t^2 - 24t^3}$$

$$B_{3,2}^3(t) = \binom{3}{3}\left(\frac{t-\frac{1}{2}}{1-\frac{1}{2}}\right)^3\left(\frac{1-t}{1-\frac{1}{2}}\right)^{3-3} = \frac{3!}{3!\,(3-3)!}(2t-1)^3 = (2t-1)^3$$
$$= \boxed{-1 + 6t - 12t^2 + 8t^3}$$

Thus,

$$\mathbf{q}^2(t) = B^3_{0,2}(t)\mathbf{q}^2_0 + B^3_{1,2}(t)\mathbf{q}^2_1 + B^3_{2,2}(t)\mathbf{q}^2_2 + B^3_{3,2}(t)\mathbf{q}^2_3$$



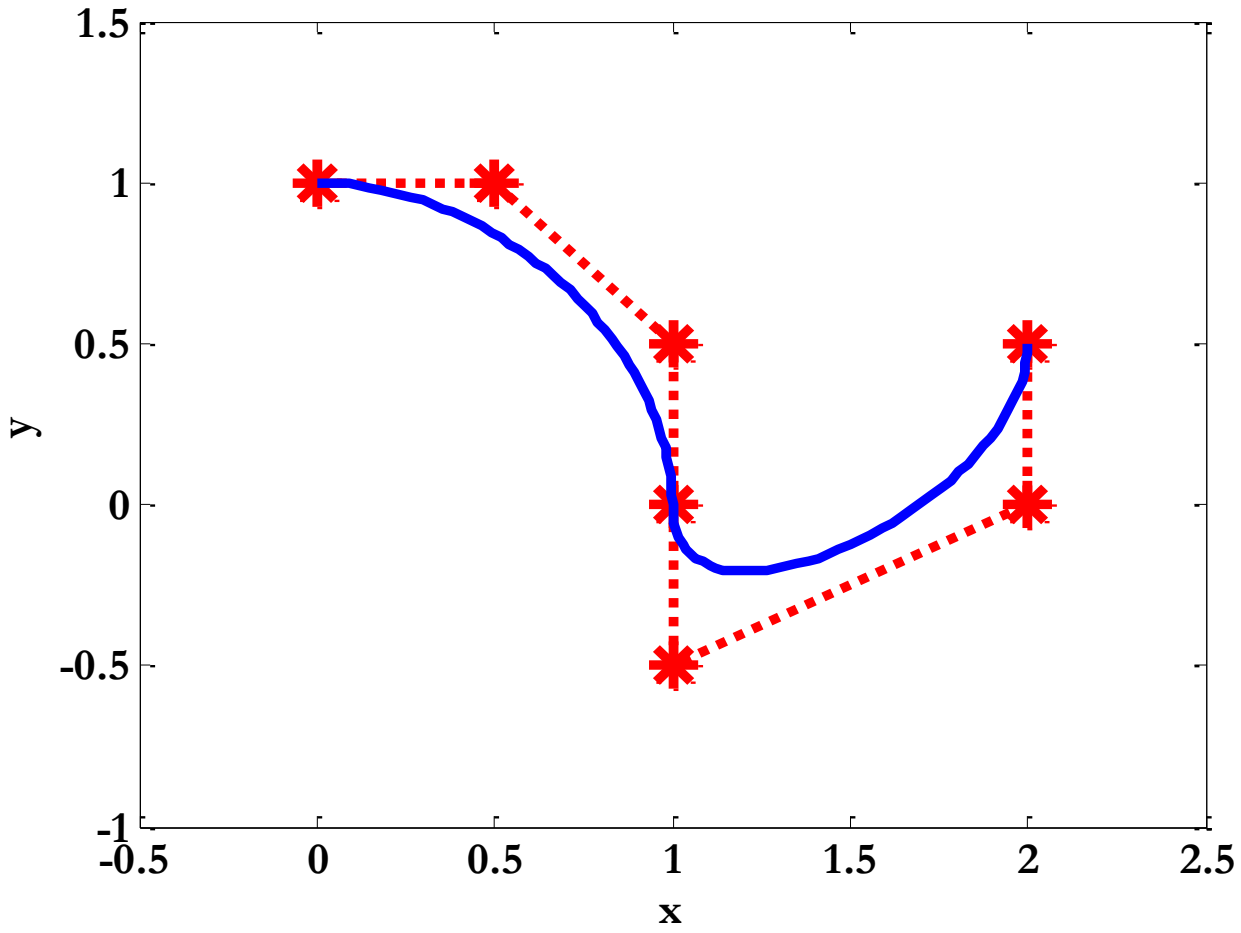
$$B^3_{0,2}(t) = 8 - 24t + 24t^2 - 8t^3$$

$$B^3_{1,2}(t) = -12 + 48t - 60t^2 + 24t^3$$

$$B^3_{2,2}(t) = 6 - 30t + 48t^2 - 24t^3$$

$$B^3_{3,2}(t) = -1 + 6t - 12t^2 + 8t^3$$

$$\mathbf{Q}(t) = \begin{cases} \mathbf{q}^1(t) & t \in [0, \frac{1}{2}) \\ \mathbf{q}^2(t) & t \in [\frac{1}{2}, 1] \end{cases}$$



Piecewise Bézier curve with two pieces, control points are shown in red, the generate curve is shown in blue

In Matlab…

```
%% our variables
% parameter
syms t;
% controlling points (four points)
% first piece
syms q10 q11 q12 q13;

% second piece
syms q20 q21 q22 q23;
```

```
%% basis functions (cubic)
% for the first piece
B30_1 = expand((1-2*t)^3);
B31_1 = expand(3*(2*t)*(1-2*t)^2);
B32_1 = expand(3*((2*t)^2)*(1-2*t));
B33_1 = expand((2*t)^3);
```

```
% for the second piece
B30_2 = expand((2-2*t)^3);
B31_2 = expand(3*(2*t-1)*((2-2*t)^2));
B32_2 = expand(3*((2*t-1)^2)*(2-2*t));
B33_2 = expand((2*t-1)^3);
```

```
%% the equation of the curve
% the first piece
q1 = B30_1*q10 + B31_1*q11 + B32_1*q12 + B33_1*q13;

% the second piece
q2 = B30_2*q20 + B31_2*q21 + B32_2*q22 + B33_2*q23;
```

In Matlab…

```
%% the actual values of the controlling points
% the first piece
q10 = [0     1]';
q11 = [0.5   1]';
q12 = [1   0.5]';
q13 = [1     0]';

% the second piece
q20 = [1     0]';
q21 = [1  -0.5]';
q22 = [2     0]';
q23 = [2   0.5]';

control_pts = [q10 q11 q12 q13 q20 q21 q22 q23];
```

```
%% generating the curve
Q = [];
i = 0;
for t = 0 : 0.01 : 1
    % evaluating the current point
    if t < 0.5
        cur_q = eval(q1);
    else
        cur_q = eval(q2);
    end
    i = i + 1;
    Q(:,i) = cur_q;

    % evaluating the individual basis functions
    basisB30_1(i)= eval(B30_1);
    basisB31_1(i)= eval(B31_1);
    basisB32_1(i)= eval(B32_1);
    basisB33_1(i)= eval(B33_1);

    % evaluating the individual basis functions
    basisB30_2(i)= eval(B30_2);
    basisB31_2(i)= eval(B31_2);
    basisB32_2(i)= eval(B32_2);
    basisB33_2(i)= eval(B33_2);
end
```

Bézier Curves Rendering

Following the construction of a Bézier curve, the next important task is to find the point $\mathbf{x}(t)$ on the curve for a particular t . A simple way is to plug t into every basis function, compute the product of each basis function and its corresponding control point, and finally add them together.

Hence, to render a Bézier curve $\mathbf{x}(t)$ we can simply choose a number of values $t_j \in [0,1], j = 0,1,\dots,m$, such that $t_j < t_{j+1}$ and render line segments from $\mathbf{x}(t_j)$ to $\mathbf{x}(t_{j+1})$, where t_j are usually selected equally spaced. as $m \to \infty$, we get closer to the correct Bézier curve.

While this works fine, it is not numerically stable (i.e., could introduce numerical errors during the course of evaluating the Bernstein polynomials). In what follows, we will discuss two main approaches which are commonly used to increase the speed at which points on Bézier curves can be evaluated and produce numerically stable results.

de Casteljau's Method

de Casteljau's method allows us to evaluate the points on a Bézier curve by repeated linear interpolation. If the control points of the curve are $\mathbf{x_k}$, for $k = 0,1,\dots,n$, which define a Bézier curve of degree n, lets denote them as $\mathbf{x_k^0}$, for $k = 0,1,\dots,n$, The 0 here indicate the initial or the 0-th iteration.

The fundamental concept of de Casteljau's algorithm is to choose a point C in line segment AB such that C divides the line segment AB in a ratio of $t\colon 1-t$ (i.e., the ratio of the distance between A and C and the distance between A and B is t). Let us find a way to determine the point C.

The vector from A to B is B - A. Since t is a ratio in the range of 0 and 1, point C is located at t (B - A). Taking the position of A into consideration, point C is A + t (B - A) = (1 - t)A + t B. Therefore, given a t, (1 - t)A + t B is the point C between A and B that divides AB in a ratio of t:1-t.

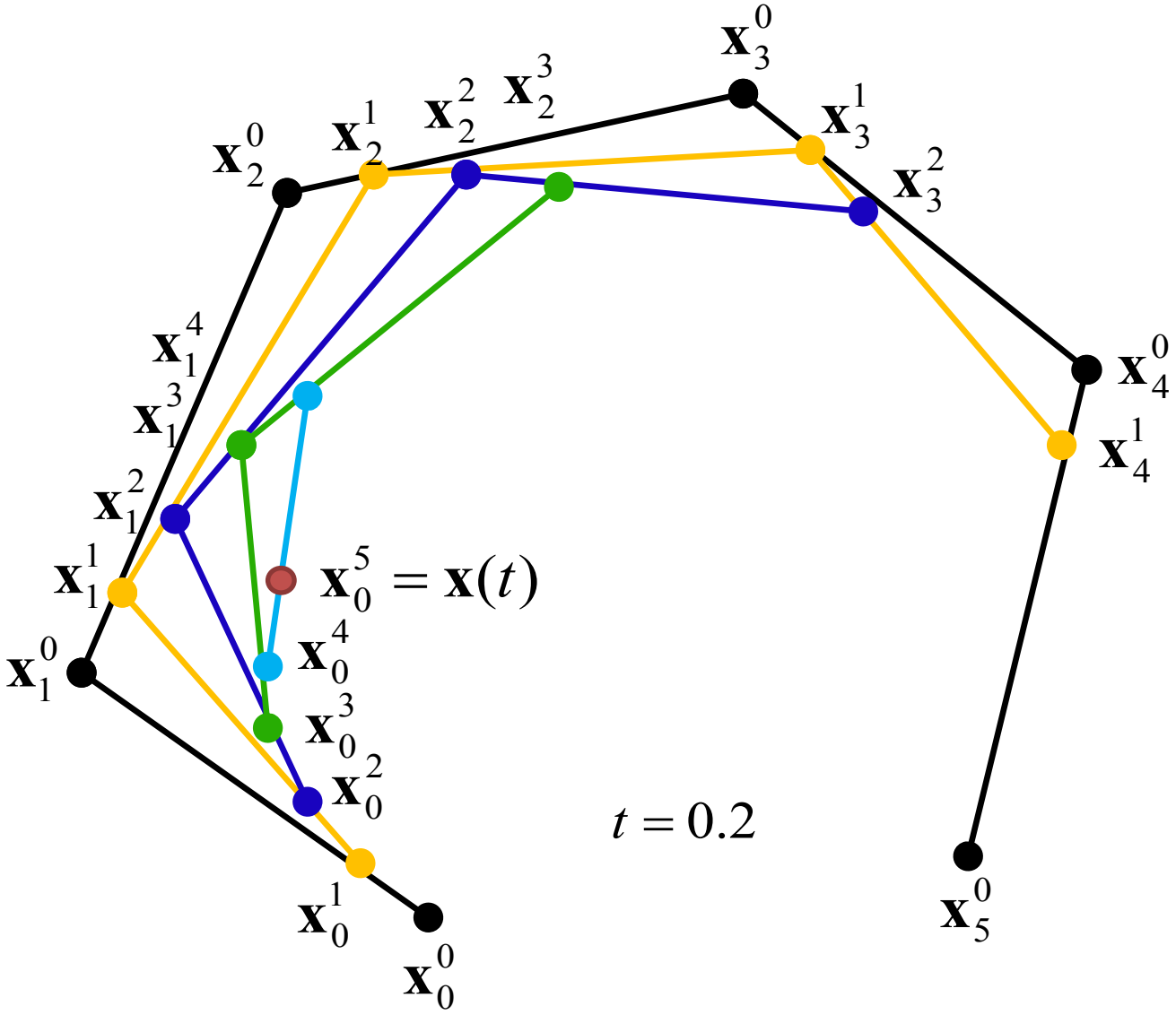
The idea of de Casteljau's algorithm goes as follows. Suppose we want to find $\mathbf{x}(t)$, where t is in [0,1]. Starting with the first polyline (set of connected line segments), $\mathbf{x_0^0} - \mathbf{x_1^0} - \cdots - \mathbf{x_n^0}$, use the above formula to find a point $\mathbf{x_k^1}$ on the leg (*i.e.* line segment) from $\mathbf{x_k^0}$ to $\mathbf{x_{k+1}^0}$ that divides the line segment connecting $\mathbf{x_k^0}$ and $\mathbf{x_{k+1}^0}$ in a ratio of $t\colon 1-t$. In this way, we will obtain n points $\mathbf{x_0^1}, \mathbf{x_1^1}, \dots, \mathbf{x_{n-1}^1}$,. They define a new polyline of n - 1 legs.

Apply the procedure to this new polyline, i.e. $\mathbf{x_0^1} - \mathbf{x_1^1} - \cdots - \mathbf{x_{n-1}^1}$ and we shall get a second polyline of n - 1 points $\mathbf{x_0^2}, \mathbf{x_1^2}, \dots, \mathbf{x_{n-2}^2}$ and n - 2 legs. Starting with this polyline, we can construct a third one of n - 2 points $\mathbf{x_0^3}, \mathbf{x_1^3}, \dots, \mathbf{x_{n-3}^3}$ and n - 3 legs. Repeating this process n times yields a single point $\mathbf{x_0^n}$. De Casteljau proved that this is the point $\mathbf{x}(t)$ on the curve that corresponds to t.

Hence we can formally define:

$$\mathbf{x}_\mathbf{k}^\mathbf{r}(t) = (1-t)\mathbf{x}_\mathbf{k}^{\mathbf{r-1}}(t) + t\mathbf{x}_{\mathbf{k+1}}^{\mathbf{r-1}}(t)$$

with $r = 1,2,\ldots,n,\ k = 0,1,\ldots,n-r$ and $\mathbf{x}_\mathbf{k}^\mathbf{0} = \mathbf{x}_\mathbf{k}$. The Bézier curve is then given by $\mathbf{x}(t) = \mathbf{x}_\mathbf{0}^\mathbf{n}(t)$.



Recursive Subdivision

Although de Casteljau's method can be numerically stable, we still do not know how many points the Bézier curve needs to be evaluated to obtain acceptable results. Next we will discuss how Bézier curves can be subdivided until a desired level of accuracy is obtained.

We now investigate a technique that allows us to break Bézier curves down into sub-portions until we have a curve that can be approximated by a line for the desired level of accuracy.

If we have a Bézier curve $\mathbf{x}(t)$ of degree n, then $\mathbf{x_1}(t) = \mathbf{x}\left(\frac{t}{2}\right)$ and $\mathbf{x_2}(t) = \mathbf{x}\left(\frac{t+1}{2}\right)$ are both Bézier curves of degree n. We have then successfully divided the curve into two curves.

Drawing a Bézier curve can now be achieved by recursively subdividing the Bézier curve into two until the portions of the Bézier curves are as close to a straight line as we need.

One way to determine if the curve is close to a straight line, is to define an error value δ. If

$$\left\|\mathbf{x}\left(\frac{1}{2}\right) - \frac{1}{2}(\mathbf{x_0} + \mathbf{x_n})\right\| < \delta$$

Then we assume the distance from the curve to the line joining the curve endpoints will be less than δ, and can thus be approximated by a straight line segment. This test may fail on occasion, another test can be used is to determine how far the interior points of the control polygon are from the line connecting the first and last control points.

Theorem 6: *Let* $\mathbf{x_1}(t) = \mathbf{x}(t_0 t)$ *and* $\mathbf{x_2}(t) = \mathbf{x}(t_0 + (1 - t_0)t)$, *where* $t_0 \in [0,1]$ *determines the parameter value at which the curve should be split. If* $t_0 = \frac{1}{2}$ *then the curve is split in two in such a way that the parameter values are divided equally between the curves. If the curve is defined over* $t \in [0,1]$, *then the subdivided curve* $\mathbf{x_1}(t)$ *will be* $\mathbf{x}(t)$ *defined over* $t \in [0, t_0]$ *and* $\mathbf{x_2}(t)$ *will be* $\mathbf{x}(t)$ *defined over* $t \in [t_0, 1]$.

(a) The curve $\mathbf{x_1}(t)$ *is equal to the degree* n *Bézier curve with control points* $\mathbf{x_0^0}, \mathbf{x_0^1}, \mathbf{x_0^2}, \dots, \mathbf{x_0^n}$, *that is*

$$\mathbf{x_1}(t) = \mathbf{x}(t_0 t) = \sum_{k=0}^{n} B_k^n(t)\mathbf{x_0^k}(t_0)$$

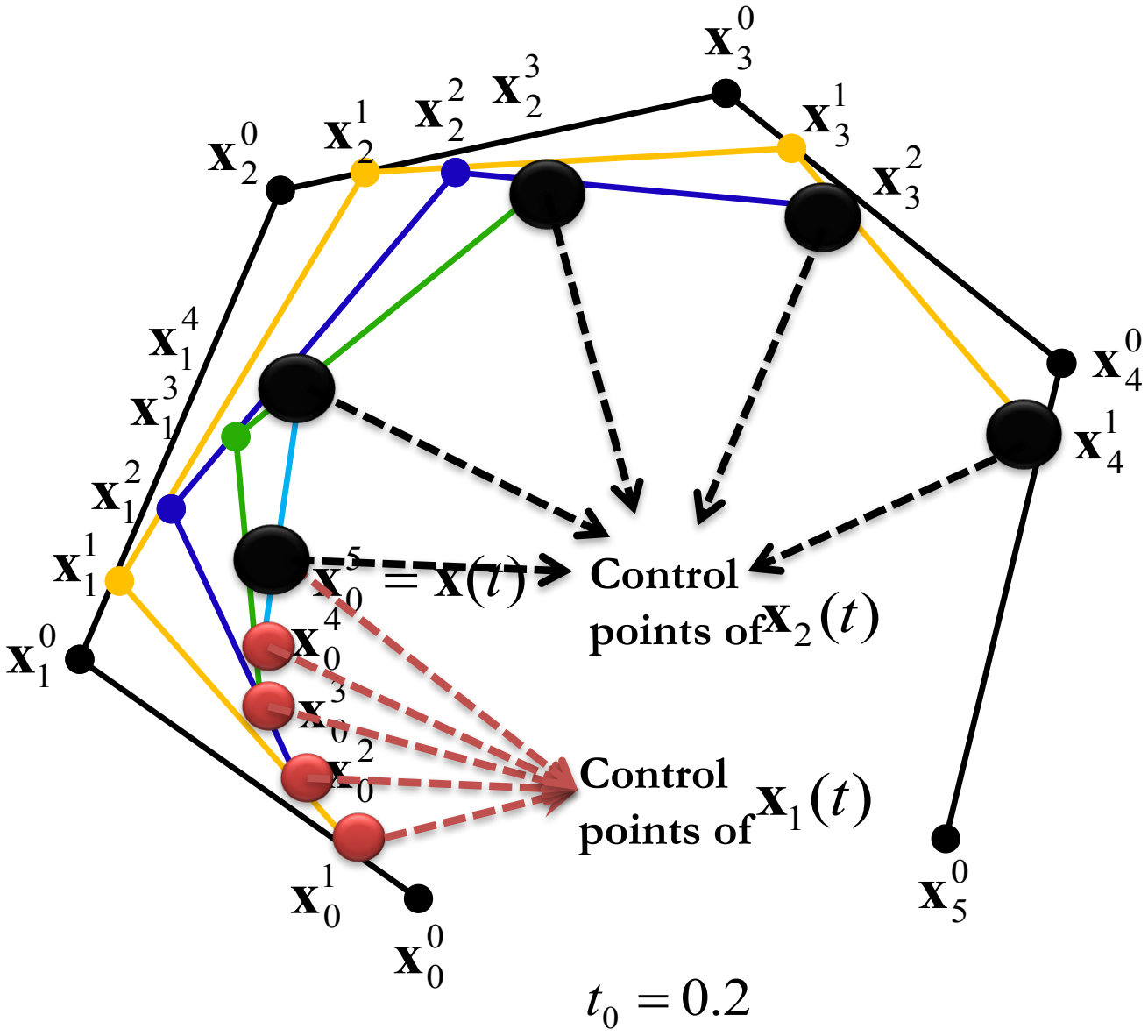
(b) The curve $\mathbf{x_2}(t)$ *is equal to the degree* n *Bézier curve with control points* $\mathbf{x_0^n}, \mathbf{x_1^{n-1}}, \mathbf{x_2^{n-2}}, \dots, \mathbf{x_n^0}$, *that is*

$$\mathbf{x_2}(t) = \mathbf{x}(t_0 + (1 - t_0)t) = \sum_{k=0}^{n} B_k^n(t)\mathbf{x_k^{n-k}}(t_0)$$

Where

$$\mathbf{x_k^r}(t) = (1 - t)\mathbf{x_k^{r-1}}(t) + t\mathbf{x_{k+1}^{r-1}}(t)$$

with $r = 1,2, \dots, n,\ k = 0,1, \dots, n - r.$


$\mathbf{x}_3^0$
$\mathbf{x}_2^3$
$\mathbf{x}_2^2$
$\mathbf{x}_2^1$
$\mathbf{x}_2^0$
$\mathbf{x}_3^1$
$\mathbf{x}_3^2$
$\mathbf{x}_1^4$
$\mathbf{x}_1^3$
$\mathbf{x}_1^2$
$\mathbf{x}_1^1$
$\mathbf{x}_1^0$
$\mathbf{x}_4^0$
$\mathbf{x}_4^1$
$\mathbf{x}_0^5 \equiv \mathbf{x}(t)$
Control points of $\mathbf{x}_2(t)$
$\mathbf{x}_0^4$
$\mathbf{x}_0^3$
$\mathbf{x}_0^2$
Control points of $\mathbf{x}_1(t)$
$\mathbf{x}_0^1$
$\mathbf{x}_0^0$
$\mathbf{x}_5^0$
$t_0 = 0.2$

Rational Bézier Curves

A Bézier curve is rational if its control points are specified by homogeneous coordinates. To do so we embed the vector space $\mathbb{R}^3$ in the vector space $\mathbb{R}^4$ to create the homogeneous coordinate system.

Definition 25: *If* $x, y, z, w \in \mathbb{R}$ and $w \neq 0$, *then* $(x, y, z, w)^{\mathrm{T}}$ *is a homogeneous coordinate representation of the point* $\left(\frac{x}{w}, \frac{y}{w}, \frac{z}{w}\right)^{\mathrm{T}}$.

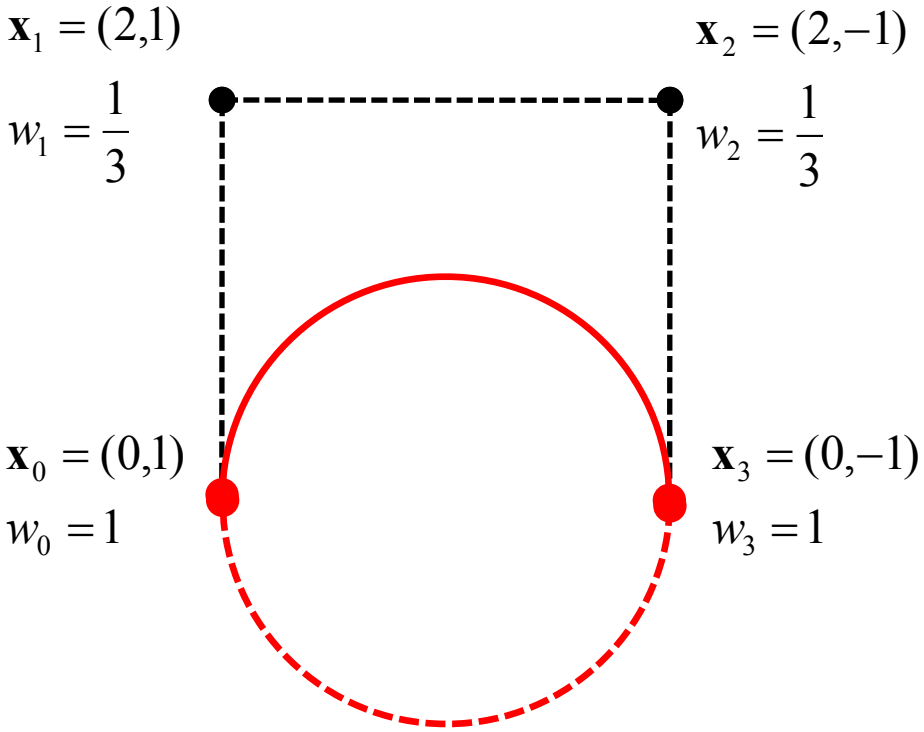
Definition 26: *If we define the Bézier curve in terms of homogeneous coordinates, then each control point* $\mathbf{x_k}$ *becomes* $(w_k \mathbf{x_k} \quad w_k)^{\mathrm{T}}$. *The Bézier is then given by:*

$$\mathbf{x}(t) = \sum_{k=0}^{n} B_k^n(t)(w_k \mathbf{x_k} \quad w_k)^{\mathrm{T}}$$

The point $\mathbf{x}(t)$ *is also a homogenous coordinate. The point that* $\mathbf{x}(t)$ *represented is given by;*

$$\frac{\sum_{k=0}^{n} w_k B_k^n(t) \mathbf{x_k}}{\sum_{k=0}^{n} w_k B_k^n(t)}$$

The w_k terms act as weighting factors, and specify how much influence a point has on the curve. These extra weighting factors allow us to draw conic sections which cannot be drawn using conventional Bézier curves.



Section of a circle, drawn with a rational Bézier curve

Surfaces

Definition 39: *A* **surface** *is a set of points, usually in* $\mathbb{R}^3$*, such that the surface can be described by a smooth function. A surface in* $\mathbb{R}^2$ *is called a* **plane surface**, *while a surface in* $\mathbb{R}^3$ *is simply denoted a surface, we call a surface in* $\mathbb{R}^d$ *where* $d > 3$ *a* **hypersurface**. *The function that we use to describe surfaces in* $\mathbb{R}^3$ *is often of the form,*

$$\mathbf{f}(u,v) = \begin{pmatrix} x(u,v) \\ y(u,v) \\ z(u,v) \end{pmatrix}$$

The surface is thus described by two parameters u *and* v*, we thus call the surface a* **parametric surface**.

An example of such parametric surface is the *sphere* where

$$\mathbf{f}(u,v) = \begin{pmatrix} \sin u \cos v \\ \sin u \sin v \\ \cos u \end{pmatrix}$$

With $0 \leq u \leq \pi$ and $0 \leq v \leq 2\pi$.

Another example of a parametric surface is the *torus* where

$$\mathbf{f}(u,v) = \begin{pmatrix} (r_1 + r_2 \cos v) \cos u \\ (r_1 + r_2 \cos v) \sin u \\ r_2 \sin v \end{pmatrix}$$

With $0 \leq u \leq 2\pi$, $0 \leq v \leq 2\pi$ and $r_2 > r_1$

Definition 40: Isocurves *are curves on a parametric surface with constant* u *or* v.

Tensor Product Surfaces

Definition 41: *Suppose we define a parametric curve with control points* $\mathbf{x}_k$ *such that,*

$$\mathbf{q}(v) = \sum_{k=1}^{n} f_k^n(v)\mathbf{x}_k$$

Where $f_k^n(v)$ *are the basis functions of the curve, now suppose that we have defined* m *such curves* $\mathbf{q}_j(v), j = 1,2,\ldots,m$ *such that* $\mathbf{q}_j(v)$ *is defined by control points* $\mathbf{x}_{j,k}$*. Then each curve can be defined as,*

$$\mathbf{q}_j\ (v) = \sum_{k=1}^{n} f_k^n(v)\mathbf{x}_{j,k}$$

If we choose a particular value for v *on each of these curves, then we end up with* m *point which we can use them to create another curve* $\mathbf{p}$*. The curve* $\mathbf{p}$ *is then defined by*

$$\mathbf{p}(u,v) = \sum_{j=1}^{m} f_j^m(u)\mathbf{q}_j\ (v) = \sum_{j=1}^{m} f_j^m(u) \sum_{k=1}^{n} f_k^n(v)\mathbf{x}_{j,k}$$

The surface defined by all such curves is known as a **tensor product surface** *and the curve of constant* v *is known as an* **isocurve**. *So we can simply write the surface as:*

$$\mathbf{p}(u,v) = \sum_{j=1}^{m}\sum_{k=1}^{n} f_j^m(u)f_k^n(v)\mathbf{x}_{j,k}$$

Tensor Product Bézier Surfaces

Definition 42: *The* **tensor product Bézier surface (Bézier patch)** *is given by,*

$$\mathbf{p}(u,v) = \sum_{j=0}^{m} \sum_{k=0}^{n} B_j^m(u) B_k^n(v) \mathbf{x}_{j,k}$$

Where B_k^n *are the Bernstein polynomials,*

$$B_k^n(v) = \binom{n}{k} v^k (1-v)^{n-k}$$

Definition 43: *The* **normal** *to the Bézier surface is given by*

$$\mathbf{n}(u,v) = \frac{\partial \mathbf{p}(u,v)}{\partial u} \times \frac{\partial \mathbf{p}(u,v)}{\partial v}$$

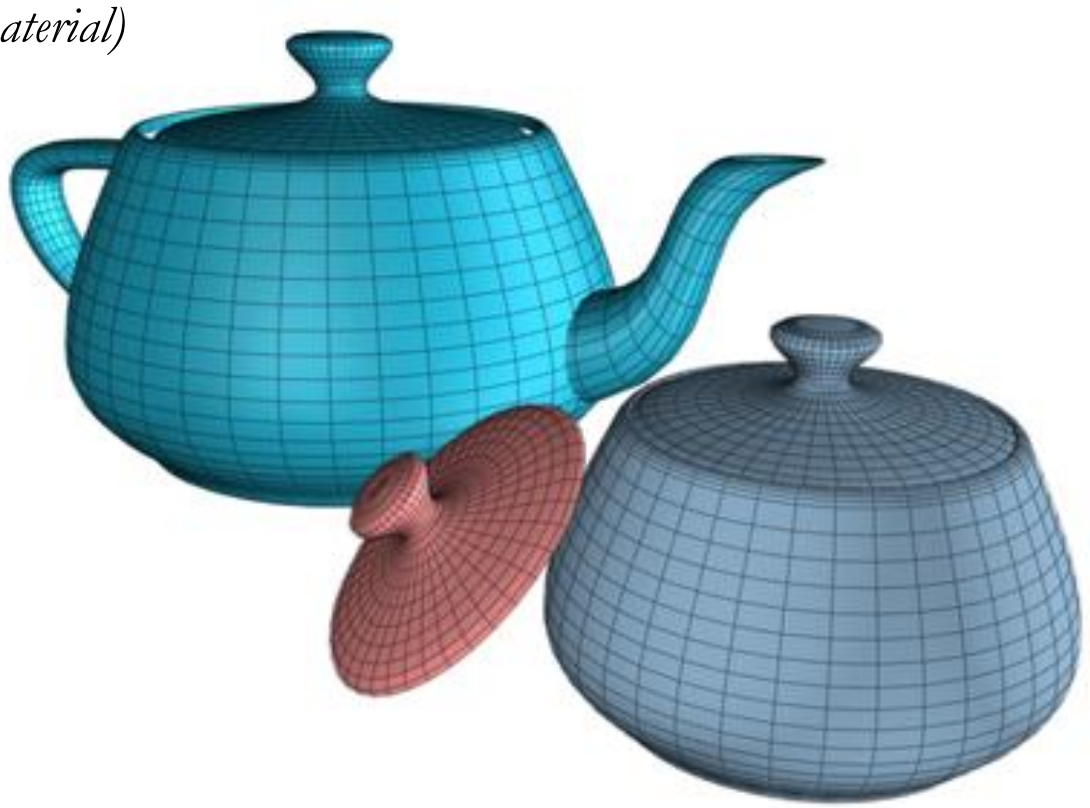
Where $\times$ *denotes the cross product and the derivatives are given by,*

$$\frac{\partial}{\partial u}\mathbf{p}(u,v) = m \sum_{j=0}^{m-1} \sum_{k=0}^{n} B_j^{m-1}(u) B_k^n(v) \left(\mathbf{x}_{j+1,k} - \mathbf{x}_{j,k}\right)$$

$$\frac{\partial}{\partial v}\mathbf{p}(u,v) = n \sum_{j=0}^{m} \sum_{k=0}^{n-1} B_j^m(u) B_k^{n-1}(v) \left(\mathbf{x}_{j,k+1} - \mathbf{x}_{j,k}\right)$$

Piecewise Continuous Bézier Surfaces

- Using the normals, we can define piece-wise continuous Bézier surfaces, where de Casteljau's algorithm can be applied to efficiently render these surfaces. (*refer to the reading material)*



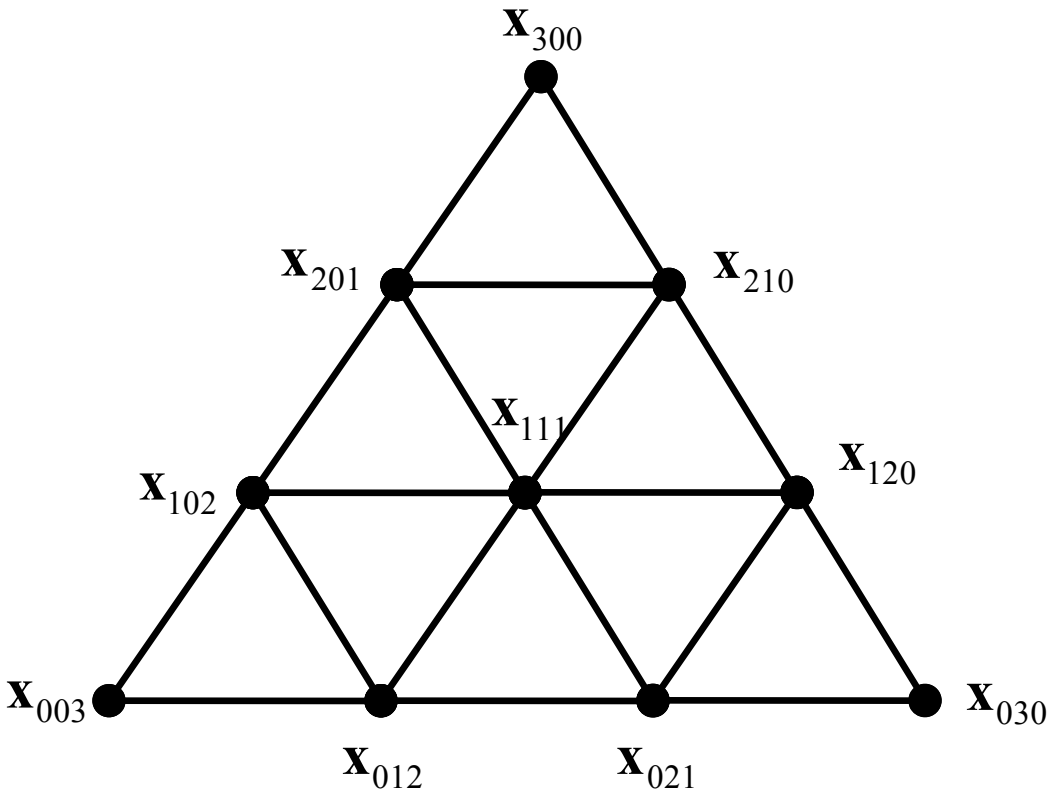
Triangular Bézier Surfaces

Instead of arranging the initial configuration of the control points in a rectangular fashion as with tensor product Bézier surfaces, we can arrange the control points in a triangle (see the following figure). If the degree of the Bézier surface is n, then there are $n+1$ control points on each side.

The control points $\mathbf{x}_{i,j,k}$ with $i+j+k=n$ and $i,j,k \geq 0$. A surface with degree n has total number of control points computed as;

$$\sum_{k=1}^{n+1} k = 1+2+3+\cdots+(n-1)+n+(n+1) = \frac{(n+1)(n+2)}{2}$$

Bézier triangles can be efficiently rendered using a variation of the de Casteljau's method, however instead of the usual parameters, we should used *barycentric coordinates*.



Definition 44: *A triangle in* $\mathbb{R}^3$ *with vertices* $\mathbf{x_0}$, $\mathbf{x_1}$ *and* $\mathbf{x_2}$ *can be described with* **barycentric coordinates** *as,*

$$\mathbf{p}(u,v) = \mathbf{x_0} + u(\mathbf{x_1} - \mathbf{x_0}) + v(\mathbf{x_2} - \mathbf{x_0}) = (1-u-v)\mathbf{x_0} + u\mathbf{x_1} + v\mathbf{x_2}$$

Where $u, v \geq 0$ *and* $u + v \leq 1$.

Definition 45: *The* **Bézier triangle**, *in terms of Bernstein polynomials, can be written as,*

$$\mathbf{p}(u,v) = \sum_{\substack{i,j,k=0 \\ i+j+k=n}}^{n} B^n_{i,j,k}(u,v)\mathbf{p}_{ijk}$$

With the Bernstein polynomials defined by,

$$B^n_{i,j,k}(u,v) = \frac{n!}{i!\,j!\,k!}u^i v^j (1-u-v)^k \quad , \qquad i+j+k=n$$

The partial derivatives are then given by,

$$\frac{\partial}{\partial u}\mathbf{p}(u,v) = \sum_{i+j+k=n-1} B^{n-1}_{i,j,k}(u,v)\mathbf{p}_{i+1,j,k}$$

$$\frac{\partial}{\partial v}\mathbf{p}(u,v) = \sum_{i+j+k=n-1} B^{n-1}_{i,j,k}(u,v)\mathbf{p}_{i,j+1,k}$$

The Bézier triangles are affine invariant, the surface interpolates the three corner control points and the surface lies within the convex hull of the control points.

Rational Bézier Surfaces

Definition 46: *The tensor product Bézier surfaces can be extended to* **rational Bézier surfaces** *to obtain further control of the surface which is then given by*

$$\mathbf{p}(u,v) = \frac{\sum_{j=0}^{m}\sum_{k=0}^{n} B_j^m(u)B_k^n(v)w_{j,k}\mathbf{x}_{j,k}}{\sum_{j=0}^{m}\sum_{k=0}^{n} B_j^m(u)B_k^n(v)w_{j,k}}$$

Where the control points are specified in the homogenous form as $\begin{pmatrix} w_{j,k}\mathbf{x}_{j,k} & w_{j,k} \end{pmatrix}^{\mathrm{T}}$. *The* $w_{j,k}$ *weights thus determine a degree of importance of each control point.*

Subdivision Surfaces

We saw before how to subdivide a Bézier curve using de Casteljau's method. With sufficient subdivision, the Bézier curve segments can be approximated by lines.

If we proceed to subdivide an infinite number of times, the limit curve is the Bézier curve.

However, this is not the only way to subdivide a curve, we can recursively introduce new control points as a function of the given control points to produce smoother curves, different subdivision methods apply different subdivision rules, i.e. how to obtain new points in terms of the points at the current iteration.

In the same way that curves can be recursively subdivided, we can subdivide a surface control mesh. The tensor product Bézier surface can be modeled in this manner, but subdivision is not limited to rectangular meshes, triangular ones can also be subdivided.

Definition 47: Subdivision curves and surfaces *is a two phase process. The* **refinement phase** *creates new vertices and new polygons from the control mesh, and the* **smoothing phase** *computes new positions for some or all of the vertices in the new control mesh.*

Definition 48: *A* **stationary** *subdivision scheme is the one that uses the same subdivision rules at every subdivision step. A* **non-stationary** *scheme changes the rules according to the subdivision step. A* **uniform** *scheme uses the same rules for each vertex or edge, while* **non-uniform** *one may depend on the valence/ neighborhood of the vertex.*

Definition 49: *The* **valence** *of a vertex* **x** *on the subdivision surface is the number of neighboring vertices, that is the number of vertices connected to* **x** *by an edge.*

Definition 50: *A vertex with valence 6 is known as* **regular** *or* **ordinary** *vertex, otherwise it is known as* **irregular** *or* **extraordinary** *one.*

Loop Subdivision

Loop's subdivision scheme works on triangular meshes. The surface is an approximating surface, that is it doesn't interpolate (passes through) its control points. A new vertex is added for each edge, at each subdivision step.

Loop's subdivision scheme can be summarized as follows;

(1) At each subdivision step k, the existing point $\mathbf{x}^k$ is updated with the scheme,

$$\mathbf{x}^{k+1} = (1 - n\beta)\mathbf{x}^k + \beta \sum_{i=1}^{n} \mathbf{y}_i^k$$

Where $\mathbf{y}_i^k$ are the n neighboring vertices and β is a constant determined by n. Loop suggests the function,

$$\beta(n) = \frac{1}{n}\left(\frac{5}{8} - \frac{\left(3 + 2\cos\left(\frac{2\pi}{n}\right)\right)^2}{64}\right)$$

While there is another alternative definition,

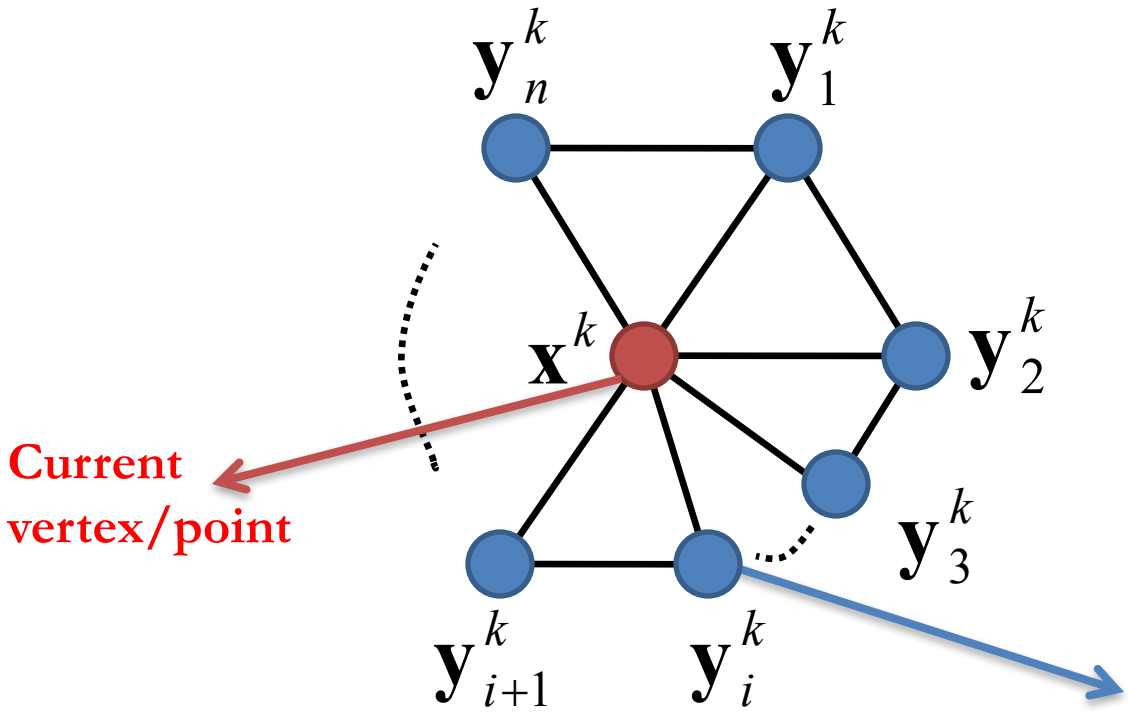
$$\beta(n) = \frac{3}{n(n+2)}$$

In both cases the surface is C^2 at vertices of regular valence and C^1 otherwise.

$$\mathbf{x}^{k+1} = (1 - n\beta)\mathbf{x}^k + \beta \sum_{i=1}^{n} \mathbf{y}_i^k$$

Where $\mathbf{y}_i^k$ are the n neighboring vertices and β is a constant determined by n. Loop suggests the function,

$$\beta(n) = \frac{1}{n}\left(\frac{5}{8} - \frac{\left(3 + 2\cos\left(\frac{2\pi}{n}\right)\right)^2}{64}\right)$$

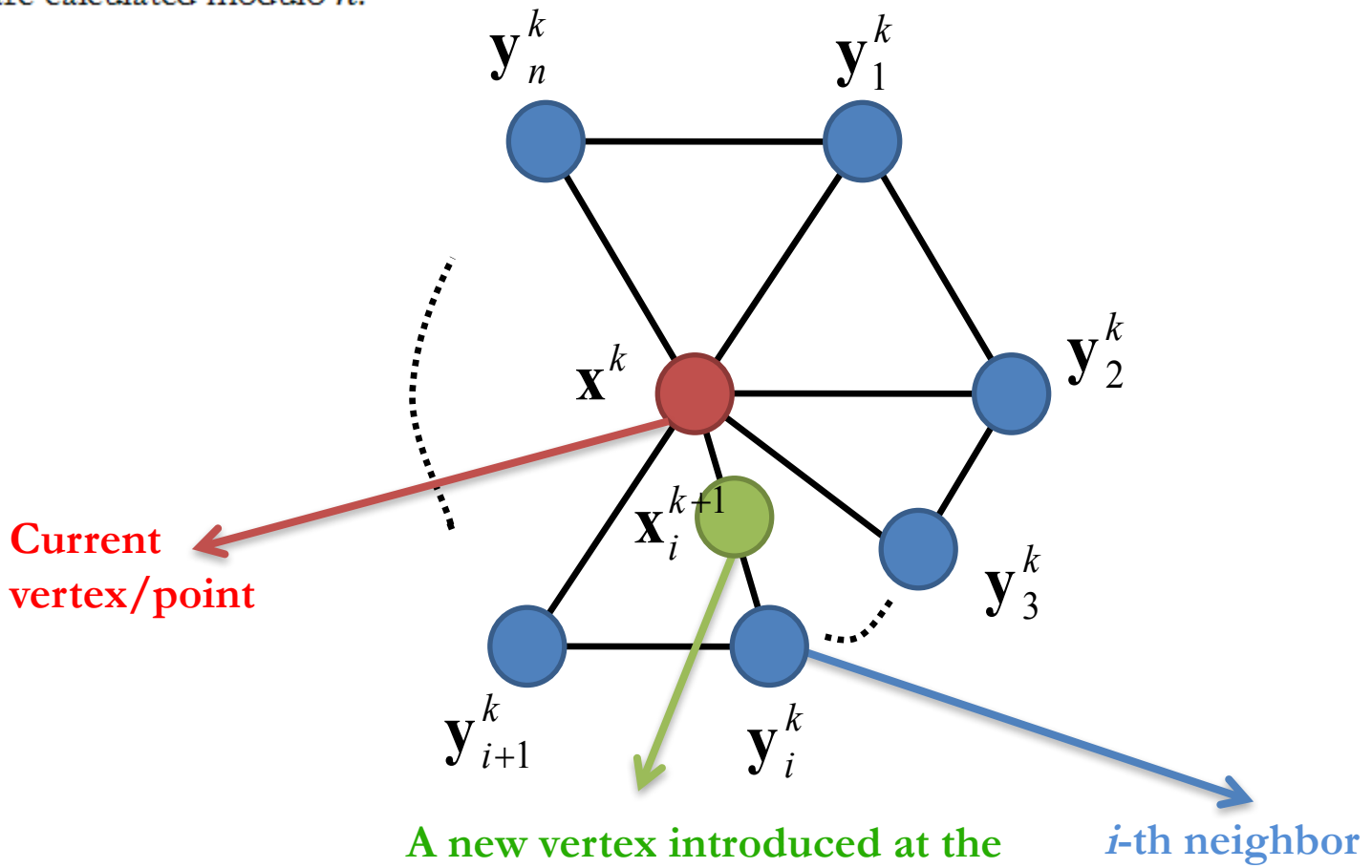


Neighborhood/valence of the current point

(2) For each edge connecting $\mathbf{x}^k$ to a neighbor, a new vertex is created via,

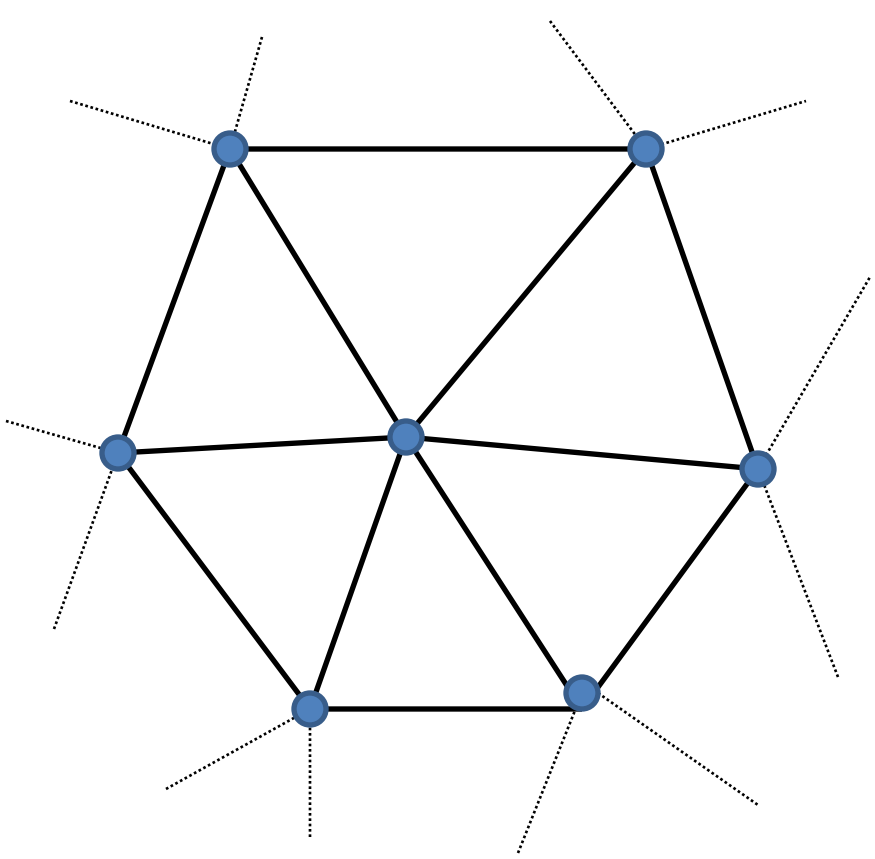
$$\mathbf{x}_i^{k+1} = \frac{3\mathbf{x}^k + 3\mathbf{y}_i^k + \mathbf{y}_{i-1}^k + \mathbf{y}_{i+1}^k}{8} \quad , \qquad i = 1,2,\ldots,n$$

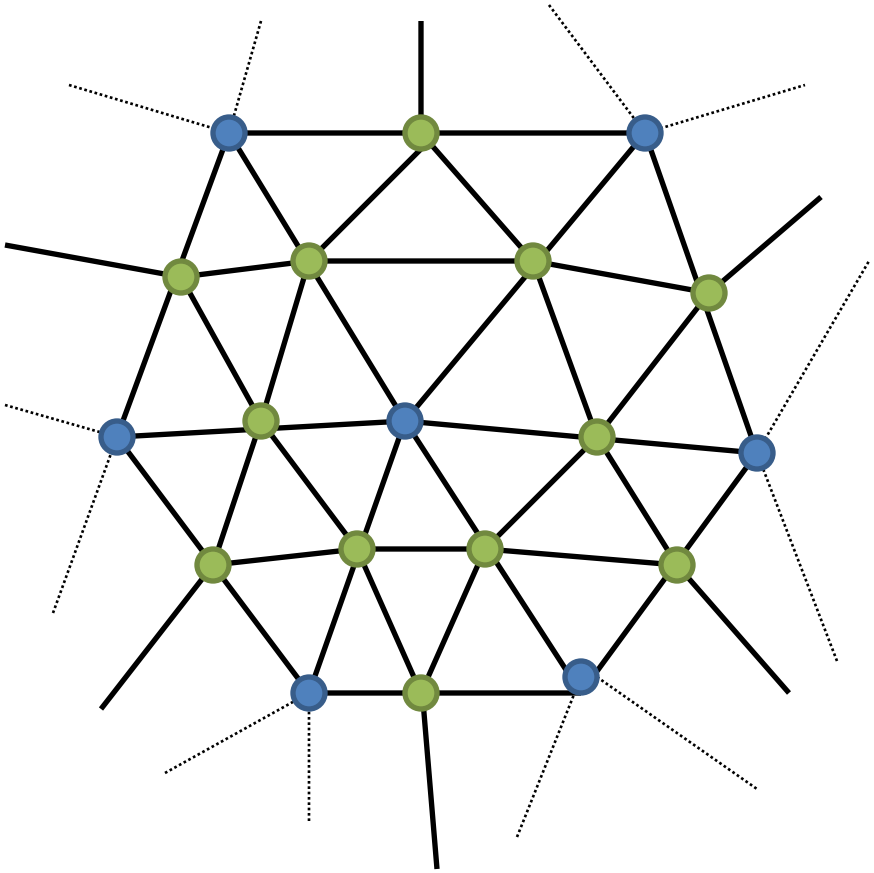
The subscripts i are calculated modulo n.



New vertex introduced on the edges connecting neighboring vertices

(3) Once the new vertices have been determined, each triangle is subdivided into four triangles.

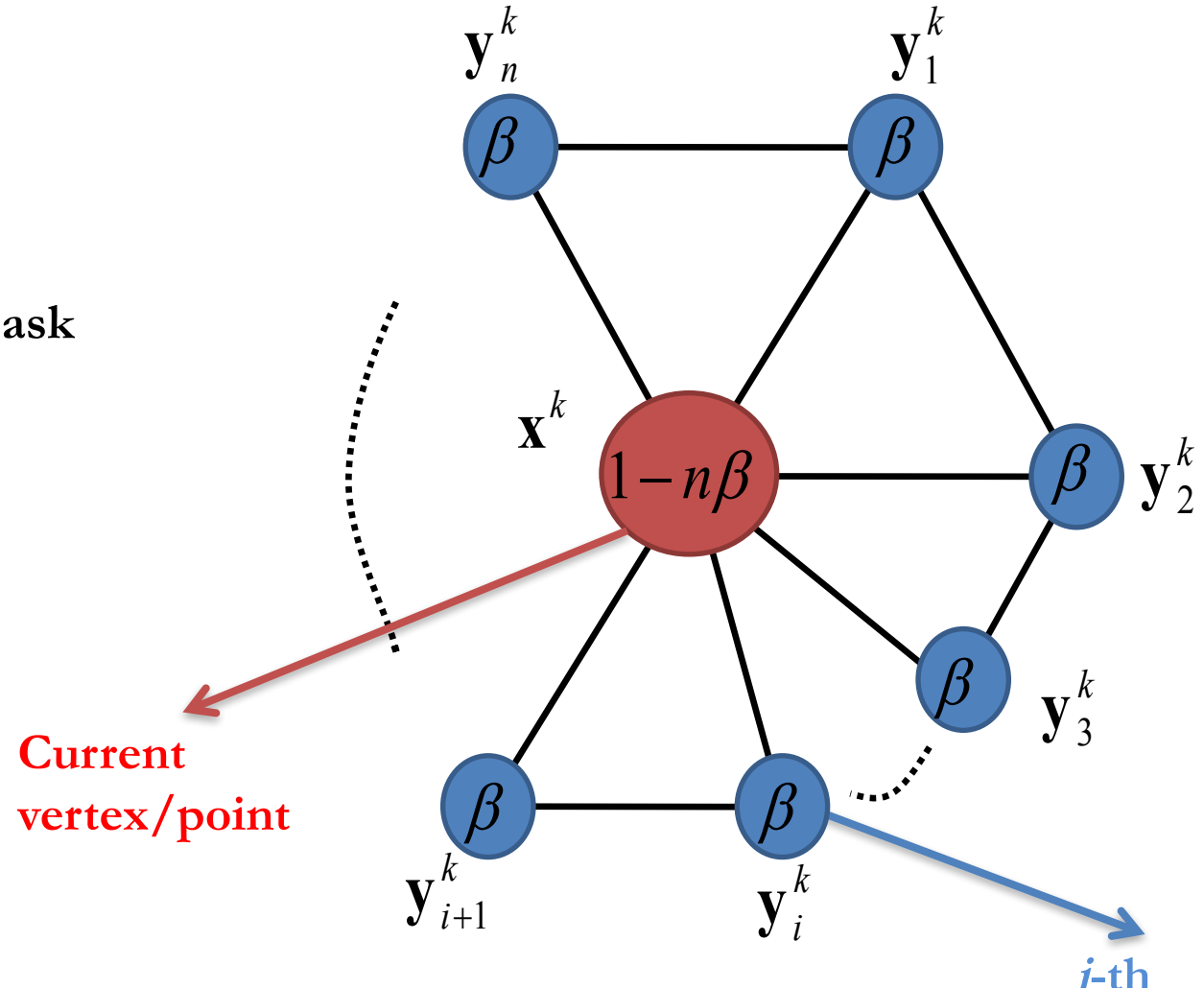




Often a mask or stencil is used to visualize the subdivision, the entries in the mask are the weights for the contributing points. Noting that the sum of the weights is one.

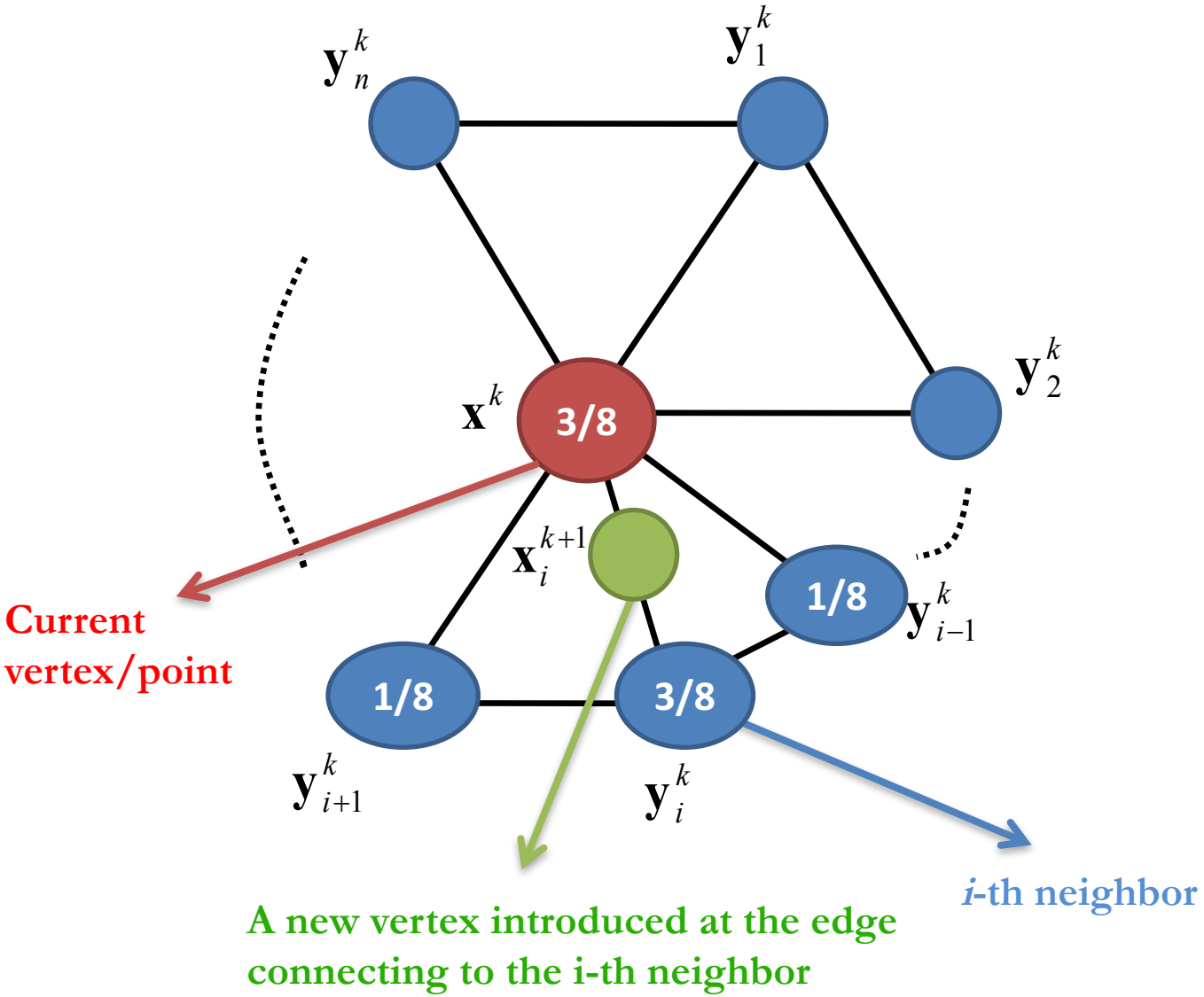
There are two types of masks, the first one is called the vertex mask which holds the weights of the current point $\mathbf{x}^k$ and its neighboring vertices $\mathbf{y}_i^k$ used to update the current point to obtain $\mathbf{x}^{k+1}$.

Vertex Mask

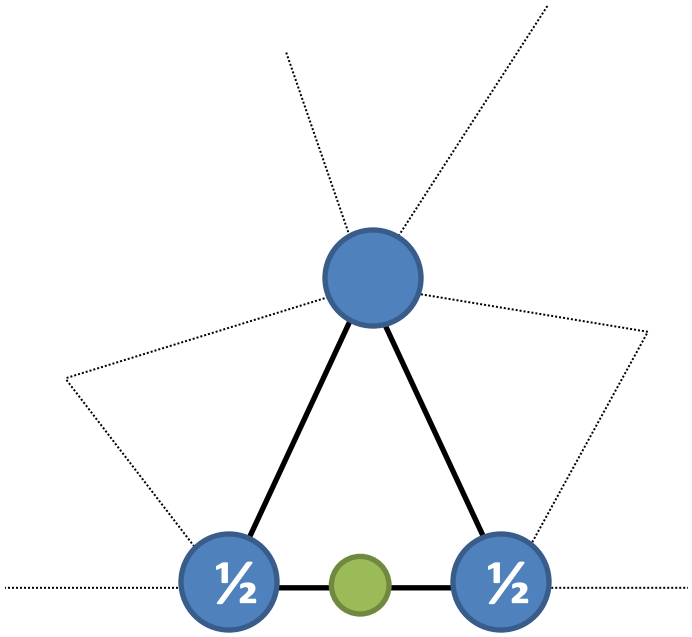


The second type of masks is the edge mask, which defines the weights of the four neighboring vertices used to generate a new vertex $\mathbf{x}_i^{k+1}$ on the edge connecting the current point $\mathbf{x}^k$ and its i-th neighbor $\mathbf{y}_i^k$.

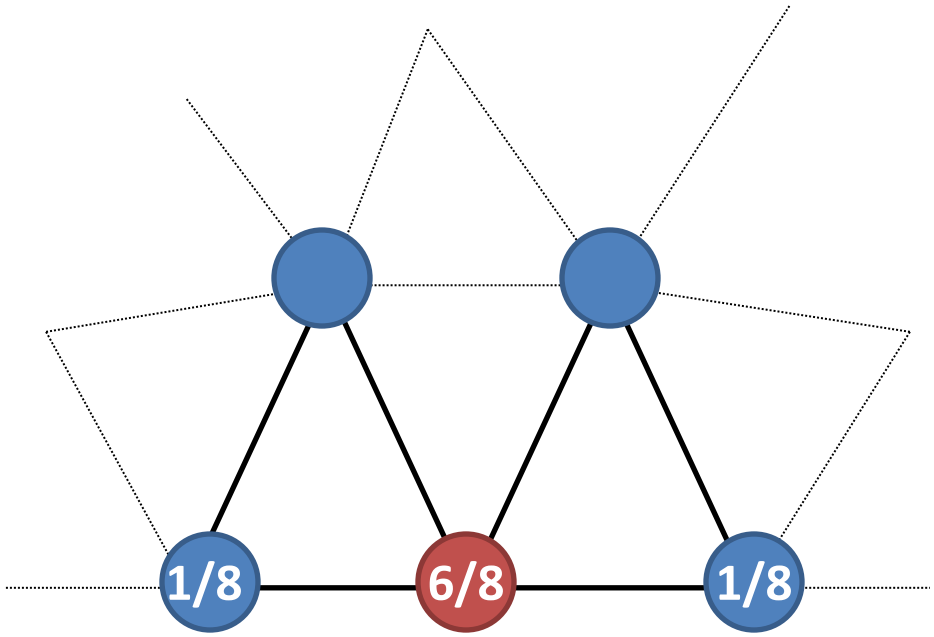
Edge Mask



The scheme listed so far can work on closed surfaces however it needs some adjustments to handle open surfaces. Edges and vertices on the boundary of an open surface have different masks



Edge mask



Vertex mask

Thank You!!!