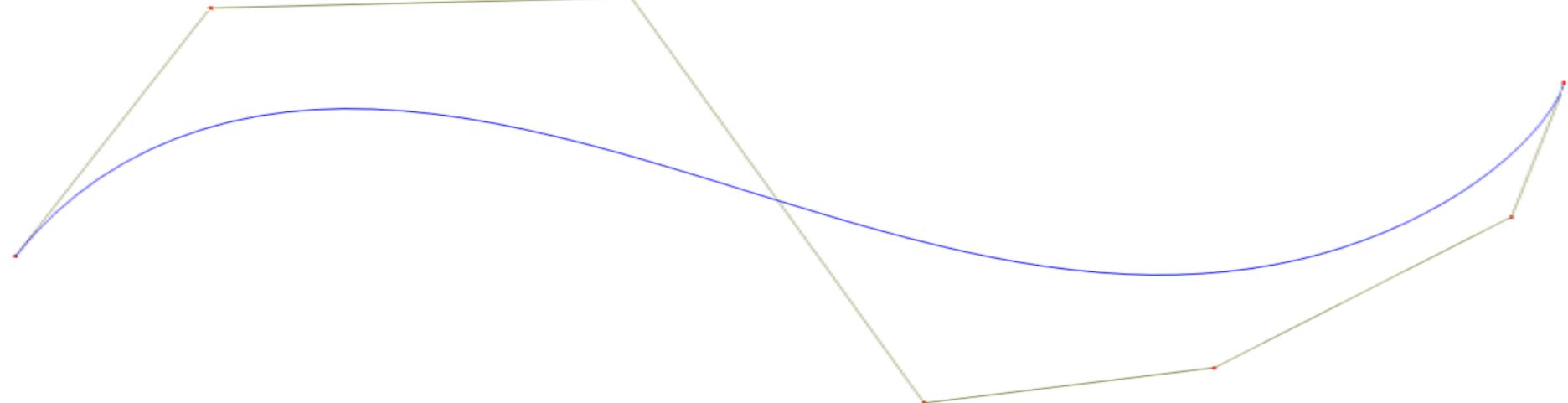


Modélisation et Programmation 3D

Courbes paramétriques



Cours du 28/01/2019

Plan

- Introduction
- Représentation d'une droite et d'une courbe
- Géométrie différentielle
- Cubiques
- Courbes de Bézier
- Courbes B-Splines / NURBS
- Dessin en OpenGL

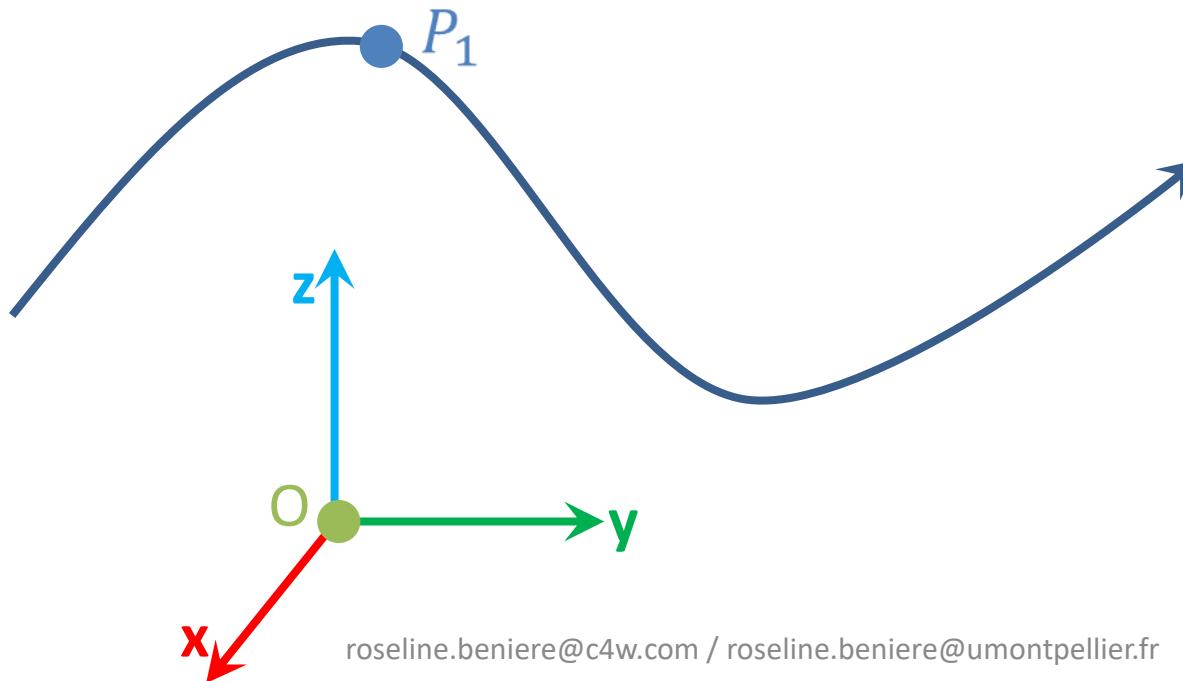
Plan

- Introduction
- Représentation d'une droite et d'une courbe
- Géométrie différentielle
- Cubiques
- Courbes de Bézier
- Courbes B-Splines / NURBS
- Dessin en OpenGL

Introduction

- Courbes 3D :

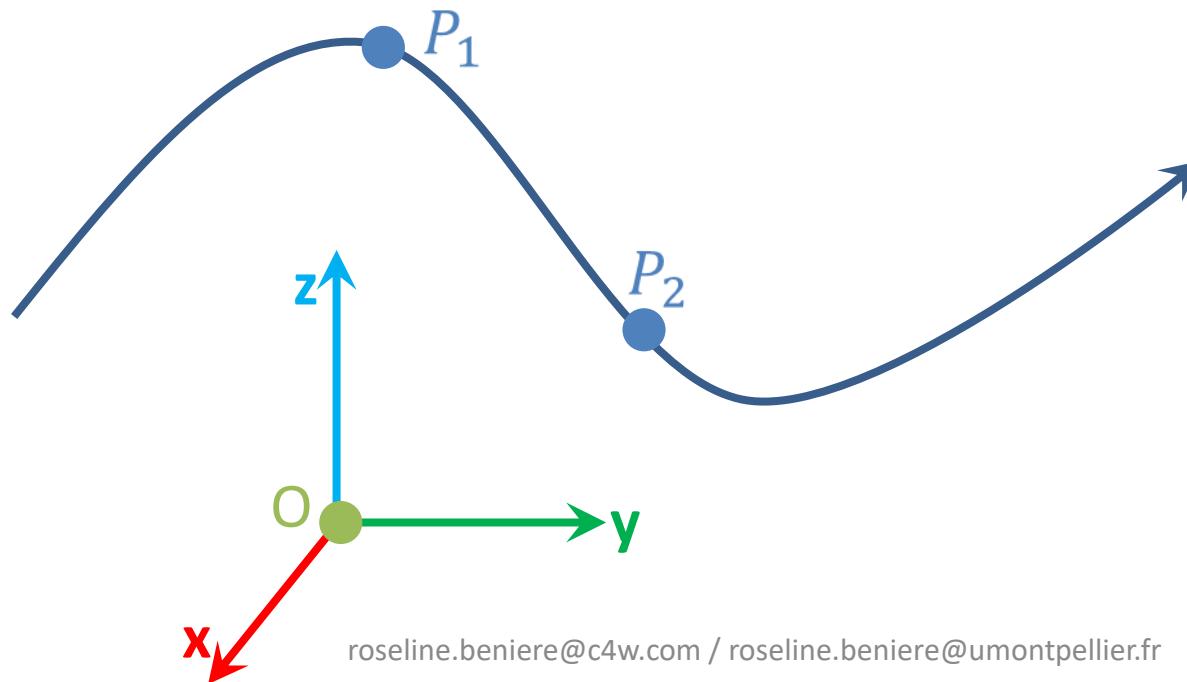
- peut être vue comme le ``déplacement'' d'un point dans l'espace,



Introduction

- Courbes 3D :

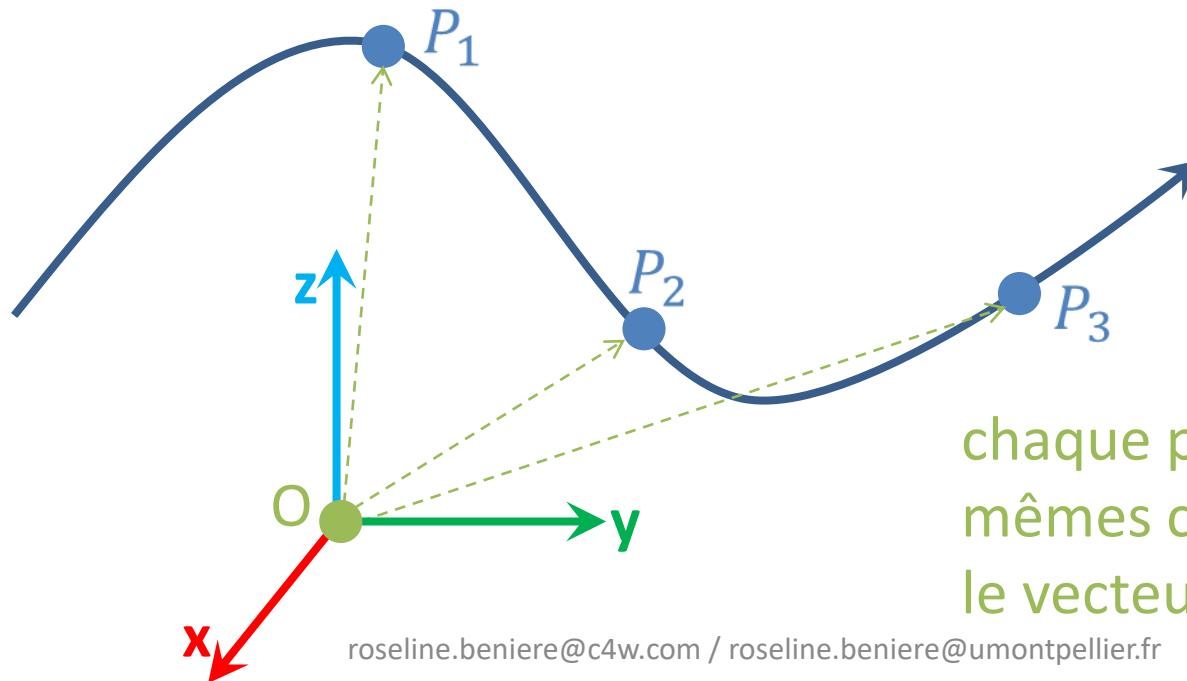
- peut être vue comme le ``déplacement'' d'un point dans l'espace,



Introduction

- Courbes 3D :

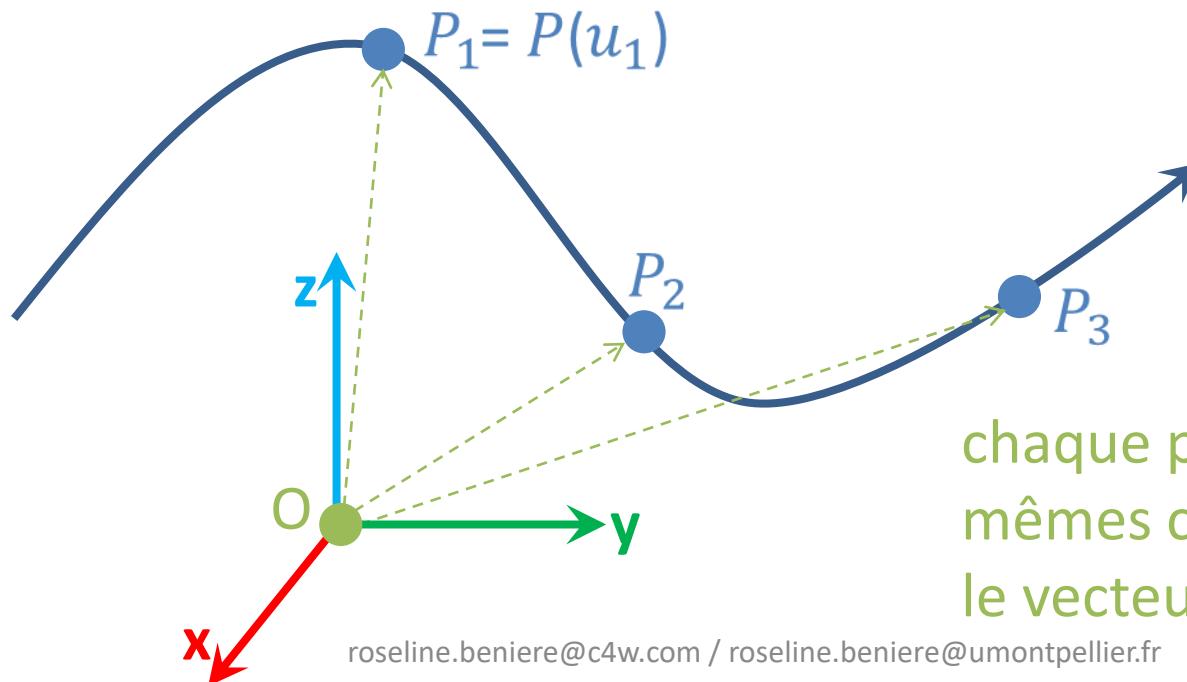
- peut être vue comme le ``déplacement'' d'un point dans l'espace,



chaque point P a les mêmes coordonnées que le vecteur \vec{OP}

Introduction

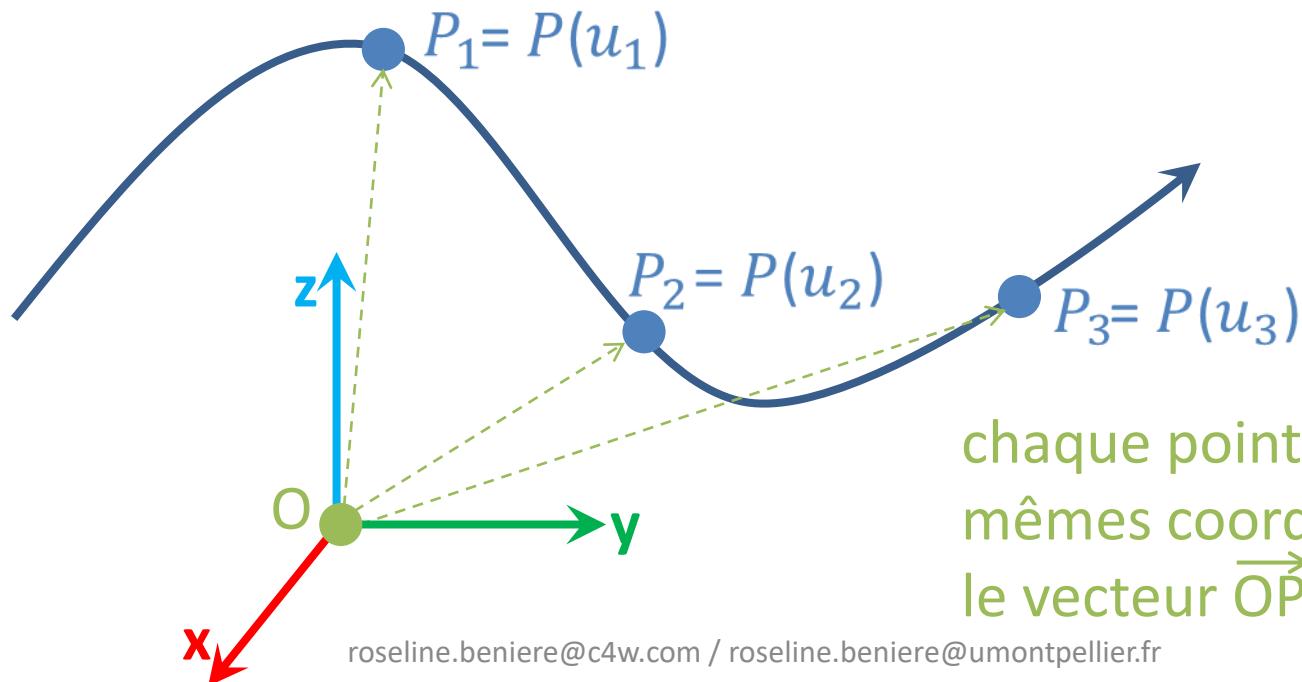
- Courbes 3D :
 - peut être vue comme le ``déplacement'' d'un point dans l'espace,
 - défini par un paramètre : un scalaire u qui représente la position du point le long de la courbe.



chaque point P a les mêmes coordonnées que le vecteur \vec{OP}

Introduction

- Courbes 3D :
 - peut être vue comme le ``déplacement'' d'un point dans l'espace,
 - défini par un paramètre : un scalaire u qui représente la position du point le long de la courbe.



Introduction

- **Courbes 3D :**

➤ définie par une fonction f dans l'espace \mathbb{R}^3

$f: \mathbb{R} \rightarrow \mathbb{R}^3$:

$$u \mapsto P(u) = \begin{cases} x(u) = f_x(u) \\ y(u) = f_y(u) \\ z(u) = f_z(u) \end{cases}$$

Introduction

- **Courbes 3D :**

➤ définie par une fonction f dans l'espace \mathbb{R}^3

$f: \mathbb{R} \rightarrow \mathbb{R}^3$:

$$u \mapsto P(u) = \begin{cases} x(u) = f_x(u) \\ y(u) = f_y(u) \\ z(u) = f_z(u) \end{cases}$$

➤ pour chaque u , on calcule indépendamment les 3 coordonnées x , y et z ;

Introduction

- **Courbes 3D :**

- définie par une fonction f dans l'espace \mathbb{R}^3

- $f: \mathbb{R} \rightarrow \mathbb{R}^3:$

$$u \mapsto P(u) = \begin{cases} x(u) = f_x(u) \\ y(u) = f_y(u) \\ z(u) = f_z(u) \end{cases}$$

- pour chaque u , on calcule indépendamment les 3 coordonnées x , y et z ;

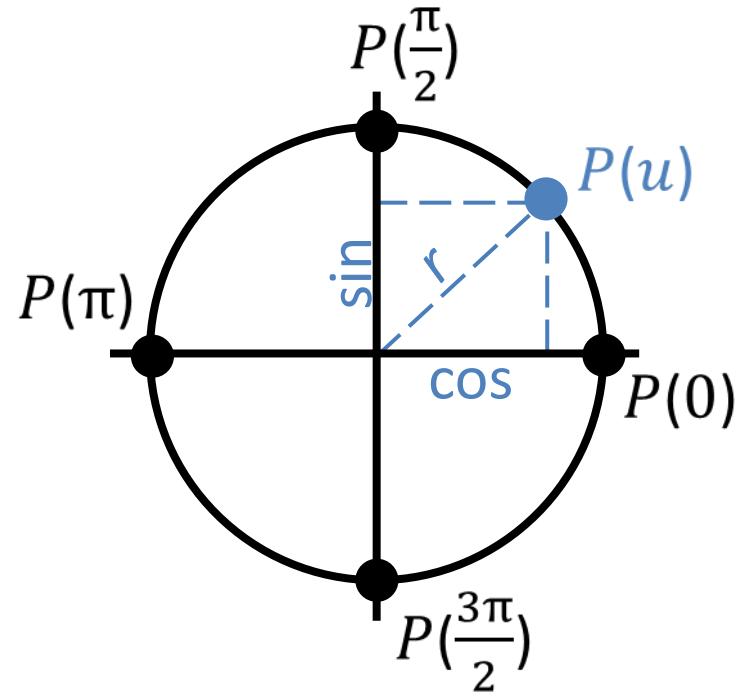
- peut être définie par plusieurs représentations différentes.

Introduction

- Exemple du cercle 2D :

➤ centré à l'origine et définie par un rayon r et un paramètre u défini dans \mathbb{R}^2 .

$$u \rightarrow P(u) = \begin{cases} x(u) = r \cdot \cos(u) \\ y(u) = r \cdot \sin(u) \end{cases}$$



Plan

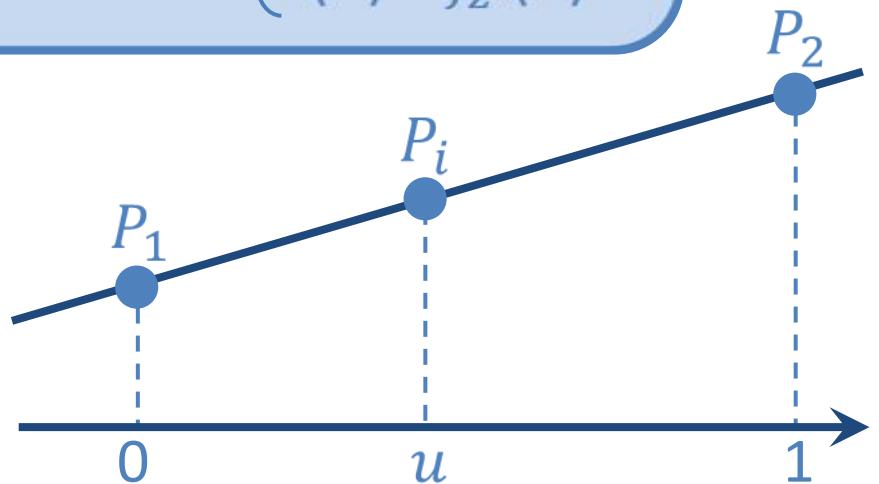
- Introduction
- Représentation d'une droite et d'une courbe
- Géométrie différentielle
- Cubiques
- Courbes de Bézier
- Courbes B-Splines / NURBS
- Dessin en OpenGL

Représentation d'une droite

- Une droite 3D :

- définie entre 2 points P_1 et P_2 :

$$P(u) = (1-u) P_1 + u P_2 \equiv P(u) \begin{cases} x(u) = f_x(u) \\ y(u) = f_y(u) \\ z(u) = f_z(u) \end{cases}$$



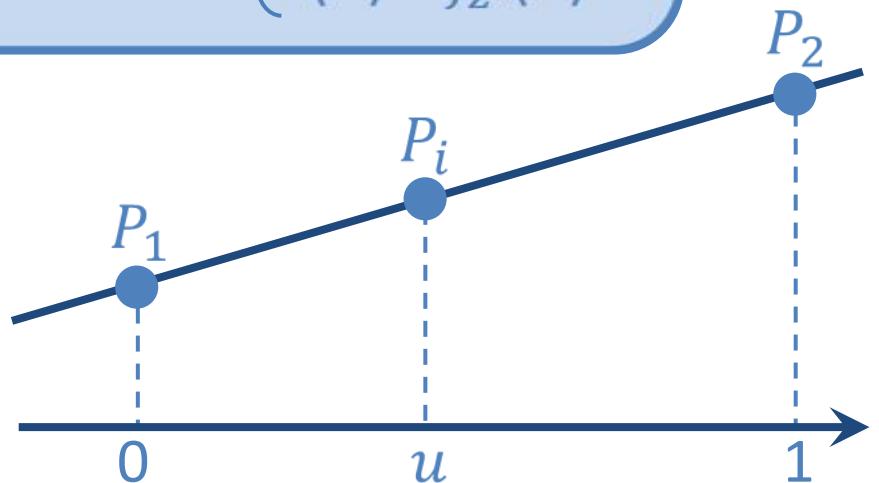
Représentation d'une droite

- Une droite 3D :

- définie entre 2 points P_1 et P_2 :

$$P(u) = (1-u) P_1 + u P_2 \equiv P(u) \begin{cases} x(u) = f_x(u) \\ y(u) = f_y(u) \\ z(u) = f_z(u) \end{cases}$$

On retrouve la notion
d'interpolation linéaire.
Si u varie entre 0 et 1 le point
 P parcourt linéairement le
segment de P_1 à P_2 .

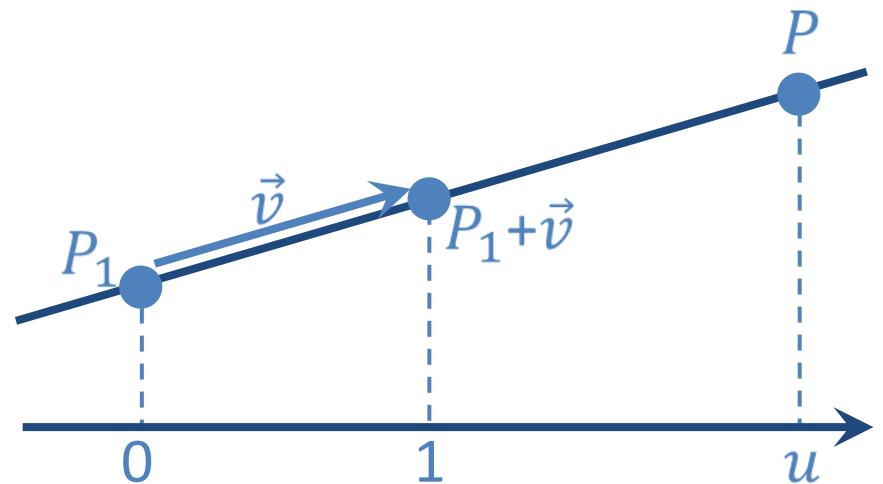


Représentation d'une droite

- Une droite 3D :

- définie par un point P_1 et un vecteur \vec{v} :

$$P(u) = P_1 + u\vec{v}$$



Représentation d'une courbe

- Une courbe 3D :

➤ définie un paramètre u :

$$P(u) = \begin{pmatrix} x(u) \\ y(u) \\ z(u) \end{pmatrix}$$

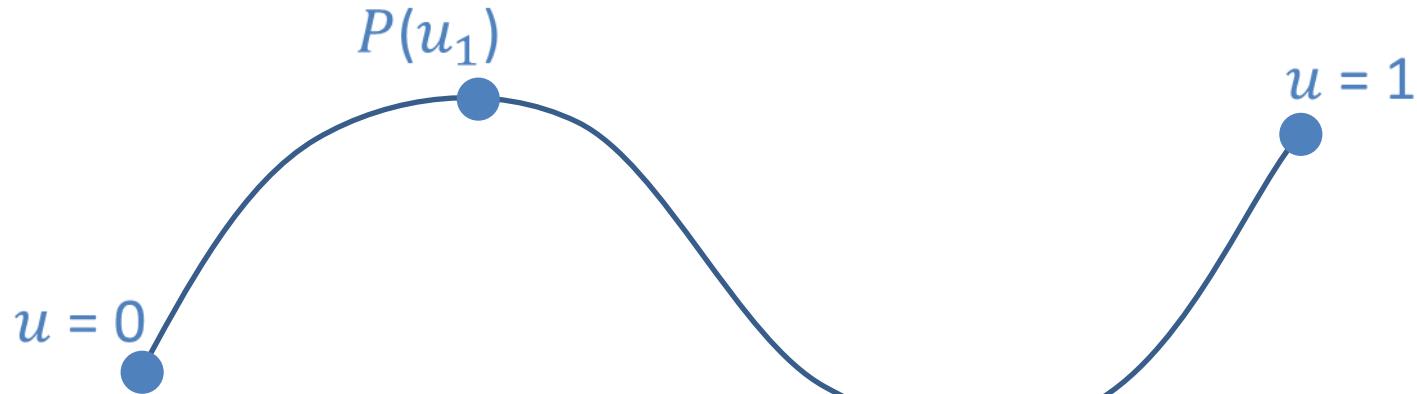
➤ en modélisation géométrique, u est bornée et normalisé : $u \in [0,1]$

Plan

- Introduction
- Représentation d'une droite et d'une courbe
- Géométrie différentielle
- Cubiques
- Courbes de Bézier
- Courbes B-Splines / NURBS
- Dessin en OpenGL

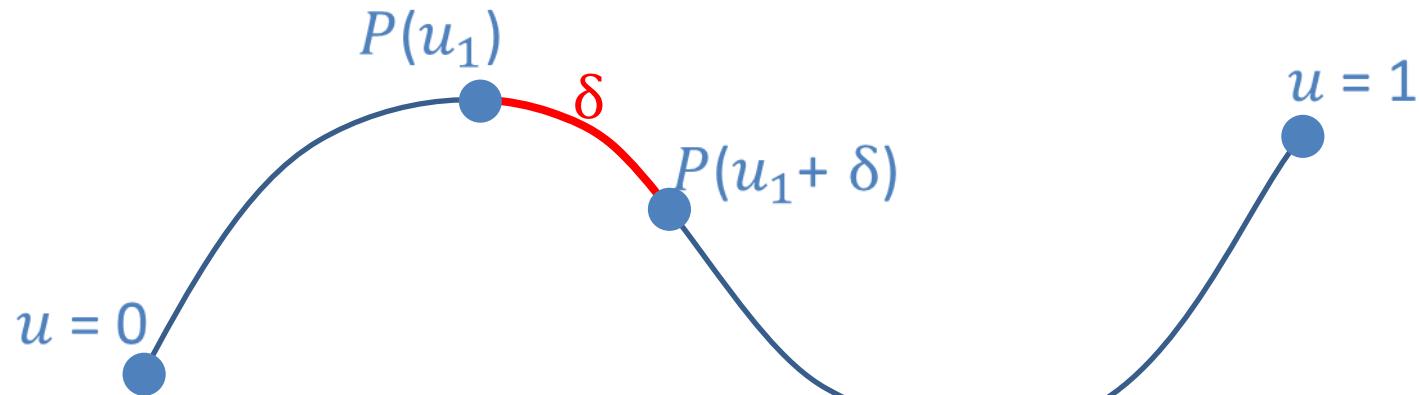
Géométrie différentielle

- Pour paramétriser une courbe :
 - avec scalaire normalisé : $u \in [0,1]$;



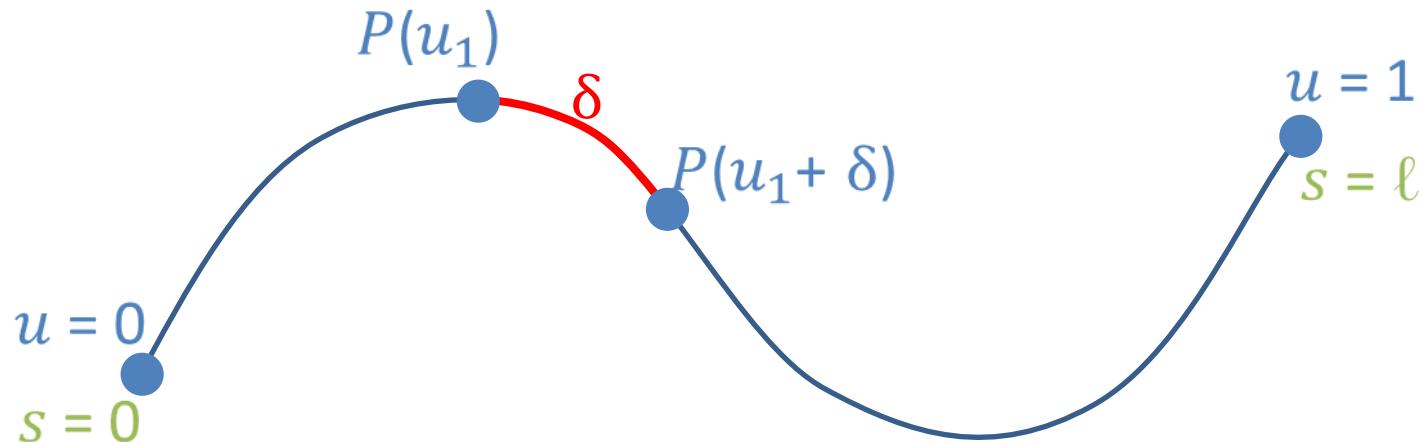
Géométrie différentielle

- Pour paramétriser une courbe :
 - avec scalaire normalisé : $u \in [0,1]$;



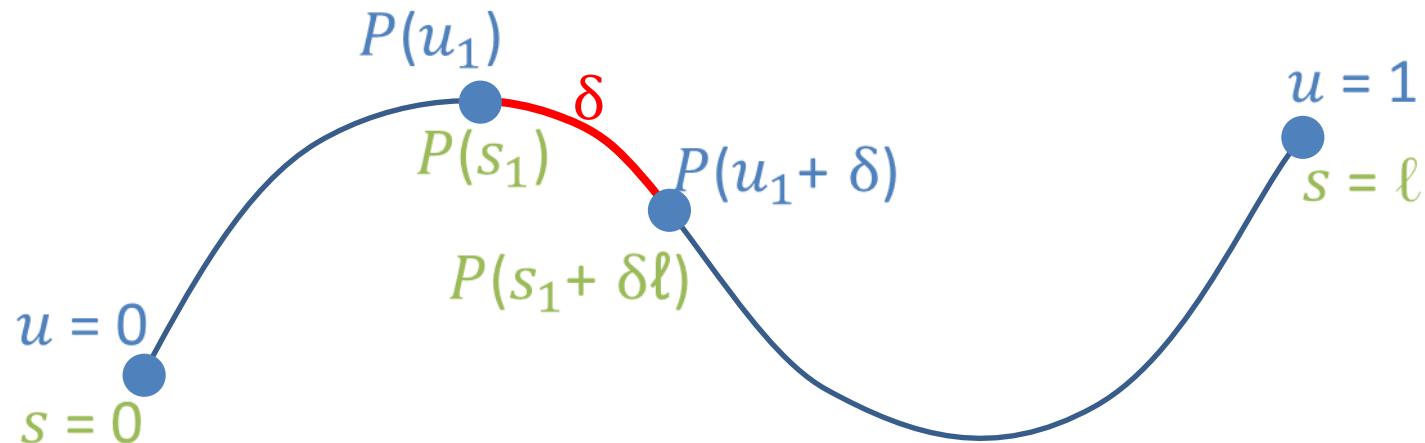
Géométrie différentielle

- Pour paramétriser une courbe :
 - avec scalaire normalisé : $u \in [0,1]$;
 - avec l'abscisse curviligne : $s \in [0,\ell]$



Géométrie différentielle

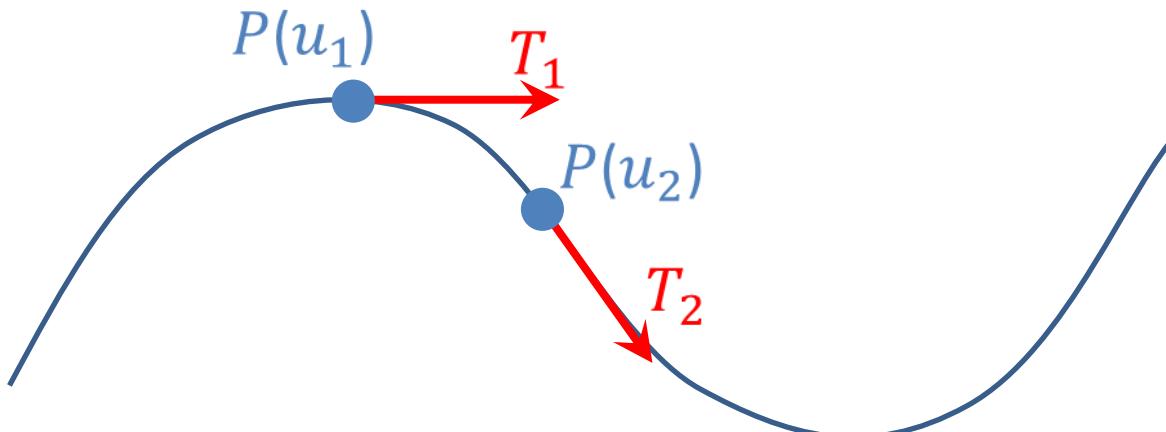
- Pour paramétriser une courbe :
 - avec scalaire normalisé : $u \in [0,1]$;
 - avec l'abscisse curviligne : $s \in [0,\ell]$
 - ℓ est la longueur parcourue le long de la courbe depuis son origine.



Géométrie différentielle

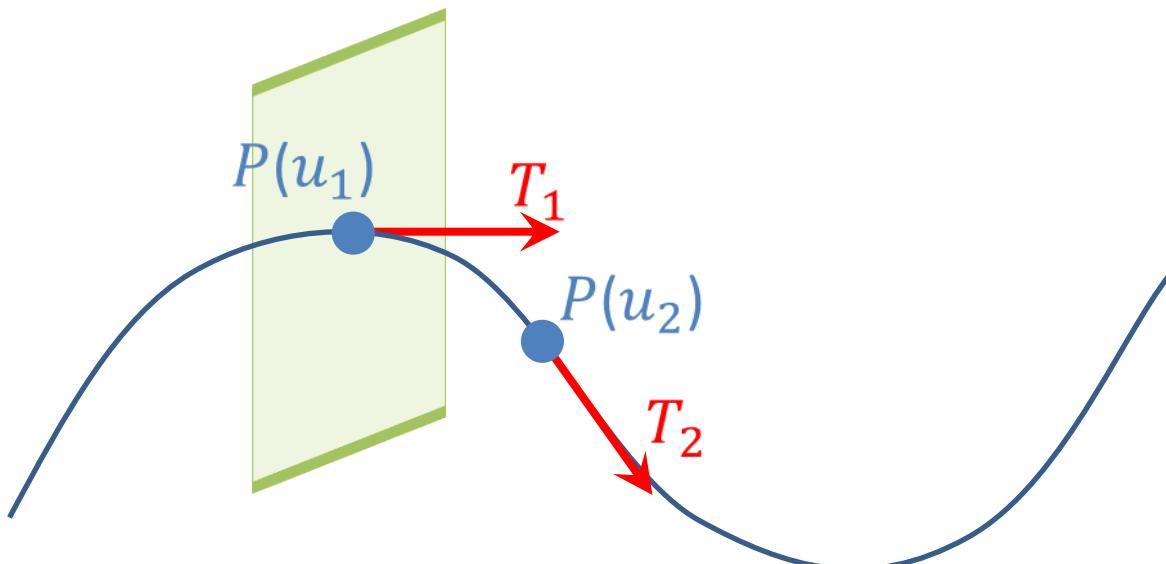
- Vecteur tangent :
 - défini pour chaque point de la courbe ;
 - correspond à la dérivée du point.

$$T = \frac{dp}{ds} = P'$$



Géométrie différentielle

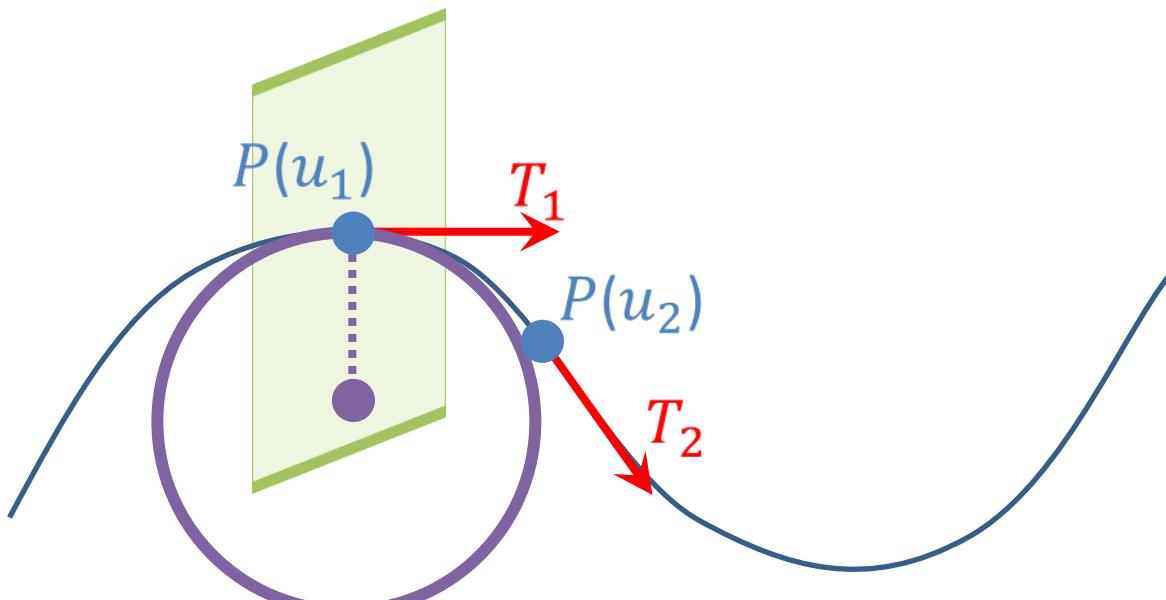
- Vecteur normal :
 - vecteur orthogonal à la tangente, mais il y en a une infinité;



Géométrie différentielle

- Vecteur normal :

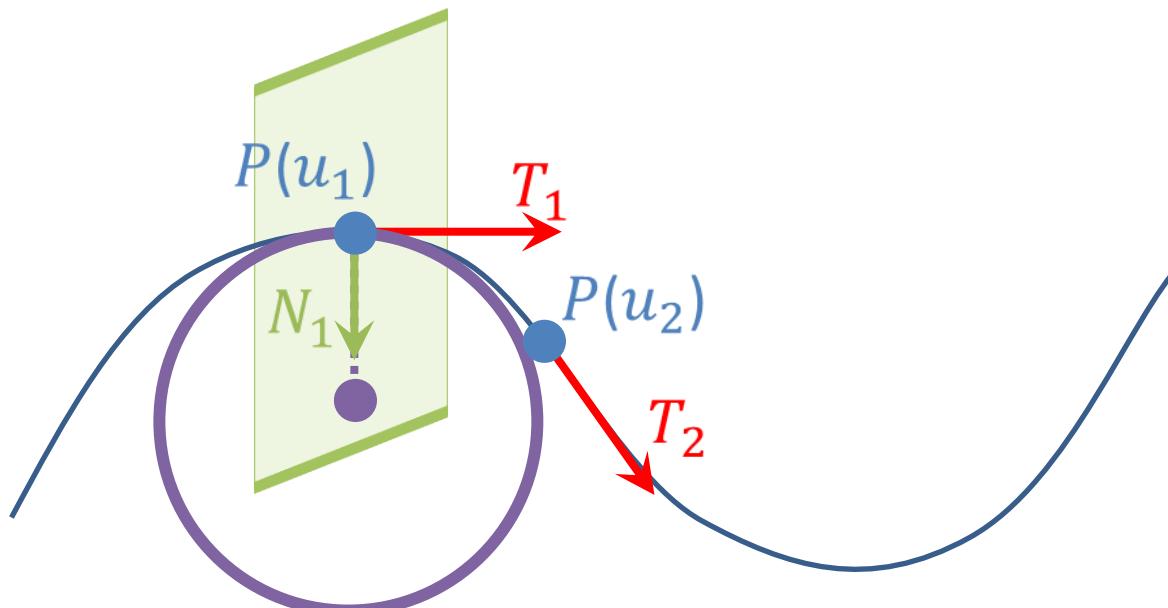
- vecteur orthogonal à la tangente, mais il y en a une infinité;
- correspond au rayon du cercle *osculateur*.



Géométrie différentielle

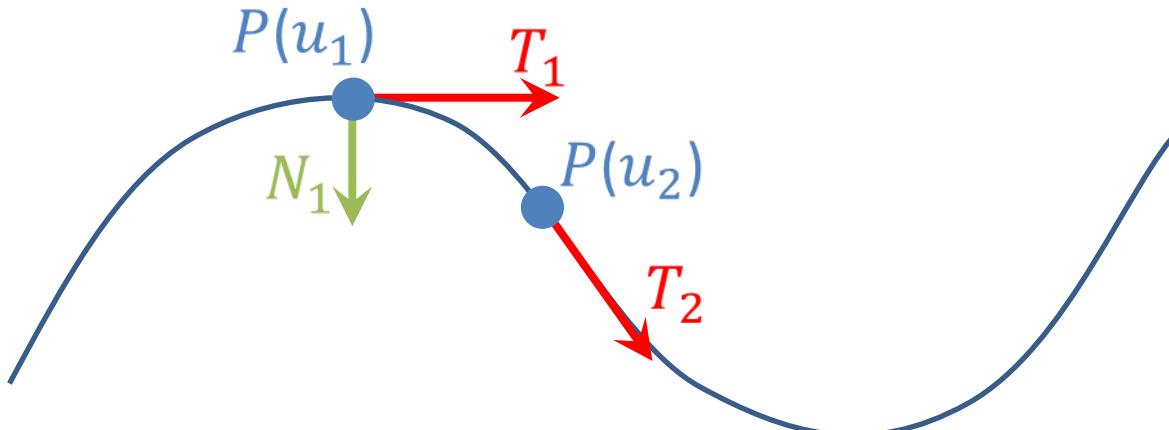
- Vecteur normal :

- vecteur orthogonal à la tangente, mais il y en a une infinité;
- correspond au rayon du cercle *osculateur*.



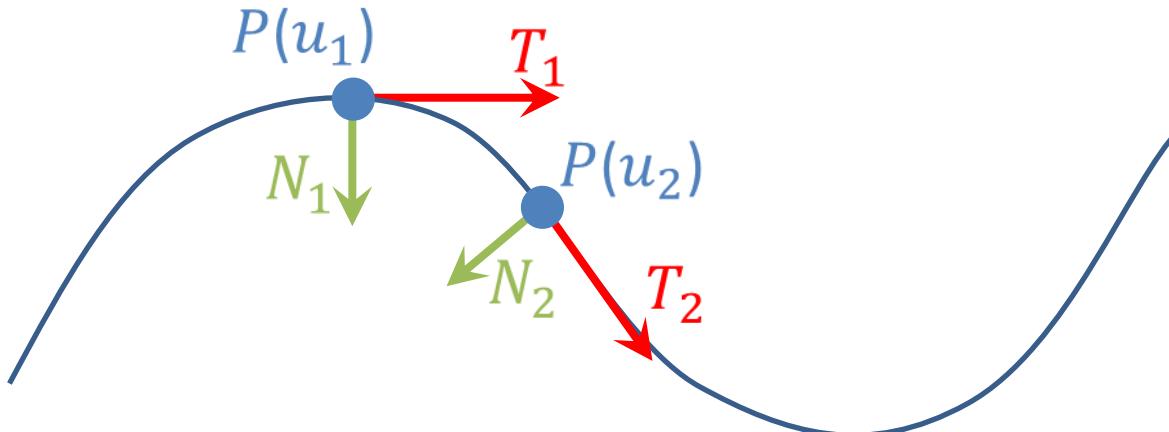
Géométrie différentielle

- Vecteur normal :
 - vecteur orthogonal à la tangente, mais il y en a une infinité;
 - correspond au rayon du cercle *osculateur*.



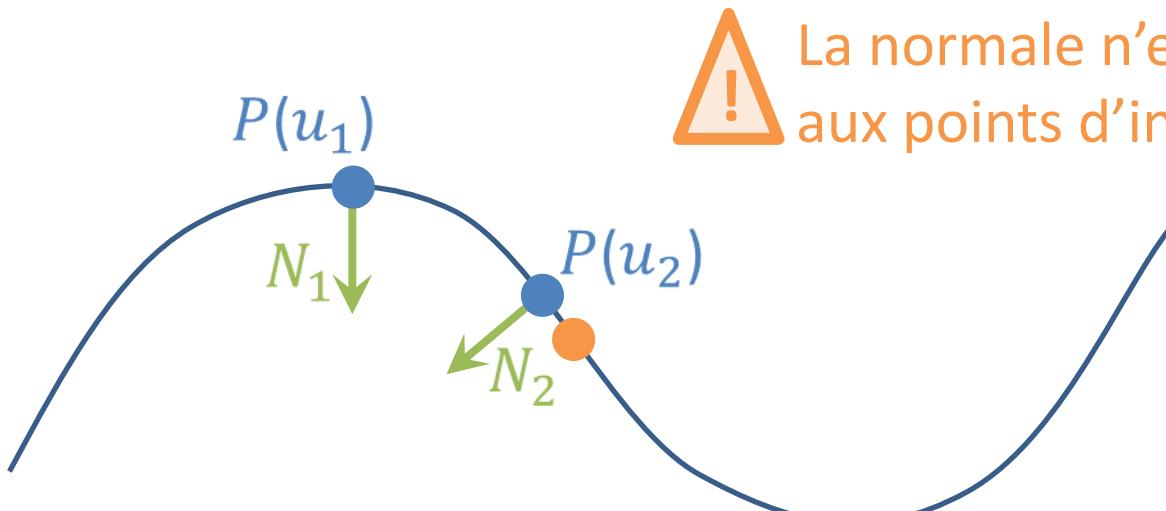
Géométrie différentielle

- Vecteur normal :
 - vecteur orthogonal à la tangente, mais il y en a une infinité;
 - correspond au rayon du cercle *osculateur*.



Géométrie différentielle

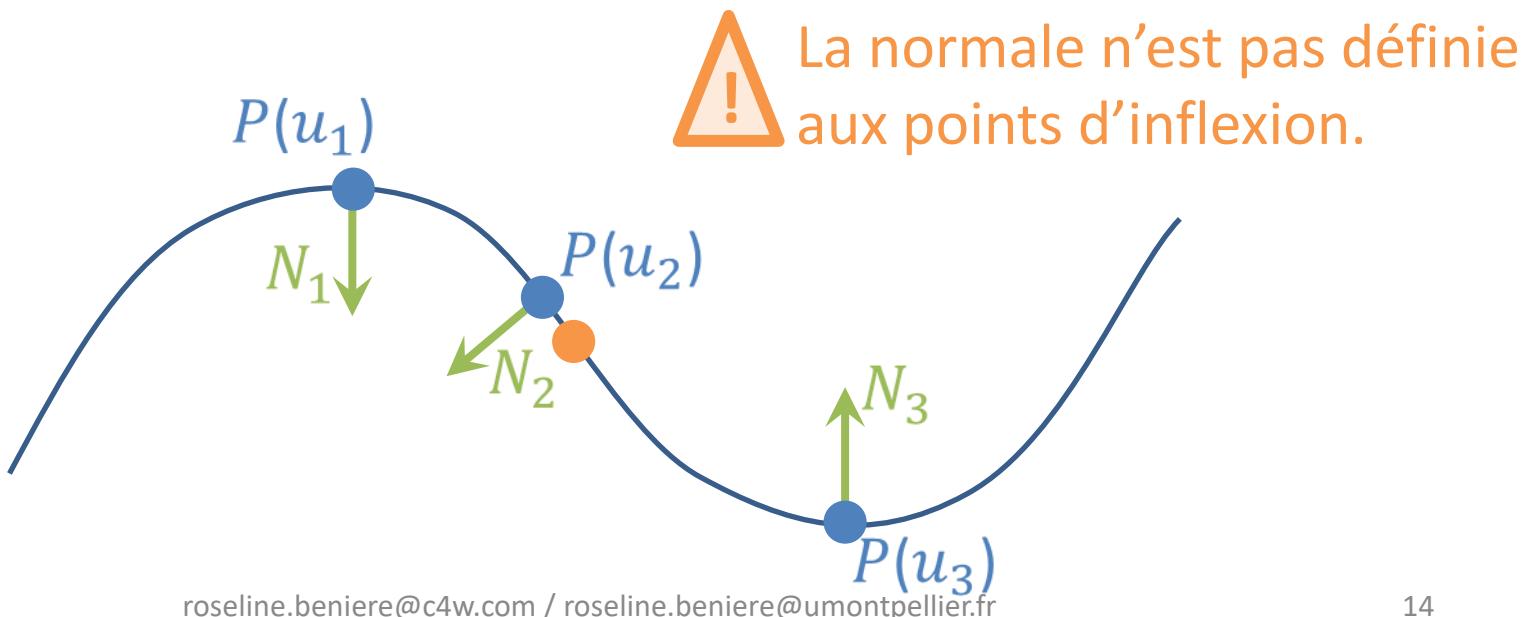
- Vecteur normal :
 - vecteur orthogonal à la tangente, mais il y en a une infinité;
 - correspond au rayon du cercle *osculateur*.



⚠ La normale n'est pas définie aux points d'inflexion.

Géométrie différentielle

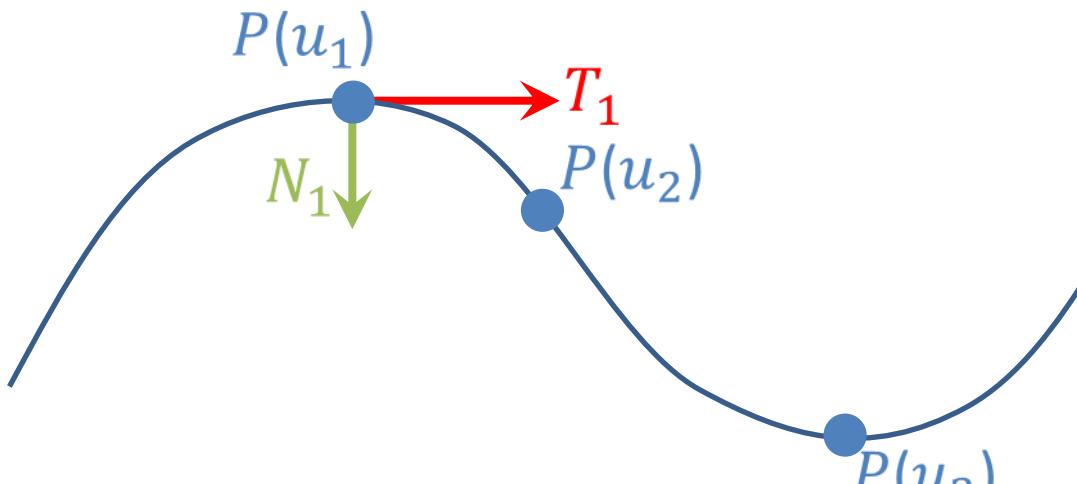
- Vecteur normal :
 - vecteur orthogonal à la tangente, mais il y en a une infinité;
 - correspond au rayon du cercle *osculateur*.



Géométrie différentielle

- **Repère de Frénet :**

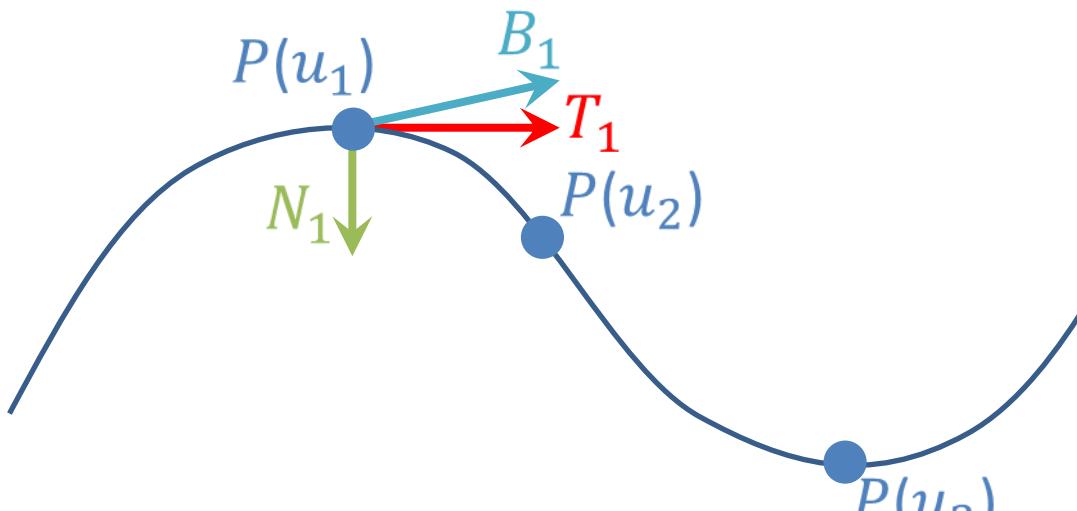
- un repère par point de la courbe,
- contient le vecteur tangent, le vecteur normal et leur produit vectoriel : B ($-B$ =dérivé seconde en P).



Géométrie différentielle

- **Repère de Frénet :**

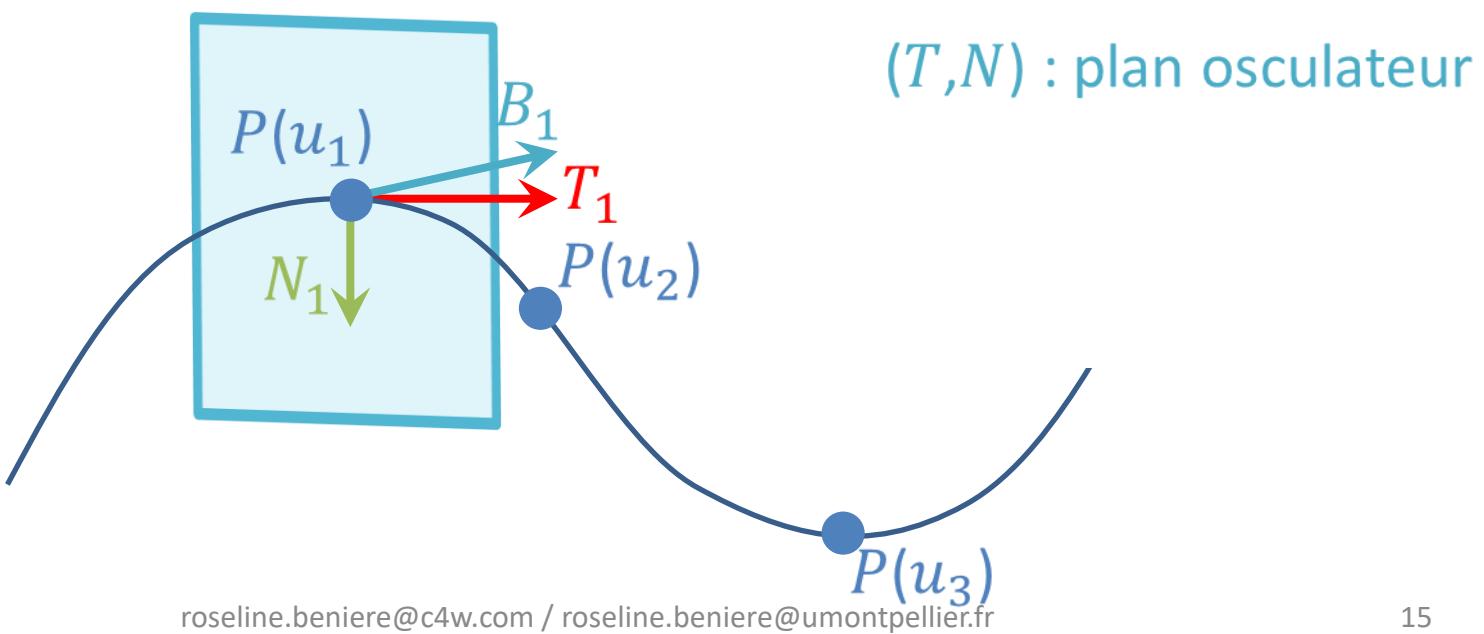
- un repère par point de la courbe,
- contient le vecteur tangent, le vecteur normal et leur produit vectoriel : B ($-B$ =dérivé seconde en P).



Géométrie différentielle

- **Repère de Frénet :**

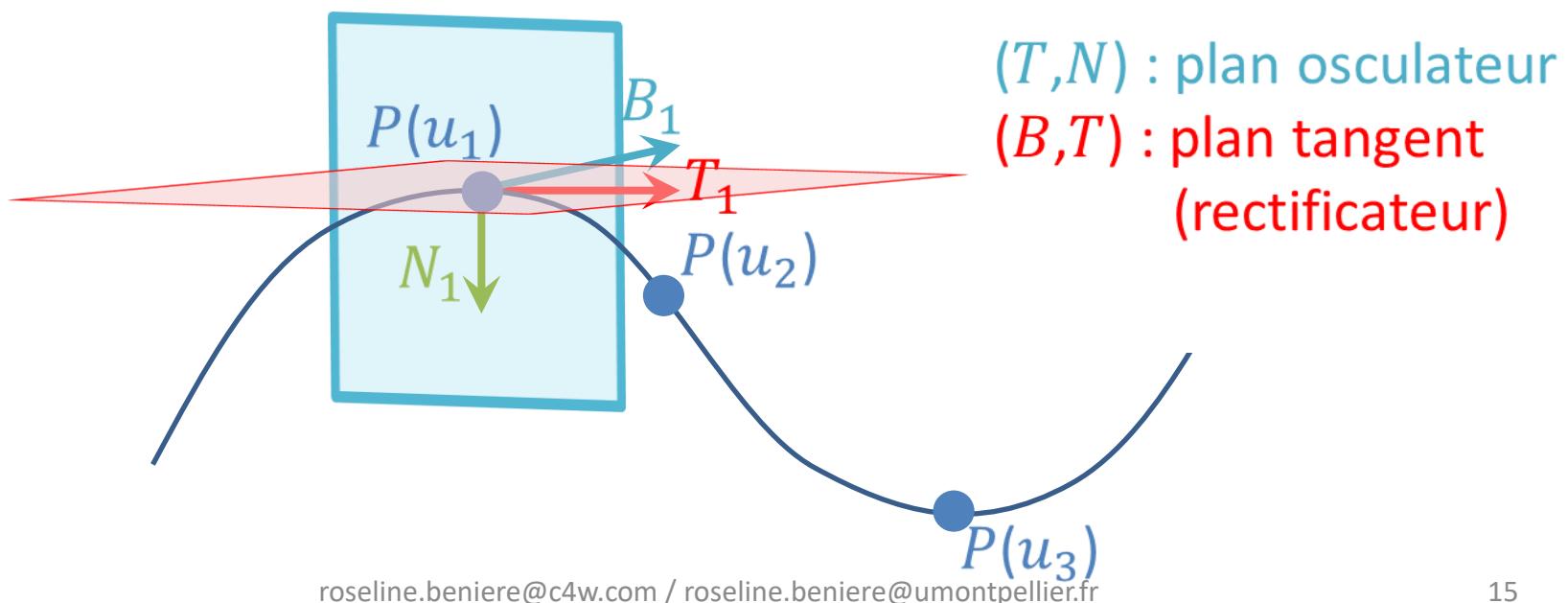
- un repère par point de la courbe,
- contient le vecteur tangent, le vecteur normal et leur produit vectoriel : B ($-B$ =dérivé seconde en P).



Géométrie différentielle

- **Repère de Frénet :**

- un repère par point de la courbe,
- contient le vecteur tangent, le vecteur normal et leur produit vectoriel : B ($-B$ =dérivé seconde en P).

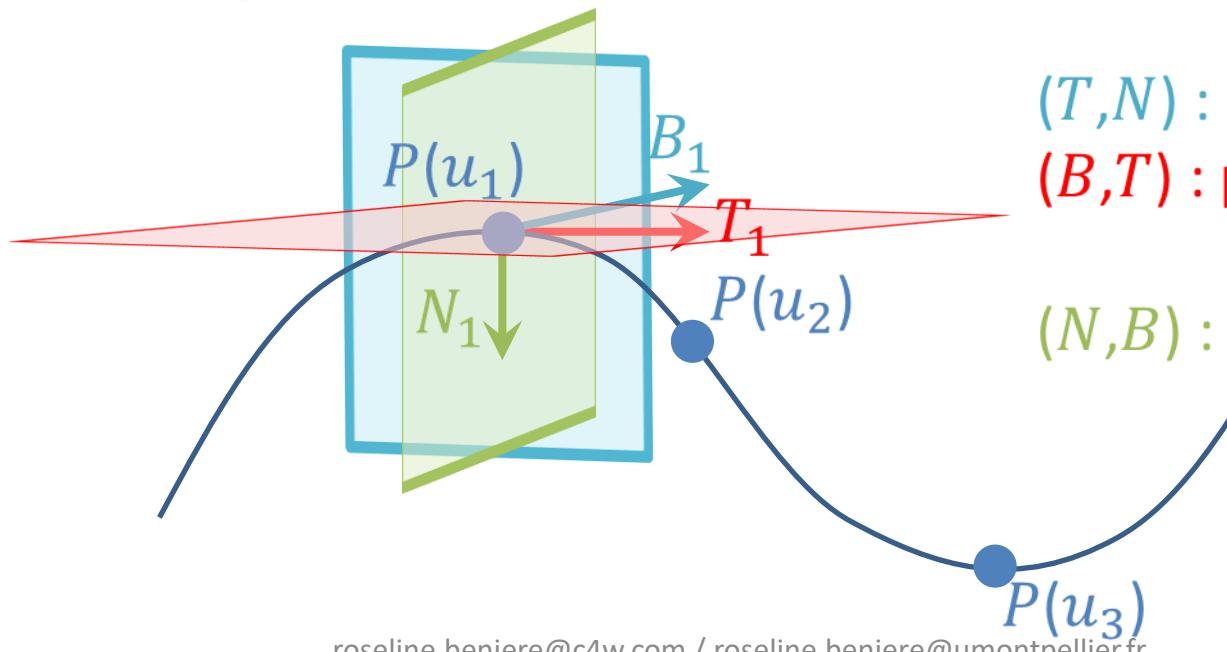


(T, N) : plan osculateur
 (B, T) : plan tangent
(rectificateur)

Géométrie différentielle

- **Repère de Frénet :**

- un repère par point de la courbe,
- contient le vecteur tangent, le vecteur normal et leur produit vectoriel : B ($-B$ =dérivé seconde en P).



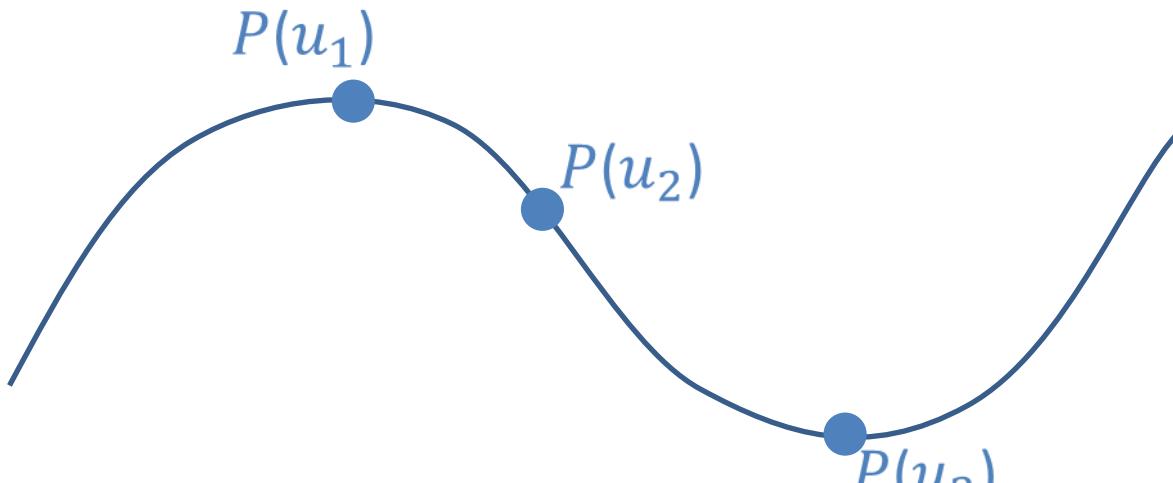
(T, N) : plan osculateur

(B, T) : plan tangent
(rectificateur)

(N, B) : plan normal

Géométrie différentielle

- Courbure :
 - définie pour chaque point de la courbe;

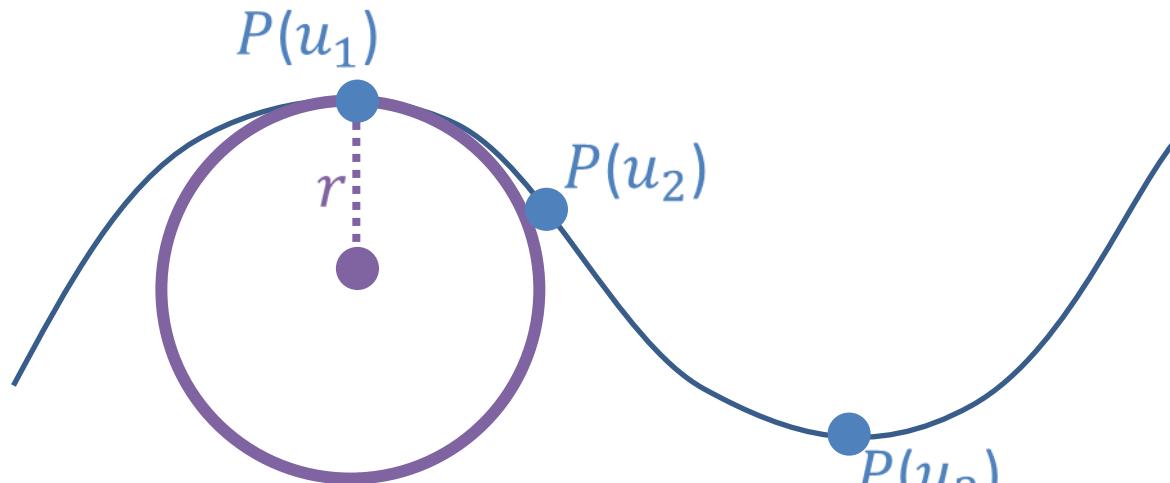


Géométrie différentielle

- Courbure :

- définie pour chaque point de la courbe;
- la valeur de la courbure k , correspond à l'inverse du rayon du cercle osculateur

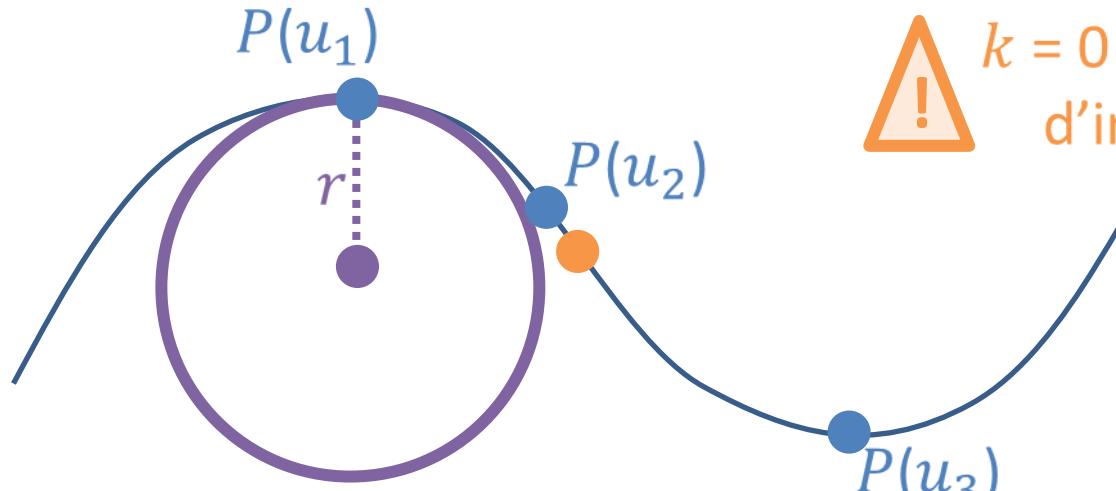
$$k = \frac{1}{r}$$



Géométrie différentielle

- Courbure :
 - définie pour chaque point de la courbe;
 - la valeur de la courbure k , correspond à l'inverse du rayon du cercle osculateur

$$k = \frac{1}{r}$$



Plan

- Introduction
- Représentation d'une droite et d'une courbe
- Géométrie différentielle
- Cubiques
- Courbes de Bézier
- Courbes B-Splines / NURBS
- Dessin en OpenGL

Cubiques

- Courbes polynomiales de degrés 3 :
 - leur représentation algébrique est :

$$P(u) = au^3 + bu^2 + cu + d$$

Cubiques

- Courbes polynomiales de degrés 3 :

➤ leur représentation algébrique est :

$$P(u) = au^3 + bu^2 + cu + d$$

➤ les coordonnées des points sont donc calculées ainsi :

$$P(u) : \begin{cases} x(u) = a_x u^3 + b_x u^2 + c_x u + d_x \\ y(u) = a_y u^3 + b_y u^2 + c_y u + d_y \\ z(u) = a_z u^3 + b_z u^2 + c_z u + d_z \end{cases}$$

Cubiques

- Courbes polynomiales de degrés 3 :

➤ leur représentation algébrique est :

$$P(u) = au^3 + bu^2 + cu + d$$

➤ les coordonnées des points sont donc calculées ainsi :

$$P(u) : \begin{cases} x(u) = a_x u^3 + b_x u^2 + c_x u + d_x \\ y(u) = a_y u^3 + b_y u^2 + c_y u + d_y \\ z(u) = a_z u^3 + b_z u^2 + c_z u + d_z \end{cases}$$

➤ impossible à ``tracer'' par un utilisateur si elle est sous cette forme.

Cubiques d'Hermite

- La courbe est définie par :
 - point de départ $P_0(u=0)$, son point d'arrivée $P_1(u=1)$ et leur tangentes respectives V_0 et V_1 ,

Cubiques d'Hermite

- La courbe est définie par :
 - point de départ $P_0(u=0)$, son point d'arrivée $P_1(u=1)$ et leur tangentes respectives V_0 et V_1 ,
 - les coefficients géométriques a, b, c et d sont calculés ainsi :

$$\begin{cases} P_0 = p(0) = d \\ P_1 = p(1) = a+b+c+d \\ V_0 = p'(0) = c \\ V_1 = p'(1) = 3a+2b+c \end{cases}$$

Cubiques d'Hermite

- La courbe est définie par :

- point de départ $P_0(u=0)$, son point d'arrivée $P_1(u=1)$ et leur tangentes respectives V_0 et V_1 ,
- les coefficients géométriques a, b, c et d sont calculés ainsi :

$$\begin{cases} P_0 = p(0) = d \\ P_1 = p(1) = a+b+c+d \\ V_0 = p'(0) = c \\ V_1 = p'(1) = 3a+2b+c \end{cases}$$

d'où

$$\begin{cases} d = P_0 \\ c = V_0 \\ b = -3P_0 + 3P_1 - 2V_0 - V_1 \\ a = 2P_0 - 2P_1 + V_0 + V_1 \end{cases}$$

Cubiques d'Hermite

- La courbe est définie par :
 - son équation $P(u) = au^3 + bu^2 + cu + d$, ce qui donne en fonction de P_0, P_1, V_0 et V_1 :

Cubiques d'Hermite

- La courbe est définie par :

➤ son équation $P(u) = au^3 + bu^2 + cu + d$, ce qui donne en fonction de P_0, P_1, V_0 et V_1 :

$$P(u) = F_1(u)P_0 + F_2(u)P_1 + F_3(u)V_0 + F_4(u)V_1$$

avec

$$\begin{cases} F_1(u) = 2u^3 - 3u^2 + 1 \\ F_2(u) = -2u^3 + 3u^2 \\ F_3(u) = u^3 - 2u^2 + u \\ F_4(u) = u^3 - u^2 \end{cases}$$

Cubiques d'Hermite

- Forme matricielle d'une cubique :

➤ chaque point est défini par :

$$p(u) = UMB = \begin{vmatrix} u^3 & u^2 & u & 1 \end{vmatrix} \begin{vmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{vmatrix} \begin{vmatrix} P_0 \\ P_1 \\ V_0 \\ V_1 \end{vmatrix}$$

➤ est appelée souvent "cubique matricielle"

Plan

- Introduction
- Représentation d'une droite et d'une courbe
- Géométrie différentielle
- Cubiques
- Courbes de Bézier
- Courbes B-Splines / NURBS
- Dessin en OpenGL

Courbes de Bézier

- **Exemple de courbes paramétriques :**

- un point de la courbe est une combinaison affine des points de contrôle P_i , si $u \in [a, b]$

$$p(u) = \sum_{i=0}^n N_i^d(u) P_i$$

avec

$$\sum_{i=0}^n N_i^d(u) = 1$$

Courbes de Bézier

- **Exemple de courbes paramétriques :**

- un point de la courbe est une combinaison affine des points de contrôle P_i , si $u \in [a, b]$

$$p(u) = \sum_{i=0}^n N_i^d(u) P_i$$

avec

$$\sum_{i=0}^n N_i^d(u) = 1$$

- la position des points de la courbe est donc liée aux points contrôles.

Courbes de Bézier

- **Polynôme de Bernstein :**

➤ on repart du développement binomial :

$$1 = (u + (1 - u))^n = \sum_{i=0}^n \binom{n}{i} u^i (1-u)^{n-i}$$

Courbes de Bézier

- **Polynôme de Bernstein :**

➤ on repart du développement binomial :

$$1 = (u + (1 - u))^n = \sum_{i=0}^n \binom{n}{i} u^i (1-u)^{n-i}$$

avec $\binom{n}{i} = \frac{n!}{i!(n-i)!}$ noté aussi C_i^n et appelé binôme de Newton

Courbes de Bézier

- **Polynôme de Bernstein :**

➤ on repart du développement binomial :

$$1 = (u + (1-u))^n = \sum_{i=0}^n \binom{n}{i} u^i (1-u)^{n-i}$$

➤ ainsi, on obtient une somme de $n+1$ polynômes appelés : **polynômes de Bernstein de degré n** :

$$B_i^n(u) = \binom{n}{i} u^i (1-u)^{n-i}, \quad i=0, \dots, n$$

avec $\binom{n}{i} = \frac{n!}{i!(n-i)!}$ noté aussi C_i^n et appelé binôme de Newton

Courbes de Bézier

- Propriétés des polynômes de Bernstein :
 - pour un degré fixé, ils sont linéairement indépendants,

Courbes de Bézier

- Propriétés des polynômes de Bernstein :
 - pour un degré fixé, ils sont linéairement indépendants,
 - ils sont symétriques :

$$B_i^n(u) = B_{n-i}^n(1-u)$$

Courbes de Bézier

- Propriétés des polynômes de Bernstein :

- pour un degré fixé, ils sont linéairement indépendants,
- ils sont symétriques :

$$B_i^n(u) = B_{n-i}^n(1-u)$$

- ils forment une partie de l'unité :

$$\sum_{i=0}^n B_i^n(u) = 1 \quad \forall u \in \mathbb{R}$$

Courbes de Bézier

- Propriétés des polynômes de Bernstein :

- pour un degré fixé, ils sont linéairement indépendants,
- ils sont symétriques :

$$B_i^n(u) = B_{n-i}^n(1-u)$$

- ils forment une partie de l'unité :

$$\sum_{i=0}^n B_i^n(u) = 1 \quad \forall u \in \mathbb{R}$$

- ils sont positifs pour u dans $[0,1]$:

$$B_i^n(u) > 0 \quad \forall u \in [0,1]$$

Courbes de Bézier

- Exemple de polynômes de Bernstein :
 - pour $n = 3$ en utilisant la formule précédente

$$B_i^n(u) = \frac{n!}{i!(n-i)!} u^i (1-u)^{n-i}, \quad i=0, \dots, n$$

Courbes de Bézier

- **Exemple de polynômes de Bernstein :**
 - pour $n = 3$ en utilisant la formule précédente

$$B_i^n(u) = \frac{n!}{i!(n-i)!} u^i (1-u)^{n-i}, \quad i=0, \dots, n$$

➤ les polynômes de Bernstein sont :

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

Courbes de Bézier

- **Exemple de polynômes de Bernstein :**
 - pour $n = 3$ en utilisant la formule précédente

$$B_i^n(u) = \frac{n!}{i!(n-i)!} u^i (1-u)^{n-i}, \quad i=0, \dots, n$$

➤ les polynômes de Bernstein sont :

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

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

Courbes de Bézier

- **Exemple de polynômes de Bernstein :**
 - pour $n = 3$ en utilisant la formule précédente

$$B_i^n(u) = \frac{n!}{i!(n-i)!} u^i (1-u)^{n-i}, \quad i=0, \dots, n$$

➤ les polynômes de Bernstein sont :

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

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

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

Courbes de Bézier

- **Exemple de polynômes de Bernstein :**
 - pour $n = 3$ en utilisant la formule précédente

$$B_i^n(u) = \frac{n!}{i!(n-i)!} u^i (1-u)^{n-i}, \quad i=0, \dots, n$$

➤ les polynômes de Bernstein sont :

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

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

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

$$B_3^3(u) = u^3$$

Courbes de Bézier

- Définition :
 - basé sur les polynômes de Bernstein :

$$p(u) = \sum_{i=0}^n B_i^n(u) P_i , \quad u \in [0,1]$$

Courbes de Bézier

- Définition :
 - basé sur les polynômes de Bernstein :

$$p(u) = \sum_{i=0}^n B_i^n(u) P_i , \quad u \in [0,1]$$

- les points P_i ($i=0 \dots n$) sont **$n+1$ points de contrôle** de la courbe,

Courbes de Bézier

- Définition :
 - basé sur les polynômes de Bernstein :

$$p(u) = \sum_{i=0}^n B_i^n(u) P_i , \quad u \in [0,1]$$

- les points P_i ($i=0 \dots n$) sont **$n+1$ points de contrôle** de la courbe,
- la courbe est d'**ordre $n+1$** et de **degré n** ,

Courbes de Bézier

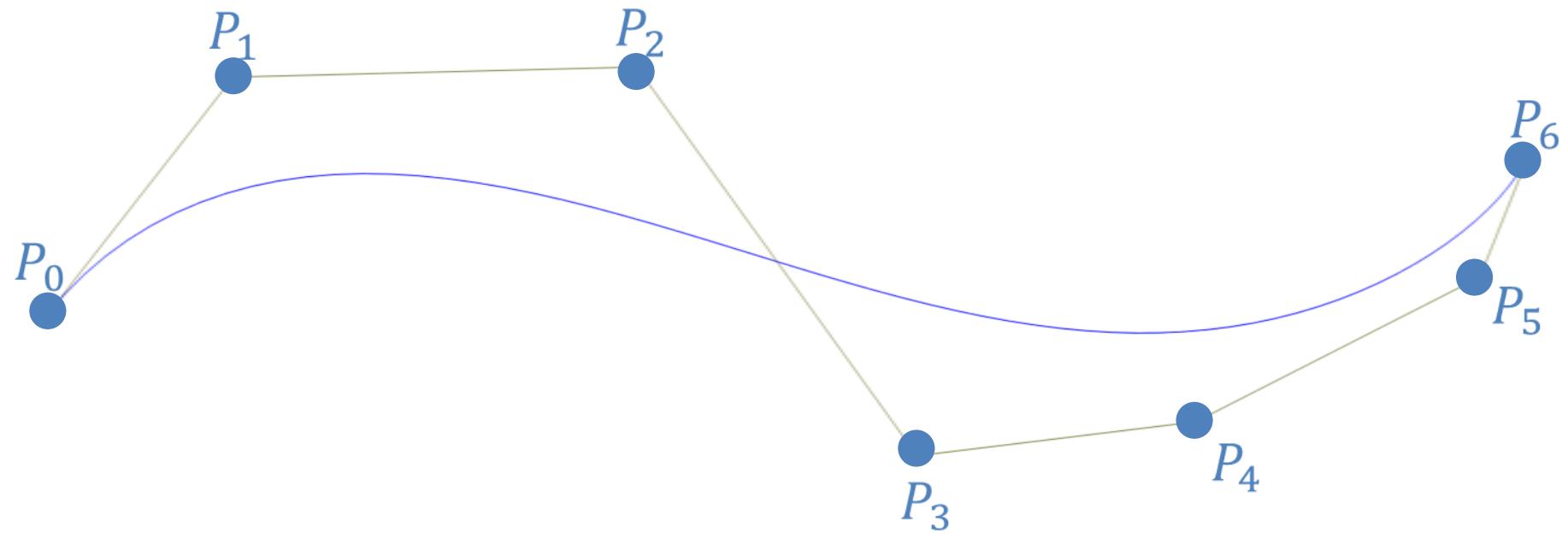
- **Définition :**
 - basé sur les polynômes de Bernstein :

$$p(u) = \sum_{i=0}^n B_i^n(u) P_i , \quad u \in [0,1]$$

- les points P_i ($i=0 \dots n$) sont **$n+1$ points de contrôle** de la courbe,
- la courbe est d'**ordre $n+1$** et de **degré n** ,
- les B_i^n sont les polynômes de Bernstein de degré n . Ils définissent les **fonctions de base** de la courbe.

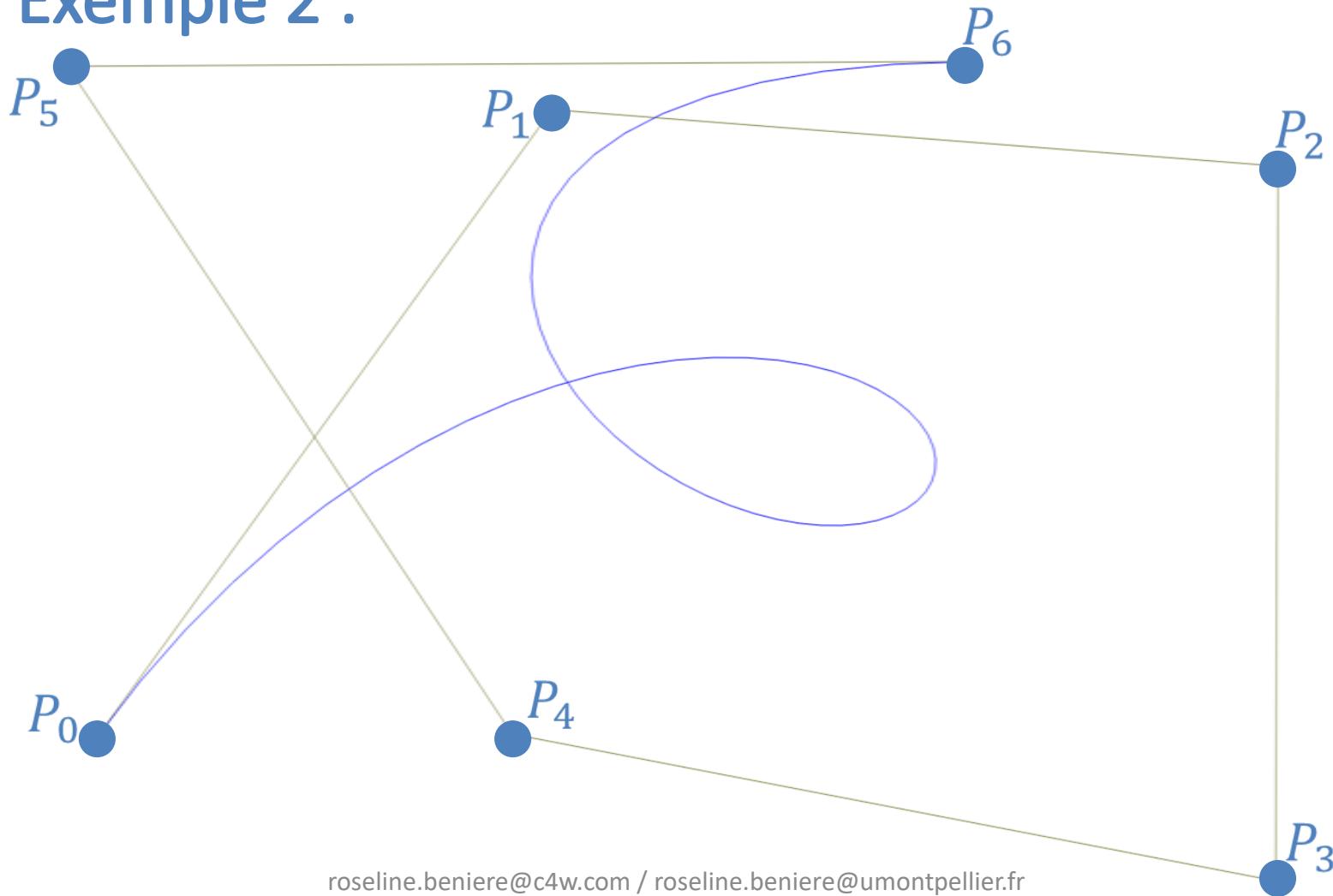
Courbes de Bézier

- Exemple 1 :



Courbes de Bézier

- Exemple 2 :



Courbes de Bézier

- Propriétés :

➤ la symétrie du polynôme de Bernstein implique :

$$p(u) = \sum_{i=0}^n B_i^n (u) P_i = \sum_{i=0}^n B_i^n (1-u) P_{n-i}$$

Courbes de Bézier

- Propriétés :

➤ la symétrie du polynôme de Bernstein implique :

$$p(u) = \sum_{i=0}^n B_i^n (u) P_i = \sum_{i=0}^n B_i^n (1-u) P_{n-i}$$

➡ la courbe est la même qu'elle soit parcourue de 0 à 1 ou de 1 à 0.

Courbes de Bézier

- Propriétés :

➤ la symétrie du polynôme de Bernstein implique :

$$p(u) = \sum_{i=0}^n B_i^n (u) P_i = \sum_{i=0}^n B_i^n (1-u) P_{n-i}$$

➡ la courbe est la même qu'elle soit parcourue de 0 à 1 ou de 1 à 0.

➤ soit $t \in [a,b]$, $t=a(1-u) + bu$ $a \neq b$ alors :

$$p(t(u)) = p(t) = \sum_{i=0}^n B_i^n (u) P_i$$

Courbes de Bézier

- Propriétés :
 - la courbe de Bézier interpolate le premier et le dernier point de contrôle ($u \in [0,1]$)

Courbes de Bézier

- Propriétés :

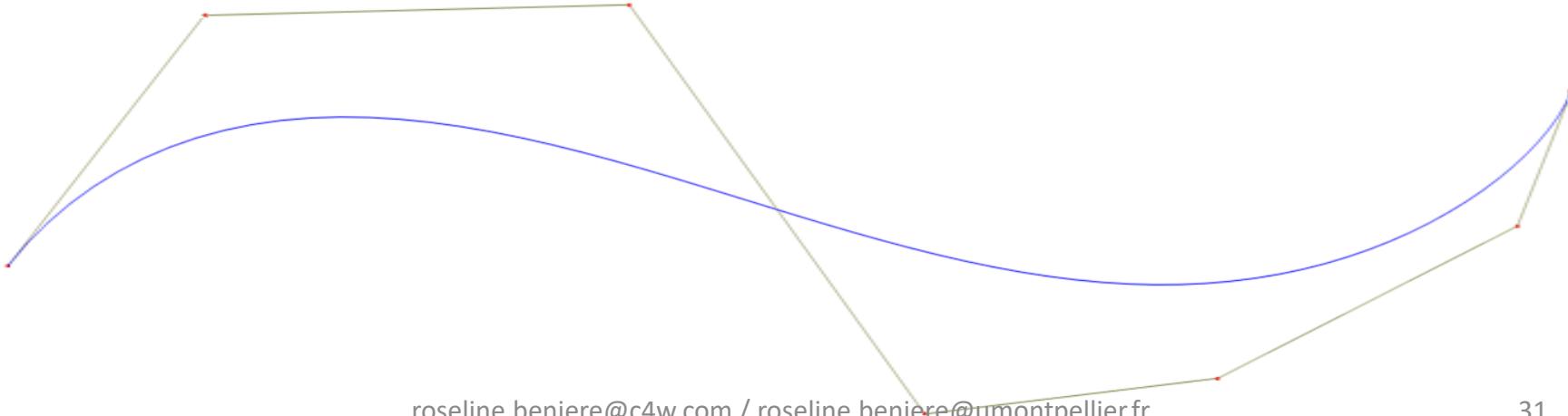
➤ la courbe de Bézier interpolate le premier et le dernier point de contrôle ($u \in [0,1]$)

$$\rightarrow p(0) = P_0 \quad p(1) = P_n$$

Courbes de Bézier

- **Propriétés :**

- la courbe de Bézier interpolate le premier et le dernier point de contrôle ($u \in [0,1]$)
→ $p(0) = P_0 \quad p(1) = P_n$
- la courbe est tangente au premier et au dernier segment du polygone de contrôle :

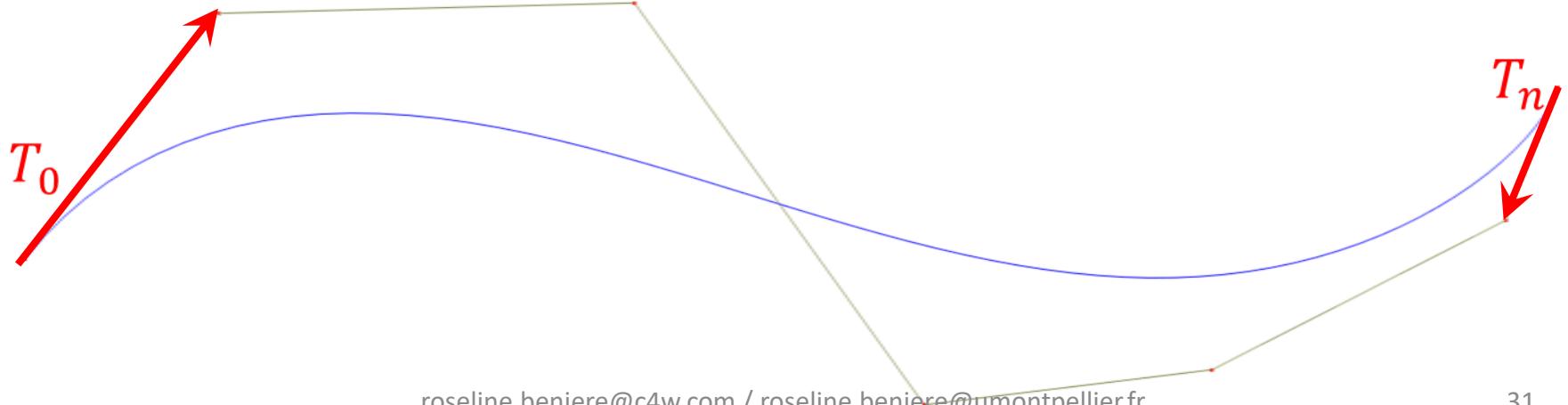


Courbes de Bézier

- Propriétés :

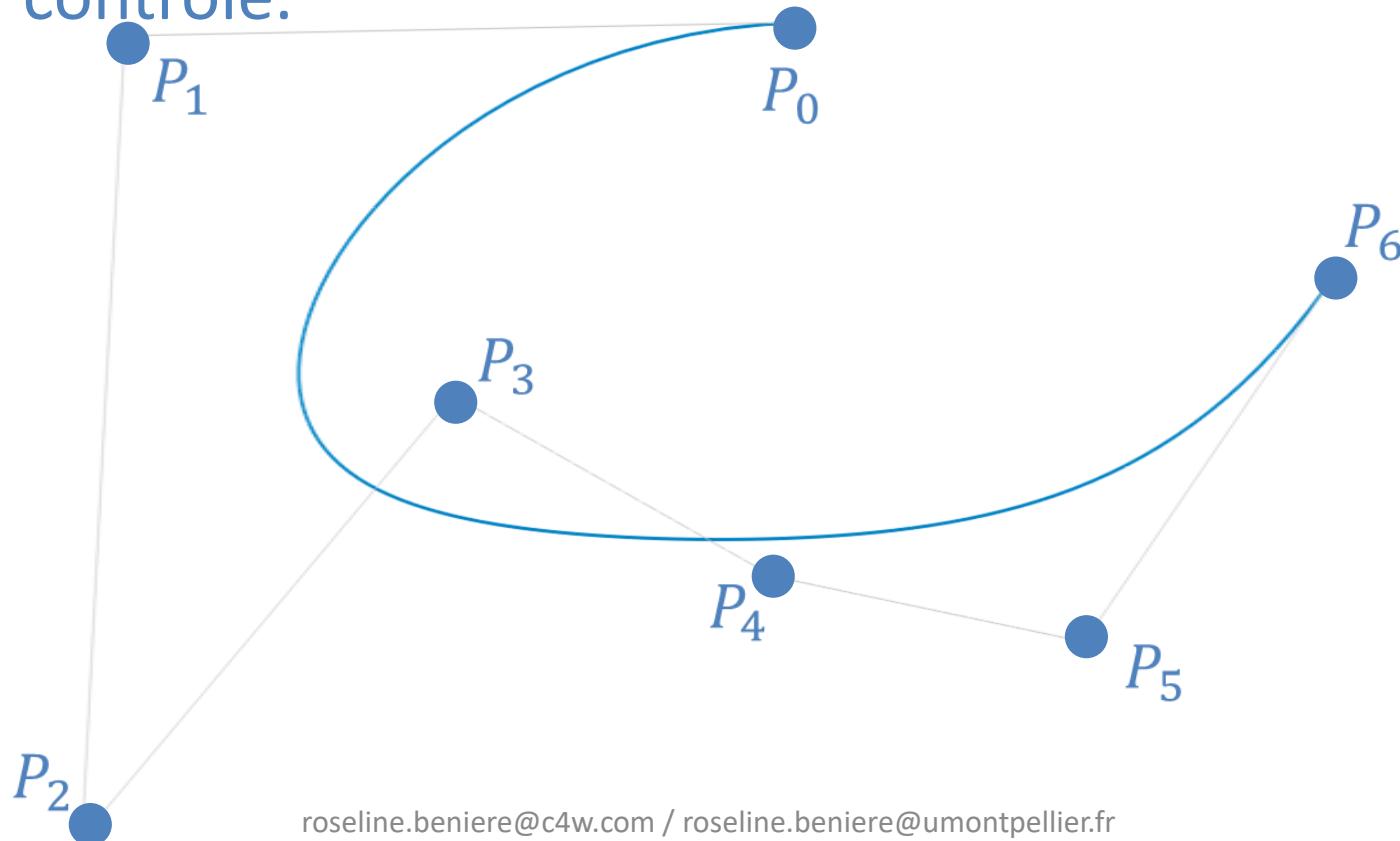
- la courbe de Bézier interpolate le premier et le dernier point de contrôle ($u \in [0,1]$)
→ $p(0) = P_0$ $p(1) = P_n$

- la courbe est tangente au premier et au dernier segment du polygone de contrôle :



Courbes de Bézier

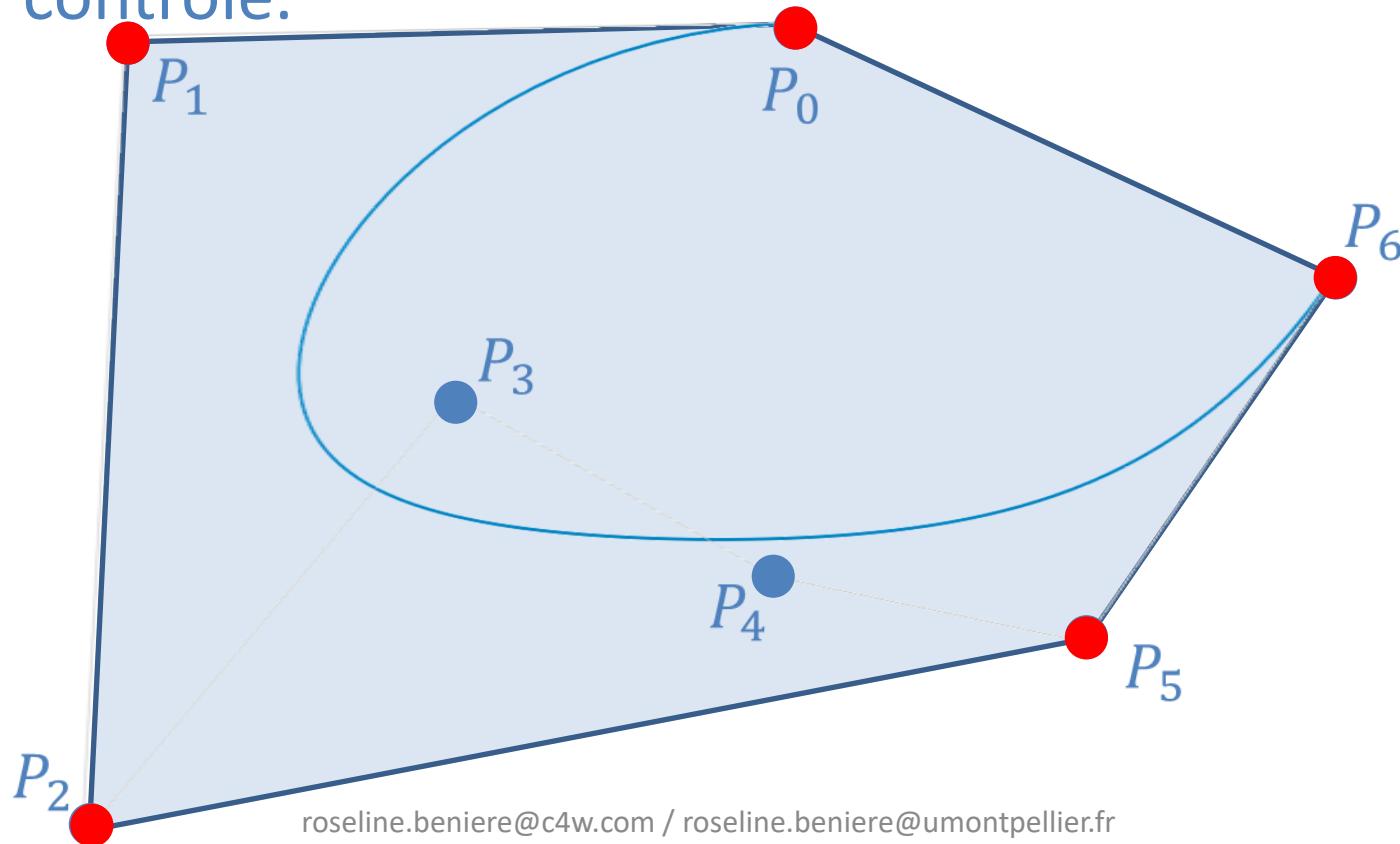
- Enveloppe convexe :
 - correspond à l'enveloppe convexe des points de contrôle.



Courbes de Bézier

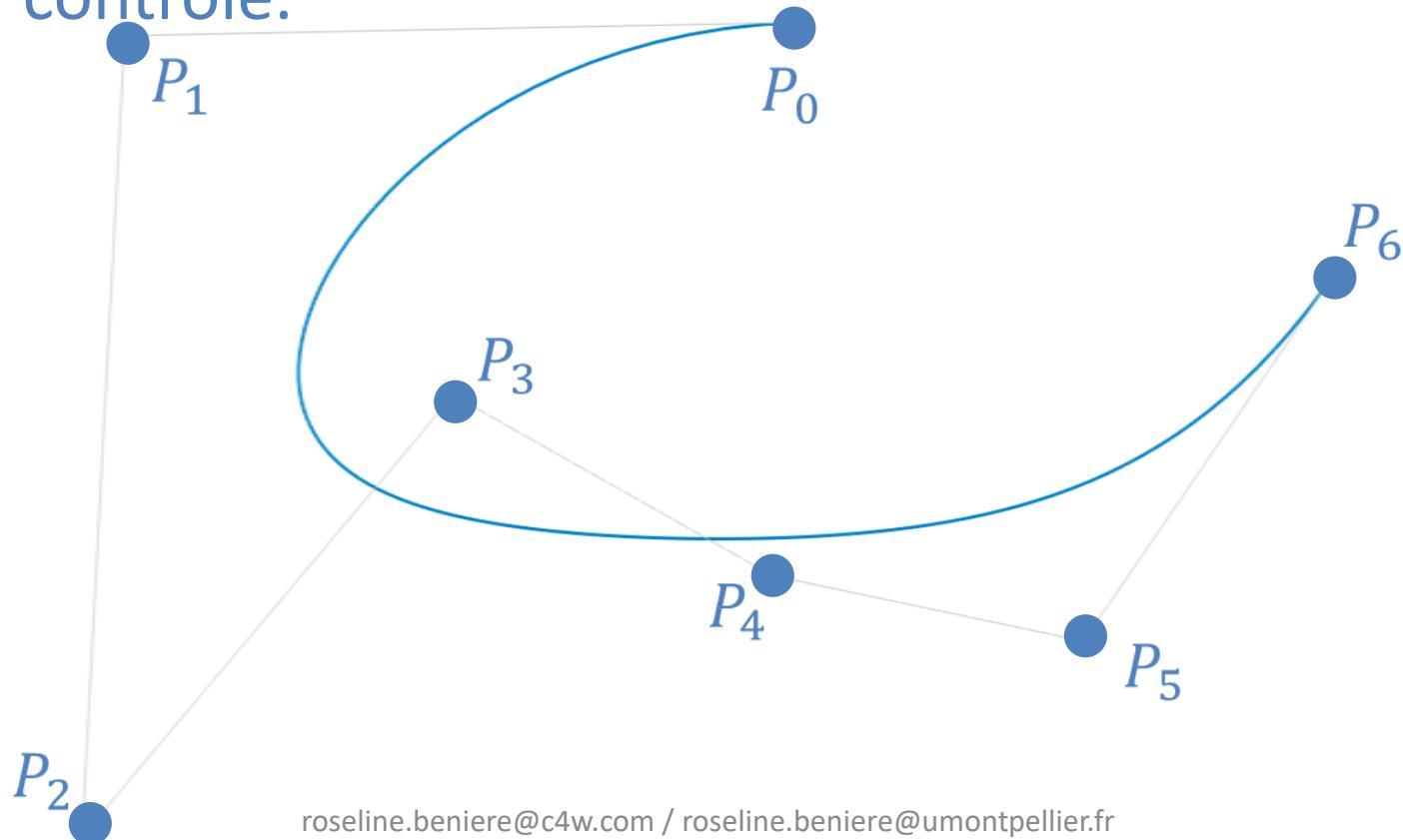
- Enveloppe convexe :

➤ correspond à l'enveloppe convexe des points de contrôle.



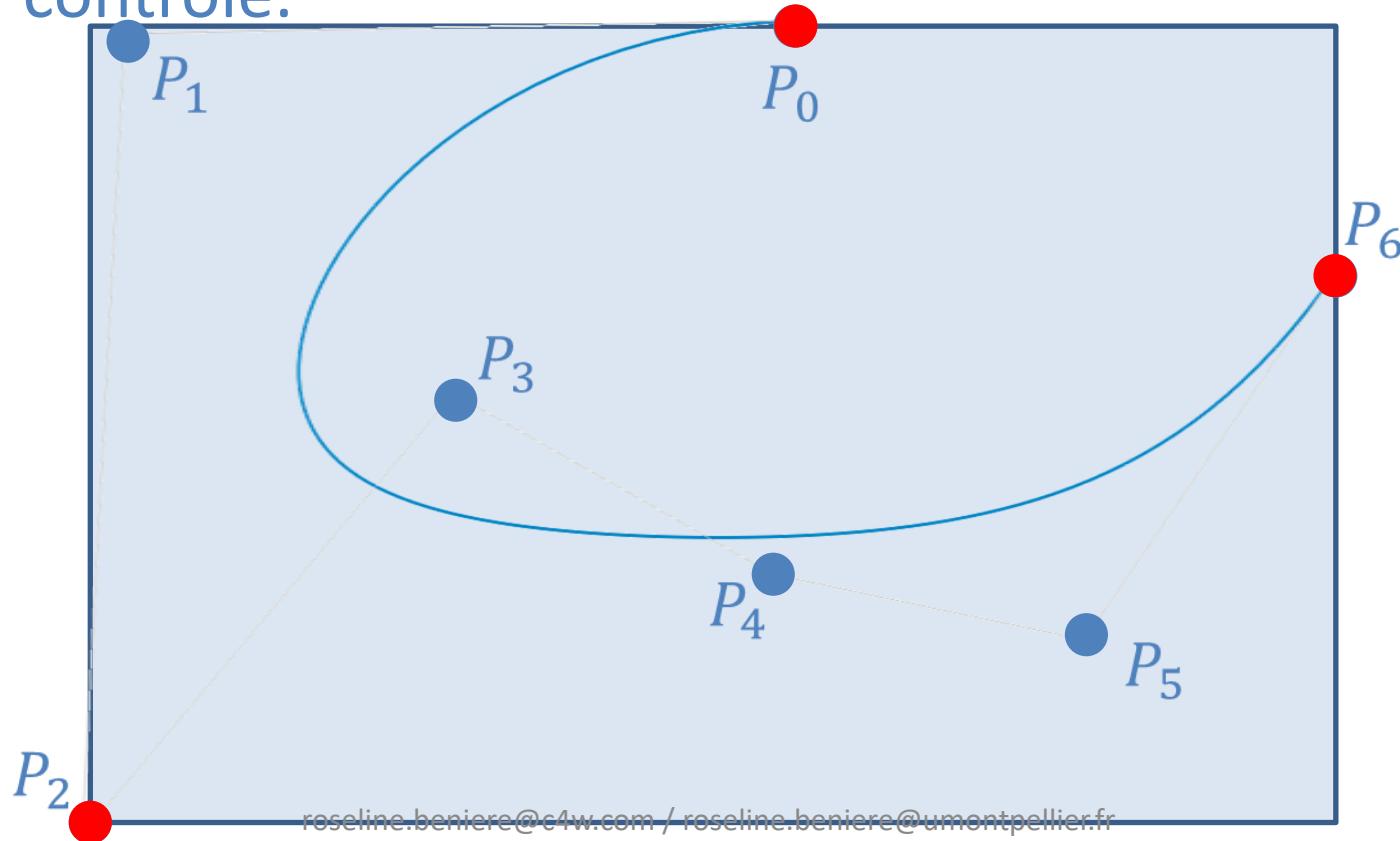
Courbes de Bézier

- Boite englobante:
 - correspond à la boite englobante des points de contrôle.



Courbes de Bézier

- Boite englobante:
 - correspond à la boite englobante des points de contrôle.



Courbes de Bézier

- Algorithme de *De Casteljau* :
 - cet algorithme s'appuie sur la formule de récurrence :

$$p(u) = \sum_{i=0}^n B_i^n(u) P_i^0 = \sum_{i=0}^{n-1} B_i^{n-1}(u) P_i^1 = \dots = \sum_{i=0}^0 B_i^0(u) P_i^n$$

Courbes de Bézier

- Algorithme de *De Casteljau* :

➤ cet algorithme s'appuie sur la formule de récurrence :

$$p(u) = \sum_{i=0}^n B_i^n(u) P_i^0 = \sum_{i=0}^{n-1} B_i^{n-1}(u) P_i^1 = \dots = \sum_{i=0}^0 B_i^0(u) P_i^n$$

→ $P_i^{k+1} = (1-u)P_i^k + uP_{i+1}^k$

Courbes de Bézier

- Algorithme de *De Casteljau* :

➤ cet algorithme s'appuie sur la formule de récurrence :

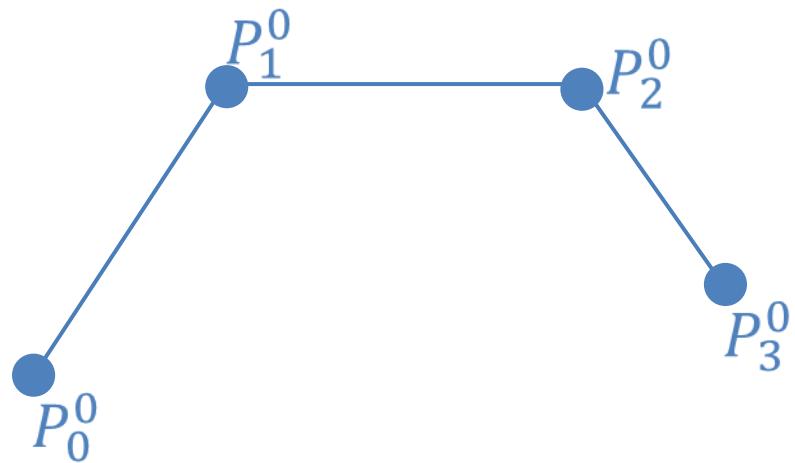
$$p(u) = \sum_{i=0}^n B_i^n (u) P_i^0 = \sum_{i=0}^{n-1} B_i^{n-1} (u) P_i^1 = \dots = \sum_{i=0}^0 B_i^0 (u) P_i^n$$

➡ $P_i^{k+1} = (1-u)P_i^k + uP_{i+1}^k$

Exemple avec :

- $n = 3$

- $u = \frac{1}{4}$



Courbes de Bézier

- Algorithme de *De Casteljau* :

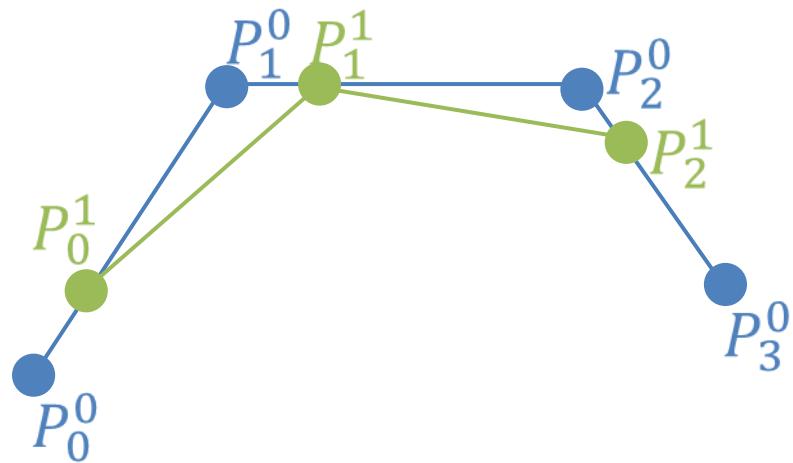
➤ cet algorithme s'appuie sur la formule de récurrence :

$$p(u) = \sum_{i=0}^n B_i^n (u) P_i^0 = \sum_{i=0}^{n-1} B_i^{n-1} (u) P_i^1 = \dots = \sum_{i=0}^0 B_i^0 (u) P_i^n$$

→ $P_i^{k+1} = (1-u)P_i^k + uP_{i+1}^k$

Exemple avec :

- $n = 3$
- $u = \frac{1}{4}$



Courbes de Bézier

- Algorithme de *De Casteljau* :

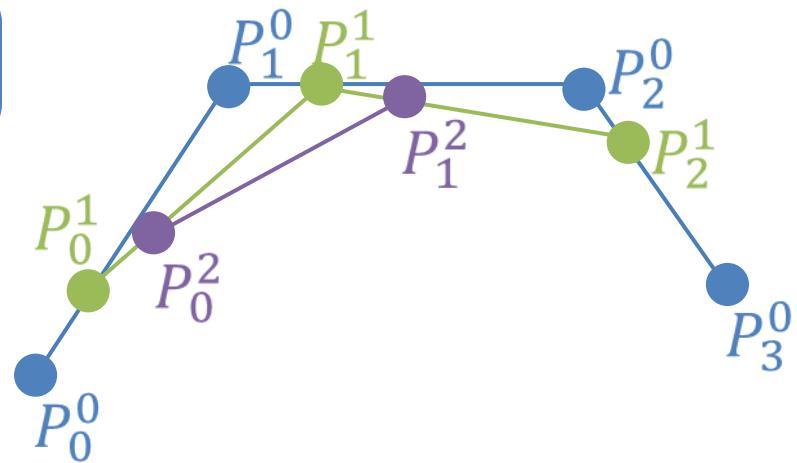
➤ cet algorithme s'appuie sur la formule de récurrence :

$$p(u) = \sum_{i=0}^n B_i^n (u) P_i^0 = \sum_{i=0}^{n-1} B_i^{n-1} (u) P_i^1 = \dots = \sum_{i=0}^0 B_i^0 (u) P_i^n$$

→ $P_i^{k+1} = (1-u)P_i^k + uP_{i+1}^k$

Exemple avec :

- $n = 3$
- $u = \frac{1}{4}$



Courbes de Bézier

- Algorithme de *De Casteljau* :

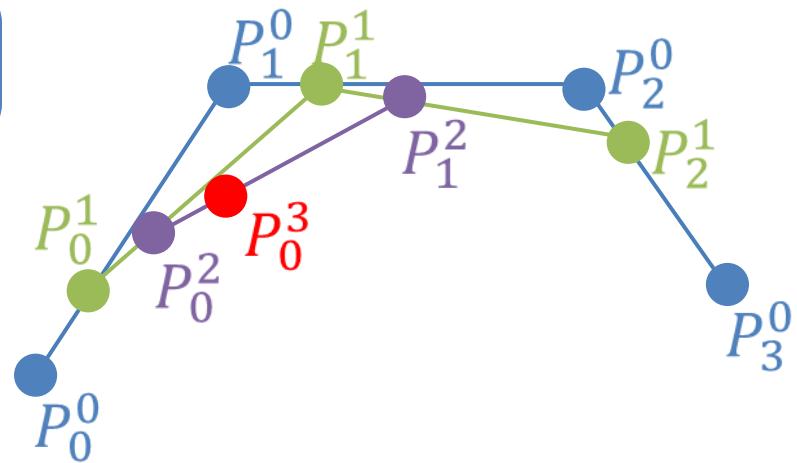
➤ cet algorithme s'appuie sur la formule de récurrence :

$$p(u) = \sum_{i=0}^n B_i^n (u) P_i^0 = \sum_{i=0}^{n-1} B_i^{n-1} (u) P_i^1 = \dots = \sum_{i=0}^0 B_i^0 (u) P_i^n$$

→ $P_i^{k+1} = (1-u)P_i^k + uP_{i+1}^k$

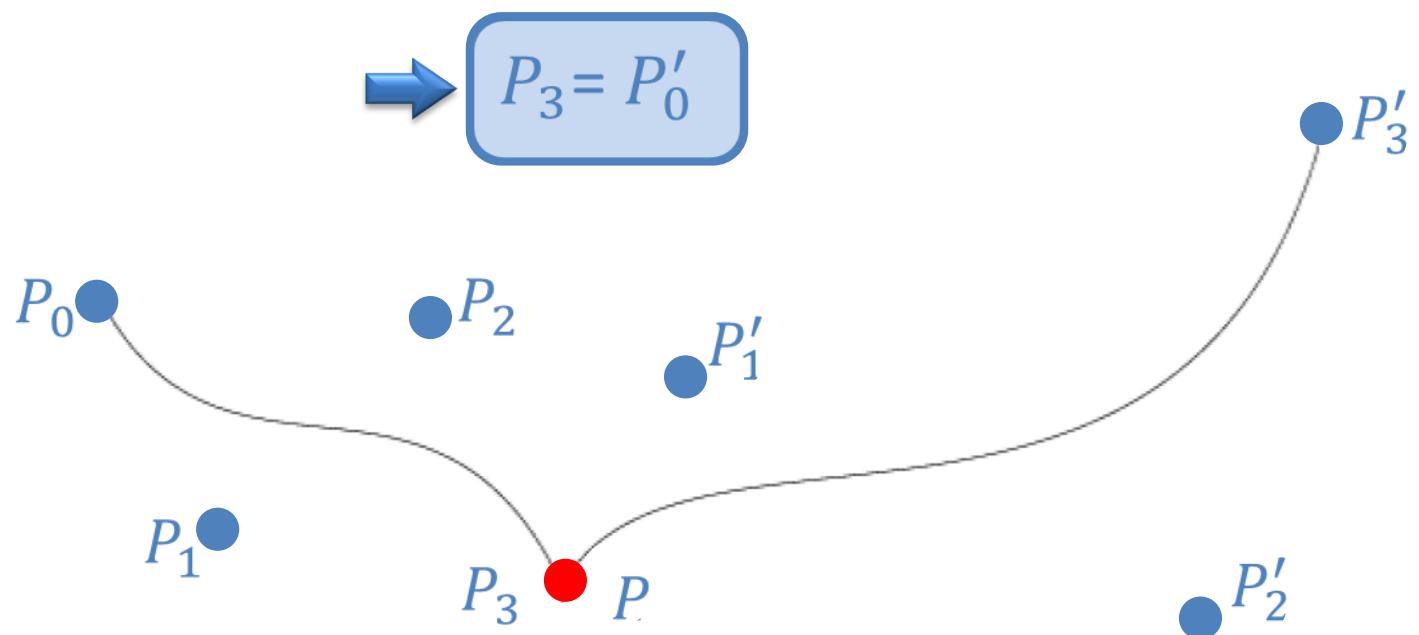
Exemple avec :

- $n = 3$
- $u = \frac{1}{4}$



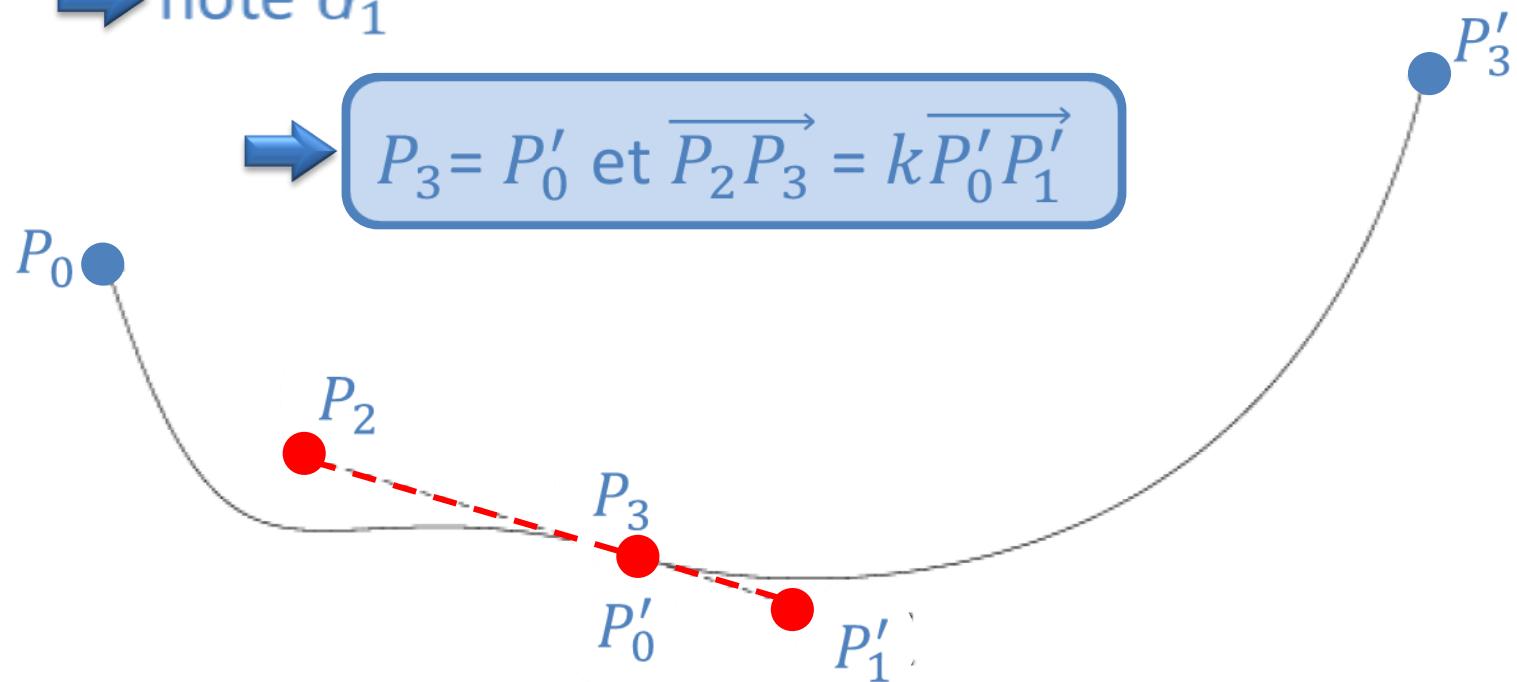
Courbes de Bézier

- Raccordement entre deux courbes bázier :
 - continuité de **position**, les deux points extrêmes sont confondus ➔ noté C_0 ou G_0



Courbes de Bézier

- Raccordement entre deux courbes bázier :
 - continuité de classe G_1 , les deux points extrêmes sont confondus et les segments sont alignés
noté G_1



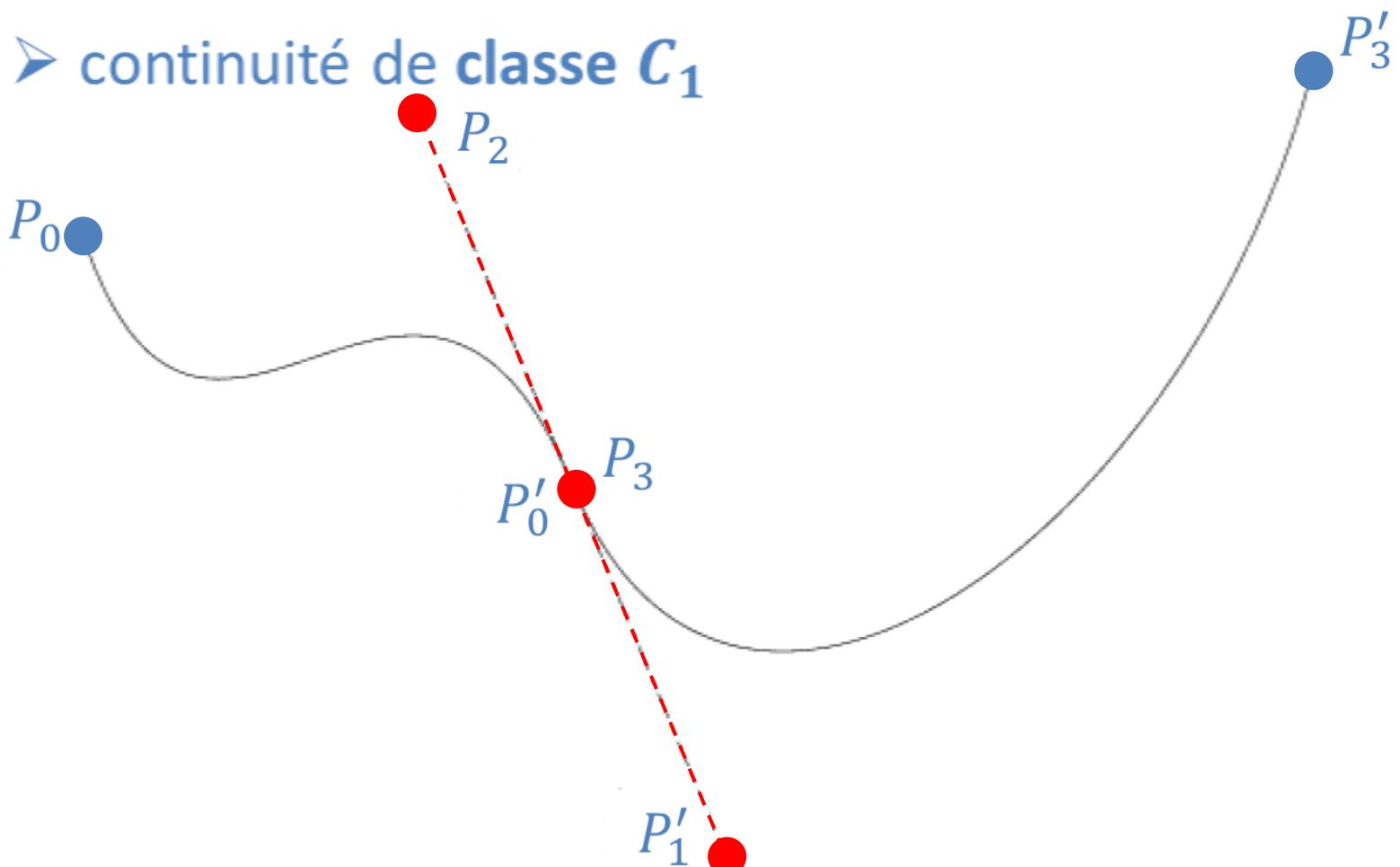
Courbes de Bézier

- Raccordement entre deux courbes b茅zier :
 - continuit茅 de classe **C_1** , les deux points extr猫mes sont confondus et situ茅s au milieu du point qui pr茅c猫de et de celui qui suit ➔ not茅 C_1

$$\rightarrow P_3 = P'_0 \text{ et } \overrightarrow{P_2 P_3} = \overrightarrow{P'_0 P'_1}$$

Courbes de Bézier

- Raccordement entre deux courbes bázier :



Courbes de Bézier

- **Raccordement entre deux courbes bázier :**
 - les continuités C_1 ou G_1 sont liés à la tangente et donc à la dérivée première de la courbe pour le point de raccord;

Courbes de Bézier

- Raccordement entre deux courbes bázier :
 - les continuités C_1 ou G_1 sont liés à la tangente et donc à la dérivée première de la courbe pour le point de raccord;
 - de la même manière on peut définir les continuités C_2 et G_2 en s'appuyant sur les dérivées secondes,
 - et ainsi de suite ...

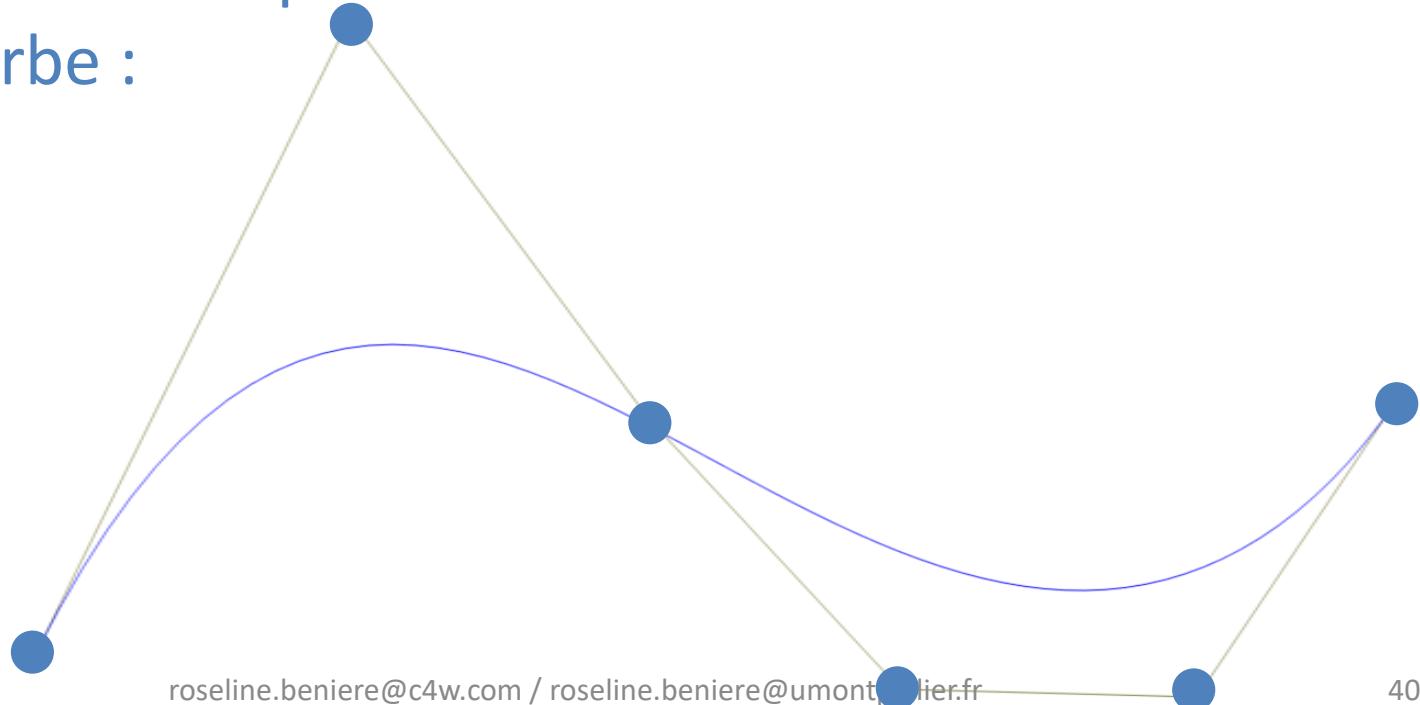
Courbes de Bézier

- Inconvénients :
 - le degré de la courbe est lié au nombre de point de contrôle;

Courbes de Bézier

- Inconvénients :

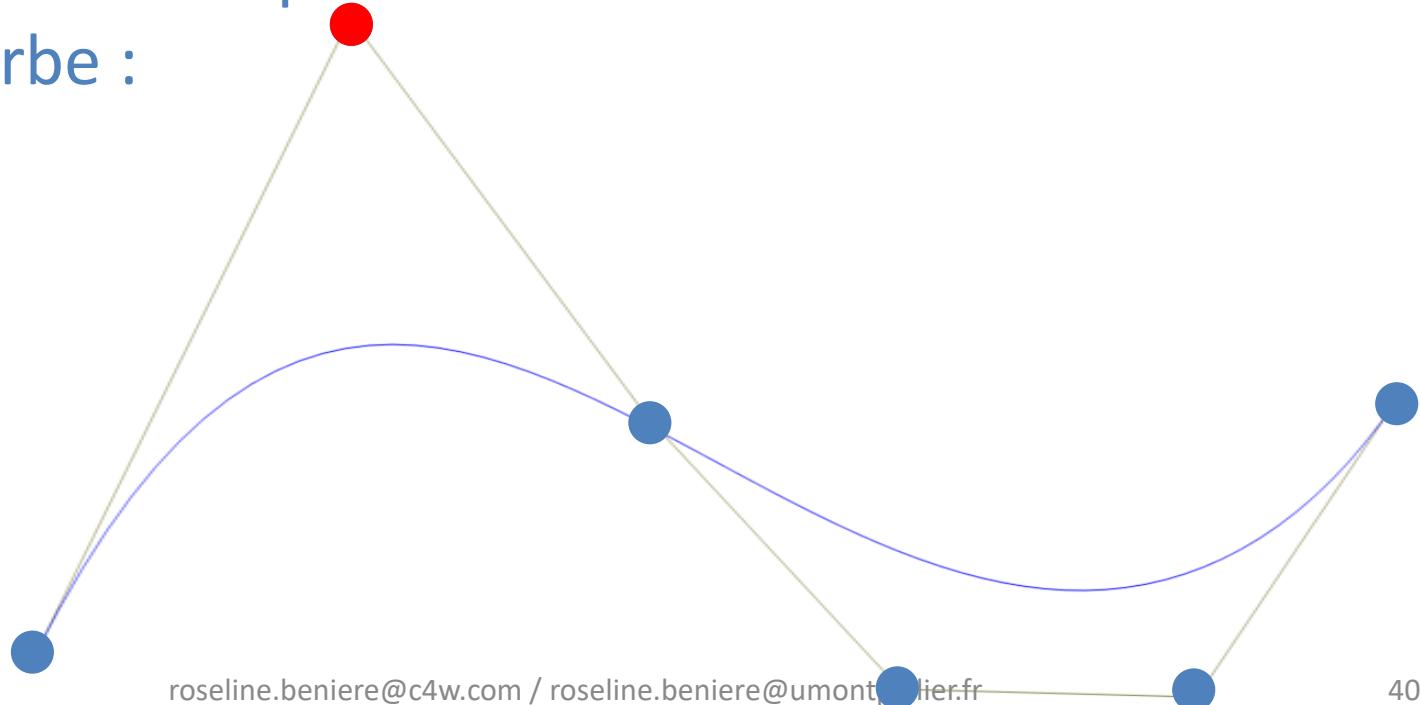
- le degré de la courbe est lié au nombre de point de contrôle;
- modifier un point de contrôle modifie toute la courbe :



Courbes de Bézier

- Inconvénients :

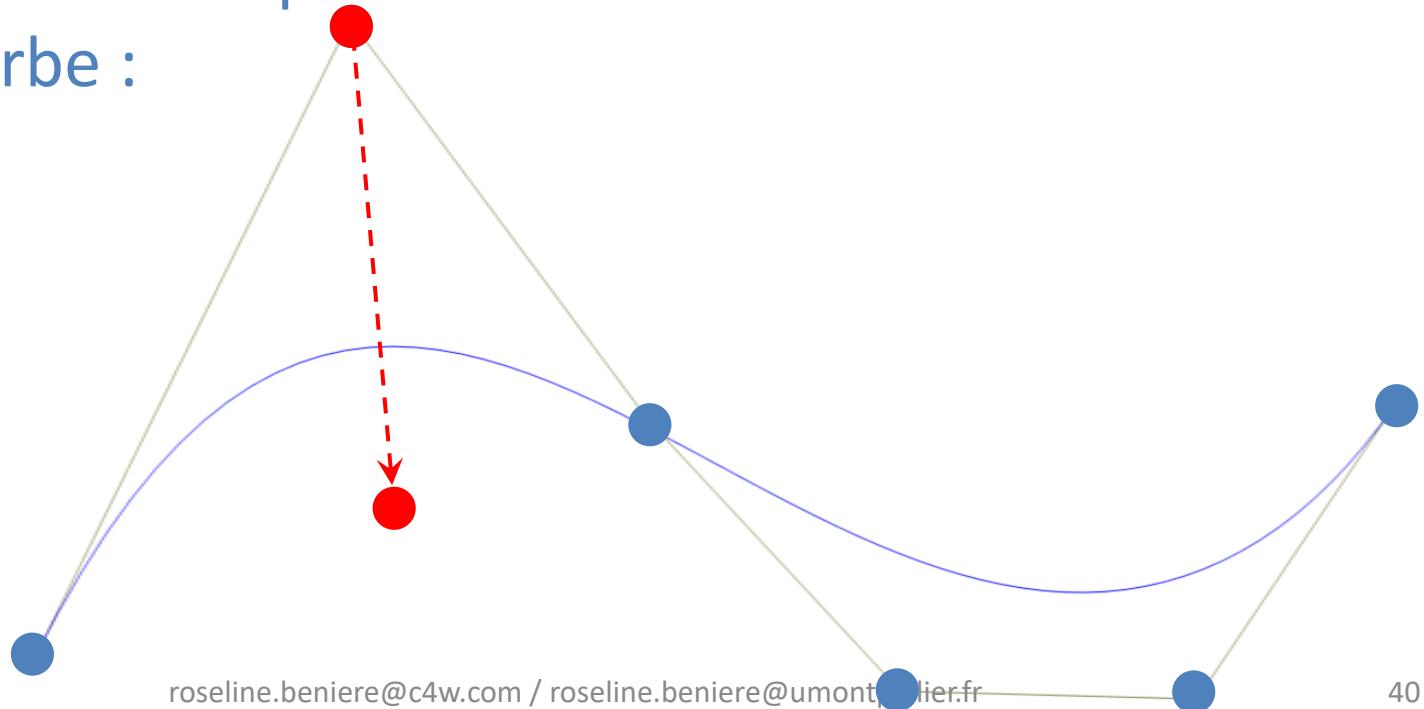
- le degré de la courbe est lié au nombre de point de contrôle;
- modifier un point de contrôle modifie toute la courbe :



Courbes de Bézier

- Inconvénients :

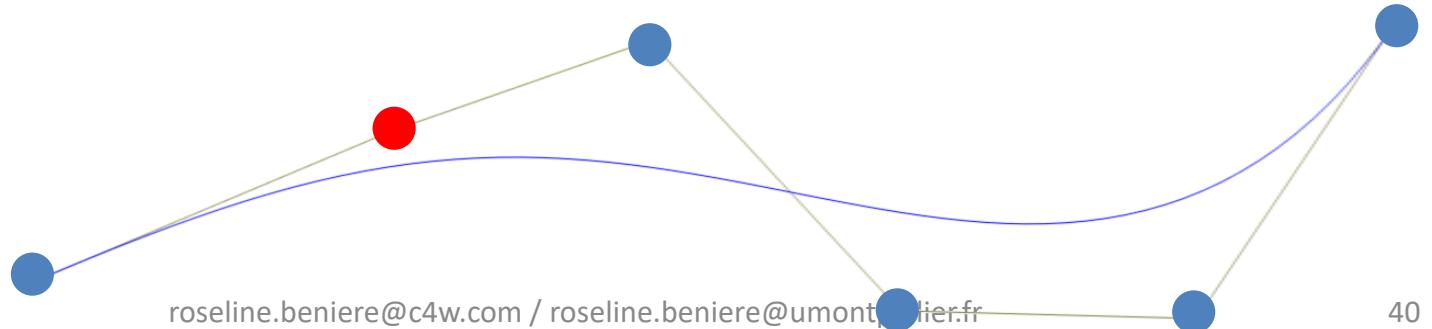
- le degré de la courbe est lié au nombre de point de contrôle;
- modifier un point de contrôle modifie toute la courbe :



Courbes de Bézier

- Inconvénients :

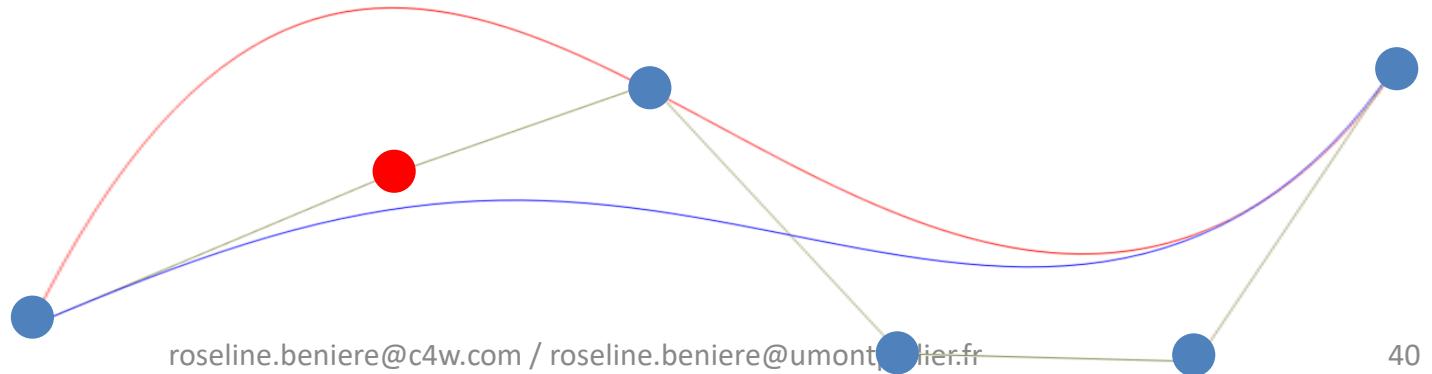
- le degré de la courbe est lié au nombre de point de contrôle;
- modifier un point de contrôle modifie toute la courbe :



Courbes de Bézier

- Inconvénients :

- le degré de la courbe est lié au nombre de point de contrôle;
- modifier un point de contrôle modifie toute la courbe :



Plan

- Introduction
- Représentation d'une droite et d'une courbe
- Géométrie différentielle
- Cubiques
- Courbes de Bézier
- Courbes B-Splines / NURBS
- Dessin en OpenGL

Courbes B-Splines

- **Une courbe B-Spline est :**
 - courbe définie par morceaux,
- **et est définie par :**
 - un ensemble de point de contrôle,
 - un vecteur de nœud,
 - un ensemble de multiplicité,
 - un ensemble de poids.

Courbes B-Splines

- Vecteur de nœud :
 - correspond à l'espace de définition de la courbe,
 - ne peut pas être infini.

Courbes B-Splines

- **Vecteur de nœud :**
 - correspond à l'espace de définition de la courbe,
 - ne peut pas être infini.
- **Multiplicité :**
 - défini pour chaque point de contrôle,
 - permet de gérer la continuité de la courbe (si multiplicité \nearrow continuité \searrow).

Courbes B-Splines

- **Vecteur de nœud :**
 - correspond à l'espace de définition de la courbe,
 - ne peut pas être infini.
- **Multiplicité :**
 - défini pour chaque point de contrôle,
 - permet de gérer la continuité de la courbe (si multiplicité \nearrow continuité \searrow).
- **Poids**
 - = 1 pour chaque point de contrôle.

Courbes B-Splines

- Pas les inconvénients d'une Bézier :
 - le degré de la courbe peut être fixé quelque soit le nombre de point de contrôle

Courbes B-Splines

- Pas les inconvénients d'une Bézier :
 - le degré de la courbe peut être fixé quelque soit le nombre de point de contrôle
- $\rightarrow \text{nbNoeud} = \text{nbPoint} + 1 + \text{degré}$

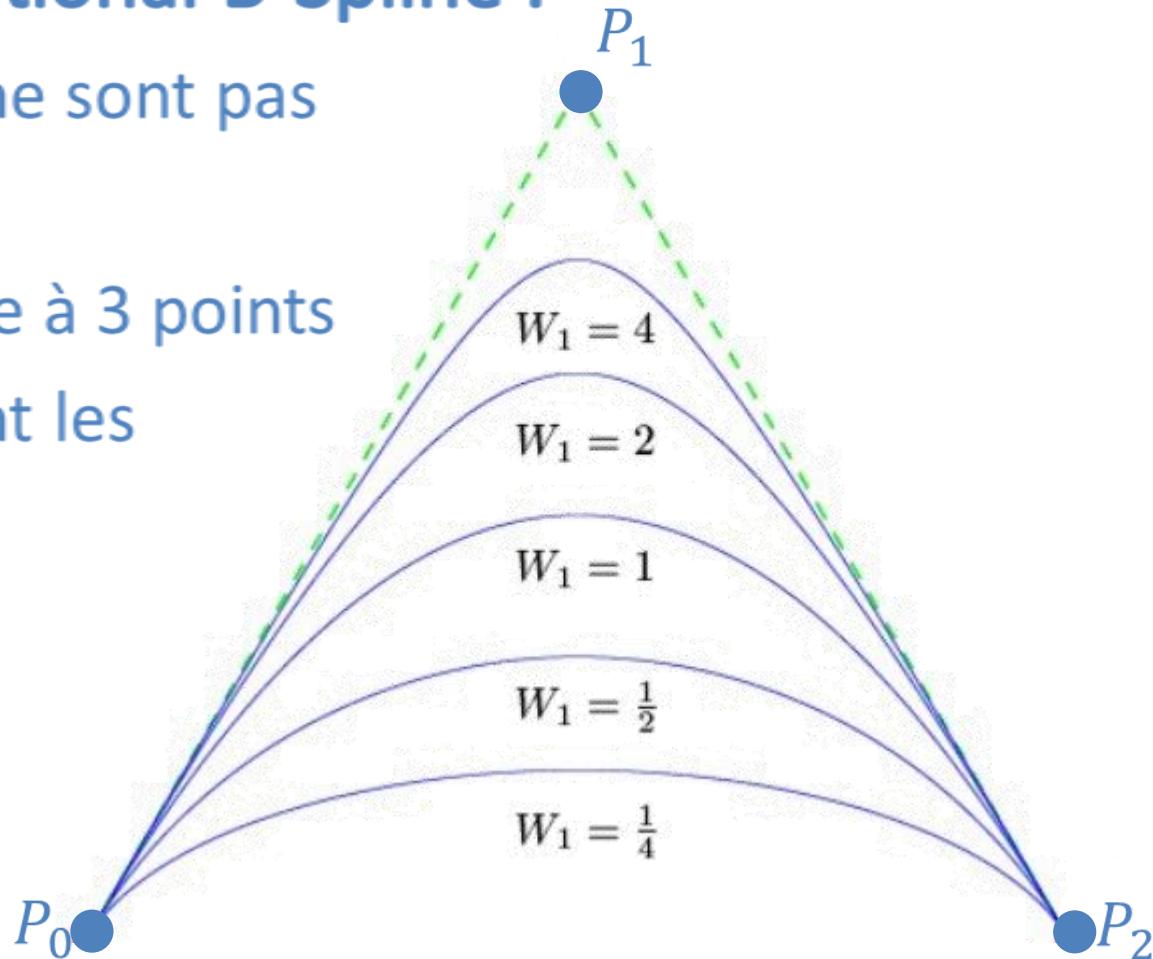
Courbes B-Splines

- Pas les inconvénients d'une Bézier :
 - le degré de la courbe peut être fixé quelque soit le nombre de point de contrôle
 - $\text{nbNoeud} = \text{nbPoint} + 1 + \text{degré}$
 - modifier un point de contrôle ne modifie qu'une partie de la courbe. Entre le point de contrôle situé à ``degré à gauche ''et celui situé à degré à droite.
 - ex : si degré = 2, modifié le point P_5 , modifie la courbe entre les points P_3 et P_7 .

NURBS

- Non-Uniform Rational B-Spline :

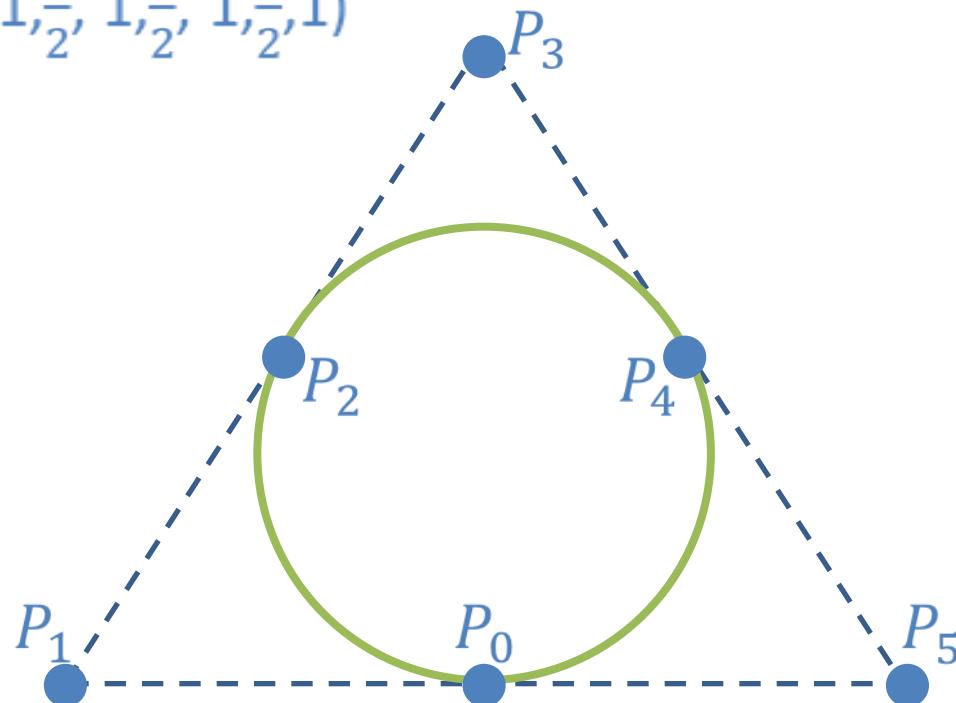
- tous les poids ne sont pas égaux à 1,
- exemple courbe à 3 points de contrôle dont les poids sont :
 - $W = (1, W_1, 1)$



NURBS

- Pour définir des courbes particulières :
 - cercle sous forme de NURBS,

$$\circ W = \left(1, \frac{1}{2}, 1, \frac{1}{2}, 1, \frac{1}{2}, 1\right)$$



Plan

- Introduction
- Représentation d'une droite et d'une courbe
- Géométrie différentielle
- Cubiques
- Courbes de Bézier
- Courbes B-Splines / NURBS
- Dessin en OpenGL

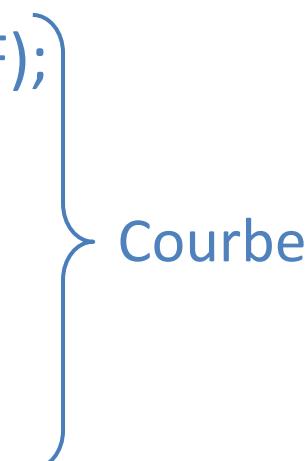
Dessin en OpenGL

- On ne peut pas donner à OpenGL une courbe continue
 - calculer un ensemble de points,
 - définir autant de u que l'on veut de points,
 - plus on aura de point et plus la courbe sera “lisse”.

Dessin en OpenGL

- On ne peut pas donner à OpenGL une courbe continue
 - calculer un ensemble de points,
 - définir autant de u que l'on veut de points,
 - plus on aura de point et plus la courbe sera "lisse".
- Pour dessiner choisir entre
 - GL_LINES ➔ donner un liste de segment
 - GL_LINE_STRIP ➔ donner une liste de points

Dessin en OpenGL

- On énonce les sommets à la suite les uns des autres :
 - `glBegin(GL_LINE_STRIP);`
 - `glColor3f(0.0F, 1.0F, 0.0F);`
 - `glVertex3f(x, y, z);`
 - `glVertex3f(x, y, z);`
 - `glVertex3f(x, y, z);`
 - `glVertex3f(x, y, z);`
 - ...
 - `glEnd();`
- 
- Courbe

Conclusion

- **Courbe paramétrique:**
 - est définie par une fonction
 - pour calculer les points on fait courir un paramètre le long de la courbe.

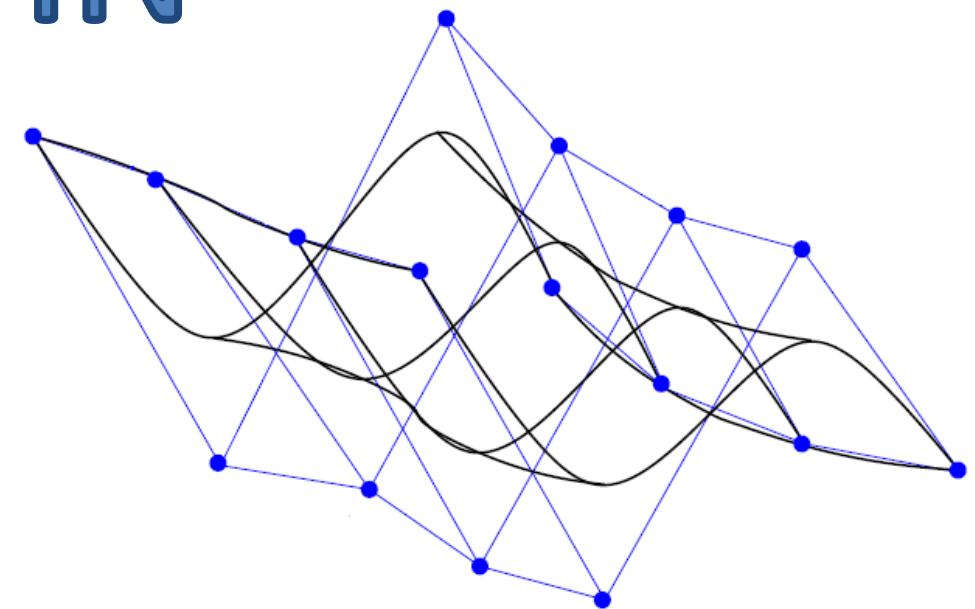
Conclusion

- **Courbe paramétrique:**
 - est définie par une fonction
 - pour calculer les points on fait courir un paramètre le long de la courbe.
- **Plusieurs types de courbes :**
 - cubiques
 - courbes de Bézier
 - courbes B-Splines ou NURBS

FIN

Surfaces paramétriques

lundi 04/02



Pour récupérer les cours et le TD/TP:
Moodle => HMIN212

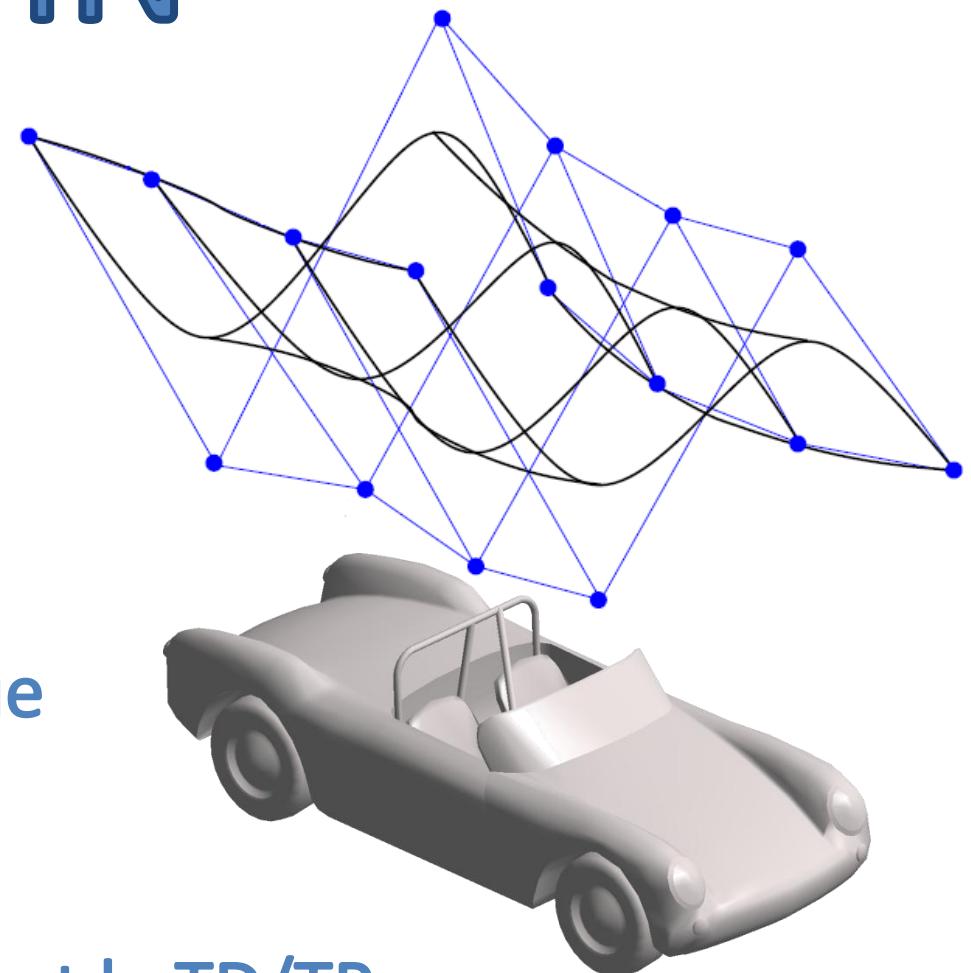
FIN

Surfaces paramétriques

lundi 04/02

Représentation surfacique

lundi 11/02



Pour récupérer les cours et le TD/TP:
Moodle => HMIN212

Sources

- Cours utilisés pour ce support :
 - Gilles Gesquière (Gamagora Lyon)
 - Loic Barthe (IRIT-UPS Toulouse)
 - Christian Jacquemin (LIMSI Paris11)
 - Marc Daniel (LSIS Marseille)
 - Vincent Legat (Louvain)