Courbes de Bézier

 —

Courbes paramétriques ? (1/7)

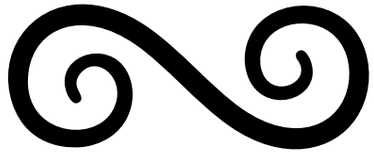
- Équations (système d'équations) permettant de représenter des courbes qui ne sont pas (que) des fonctions

    - Pour rappel une **fonction** d'un point de vue ensembliste est une application entre deux ensembles qui met (ou peut mettre) en relation un élément de l'ensemble de départ avec au maximum un élément de l'ensemble d'arrivée

      $\Rightarrow$ une image ne peut avoir qu'un seul antécédent.

      Pour une espace bidimensionnel on écrira $y = f(x)$ si f est définie pour la valeur x.
      Ainsi, par exemple, un cercle n'est pas une fonction mais plutôt l'assemblage du résultat de deux fonctions.

    - Ce dessin n'est donc pas représentable à l'aide d'une fonction

    - Comment faire donc pour représenter une telle courbe ?
      💡 imaginer que cette courbe est le résultat du déplacement d'un **point** $\mathscr{P}$ au cours du temps

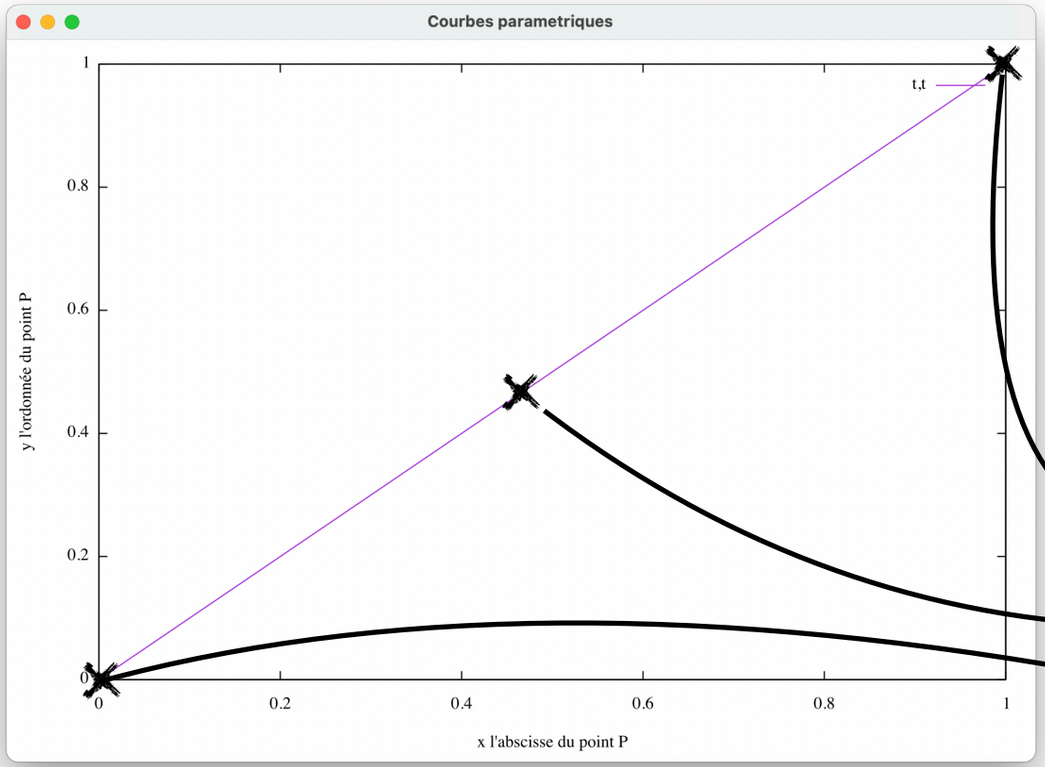
    - $\Rightarrow$ chaque coordonnée de ce **point** $\mathscr{P}$ serait donc une fonction dépendante d'un paramètre, disons le temps, et notons-le t. Ceci est vrai quelque soit la dimension de l'espace considéré.

Courbes paramétriques ? (2/7)

- Sans obligation, nous prendrons $t \in [0; 1]$ afin de normaliser le début de la courbe à $t = 0$ et sa fin à $t = 1$
- La courbe paramétrique serait la succession des positions du point $\mathscr{P}$ (qu'on pourrait aussi appelé « le mobile $\mathscr{M}$ ») pour $t : 0 \to 1$
  - Si le point est défini dans un espace 2D, on peut écrire :
    pour $\mathscr{P}\begin{pmatrix} x \\ y \end{pmatrix} \in \mathscr{C}$, la courbe $\mathscr{C}$ est obtenue par le système
    $$\forall t \in [0; 1], \begin{cases} x = f(t) \\ y = g(t) \end{cases}$$
    où f et g sont des fonctions définies dans l'intervalle $[0; 1]$
  - Si le point est défini dans un espace 3D :
    pour $\mathscr{P}\begin{pmatrix} x \\ y \\ z \end{pmatrix} \in \mathscr{C}$, la courbe $\mathscr{C}$ est obtenue par le système
    $$\forall t \in [0; 1], \begin{cases} x = f(t) \\ y = g(t) \\ z = h(t) \end{cases}$$
    où f, g et h sont des fonctions définies dans l'intervalle $[0; 1]$
  - …

Courbes paramétriques ? (3/7)

- Exemple (très) simple de courbe paramétrique :
  - $\mathscr{C}$ telle que : $\forall t \in [0;1], \begin{cases} x = t \\ y = t \end{cases}$



Produit avec gnuplot

```
set parametric
set trange[0:1]
set xlabel "x l'abscisse du point P"
set ylabel "y l'ordonnée du point P"
plot t,t
```

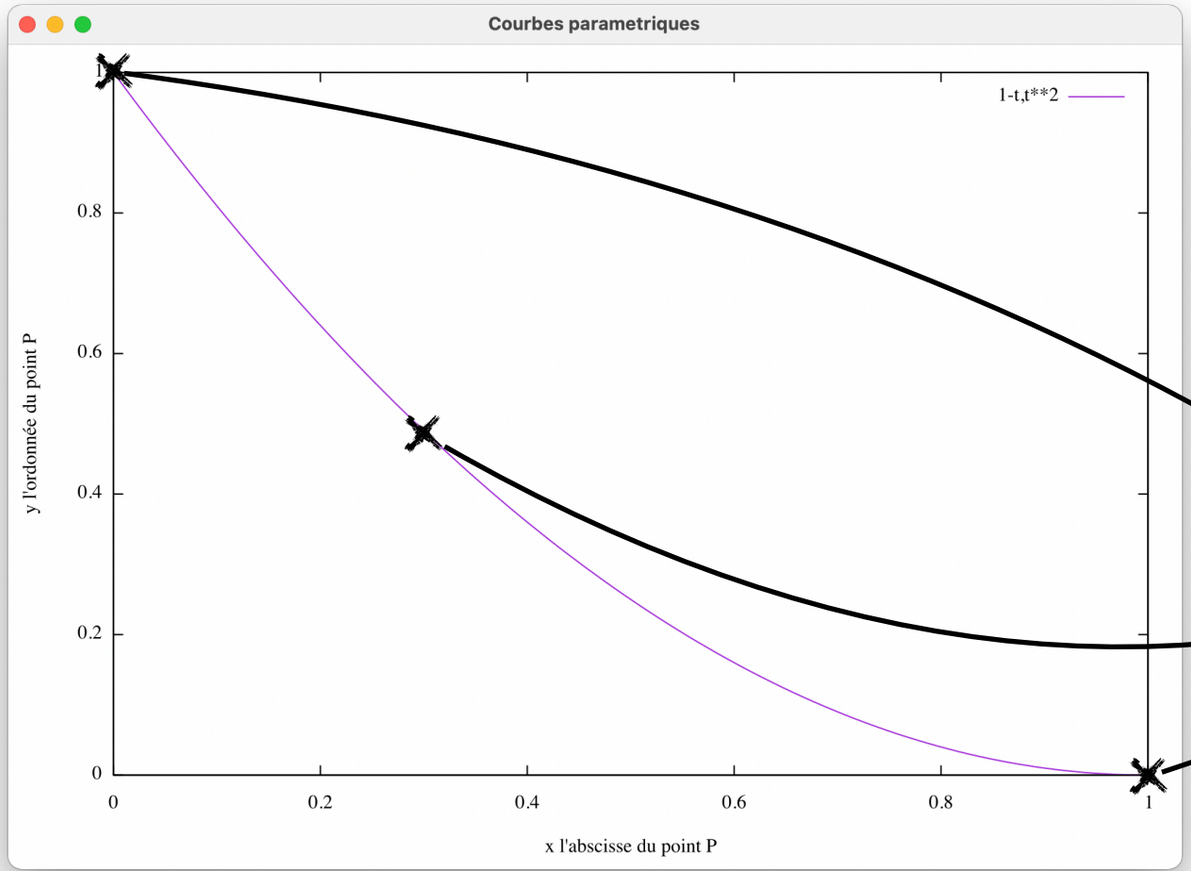
$t = 1$

$t = 0.5$

$t = 0$

Courbes paramétriques ? (4/7)

- Exemple simple de courbe paramétrique :
  - $\mathscr{C}$ telle que : $\forall t \in [0;1], \begin{cases} x = 1 - t \\ y = t^2 \end{cases}$



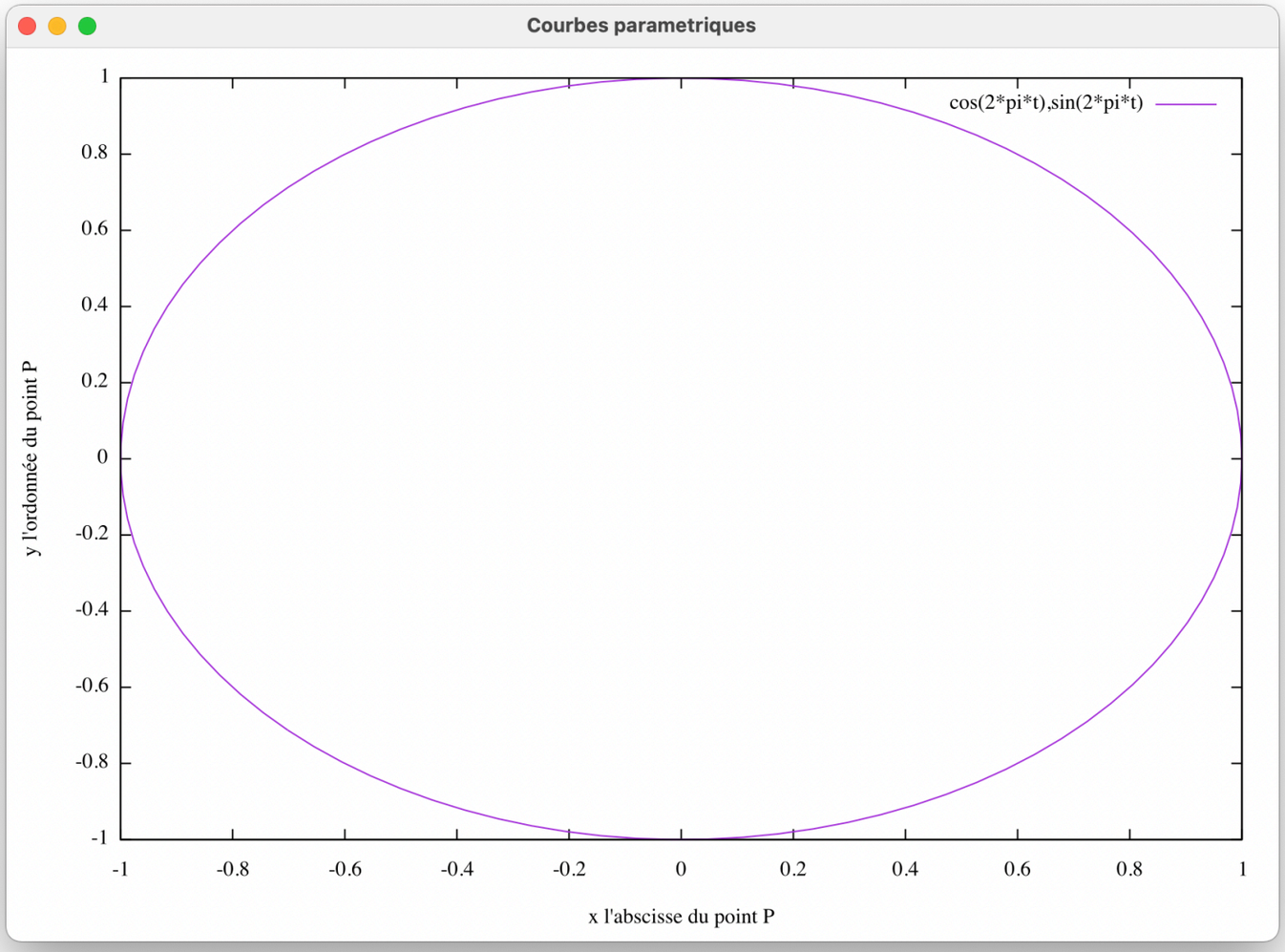
Produit avec gnuplot

```
set parametric
set trange[0:1]
set xlabel "x l'abscisse du point P"
set ylabel "y l'ordonnée du point P"
plot 1-t,t**2
```

$t = 1$ $t = 0.7$ $t = 0$

Courbes paramétriques ? (5/7)

- Exemple de courbe paramétrique :
  - $\mathscr{C}$ telle que : $\forall t \in [0;1], \begin{cases} x = cos(2 \times \pi \times t) \\ y = sin(2 \times \pi \times t) \end{cases}$

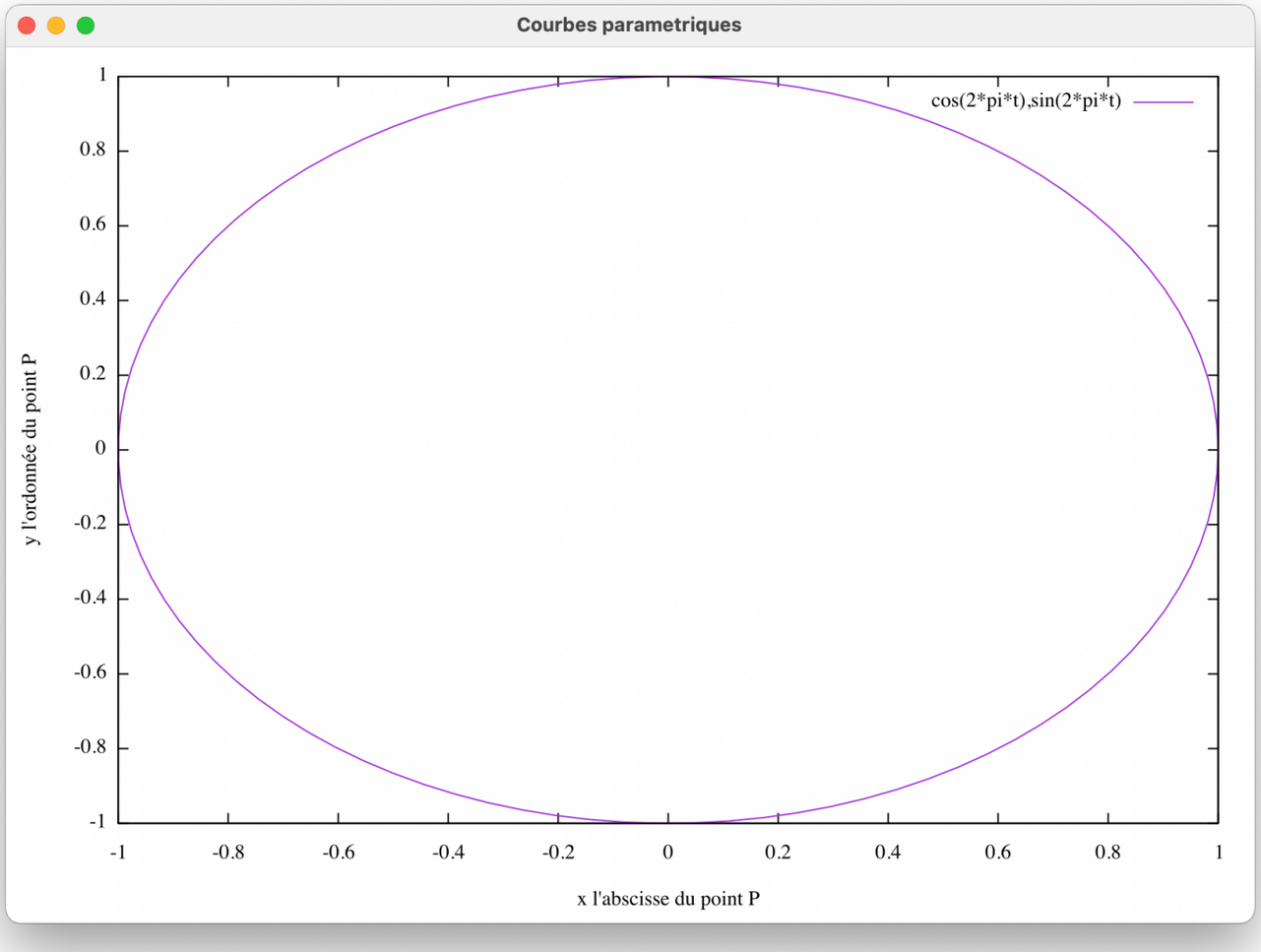


Produit avec gnuplot

```
set parametric
set trange[0:1]
set xlabel "x l'abscisse du point P"
set ylabel "y l'ordonnée du point P"
plot cos(2*pi*t),sin(2*pi*t)
```


Courbes paramétriques ? (6/7)

- Exemple de courbe paramétrique :
  - $\mathscr{C}$ telle que : $\forall t \in [0;1], \begin{cases} x = cos(2 \times \pi \times t) \\ y = sin(2 \times \pi \times t) \end{cases}$

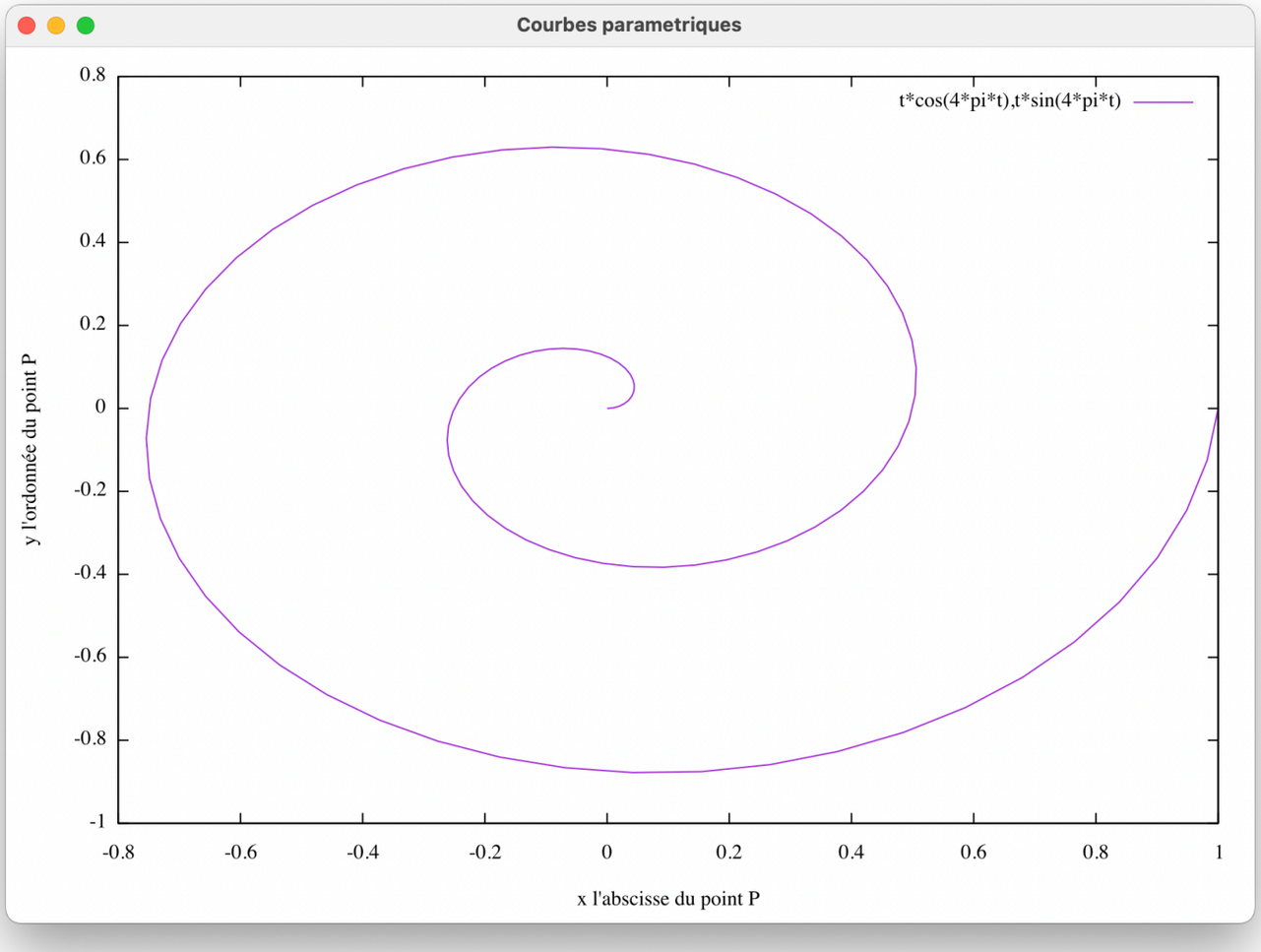


Produit avec gnuplot

```
set parametric
set trange[0:1]
set xlabel "x l'abscisse du point P"
set ylabel "y l'ordonnée du point P"
plot cos(2*pi*t),sin(2*pi*t)
```

Courbes paramétriques ? (7/7)

- Exemple de courbe paramétrique :
  - $\mathscr{C}$ telle que : $\forall t \in [0;1], \begin{cases} x = t \times cos(4 \times \pi \times t) \\ y = t \times sin(4 \times \pi \times t) \end{cases}$



Produit avec gnuplot

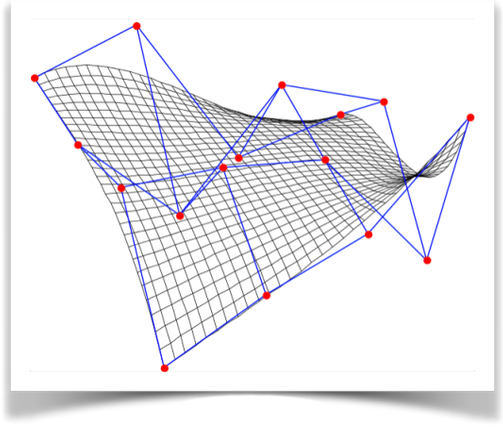
```
set parametric
set trange[0:1]
set xlabel "x l'abscisse du point P"
set ylabel "y l'ordonnée du point P"
plot t*cos(4*pi*t),t*sin(4*pi*t)
```

Défaut : la forme générale des courbes paramétrique est difficilement contrôlable

⇒ Courbes de Bézier

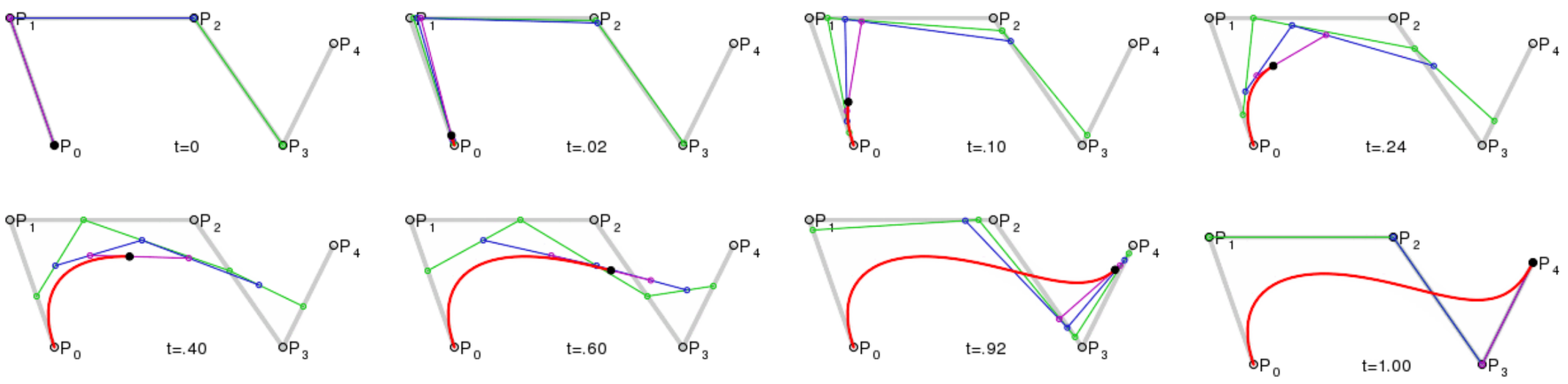
Courbes de Bézier ?

- Problème de CAO (carrosserie automobile → surfaces)
  - Pierre Bézier (en 1962 pour Renault)
  - Paul de Casteljau (en 1959 pour Citroën) 🤫
- → Courbes **polynomiales** paramétriques



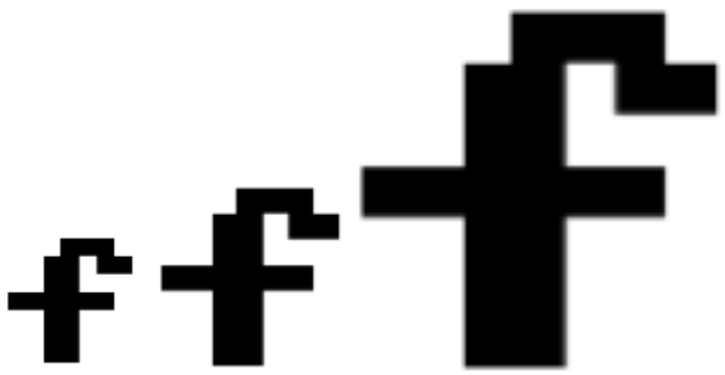
Surface de Bézier (source Wikipédia)

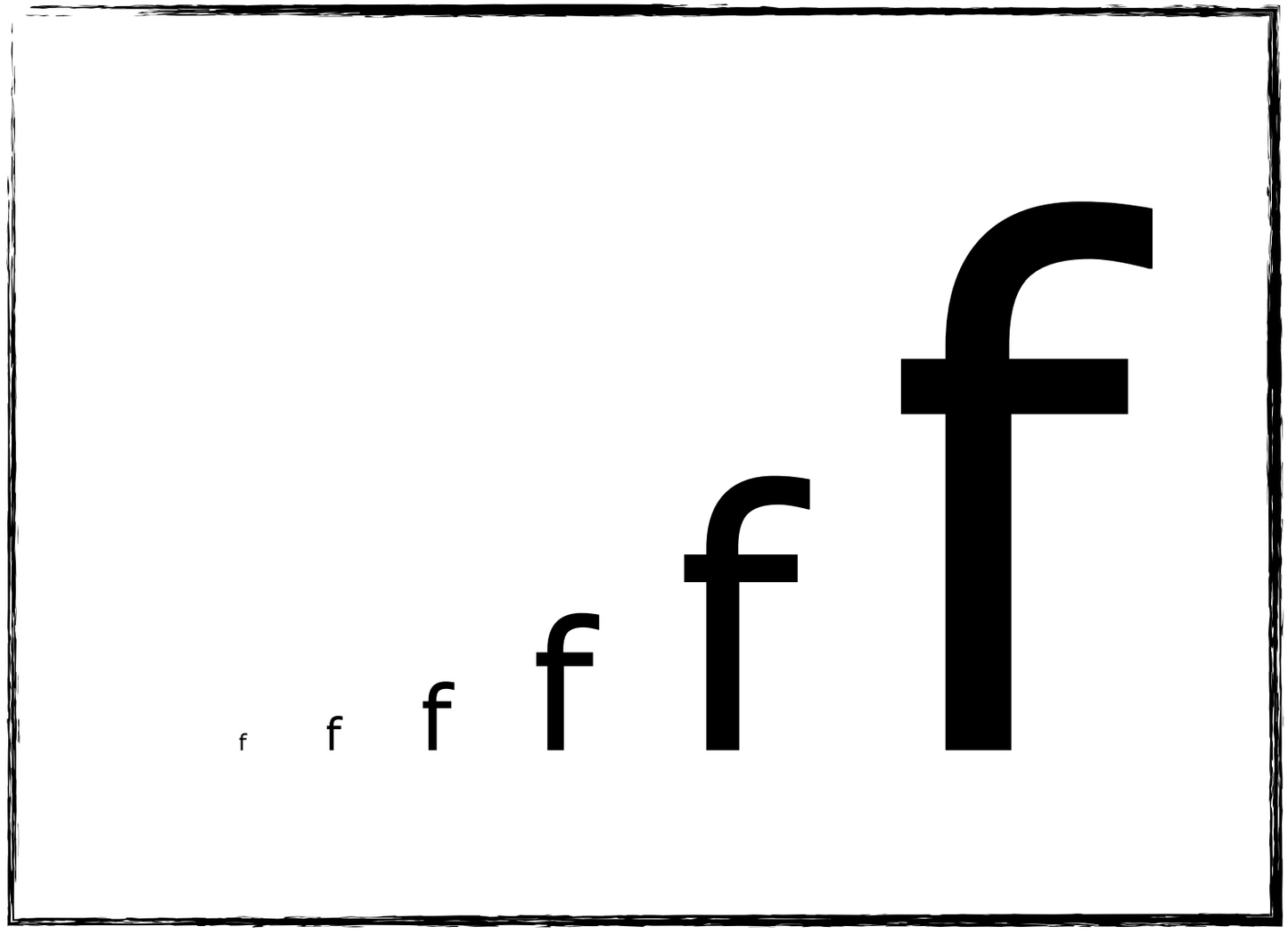
- La base est de produire une interpolation entre des paires de points tout en calculant des barycentre successifs



https://upload.wikimedia.org/wikipedia/commons/c/c6/Bezier_forth_anim.gif

Les courbes de Bézier sont partout !





Vers 1982, John Warnock travaille sur un système de description des polices de caractères en se basant sur des courbes de Bézier (les *B-splines*) → naissance du format **PostScript** (format — c'est un langage — non compressé du pdf, directement interprétable par les imprimantes) et d'**Adobe** → naissance du TTF

Représentation paramétrique de la courbe de Bézier

- Sous la forme d'un polynôme (tout en ayant une forme paramétrique $\mathscr{P}(t)$, soit on cherche les positions du point $\mathscr{P}$ en fonction de t)

- Ce polynôme, de degré n, dépend de $(n+1)$ points, notés $\mathscr{P}_i$, qui donnent la forme de la courbe. Ces points sont appelés « points de contrôle » ; ils ne sont pas tous atteignables.

- En pratique, on a la garantie que le premier point $\mathscr{P}_0$, ainsi que le dernier $\mathscr{P}_n$, sont atteignables. Pour les autres, ceci peut se produire s'il y a un alignement.

- La forme générale de cette représentation paramétrique est :

$$\mathscr{P}(t) = \sum_{i=0}^{n} \mathscr{B}_i^n(t) \times \mathscr{P}_i$$

avec $t \in [0;1]$ et $\mathscr{B}_i(t)$ est un polynôme de Bernstein de degré n

Polynômes de Bernstein

- Fonction polynomiale utilisant les coefficients binomiaux :

$$C_n^p = \frac{n!}{p!(n-p)!}$$

- Elle s’écrit :

$$B_i^n(t) = C_n^i \times t^i \times (1-t)^{n-i}$$

Comment produire une courbe de Bézier ?

1. Proposer $(n+1)$ points de contrôle pour produire une courbe d'ordre n, puis, pour un t qui varie de $0 \to 1$, soit :

    1. Calculer et afficher tous les $\mathscr{P}(t)$ obtenus ;
    2. Réaliser deux à deux les calculs d'interpolation (soit (n) moyennes pondérées par $(1-t)$ et t entre deux points, cf. cours précédent) entre les $(n+1)$ points de contrôle successifs ($\mathscr{P}_0$ avec $\mathscr{P}_1$, $\mathscr{P}_1$ avec $\mathscr{P}_2$, …, $\mathscr{P}_{n-1}$ avec $\mathscr{P}_n$).
    Le résultat est (n) **nouveaux points !**
    → répéter l'opération avec les nouveaux points … et encore … et encore (donc récursivement)
    → **JUSQU'À** obtenir un dernier unique point (l'interpolation des interpolations successives) : **ce dernier point est donc celui qui est sur la courbe de Bézier à l'instant t !**

Aller plus loin

1. Autre solution pour dessiner une courbe de Bézier :
   → Voir l'algorithme récursif de De Casteljau

2. Voir Aussi :

   1. Les *B-Splines*

   2. Les *NURBS* ...

Pour finir (DM)

- Partir du code https://expreg.org/amsi/C/APG2223S1/code/sc_00_02_bezier-0.1.tgz permettant de gérer et déplacer des points de contrôle (ici 4)
- Implémentez la méthode de votre choix de calcul de la courbe de Bézier
- (BONUS) Vous vous donner la possibilité d'ajouter d'autres points
- Vous déposez votre travail sur moodle