Tracé interactif de courbes

Utilisation en Conception Assistée par Ordinateur

1 Courbes paramétriques

1.1 *Définition*

Soient f et g deux fonctions définies et continues respectivement sur les intervalles D_f et D_g de R. Soit une variable réelle t. Les points M(x,y) définis par $\begin{cases} x = f(t) \\ y = g(t) \end{cases}$ pour toutes les valeurs de t appartenant à $Df \cap Dg$, définissent une courbe. Le système d'équations ci-dessus en est la représentation paramétrique et la variable t le paramètre.

1.2 *Les classiques*

Les courbes paramétriques les plus classiques sont les courbes polynomiales comme la droite ou les coniques : ellipses, paraboles, hyperboles.

1.2.1 Travail à faire

Avec MatLab tracer les graphiques des courbes définies par les équations paramétriques ci-dessous.

$\begin{cases} x(t) = t \\ y(t) = 1 - t \end{cases}$ pour 0<t<5

$\begin{cases} x(t) = \sin(3t) \\ y(t) = \sin(4t) \end{cases}$ pour 0<t<6.3

$\begin{cases} x(t) = \cos(t)^3 \\ y(t) = \sin(t)^3 \end{cases}$ pour -5<t<5

$\begin{cases} x(t) = \cos(t) \\ y(t) = t/2 + \sin(t) \end{cases}$ pour 0<t<2π

$\begin{cases} x(t) = \cos(t) \\ y(t) = \sin(t)^2 / (2 + \sin(t)) \end{cases}$ pour 0<t<12

Pour pouvoir observer la construction de la courbe, on trace un point après l'autre et on rafraichit la fenêtre à chaque tracer. Utiliser la portion de code suivante :

```
% tableau des valeurs de t : prendre 500 points pour visualiser la courbe
 t = ...
for i=1 : length(t)
   tt = t(i); % un seul point à la fois
   x =...;
   y =...;
   plot(x,y,'r.')
   drawnow  % force le rafraichissement de la fenêtre graphique
end
```

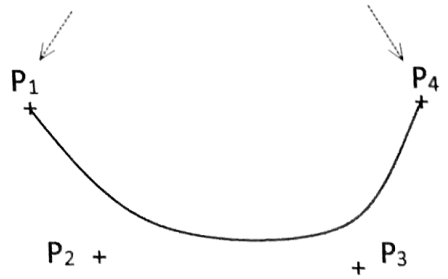
2 Courbe interpolée - courbes à attracteurs

Nous avons vu que, avec l'interpolation polynomiale, on pouvait obtenir la courbe d'un polynôme **passant par des points de collocation**. Ces courbes sont des courbes interpolées.

Il existe un autre moyen pour obtenir une courbe, toujours à partir de points de contrôle, mais **sans que la courbe passe par tous les points**. Ce sont les courbes de Bézier.

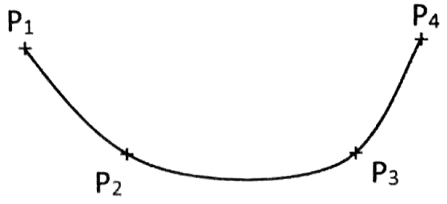
Courbe à attracteurs : Courbe de Bézier

Deux points de contrôle sur la courbe



Deux points de contrôle en dehors de la courbe

Courbe interpolées: Lagrange, splines interpolées etc.



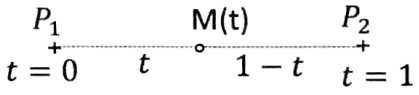
Tous les points de contrôle sont sur la courbe

3 Construction des Courbes de Bézier

Les courbes de Bézier sont des courbes polynomiales paramétriques décrites pour la première fois en 1962 par l'ingénieur français Pierre Bézier qui les utilisa pour concevoir des pièces d'automobiles à l'aide d'ordinateurs. Elles ont de nombreuses applications dans la synthèse d'images et le rendu de polices de caractères. Comme nous allons le voir, une courbe de Bézier est une application pratique de la notion bien connue de Barycentre.

3.1 Définition récursive de Casteljau

Géométriquement, une courbe de Bézier peut se définir comme une construction récursive de barycentres avec les poids (1 - t) sur l'un des points et t sur l'autre point, voir ci-dessous.
On notera que t est toujours compris entre 0 et 1.



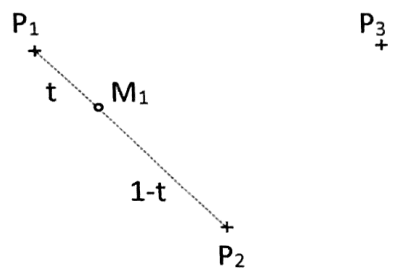
On notera que
si t=0, P_1 et M sont confondus
si t=1, P_2 et M sont confondus

3.2 Courbe de Bézier sur 3 points

3.2.1 Interprétation graphique

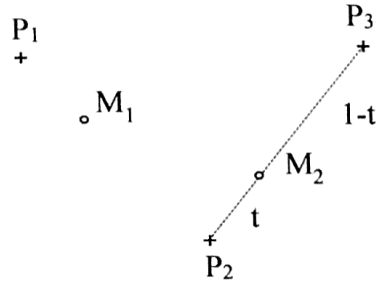
Soient trois points $P_1(x_1(t),y_1(t))$, $P_2(x_2(t),y_2(t))$ et $P_3(x_3(t),y_3(t))$

Etape 1 : construction de M_1



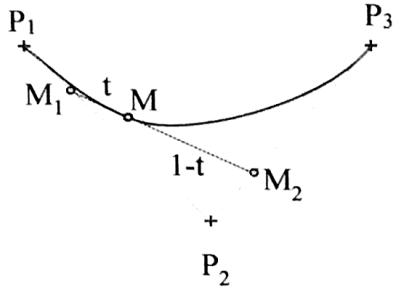
On notera que
si t=0, P_1 et M_1 sont confondus
si t=1, P_2 et M_1 sont confondus

Etape 2 : construction de M_2



On notera que :
si t=0, P_2 et M_2 sont confondus
si t=1, P_3 et M_2 sont confondus

Etape 3 : construction de M



3.2.2 Développement mathématique

Première construction avec P_1 et P_2	Deuxième construction avec P_2 et P_3
On construit la droite joignant P_1 à P_2 et on détermine un point M_1 sur cette droite tel que : $$P_1(t) = (1-t)P_1 + tP_2$$ On notera que pour t=0, M_1 est confondu avec P_1 et que si t=1, M_1 est confondu avec P_2	On construit la droite joignant P_2 à P_3 et on détermine un point M_2 sur cette droite tel que : $$P_2(t) = (1-t)P_2 + tP_3$$ On notera que pour t=0, M_2 est confondu avec P_2 et que si t=1, M_2 est confondu avec P_3

Troisième construction avec M_1 et M_2
On construit la droite joignant M_1 et M_2 et on détermine un point sur cette droite tel que : $$M(t) = (1-t)M_1 + tM_2$$ On notera que pour t=0, M est confondu avec M_1 et que si t=1, M est confondu avec M_2.

$M(t)$ est un des points de la courbe qui se construit entièrement en faisant varier t.
On notera également que le segment $[M_1(t), M_2(t)]$ est tangent à la courbe en $M(t)$.
En développant $M(t)$ on obtient l'équation paramétrique de la courbe. Les coordonnées de tous les points de la courbe sont données par la relation :

$$M(t) = (1-t)^2 P_1 + 2t(1-t)P_2 + t^2 P_3$$

On notera que le polynôme est d'ordre 2.

3.2.2.1.1 Formulation matricielle

En développant la forme ci-dessus on montre que l'on peut écrire les coordonnées d'un point de la courbe comme le produit d'une matrice contenant les monômes du paramètre t, d'une matrice de coefficients et une matrice de points de contrôle P_1, P_2, P_3.

$$M(t) = [t^2\ t^1\ t^0] \begin{bmatrix} 1 & -2 & 1 \\ -2 & 2 & 0 \\ 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} P_1 \\ P_2 \\ P_3 \end{bmatrix}$$

3.3 Courbe de Bézier sur 4 points

La construction est identique, avec un niveau de plus.

a) Au premier niveau :

Avec (P1, P2) : $M_1(t) = (1-t)P_1 + tP_2$

Avec (P2, P3) : $M_2(t) = (1-t)P_2 + tP_3$

Avec (P3, P4) : $M_3(t) = (1-t)P_3 + tP_4$

b) Au second niveau :

Avec (M_1, M_2):

$$M_{12}(t) = (1-t)M_1 + tM_2 = (1-t)^2 P_1 + 2t(1-t)P_2 + t^2 P_3$$

Avec (M_2, M_3):

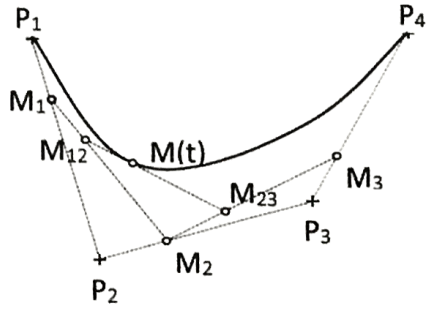
$$M_{23}(t) = (1-t)M_2 + tM_3 = (1-t)^2 P_2 + 2t(1-t)P_3 + t^2 P_4$$

c) Au troisième niveau :

Avec (M_{12}, M_{23}) :

$$M(t) = (1-t)M_{12} + tM_{23} = (1-t)^3 P_1 + 3t(1-t)^2 P_2 + 3t^2(1-t)P_3 + t^3 P_4$$

Construction géométrique d'une courbe de Bézier à quatre



On notera que :

- Le segment [M_{12}(t),M_{23}(t)] est tangent à la courbe en M(t)
- Le polynôme est d'ordre 3

3.3.1 Formulation matricielle

En développant la forme ci-dessus on montre que l'on peut écrire les coordonnées d'un point de la courbe comme le produit d'une matrice contenant les monômes du paramètre t, d'une matrice de coefficients et une matrice de points de contrôle P_1, P_2, P_3, P_4.

$$M(t) = [t^3\ t^2\ t\ \ 1]\begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}\begin{bmatrix} P_1 \\ P_2 \\ P_3 \\ P_4 \end{bmatrix} \quad (1)$$

3.3.2 Travail à faire

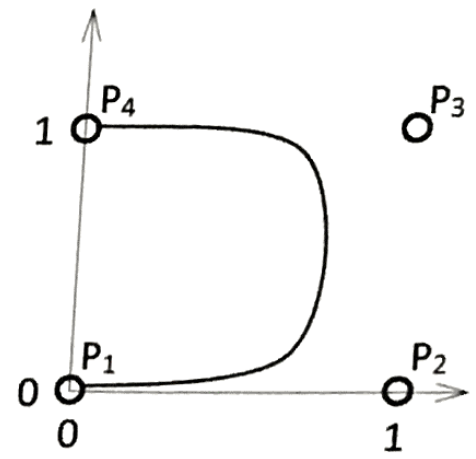
3.3.2.1 Préparation

1. **Créer** une fonction Bezier4 dans le fichier ***Bezier4.m***
   - L'entête de fonction peut être la suivante : ***function M = Bezier4(P1,P2,P3,P4,t)***
   - La fonction reçoit les quatre points de contrôle **P1 à P4** et le tableau **t**
     - **P1, P2, P3 et P4** sont des vecteurs à 2 coordonnées
     - **t** sera un tableau de 100 valeurs
   - Elle renvoie **M**, un tableau de 100 points. Deux coordonnées par point calculées pour toutes les valeurs de **t**
   - La fonction implémente le calcul des coordonnées du point **M(t)** comme indiqué dans la formule (1) du paragraphe précédent

2. **Créer** un script principal ***Bezier.m***

Le but de ce script est de dessiner une simple courbe de Bézier à 4 points de contrôle.

```
clc
clear all
close all
Définir le tableau t avec 100 points
%Définir les 4 points de contrôle
P1=[0 0];
P2=[1 0];
P3=[1 1];
P4=[0 1];
% Tracer les 4 points de contrôle avec de « o »
%Utiliser la fct xy pour récupérer les coordonnées
plot(...,...,'or');
% Appeler Bezier4
M = Bezier4(...)
% utiliser la fct window
window(P1, P2, P3, P4)
% Tracer la courbe des points M
plot(M(:,1),M(:,2))
```



3.3.2.2 Tracer interactif

3. Recopier le code de la fonction xy ci-dessous dans le fichier ***xy.m***

```
function [x,y]= xy(P1,P2,P3,P4)
% crée deux tableaux x et y à partir des coordonnées des deux points
```

```
x=[P1(1) P2(1) P3(1) P4(1)];
y=[P1(2) P2(2) P3(2) P4(2)];
```

Cette fonction renvoie deux tableaux « x » et « y » contenant chacun les coordonnées x et y des 4 points passés en paramètre.

4. Recopier le code de la fonction GetPoint ci-dessous dans le fichier ***GetPoint.m***

```
function [p,Bt] = GetPoint(P1, P2, P3, P4)
[x y]=xy(P1,P2,P3,P4);   % récupère les coordonnées des 4 points
s=20
dx = (max(x)-min(x))/s;  % taille de la zone de recherche en x
dy = (max(y)-min(y))/s;  % taille de la zone de recherche en y
[xs,ys,Bt]=ginput(1);   %  appel du curseur et récupération des coordonnées du pointé
p = find(abs(xs-x)<dx & abs(ys-y)<dy);  % recherche du point P(1,2,3 ou 4)
if size(p)~=0  % si trouvé
 [x y]=xy(P1,P2,P3,P4);
text(min(x),max(y)+0.05*(max(y)-min(y)),['Point ', num2str(p)],'FontSize',12, 'Color',[1 0 1]);
switch p
    case 1
        plot(P1(:,1),P1(:,2),'bo');  % retrace le point P1
    case 2
      plot(P2(:,1),P2(:,2),'bo');  % retrace le point P2
    case 3
      plot(P3(:,1),P3(:,2),'bo');  % retrace le point P3
    case 4
        plot(P4(:,1),P4(:,2),'bo');  % retrace le point P4
    end
else
    p=0;  % si pas trouvé
end
```

Après avoir effectué un pointé dans la fenêtre graphique, cette fonction la fonction essaye de trouver un des 4 points de contrôle. Si le point a été trouvé, la fonction redessine le point et renvoie le N° du point « p » ainsi que le code du bouton de la souris qui a effectué la validation « Bt ».

5. Recopier le code de la fonction window ci-dessous dans le fichier ***window.m***.

```
function window(P1, P2, P3, P4)
[x y]=xy(P1,P2,P3,P4);
dx = max(x)-min(x);
dy = max(y)-min(y);

axis([min(x)-0.1*dx max(x)+0.1*dx min(y)-0.1*dy max(y)+0.1*dy]);
axis square
text(min(x)-0.1*dx,min(y)-0.05*dy,'Bouton droit pour quitter','FontSize',14);
```

Le but de cette fonction est de calculer une zone de tracer avec une marge suffisamment grande pour visualiser correctement la courbe et ses points de contrôle.

6. **Créer** une fonction MovePoint dans le fichier ***MovePoint.m***
   - L'entête de fonction est la suivante : ***function [P1,P2,P3,P4,Bt] = MovePoint(P1,P2,P3,P4)***

- La fonction appelle la fonction ***GetPoint*** pour obtenir un point à modifier. Si un point a été trouvé on rappel ***ginput*** pour faire un nouveau pointé donnant la nouvelle position du point.
- La fonction renvoie les quatre points de contrôle dont un a pu être mis à jour après un pointé ainsi que le code du bouton de la souris qui a effectué la validation « Bt »

L'algorithme peut être le suivant :

```
function [P1,P2,P3,P4,Bt] = MovePoint(P1,P2,P3,P4)
Appeler la fonction Getpoint qui renvoie le N° du point et le N° de bouton de la souris

Si un point a été trouvé (tester p)
    Appeler ginput  % nouvelle position
    switch ...
        case ...
                Mettre à jour le point sélectionné avec les coordonnées renvoyées par ginput,
    end
Sinon
    % afficher un message « 'Point non trouvé » dans la fenêtre
    if(Bt ~= 3)
        [x y]=xy(P1,P2,P3,P4);
        text(min(x),max(y)+0.05*(max(y)-min(y)), 'non trouvé','FontSize',12, 'Color',[1 0 1]);
        pause(0.5);
    end
end
```

7. **Modifier** le script principal ***Bezier.m***

Le but de ce nouveau script est de créer un outil simple de modification interactif des points de contrôle d'une courbe de Bézier. On pourra ainsi observer comment la courbe se modifie.

L'algorithme peut être le suivant :

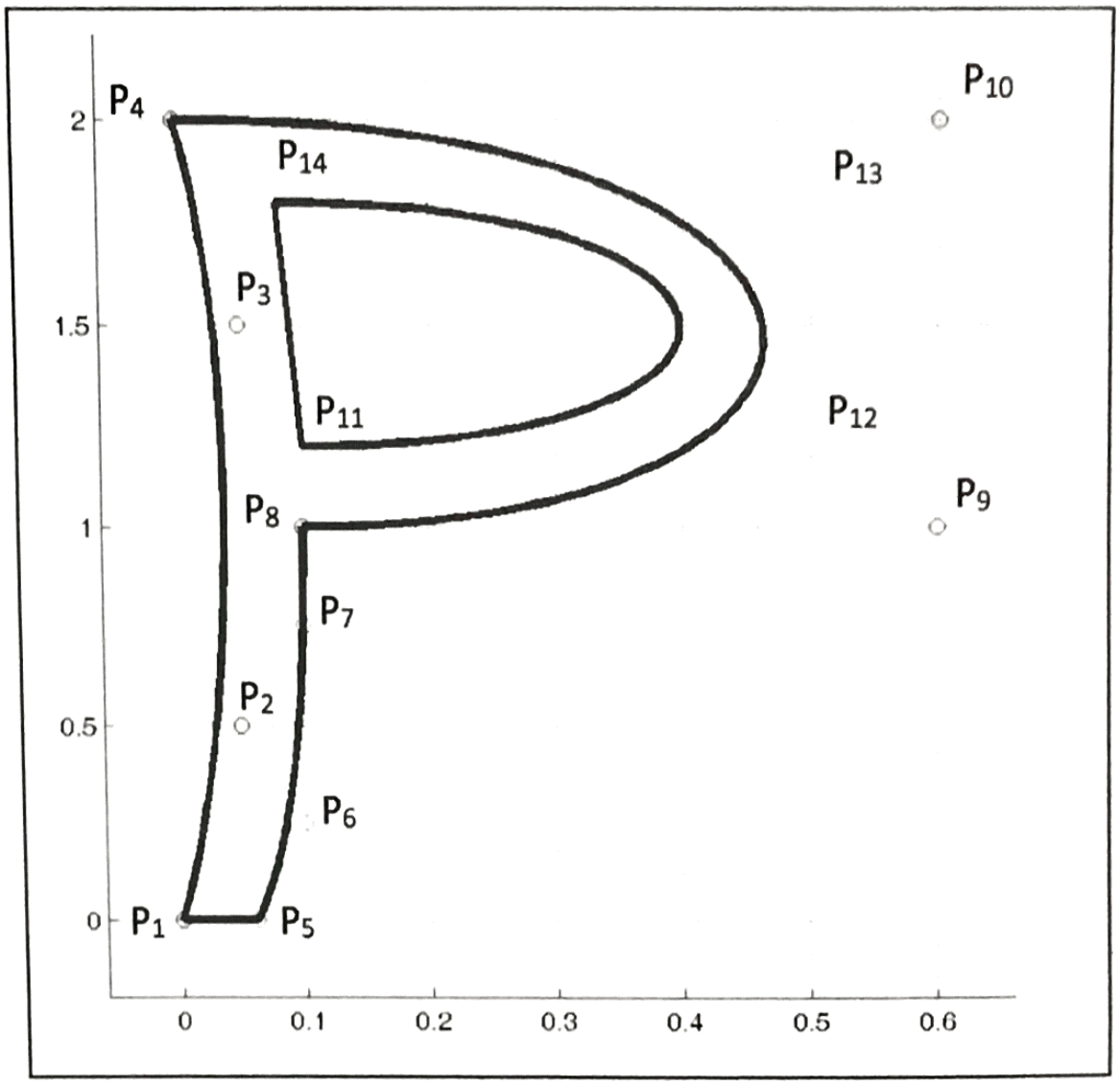
```
Faire le ménage
Définir le tableau t
Définir les 4 points de contrôle
P1=[0 0];
P2=[1 0];
P3=[1 1];
P4=[0 1];
Tracer les 4 points de contrôle
Utiliser la fonction window
Calculer avec Bezier4 les points pour le tracer de la courbe de Bézier
Tracer la courbe

Forcer l'entrer dans la boucle
Tant que (Bt ~= 3) (tant que le bouton de la souris n'est pas le bouton droit)
    Appeler MovePoint
    clf
    hold on
    grid on
    Appeler xy
    Tracer les points de contrôle actualisés
```

Utiliser la fonction window
Calculer avec Bezier4 les points pour le tracer de la courbe de Bézier
Tracer la courbe
Fin Tant que

3.3.2.3 Tracer d'une lettre de l'alphabet

Dans cette partie, vous allez tracer la lettre P avec des courbes de Bézier à 4 points.
Les coordonnées des points sont les suivantes :

P1=[0 0];
P2=[0.05 0.5];
P3=[0.05 1.5];
P4=[0 2];
P5=[0.06 0];
P6=[0.1 0.25];
P7=[0.1 0.75];
P8=[0.1 1];
P9=[.6 1];
P10=[.6 2];
P11=[0.1 1.2];
P12=[0.5 1.2];
P13=[0.5 1.8];
P14=[0.08 1.8];

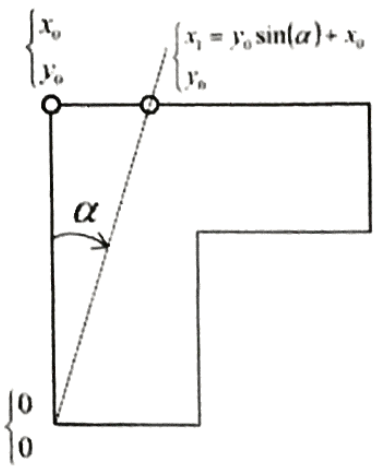


Seule exception, pour simplifier, les segments [P_1 P_5] et [P_{11} P_{14}] seront des droites.

8. Créer un script ***Lettre.m*** pour tracer la lettre P.

3.3.2.4 Tracer en italique

Pour tracer la lettre P en italique on supposera que la transformation ci-contre peut s'appliquer.



9. Ecrire une fonction MatLab qui transforme un point selon la méthode ci-contre. Utiliser cette fonction pour modifier le programme précédent et tracer la lettre P en italique.