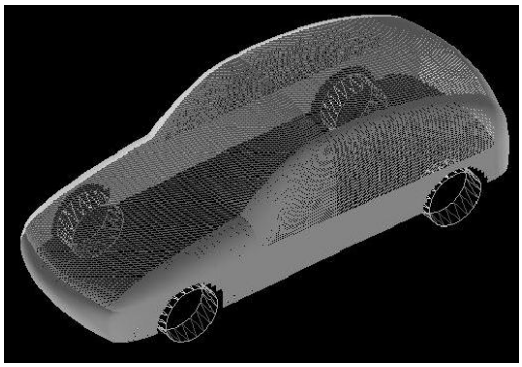
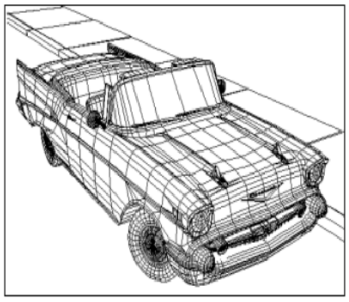
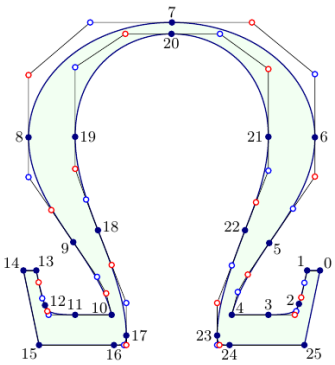
DM n°6 ~ Courbes de Béziers

Pierre Béziers, ingénieur chez Renault, décrit pour la première fois ses courbes en 1962, dans le but de concevoir des pièces automobiles à l'aide d'ordinateur. Aujourd'hui ces courbes sont utilisées dans la plupart des logiciels de dessin ou de synthèse d'images. En voici quelques exemples :

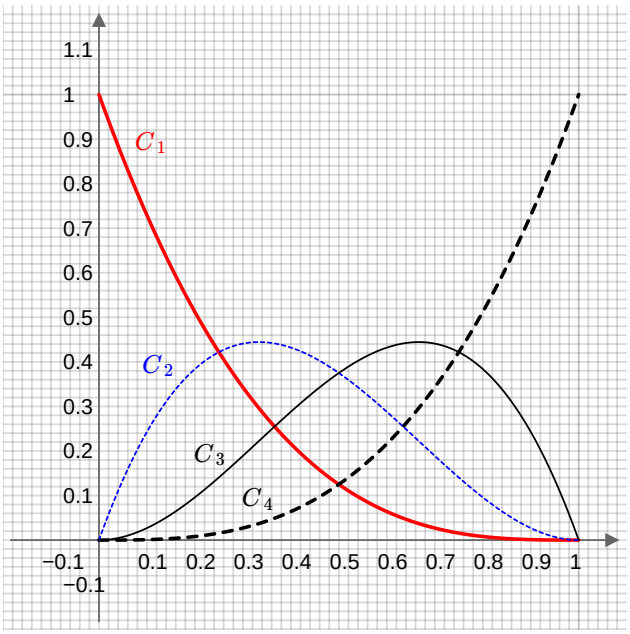




1 Les polynômes de Bernstein du troisième degré

Pour tout réel t on note :

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

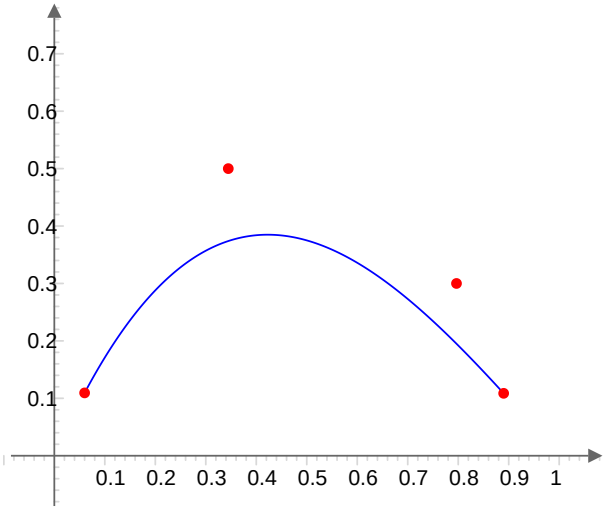
les quatre polynômes de Bernstein du troisième degré. Ils seront à la base de la construction des courbes de Béziers. Nous étudions quelques une de leurs propriétés dans cette partie.



1. Dresser les tableaux de variations des polynômes B_0 , B_1 , B_2 et B_3 sur $[0\,;1]$.
2. Déterminer les racines des polynômes B_0 , B_1 , B_2 et B_3 sur $[0\,;1]$.
3. Associer à chacune des courbes ci-contre le polynôme de Bernstein du troisième degré correspondant.

2 Tracé de courbes de Béziers

L'idée de Pierre Béziers est de permettre de construire des surfaces de la façon la plus simple possible en étant assisté d'un ordinateur. Nous nous limiterons ici à la construction de courbes.
L'objectif est de tracer une courbe la plus « lisse » possible à partir de peu de points, que l'on appelle points de contrôle. La courbe part du premier point de contrôle pour rejoindre le dernier, et en chemin elle se fait « attirer » par les autres points comme dans l'exemple ci-contre.
Les polynômes de Bernstein prennent ici tout leur intérêt puisque les maximum de chacun d'eux sont régulièrement répartis sur l'intervalle $[0\,;1]$.



Définition 1

Soient $P_0(x_0\,;y_0)$, $P_1(x_1\,;y_2)$, $P_2(x_2\,;y_2)$ et $P_3(x_3\,;y_3)$ trois points d'un repère du plan et soient B_0, B_1, B_2 et B_3 les quatre polynômes de Bernstein de degré 3.
La courbe de Béziers associée aux quatre points de contrôle P_0, P_1, P_2 et P_3, est l'ensemble des points $M(x\,;y)$ du plan vérifiant :

$$\begin{cases} x &= x_0B_0(t)+x_1B_1(t)+x_2B_2(t)+x_3B_3(t) \\ y &= y_0B_0(t)+y_1B_1(t)+y_2B_2(t)+y_3B_3(t) \end{cases}, t\in[0\,;1].$$

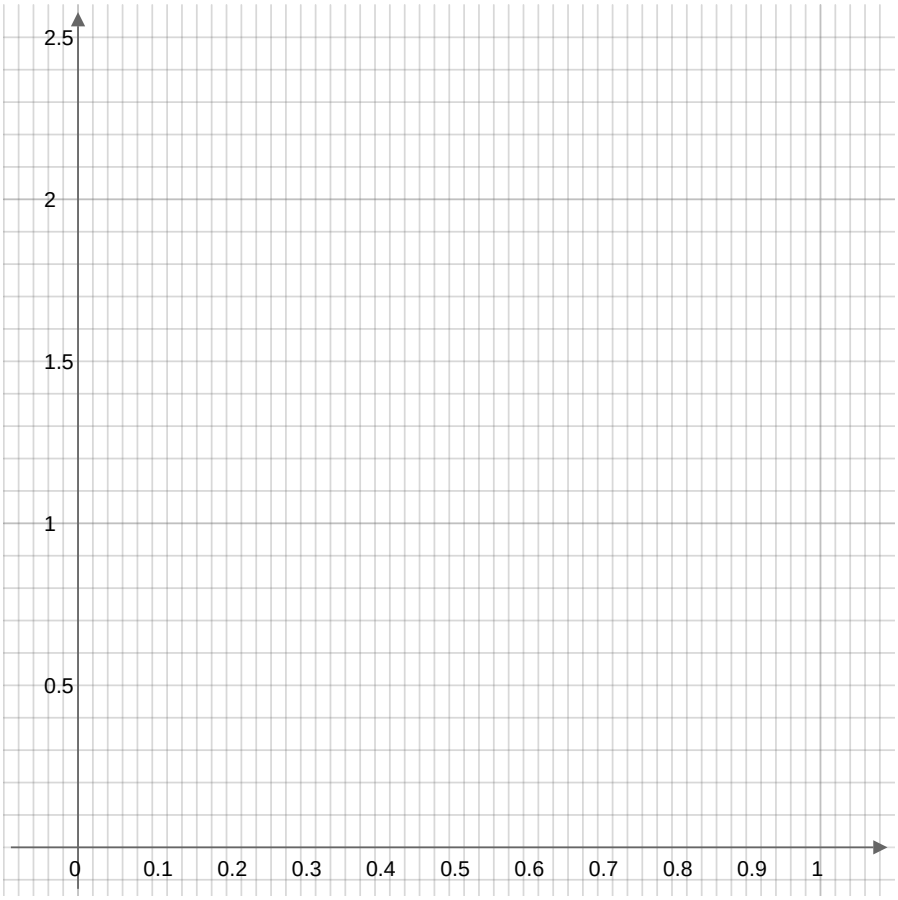
1. Montrer que toute courbe de Béziers passe par P_0 et P_3.
2. On considère la courbe de Béziers associée aux points de contrôle : $P_0(0,5\,;0,5)$, $P_1(0\,;2)$, $P_2(0,5\,;2,5)$ et $P_3(0,5\,;1,5)$.
   a. Montrer que la paramétrisation de cette courbe de Béziers est :

$$\begin{cases} x &= -1,5t^3+3t^2-1,5t+0,5 \\ y &= -0,5t^3-3t^2+4,5t+0,5 \end{cases}, t\in[0\,;1].$$

   b. À l'aide d'une calculatrice compléter le tableau suivant :

t	0	$0,1$	$0,2$	$0,3$	$0,4$	$0,5$	$0,6$	$0,7$	$0,8$	$0,9$	1
$x(t)$											
$y(t)$											

   c. Dans le repère ci-dessous construire les points de contrôle.
   d. Tracer ensuite l'ensemble des points du tableau précédent et les relier par une courbe des plus lisses possibles.



   e. Tracer dans le même repère la courbe de Béziers associée aux points de contrôle $P_0(0,5\,;0,5)$, $P_1(1\,;2)$, $P_2(0,5\,;2,5)$ et $P_3(0,5\,;1,5)$.

On déterminera tout d'abord la paramétrisation correspondante et on pourra construire un tableau de valeurs.
La jonction de ces deux courbes devrait faire apparaître un joli dessin.

3. Dans l'algorithme ci-dessous, en langage apparenté au javascript, on définit la fenêtre graphique entre les ligne 1 et 4, puis des points (qui correspondent aux points de contrôle précédents).
Les fonctions définies entre les lignes 16 et 27 correspondent aux polynômes de Bernstein de degré 3.
La fonction *bezier*, définie à la ligne 29, permet de construire une courbe de Bézier en lui donnant quatre points de contrôle.
Les lignes 45 et 46 permettent d'exécuter la fonction *beziers* avec des points de contrôle différents.
Compléter les lignes 23 et 40 pour que l'algorithme fonctionne.

```
1  Xmin = -0.1
2  Xmax = 1.1
3  Ymin = -0.2
4  Ymax = 2.6
5
6  P0 = [0.5,0.5]
7  P1 = [0,2]
8  P2 = [0.5,2.5]
9  P3 = [0.5,1.5]
10
11 Q0 = [0.5,0.5]
12 Q1 = [1,2]
13 Q2 = [0.5,2.5]
14 Q3 = [0.5,1.5]
15
16 function B0(t){
17     return -t*t*t+3*t*t-3*t+1
18 }
19 function B1(t){
20    return 3*t*t*t-6*t*t+3*t
21 }
22 function B2(t){
23     return
24 }
25 function B3(t){
26     return t*t*t
27 }
28
29 function beziers(P0,P1,P2,P3){
30     trait = 1
31     couleur = rouge
32     point(P0)
33     point(P1)
34     point(P2)
35     point(P3)
36     couleur = noir
37     trait = 0.5
38     for(i=0; i <= 1; i = i+0.01){
39         x = P0[0]*B0(i)+P1[0]*B1(i)+P2[0]*B2(i)+P3[0]*B3(i)
40         y =
41         point([x,y])
42     }
43 }
44
45 beziers(P0,P1,P2,P3)
46 beziers(Q0,Q1,Q2,Q3)
```

4. Donner des points de contrôle pour obtenir une courbe issue de votre imagination.

3 Généralisation

Pour définir des courbes de Béziers avec un nombre de points plus important, on utilise des polynômes de Bernstein de degré supérieur.

Définition 2

Soit n un entier naturel et $k \in [\![0;n]\!]$.
Le k^{e} polynôme de Bernstein de degré n est défini pour tout réel t par :

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

1. Déterminer les expressions développées des cinq polynômes de Bernstein de degré 4.
2. Pour tout $n \in \mathbb{N}$, pour tout $k \in [\![0;n]\!]$ et pour tout $t \in [0\,;1]$, justifier que $B_k^n(t) \geq 0$.
3. Pour tout $n \in \mathbb{N}$, pour tout $k \in [\![0;n]\!]$ et pour tout $t \in [0\,;1]$, justifier que $(B_k^n)'(t) = \binom{n}{k} t^{k-1}(1-t)^{n-k-1}(k-nt)$.
4. En déduire que le polynôme B_k^n atteint son maximum sur $[0\,;1]$ en $\dfrac{k}{n}$.
5. Soient $n \in \mathbb{N}$, $k \in [\![0;n]\!]$ et $t \in [0\,;1]$. En utilisant un résultat du cours de probabilités, déterminer la valeur de $\sum_{k=0}^{n} B_k^n(t)$.
6. Expliquer comment l'algorithme ci-dessous généralise la méthode de la partie 2.

```
Xmin = -1.1
Xmax = 2.1
Ymin = -1.1
Ymax = 2.6

function facto(n){
    f = 1
    for(i=2; i <= n; i++){
        f = f*i
    }
    return f
}

function binom(n,k){
    return facto(n)/(facto(k)*facto(n-k))
}

function B(n,k,t){
    return binom(n,k)*puissance(t,k)*puissance(1-t,n-k)
}

function beziers(listePoints){
    n = listePoints.length
    trait = 1
    couleur = rouge
    for( i = 0; i < n; i++){
        point(listePoints[i])
    }
    couleur = noir
    trait = 0.2
    for( t = 0 ; t <= 1; t = t+0.002){
        x = 0
        y = 0
        for(j=0; j < n; j++){
            x = x+B(n-1,j,t)*listePoints[j][0]
            y = y+B(n-1,j,t)*listePoints[j][1]
        }
        point([x,y])
        console.log(x+","+y)
    }
}

L1 = [ [-0.2,0], [-0.5,1], [0.2,2], [0.7,2.5], [1.9,-1], [-0.9,-1], [0.3,2.5], [0.8,2], [1.5,1],
beziers(L1)
```