Rapport du projet :

Recherche de fonctions de base pour la résolution de problème d'optimisation basée sur l'analyse par intervalles

Réalisé par :
Rayhane BOUACHRINE, Abdelali TEMSAMANI
Encadré par :
M. Francois BOUCHON, M. Sebastien LENGAGNE
Filière :
Ingénierie Mathématique et Data Science

Année universitaire :
2021-2022

Résumé

Dans ce projet, on s'intéresse aux problèmes de la robotique afin de proposer des solutions pour un problème d'optimisation sous contraintes.
La résolution de ces problèmes sera basée sur la méthode de l'analyse par intervalles et l'algorithme de la bissection qui consiste à diviser un domaine d'entrée en plusieurs sous-domaines. L'objectif de ce projet est donc d'approximer les extremas d'une fonction polynomiale sur un sous domaine avec les points de côntrole. Pour ce fait, nous utiliserons les propriétés des fonctions B-splines qui ont la capacité de résoudre le problème du pessimisme causé par la méthode de l'analyse par intervalles. Nous aborderons aussi dans ce rapport d'autres fonctions de base qui peuvent amener à de meilleurs résultats.

Mots-clés : Robotique, optimisation, analyse par intervalle, bissection, fonctions de base, B-splines.

Abstract

In this project, we have interest in robotics problems in order to propose solutions to solve constrained optimization problems.
The resolution of these problems will be based on the method of analysis by intervals and the bisection which consists in dividing the input domain into several sub-domains. Thus, the goal of this project is to approximate the extremes of a polynomial function on sub-domains by control points. For this we will use the properties of B-spline functions which have the ability to solve the problem of pessimism caused by interval analysis. We will also discuss in this report other basic functions leading to better results.

Keywords : Robotics, optimization, analysis by intervals, bisection, basic functions, B-spline.

Table des matières

Table des figures

Introduction

Dans les problèmes d'optimisation, on cherche souvent à évaluer les valeurs d'une fonction à fin de trouver la meilleure solution aux problèmes initiales. Dans ce projet, nous traiterons des problèmes d'optimisation sous contraintes qui sont des conditions généralement décrites par des équations, et qui doivent être satisfaites quelle que soit la solution.
On étudiera donc des outils mathématiques, tout en se basant sur l'analyse par intervalles, pour répondre à ces problèmes d'optimisation. Nous allons aborder dans un premier temps
l'algorithme de la bissection qui consiste à encadrer, après chaque itération, la solution recherchée. Nous verrons aussi les fonctions B-splines qui aident à avoir une évaluation, contrairement à l'analyse par intervalles connue par la surestimation de l'image d'un intervalle, moins pessimiste des fonctions mathématiques grâce à ses propriétés.
Le but de ce projet est donc d'implémenter l'algorithme de la bissection et les fonctions B-splines à l'aide du langage Python, avant de faire une comparaison avec d'autres fonctions de base qui peuvent amener à des résultats plus optimales.

Chapitre 1

Problèmes d'optimisation basé sur l'analyse par intervalles

On commence à travers ce chapitre par introduire notre problème étudié ainsi que la définition de l'outil de l'analyse par intervalles qui nous permettera de répondre à ce problème.
Il est à noter qu'une grande partie de ce chapitre ainsi que des figures utilisées sont tirés de la thèse de M. Rawan Kalawoun **[1]**.

1.1 Analyse par intervalles

Dans le but de présenter l'outil de l'analyse par intervalles, nous décrivons dans cette partie les notations et les opérations arithmétiques liées aux intervalles.

1.1.1 Définition et propriétés

Soit $\zeta(\mathbb{R})$ l'ensemble de tous les sous-ensembles non vides, compacts et convexes de $\mathbb{R}$.
Soient A, B $\in \zeta(\mathbb{R})$ et soit $\odot \in \{+,-,\cdot,/\}$ une opération binaire de $\mathbb{R}$, i.e. addition, soustraction, multiplication et division.
On note par ξ l'ensemble des intervalles fermés et bornés de $\mathbb{R}$. Dans l'ensemble de ce rapport, pour un intervalle A on note A = [a_i , a_s] où a_i et a_s sont respectivement la valeur inférieure et superieure de A.

Opérations sur les intervalles

Comme les intervalles sont essentiellement des ensembles de nombres réels, il est souvent utile de définir des opérations liées à leur comportement tant qu'ensembles.
Soient A,B $\in \xi$. L'intersection des deux intervalles A et B est vide si et seulement si $b_s < a_i$ ou $a_s < b_i$. Sinon on a :

$$A \cap\ B = \{x : x \in\ A \wedge\ x \in\ B\} = [\max\{a_i, b_i\}, \min\{a_s, b_s\}] \tag{1.1}$$

En particulier, l'intersection joue un rôle très important dans la technique de l'analyse par intervalles. Si on a par exemple deux intervalles qui contiennent un résultat d'intérêt, alors l'intersection de ces deux intervalles, qui est éventuellement plus étroit (Figure 1.1) contient aussi le résultat.

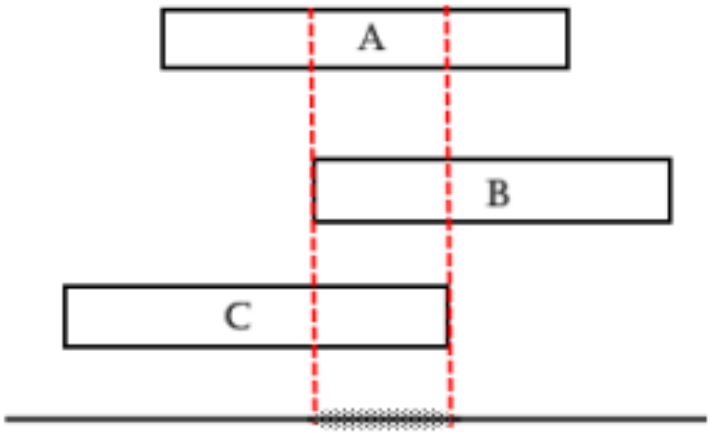


FIGURE 1.1 : L'intersection de trois intervalles A,B et C

Propriétés

Toujours avec la notation A = [a_i , a_s] on peut définir quelques propriétés principales des intervalles (Figure 1.2).
Soit A $\in \xi$. On définit la longuer de A par :

$$l(A) = a_s - a_i \tag{1.2}$$

Soit A $\in \xi$. On définit la valeur absolue de A par :

$$|A| = \max\{|a_i|, |a_s|\} \tag{1.3}$$

Soit A $\in \xi$. Le milieu de A est donné par :

$$m(A) = \frac{a_i + a_s}{2} \tag{1.4}$$

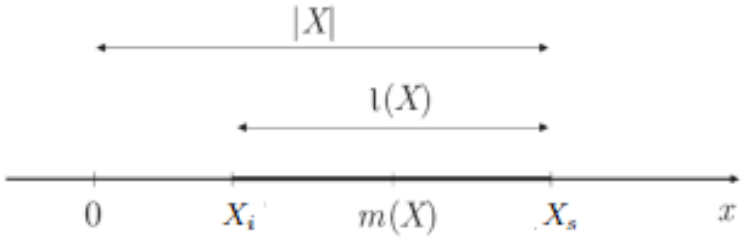


FIGURE 1.2 : Propriétés géométriques d'un intervalle

Opérations arithmétiques

Soient A,B $\in \xi$ subset [fe([x]).t soit $\odot \in \{+,-,\cdot,/\}$ une opération binaire de $\mathbb{R}$, i.e. addition, soustraction, multiplication et division. L'opération A $\odot$ B est définie par :

$$A \odot B = \{a \odot b : a \in A, b \in B\} \tag{1.5}$$

Par exemple, l'addition de deux intervalles A et B peut être explicitement écrit comme suit :

$$A + B = [a_i + b_i, a_s + b_s] \tag{1.6}$$

De la même manière, on peut écrire toutes les autres opérations arithmétiques entre A et B.

1.1.2 Fonctions d'inclusion

Définition

On considère une fonction f de $\mathbb{R}^n$ dans $\mathbb{R}^m$ où n,$m \in \mathbb{N}$.
La fonction $[f]$ de $\mathbb{R}^n$ dans $\mathbb{R}^m$ est une fonction d'inclusion de f si :

$$\forall\ [x] \in \mathbb{R}^n,\ f([x]) \subset [f]([x]). \tag{1.7}$$

Où [x] est une boîte de dimension n définie par :

$$[x] = [x_1] \times [x_2] \times ... \times [x_n] \tag{1.8}$$

avec $[x_j] = [x_{j_i}, x_{j_s}]$ la projection de [x] sur le $j^{\text{ème}}$ axe, $\forall j \in [\![1, n]\!]$.

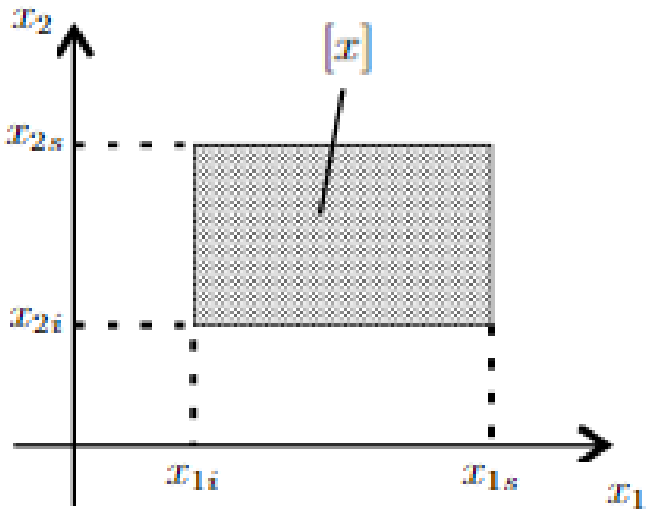


FIGURE 1.3 : Une boîte $[x]$ en $\mathbb{R}^2$ **[1]**

La fonction d'inclusion $[f]$ est minimale et on la note $[f]^*$ si pour tout $[x]$, $[f]([x])$ est la plus petite boîte contenant $f([x])$. Autrement dit :

$$\forall [f],\ [f]^*([x]) \subseteq [f]([x]) \tag{1.9}$$

De plus, il est évident que $[f]^*$ est unique.

Quelle que soit la forme de $f([x])$, il est possible de trouver une fonction d'inclusion $[f]$ de f pour générer une boîte $[f]([x])$ qui la contient (Figure 1.4).

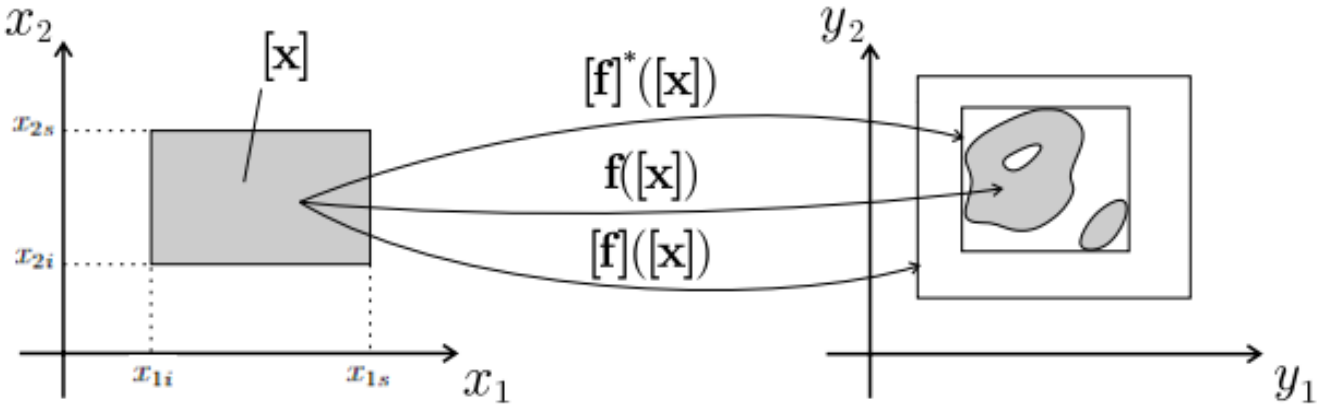


FIGURE 1.4 : Les images d'une boîte par une fonction f et ses deux fonctions d'inclusion $[f]$ et $[f]^*$ **[1]**

En effet, parmi les objectifs de l'analyse par intervalles, c'est de fournir pour un large ensemble de fonctions f, des fonctions d'inclusion qui peuvent être évaluées rapidement et de sorte que $[f]([x])$ ne soit pas trop étendue.

1.1.3 Problème du pessimisme

Le principal inconvénient de l'analyse par intervalles est la surestimation des solutions ou de l'image d'un intervalle par sa fonction d'inclusion, c'est ce qu'on appelle le pessimisme.
Ce problème de pessimisme est généralement causé par l'occurence multiple des variables dans les équations. Chaque occurence de la même variable est considérée comme une nouvelle variable qui s'appuie sur le même intervalle.
De ce fait, L'utilisation des fonctions d'inclusions naturelles citées avant n'est pas recommandé puisqu'elles dépendent des occurences des variables. Pour mieux comprendre ce point, on propose l'exemple suivant :

Soit $[x] = [-2, 2]$ et $f : \mathbb{R} \longmapsto \mathbb{R}$. On considére 4 expressions diffèrentes de f et pour chaque expression on écrit l'image de $[x]$ par la fonction d'inclusion correspondante :

$$f_1(a) = a(a+1) \;\Rightarrow\; [f_1]([a]) = [a]([a]+1) = [-2, 2]$$

$$f_2(a) = a * a + a \;\Rightarrow\; [f_2]([a]) = [a] * [a] + [a] = [-2, 2]$$

$$f_3(a) = a^2 + a \;\Rightarrow\; [f_3]([a]) = [a]^2 + [a] = [-1, 2]$$

$$f_4(a) = (a + \frac{1}{2})^2 - \frac{1}{4} \;\Rightarrow\; [f_4]([a]) = ([a] + \frac{1}{2})^2 - \frac{1}{4} = [\frac{-1}{4}, 2]$$

On peut remarquer que pour la même fonction f on obtient des intervalles différents et qui sont calculés avec quatre fonctions d'inclusions diffèrentes (Figure 1.5).

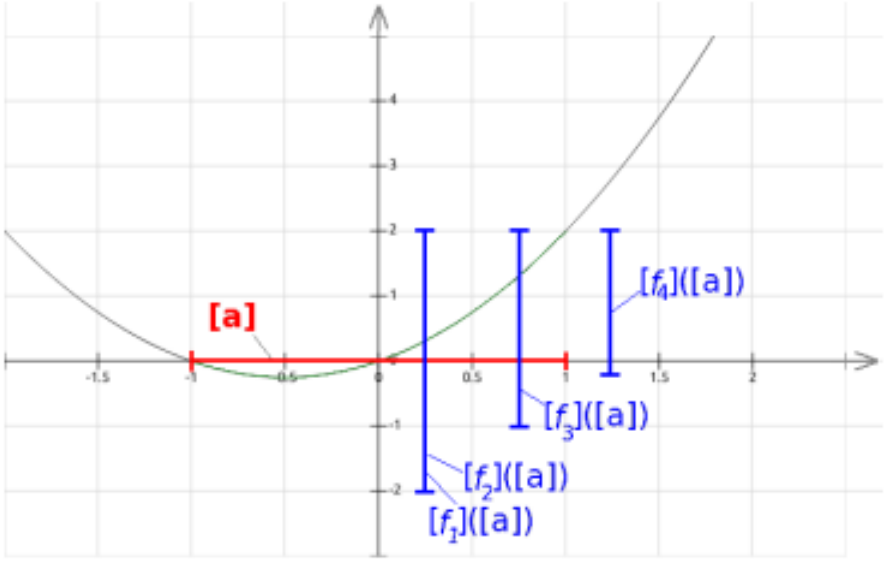


FIGURE 1.5 : Quatre fonctions d'inclusions diffèrentes pour la même fonction f **[1]**

Cette différence entre les résultats est causée par le problème du pessimisme. Nous reviendrons plus tard vers ce problème dans le deuxième chapitre pour voir qu'il est possible d'en faire face grâce aux fonctions B-splines.

1.2 Résolution des POC avec l'AI

La résolution des problèmes d'optimisation (POC) sous contraintes consiste à trouver l'ensemble des variables qui optimisent une fonction critère et qui satisfont un ensemble de conditions.

1.2.1 Problème d'optimisation sous contraintes (POC)

Les POC sont très populaires dans le domaine de la robotique où les contraintes peuvent être sous plusieurs formes, à savoir les limites articulaires et dynamiques des robots.
L'outil de l'analyse par intervalles (AI) est un outil mathématique très puissant qui peut être utilisé dans la résolution des POC. Les techniques d'optimisation basées sur l'AI donnent des résultats optimals tout en satisfaisant les contraintes du problème. L'AI est l'outil utilisé dans la résolution des POC pour assurer la fiabilité et faire face aux incertitudes de la structure mécanique des robots. La résolution d'un problème d'optimisation sous contraintes peut être définie par l'équation suivante :

$$\text{Trouver } [x] \subset \mathbb{Q} \subset \mathbb{R}^n$$

$$\text{tel que, } \min_x \mathcal{F}([x]) \tag{1.10}$$

$$\text{Avec } \forall j \in \{1, ..., m\} \ \varphi([x]) \in [g_{j_i}, g_{j_s}]$$

Où $\mathcal{F}([x])$ est la fonction critère et :

- n est le nombre de variables de l'ensemble x
- m est le nombre de contraintes
- x = $\{x_1, ..., x_n\}$ est un ensemble de n variables
- $\mathbb{Q} = \{[x_{1_i}, x_{1_s}], ..., [x_{n_i}, x_{n_s}]\} \subset \mathbb{R}^n$
- $\varphi([x])$ est un ensemble de m contraintes. Avec $\varphi_j([x]) \subset [x_{j_i}, x_{j_s}]$, $\forall j \in \{1, ..., m\}$.

1.2.2 La bissection

La bissection est une méthode itérative qui décompose un domaine d'entrée en plusieurs sous-domaines à fin de borner la solution otimale avec une grande précision.
La méthode de la bissection est généralement utilisé pour réduire le problème de la surestimation puisque le pessimisme dépende de la largeur du domaine d'entrée.

Principe

Dans les POC, les contraintes et la fonction critère sont évaluées sous forme de boîtes. Après chaque itration il faut prendre une décision : soit jeter la boîte courante ou appliquer la bissection pour la diviser encore en boîtes de tailles plus petites. La décision de jeter une boîte est prise suite à la violation d'une des contraintes du problème ou si la fonction critère est très large. Une contrainte est violée si l'intersection de son intervalle évalué et les limites $[x_j]$ est vide. Le processus se termine une fois qu'on obtient une boîte de taille plus petite qu'un seuil pré-défini. Cette méthode a comme objectif d'obtenir une évaluation plus étroite des contraintes et de la fonction critère.

Algorithme

Entrée: Q l'espace de recherche initial
ϵ : la présicion désirée
Sortie : Le boite optimale $\bar{x}$
Initialisation Q.empiler(**Q**), $\bar{f} = \infty$
while Q *non vide* **do**
  $[x]$ = Q.dépiler()
  $[f] = [f_t, f_s] = \mathcal{F}[x]$
  if $f_t < \bar{f}$ **then**
    Evaluer les contraintes : $\forall j [g_j] = \mathcal{G}_j([x])$
    if $\forall j [g_j] \cap [g_t, g_s] = \emptyset$ **then**
      if $[g_j] \cap [g_t, g_s] = [g_j] and f_s < \bar{f}$ **then**
        $\bar{f} = f_s$
        $\bar{x} = x$
      end if
      if $diam([x]) > \epsilon$ **then**
        $\{x_1, x_2\} = bissection(x)$
        Q.empiler(x_1)
        Q.empiler(x_2)
      end if
    end if
  end if
end while
return $\bar{x}$

FIGURE 1.6 : L'algorithme de la bissection **[1]**

Code python

```
def __bisection__(intervalle, fonction ):
    ## initialisations
    max = 100
    EPS= 1e-4
    pile = deque()   # on cree une pile
    pile.append(intervalle) # on ajoute l'intervalle dedans
    empty = False
    ni = 0   # nombre d'iterations
    out = it.Intervalle(-10, 10)
    ## bisection
    while (empty == False ):
        ni += 1   #nombre d'iteration
        b = pile.pop()  #renvoie le dernier element et le retire du deque
        # A faire
        #z = fonction(b, poly)
        z = fonction(b)

        if (z.inf < max):
            if (z.sup < max):
                max = z.sup
                intervalle_final = b
                out = z

            if (b.__diam__() > EPS):
                b1 = b.__divide_interval_left__()
                b2 = b.__divide_interval_right__()
                pile.append(b1)
                pile.append(b2)


        if (len(pile) == 0):
            empty = True

    print ("Intervalle : " + intervalle_final.__to_string__())
    print ("Valeur min : " + out.__to_string__())
    print ("Number of Iterations : " + str(ni))
    return intervalle_final
```

FIGURE 1.7 : L'implementation de l'algorithme de bissection en langage python **[1]**

1.2.3 Exemple

Pour mieux comprendre le principe de la méthode de la bissection, on prend l'exemple suivant : On considère deux variables $[x] = [x_i, x_s]$ et $[y] = [y_i, y_s]$. On cherche à trouver l'ensemble des boîtes qui respectent une contrainte C représentée en vert dans le Figure 1.7.
En utilisant l'algorithme de la bissection, on obtient trois types de boîtes : les boîtes non faisables (boîtes blanches), les boîtes semi faisables (boîtes rouges) et les boîtes faisables (boîtes vertes) (Figure 1.7).

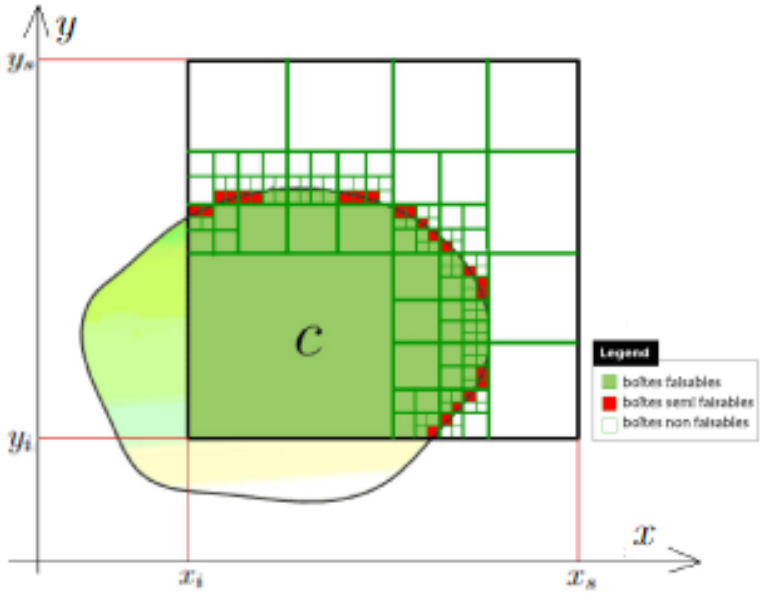


FIGURE 1.8 : L'algorithme de la bissection appliqué à une contrainte C en 2D **[1]**

1.3 Résolution du problème du pessimisme

L'un des intérêts les plus importants de la méthode de la bissection est la réduction du temps de calcul. Cependant, elle fait appel aux fonctions d'inclusions, ce qui veut dire qu'elle a aussi le problème du pessimisme.
C'est la raison pour laquelle on introduit les fonctions B-splines dans le prochain chapitre, qui peuvent grâce à leurs propriétés, résoudre ce problème du pessimisme.

Chapitre 2

Les fonctions B-splines

2.1 Définition

Une fonction BSpline de degré n est la somme pondérée de plusieurs fonctions de base de degré n. Elle est définie par m points de contrôle comme suit :

$$F(q) = \sum_{i=1}^{m} B_{i,n}(q) \times P_i \tag{2.1}$$

Avec $\forall q \in [\underline{q}, \bar{q}]$:

$$\sum_{i=1}^{m} B_{i,n}(q) = 1 \tag{2.2}$$

On peut obtenir un encadrement de F(q) en calculant le minimum et le maximum des points de controle et en utilisant la propriété ci-dessus (2.2) :

$$\forall i \in [\![1, m]\!], \forall q \in [\underline{q}, \bar{q}] : \underline{F} \leq P_i \leq \bar{F} \implies \underline{F} \leq F(q) \times P_i \leq \bar{F} \tag{2.3}$$

2.1.1 Fonctions de bases

On définit un vecteur $\{t_0, t_1, \ldots, t_m\}$, nommé vecteur noeudal , avec $m = k + n + 1$ et k le degré de la fonction.

pour $j = 0, ..., m - 1$:

$$B_{j,0}(t) := \begin{cases} 1 \ si \ t_j \leq t \leq t_{j+1} \\ 0 \ sinon \end{cases} \tag{2.4}$$

pour $k \geq 1$ and $i = 0, .., m - k - 1$:

$$B_{j,n}(t) := \frac{t - t_j}{t_{j+n} - t_j} B_{j,n-1}(t) + \frac{t_{j+n+1} - t}{t_{j+n+1} - t_{j+1}} B_{j+1,n-1}(t). \tag{2.5}$$

Cette relation de récurrence est représentée par le schéma de Cox-DeBoor suivant :

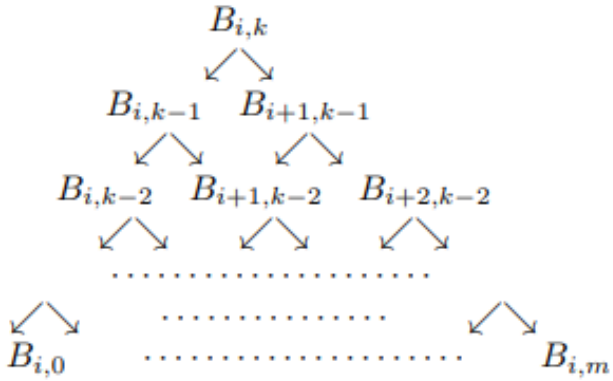


FIGURE 2.1 : Schéma de Cox-DeBoor, récursivité des Bspline

2.1.2 Quelques Propriétés

On citera ci-dessous quelques propriétés des B-splines :

- $\{t_0, t_1, \ldots, t_m\}$ sont des points contenus dans l'intervalle $[q] = [\underline{q}, \bar{q}]$.
- $\forall t \in]t_j; t_{j+n}[$, on a $0 < B_{j,n}(t) \leq 1$
- $B_{j,n}(t)$ est une fonction de degré k $\forall t \in [t_j; t_{j+n}[$
- $\sum_{i=j-k}^{j} B_{i,k}(t) = 1 \ \forall t \in [t_j; t_{j+n}[$
- $\sum_{i=0}^{m-k-1} B_{i,k}(t) = 1 \ \forall t \in [t_k; t_{m-k}[$
- $\sum_{i=0}^{n-1} B_{i,k}(t) = 1 \ \forall t \in [t_k; t_n[$ tel que $n \leq m - k$

2.2 Construction des B-splines

2.2.1 Algorithme de construction

Les étapes expliquées dans la partie définition sont mises en oeuvre dans l'algorithme suivant :

Algorithm 1: Calcul d'une B-spline

Entrée: un réel x , des entiers i et k, et un vecteur noeudal t
Sortie : B(x, k, i, t)
if $k = 0$ **then**
  if $t[i] \leq x$ *and* $x \geq t[i+1]$ **then**
    return 1
    else
      return 0
    end if
  end if
  else
    return $\frac{x-t[i]}{t[i+k]-t[i]} \times B(x, k-1, i, t) + \frac{t[i+k+1]-x}{t[i+k+1]-t[i+1]} \times B(x, k-1, i+1, t)$
  end if
end if

Code python :

```python
def B(x, k, i, t):
    if k == 0:
        return 1.0 if t[i] <= x < t[i+1] else 0.0
    if t[i+k] == t[i]:
        c1 = 0.0
    else:
        c1 = (x - t[i])/(t[i+k] - t[i]) * B(x, k-1, i, t)
    if t[i+k+1] == t[i+1]:
        c2 = 0.0
    else:
        c2 = (t[i+k+1] - x)/(t[i+k+1] - t[i+1]) * B(x, k-1, i+1, t)
    return c1 + c2
```

FIGURE 2.2 : Algorithme écrit en langage Python

Vérification :

```python
#on considère le vecteur noeudal suivant :
t=[0,1,2,3,4,5]

#on prend une valeur x qui appartient à [t[0],t[5]] et un degré k=2
x=1.5
k=2
i=0
print("le résultat est : " ,B(x,k,i,t))
```

```
le résultat est :  0.75
```

pour t dans l'intervalle [1,2[:

$$\begin{aligned} B_{0,2}(t) &= \frac{1}{2}[-t^2(t-1)^2 + 2(1-t) + 2] \\ &= \frac{1}{2}[-1.5^2(1.5-1)^2 + 2(1-1.5) + 2] \\ &= 0.75 \end{aligned}$$

```python
def bspline(x, t, c, k):
    n = len(t) - k - 1
    assert (n >= k+1) and (len(c) >= n)
    return sum(c[i] * B(x, k, i, t) for i in range(n))
```

FIGURE 2.3 : Fonction qui calcule les Bsplines en langage Python

2.2.2 Calcul analytique des fonctions de base

On considère une B-spline de degré k=2 et un vecteur noeudal tnoeud=[0,1,2,3,4,5] , Calculons par exemple $B_{0,2}$ sur l'intervalle $[0;3]$:

$$B_{0,2}(t) = \frac{t-0}{2-0}B_{0,1}(t) + \frac{3-t}{3-1}B_{1,1}(t) \tag{2.6}$$

Avec :

$$B_{0,1}(t) = \frac{t-0}{1-0}B_{0,0}(t) + \frac{2-t}{2-1}B_{1,0}(t) \tag{2.7}$$

Et :

$$B_{1,1}(t) = \frac{t-1}{2-1}B_{1,0}(t) + \frac{3-t}{3-1}B_{2,0}(t) \tag{2.8}$$

Donc :

$$B_{0,2}(t) = \frac{1}{2}[t^2 B_{0,0}(t) + t(2-t)B_{1,0}(t) + (3-t)(t-1)B_{1,0}(t) + (3-t)^2 B_{2,0}(t)] \tag{2.9}$$

Calcul de $B_{0,0}(t)$ $B_{1,0}(t)$ $B_{2,0}(t)$:

- Pour t $\in [0;1[$, $B_{0,0}(t) = 1$ $B_{1,0}(t) = 0$ $B_{2,0}(t) = 0$
- Pour t $\in [1;2[$, $B_{0,0}(t) = 0$ $B_{1,0}(t) = 1$ $B_{2,0}(t) = 0$
- Pour t $\in [2;3[$, $B_{0,0}(t) = 0$ $B_{1,0}(t) = 1$ $B_{2,0}(t) = 1$

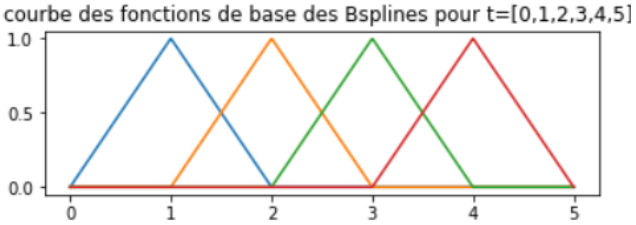
Résultat Final :

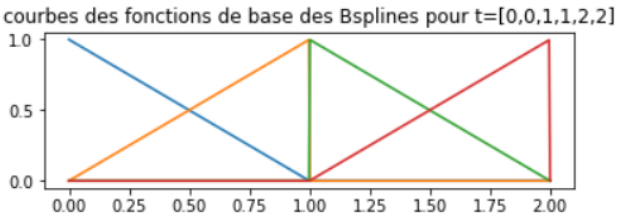
$$B_{0,2}(t) = \begin{cases} \frac{1}{2}t^2 & \text{si t} \in [0;1[\\ \frac{1}{2}[-t^2(t-1)^2 + 2(1-t) + 2] & \text{si t} \in [1;2[\\ \frac{1}{2}[(t-2)^2 - 2(t-2) + 1] & \text{si t} \in [2;3[\end{cases}$$

2.2.3 Quelques présentations graphiques

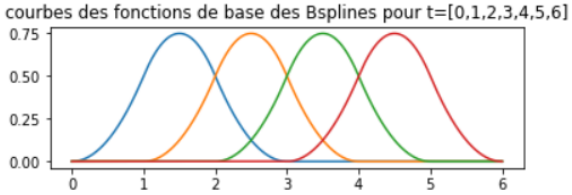
On a tracé des courbes des fonctions de base de B-splines pour différents degrés et différents vecteurs noeudals et ça a donné les courbes suivantes :

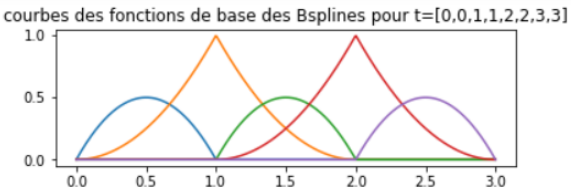
Pour K=1 :



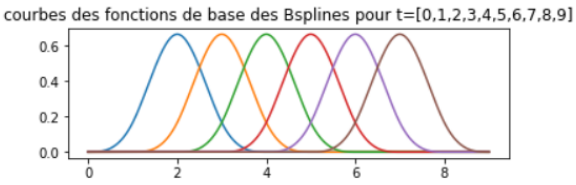


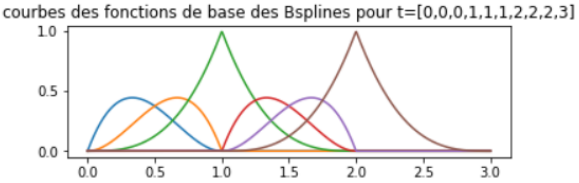
Pour K=2 :





Pour K=3 :





2.3 Points de contrôle

On suppose que les fonctions utilisées sont toutes polynômiales, on a alors :

$$F(q) = [1, q, \ldots, q^n] \times [a_0, a_1, \ldots, a_n]^T \tag{2.10}$$

Les Pi forment l'ensemble des points de controle qui décrivent la courbe et sont en lien avec les coefficients du polynomes donc :

$$F(q) = [1, q, \ldots, q^n] \times B \times [p_0, p_1, \ldots, p_n]^T \tag{2.11}$$

Ce qui donne :

$$[p_0, p_1, \ldots, p_n]^T = B^{-1} \times [a_0, a_1, \ldots, a_n]^T \tag{2.12}$$

Avec :

- $\{a_0, a_1, \ldots, a_n\}$ sont les coefficients du polynôme.
- B est la matrice contenant les coefficients des fonctions de base.
- $\{p_0, p_1, \ldots, p_n\}$ sont les points de contrôle.

Dans le cas multidimensionnel, la matrice B peut être écrite comme suit :

$$B = B_1 \otimes B_2 \otimes B_3 \otimes \cdots \otimes B_p \tag{2.13}$$

Où :

- p est la dimension de la fonciton F soit le nombre de variables de la fonciton F
- $\forall i \in [\![1, p]\!]$, B_i est la matrice des fonctions de base de F suivant la variable q_i
- $\otimes$ est l'opérateur du produit de Kronecker

On pose $P = [p_0, p_1, \ldots, p_n]^T$ et $X = [a_0, a_1, \ldots, a_n]^T$, on a :

$$P = (B_1 \otimes B_2 \otimes B_3 \otimes \cdots \otimes B_p)^{-1} \times X \tag{2.14}$$

En utilisant la propriété de commutativité de l'inverse d'un produit de Kronecker (i.e. : $(A \otimes B)^{-1} = A^{-1} \otimes B^{-1}$) **[4]**, l'équation (2.10) devient :

$$P = (B_1^{-1} \otimes B_2^{-1} \otimes B_3^{-1} \otimes \cdots \otimes B_p^{-1}) \times X \tag{2.15}$$

Chapitre 3

Autres fonctions de base

Dans ce chapitre, nous aborderons d'autres fonctions de base, notamment les fonctions de Bernstein et les fonctions MINVO, qui comme les fonctions B-splines, ont comme objectif de trouver des simplexes avec des courbes polynomiales contenant un volume minimale. Dans le but de faire une comparaison de performance entre les 3 types de fonctions dans le prochain chapitre.

3.1 Problématique

A priori, l'enveloppe convexe de l'ensemble des points de côntrole vus précédemment contient la courbe des fonctions de base. Il est donc nécessaire de trouver le meilleur enveloppe convexe de l'ensemble de ces points et de donner une approximation minimale au simplex de la courbe polynomiale.
Cependant, et malgré que les fonctions B-splines fournissent des solutions pour ces problèmes, mais elles ne sont pas les plus optimales puisque les enveloppes convexes contenant une fonction polynomiale calculées à l'aide des fonctions B-splines ne sont pas minimales. Pour ce fait, nous étudierons les propriétés des fonctions de Bernstein et les fonctions MINVO qui ont le même objectif et qui peuvent éventuellement amener à de meilleurs performances.

3.2 Les fonctions de Bernstein

3.2.1 Polynôme de Bernstein

Les polynômes de Bernstein, nommés ainsi en l'honneur du mathématicien russe Sergeï Bernstein (1880-1968) et qui sont utilisés dans la formulation générale des courbes de Bézier.

Pour un degré $n \geq 0$, sur un intervalle $[0\,;1]$, on définit les polynomes de Bernstein par :

$$Be_{i,n}(t) = \binom{n}{i} t^i (1-t)^{n-i} \tag{3.1}$$

avec i $\in [0; n]$ et Les n+1 polynômes de Bernstein forment une base des polynômes de degré au plus n.

Exemple

On prend un degré $n = 2$ et on obtient :

- $Be_{0,0}(t) = 1$, $Be_{0,1}(t) = 1 - t$
- $Be_{1,1}(t) = t$, $Be_{0,2}(t) = (1 - t)^2$
- $Be_{1,2}(t) = 2(1 - t)t$, $Be_{2,2}(t) = t$
- $Be_{0,3}(t) = (1 - t)^3$, $Be_{1,3}(t) = 3(1 - t)^2 t$
- $Be_{2,3}(t) = 3(1 - t)t^2$, $Be_{3,3}(t) = t^3$

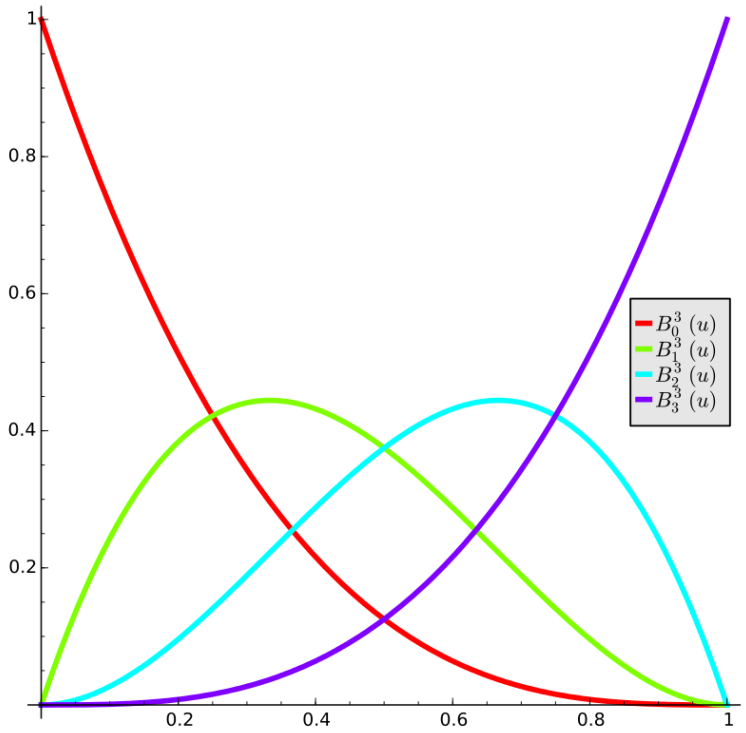


FIGURE 3.1 : Représentation graphique des polynômes de Bernstein Pour n=3.

Quelques propriétés

On citera ci-dessous quelques propriétés des polynomes de Bernstein $\forall t \in [0; 1]$:

- Normalisation : $\sum_{i=0}^{n} Be_{i,n}(t) = 1$
- Positivité : $\forall i \in \{0, 1, \ldots, n\}, Be_{i,n}(t) \geq 0$
- Symétrie : $\forall i \in \{0, 1, \ldots, n\}, Be_{i,n}(t) = Be_{n-i,n}(1 - t)$

3.2.2 Courbes de Bézier

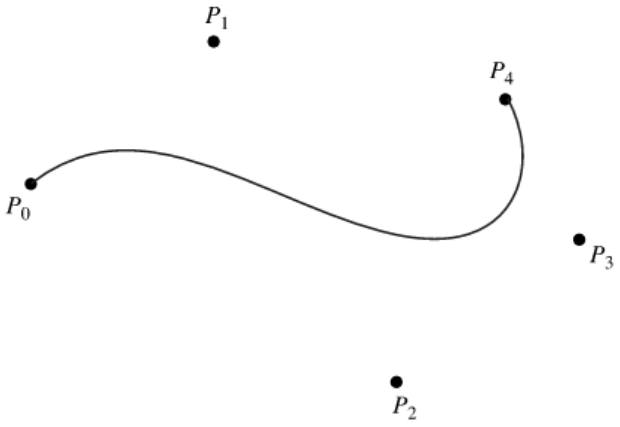
Les courbes de Bézier sont des courbes polynomiales que pour des n+1 points de contrôle $P_0, P_1, .., P_n$ sont définies par :

$$C(t) = \sum_{i=0}^{n} P_i Be_{i,n}(t) \tag{3.2}$$

avec t $\in [0; 1]$ et Les $Be_{i,n}$ sont les polynômes de Bernstein .

Quelques propriétés

- La courbe est à l'intérieur de l'enveloppe convexe des points de contrôle.
- La courbe commence par le point P_0 et se termine par le point P_n, mais ne passe pas a priori par les autres points de contrôle.
- Une courbe de Bézier est infiniment dérivable (de classe C^∞).
- Chaque restriction d'une courbe de Bézier est aussi une courbe de Bézier.



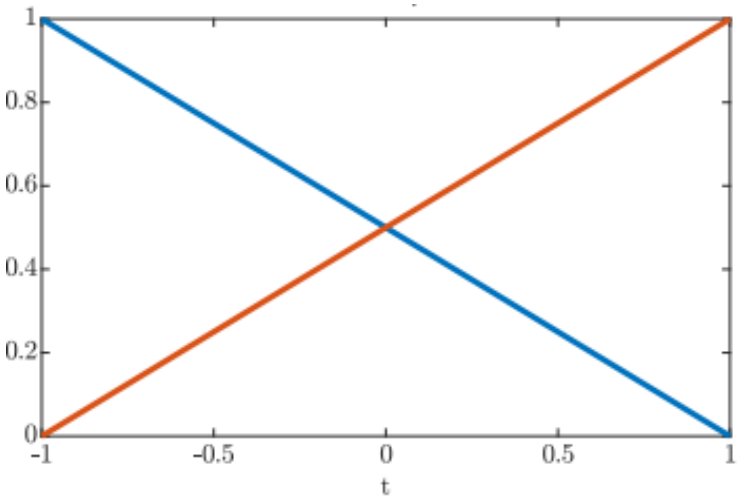
3.3 Les fonctions MINVO

les fonctions MINVO sont la somme pondérée de plusieurs fonctions de base de degré n qui sont utilisés pour obtenir une courbe paramétré pour des points de controle $P_0, P_1, .., P_n$, ces fonctions ont les memes fonctionalités que les B-splines et Bernstein , ce qui nous mène à les comparer et chercher la différence entre eux dans ce qui suit.

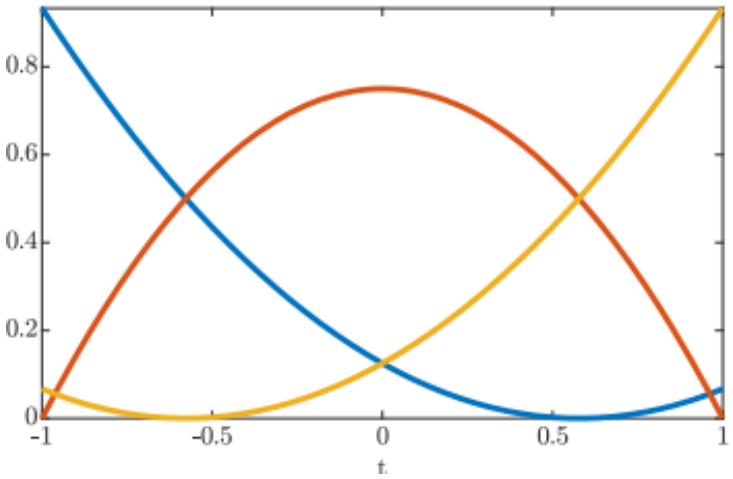
Représentation graphique des MINVO

Voici quelques représentation graphiques des courbes des MINVO pour différents degrés n :

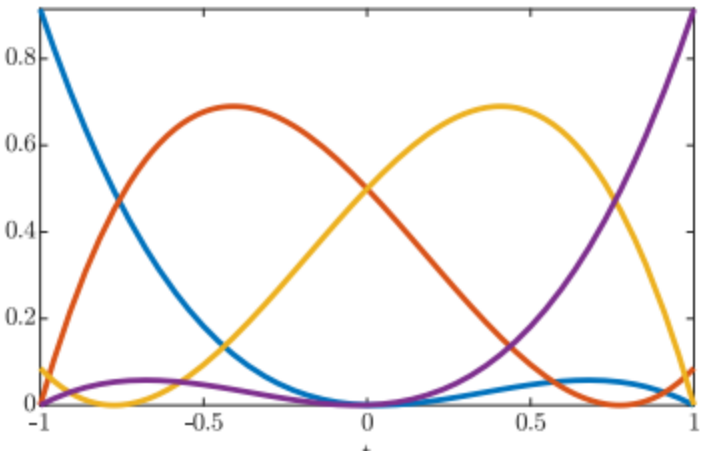
Pour n=1 :



Pour n=2 :



Pour n=3 :



Chapitre 4

Résultats et gestion du projet

4.1 Exploitation de résultats

4.1.1 Bissection

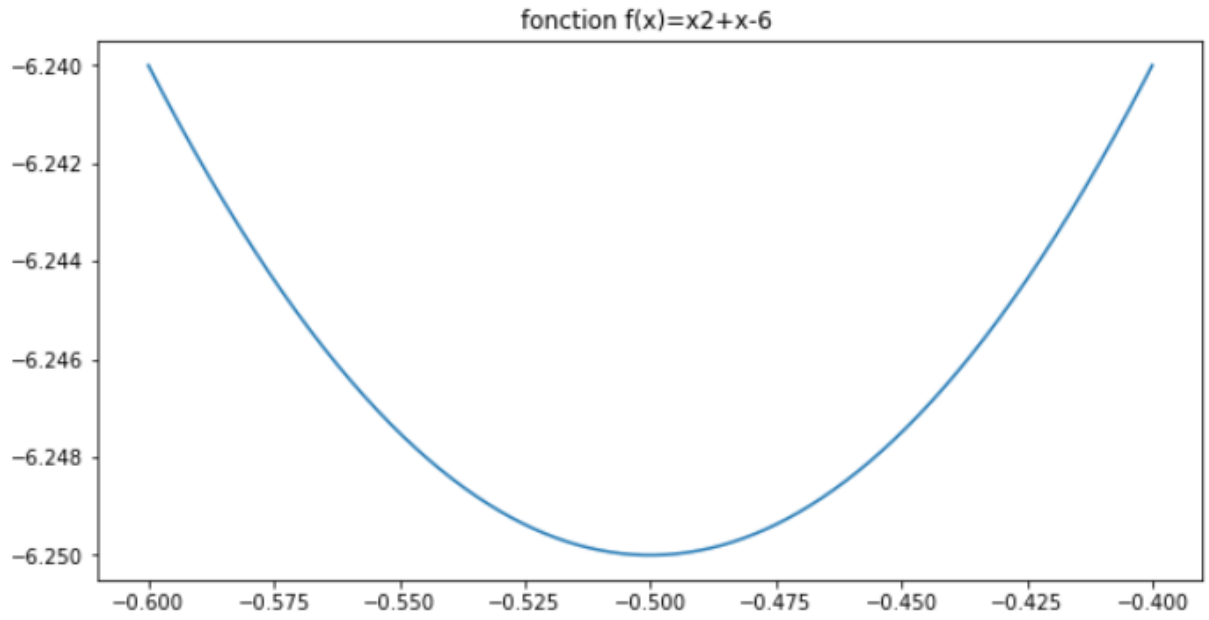
Fonction de degré 2

On choisit la fonction $f(x) = x^2 + x - 6$ de degré $n = 2$ et on applique notre algorithme de bissection sur l'intervalle $[-1; 1]$:

```
# on pose l intervalle [-1,1]
i1 = it.Intervalle(-1, 1)

# Definition de la fonction f(x)=x2+x-6
def fonction(it):
    i=it.__pow__(2)+it.__add_n__(-6)
    return i

__bisection__(i1, fonction)

#tracage de la fonction
x = np.linspace(-0.6, -0.4, 1000)
y = x ** 2+x-6
fig = plt.figure(figsize = (10, 5))
plt.plot(x, y)
plt.title("fonction f(x)=x2+x-6 ")
plt.show()
```

Visualisation du résultat :

```
Intervalle : [-0.5 , -0.4921875]
Number of Iterations : 441
```



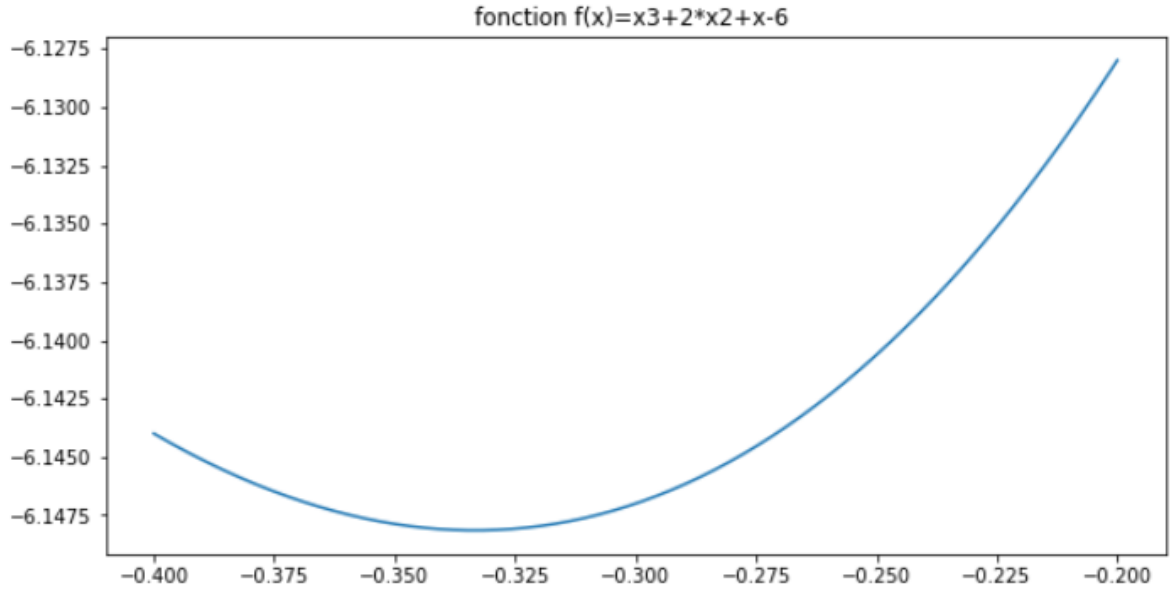
Fonction de degré 3

On choisit la fonction $f(x) = x^3 + 2x^2 + x - 6$ de degré $n = 3$ et on applique notre algorithme de bissection sur l'intervalle $[-1; 1]$:

```
#definition d'une fonction d'ordre 3 : f(x)=x3+2*x2+x-6
def fonction3(it):
    i=it.__pow__(3)+it.__pow__(2).__mul_n__(2)+it.__add_n__(-6)
    return i


__bisection__(i1, fonction3)

#tracage de la fonction
x = np.linspace(-0.4, -0.2, 1000)
y = x ** 3+2*x**2+x-6
fig = plt.figure(figsize = (10, 5))
plt.plot(x, y)
plt.title("fonction f(x)=x3+2*x2+x-6 ")
plt.show()
```

Visualisation du résultat :

```
Intervalle : [-0.328125 , -0.3203125]
Number of Iterations : 441
```



4.1.2 Comparaison des fonctions de base

Pour faire une comparaison entre les fonctions MINVO, Bernstein et B-splines, nous allons prendre un exemple en 2D et un autre en 3D d'une courbe polynomiale et nous regardons le simplex obtenu pour cette courbe par chaque fonction ainsi que son volume. Les résultats obtenus pour cet exemple peuvent se généraliser à toute courbe polynomiale.
Les exemples utilisés dans cette partie ainsi que les figures sont tirés de l'article de M. Jésus Tordesillas **[2]**.

Exemple 1 (en 2D)

Pour une courbe polynomiale donné, on peut remarquer que les fonctions MINVO donnent un simplex qui est 1.3 et 5.2 fois plus petit que celui trouvé par les fonctions Bernstein et B-splines respectivement (Figure 4.1).

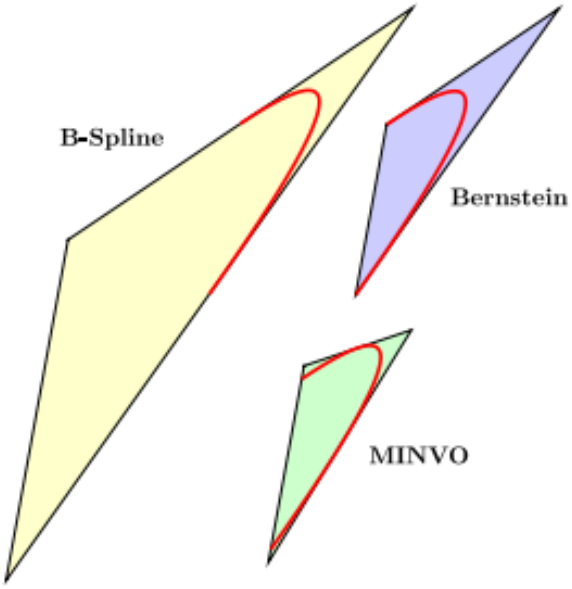


FIGURE 4.1 : Le simplex donné par les MINVO, Bernstein et B-splines en 2D

Dans le sens contraire, pour un simplex donné, les fonctions MINVO donnent une courbe polynomiale contenue dans le simplex, dont l'enveloppe convexe est 1.3 et 5.2 fois plus petit que celui trouvé par les fonctions Bernstein et B-splines respectivement (Figure 4.2).

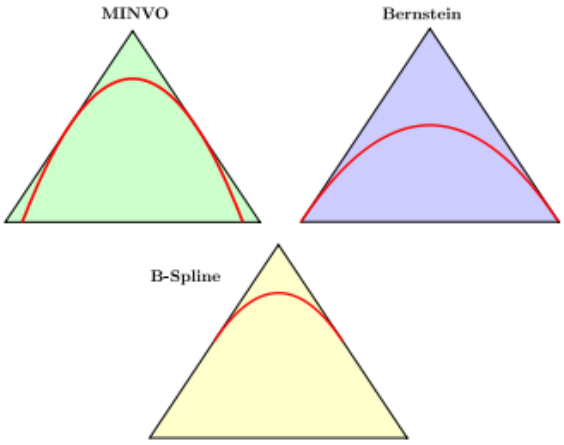


FIGURE 4.2 : L'enveloppe convexe donné par les MINVO, Bernstein et B-splines en 2D

Exemple 2 (en 3D)

Pour une courbe polynomiale donné, on peut remarquer que les fonctions MINVO donnent un simplex qui est 2.36 et 254.9 fois plus petit que celui trouvé par les fonctions Bernstein et B-splines respectivement (Figure 4.3).

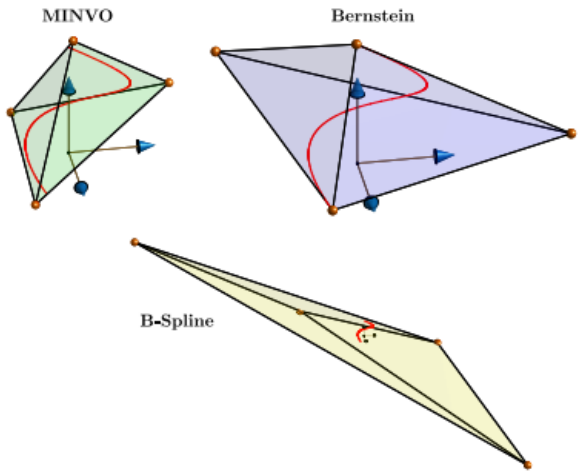


FIGURE 4.3 : Le simplex donné par les MINVO, Bernstein et B-splines en 3D

Dans le sens contraire, pour un simplex donné, les fonctions MINVO donnent une courbe polynomiale contenue dans le simplex, dont l'enveloppe convexe est 2.36 et 254.9 fois plus petit que celui trouvé par les fonctions Bernstein et B-splines respectivement (Figure 4.4).

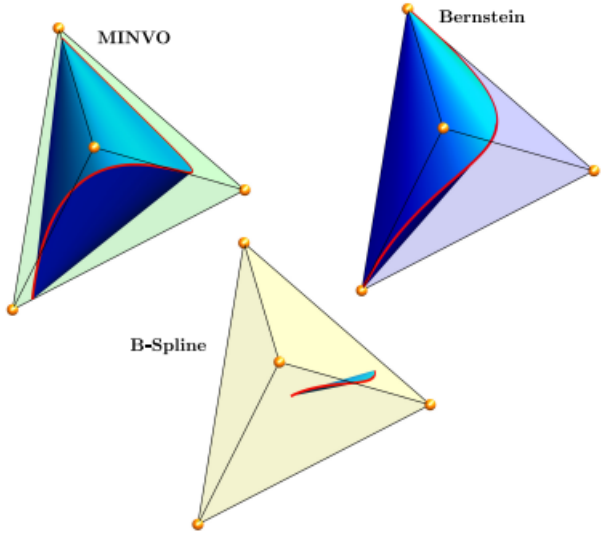


FIGURE 4.4 : L'enveloppe convexe donné par les MINVO, Bernstein et B-splines en 3D

Conclusion

Ainsi, il est clair que les fonctions MINVO fournissent des résultats largement meilleurs que les fonctions Bernstein et les fonctions B-splines surtout. Il est donc recommandé d'utiliser les fonctions MINVO dans la résolution des problèmes d'optimisation pour une meilleure performance.

4.2 Gestion du projet

4.2.1 Organisation du travail

L'organisation du travail ne nous a pas posé de problème puisque c'est pas la première fois qu'on travaille par binome sur un projet , donc pour bien gérer le travail on a fait des réunions hebdomadaires, d'abord entre nous les deux pour discuter nos résultats de recherche , et après avec nos tuteurs M. Bouchon et M. Lengagne afin de lever nos doutes et d'avancer plus sereinement dans ce projet , on utilisait la platforme Forge qui nous a faciliter la collaboration continue à distance de notre code.

Diagramme de GANTT

Ci-dessous le diagramme de Gantt modélisant le déroulement de notre travail :

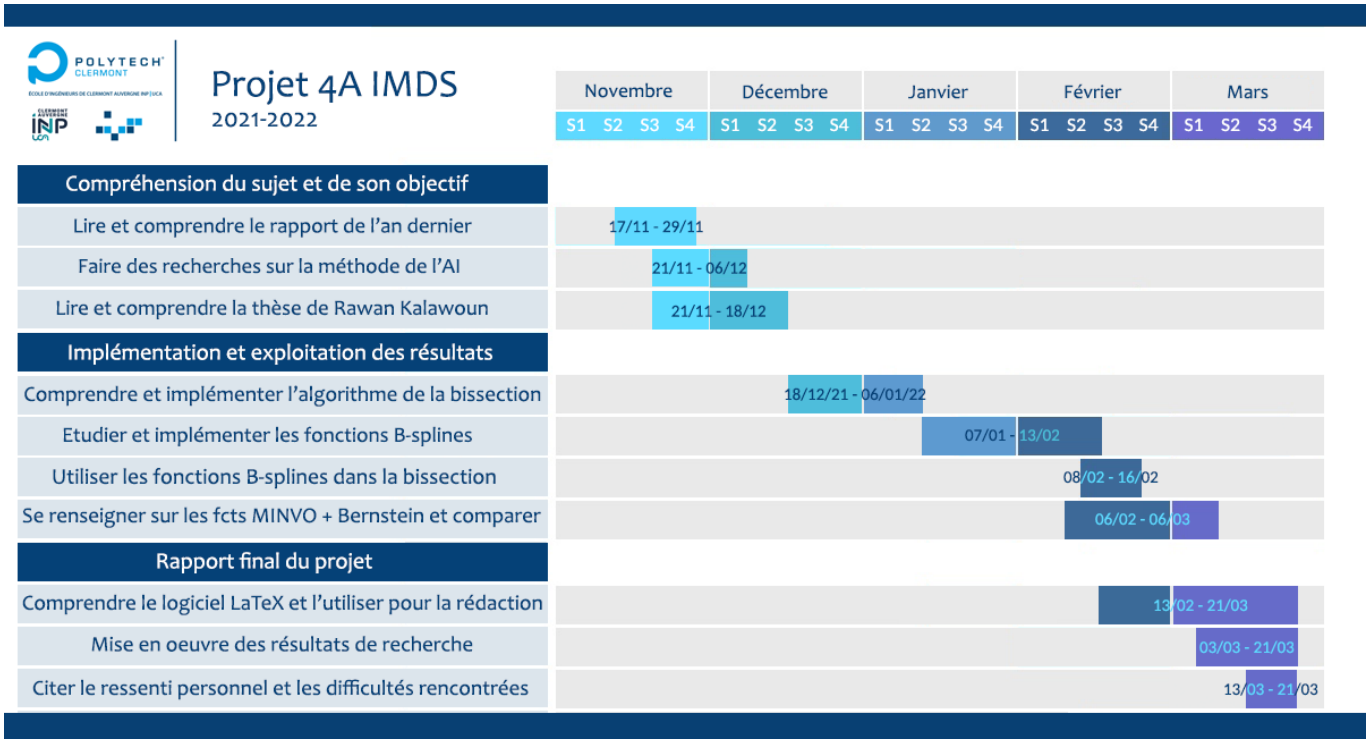


FIGURE 4.5 : Diagramme de GANTT

4.2.2 Difficultés rencontrées

Pendant la réalisation de ce projet et de ce rapport, on a rencontré pas mal de difficultés qu'on a assayé de surmonter. Parmi ces difficultés :

- **Mauvaise gestion du temps de travail :** Avant de commencer le projet et notre recherche, on avait fixer un plan pour ceci, mais plus tard on a rendu compte que le temps qu'on a consacré pour chaque tâche n'était pas suffisant surtout avec tous les examens et les autres projets qu'on avait. Cette mauvaise gestion avait ses conséquences bien sûr ; par exemple on a pas réussi à respecter nos échéances. Aussi on a l'impression qu'on pouvait

faire mieux dans ce projet si on avait prévu un meilleur plan et une meilleure gestion du temps de travail.

- **Manque de ressources :** Tout au long de notre travail, on a fait beaucoup de recherches sur internet pour essayer de comprendre le sujet, de proposer des améliorations et de réaliser l'objectif de ce projet. Cependant, on a galéré plusieurs fois pour trouver ce qu'on cherchait, et surtout dans la partie des fonctions MINVO où on n'est pas arrivé à tomber sur des résultat là dessus, à part l'article **2**.

- **Implémentation de la bissection avec les B-splines :** On a aussi trouver des difficultés dans la partie de la programmation, et en particulier l'utilisation des fonctions B-splines et MINVO dans l'algorithme de la bissection. On a consacré pas mal du temps pour ceci mais malheureusement on n'a pas réussi à y arriver.

Malgré toutes ces difficultés, on a assayé tout au long du projet à proposer des solutions pour qu'elles ne nous empêchent pas à réaliser notre objectif. Ainsi, et à part quelques erreurs qu'on pouvait éviter, on peut sincérement dire qu'on est satisfait de notre travail ‘

4.2.3 Ressenti personnel

- **Abdelali :** Personnellement, ce projet était une éxperience largement positive, puisqu'il m'a permis de développer mes compétences dans la recherche scientifique ainsi que l'utilisation de nouveaux outils de travail à savoir les logiciels Git et LaTeX avec lequel on a rédigé ce rapport. De plus, j'ai pu à travers ce projet, développer ma capacité à faire face aux difficultés et avoir le réflexe pour rapidement proposer des solutions.

- **Rayhane :** Pour ma part, j'ai trouvé que le fait de réaliser un projet mathématique avec une approche informatique très interessant , ça m'a permis de développer les compétences acquises pendant mon parcours universitaire . Ainsi que j'ai eu la chance d'apprendre de nouvelles choses comme l'utilisation du Git et LaTeX et sans oublier quelques nouvelles notions sur Python .

‘

4.2.4 Remerciements

Nous tenons à remercier d'abord nos tuteurs, M. Bouchon (Polytech et Laboratoire de Mathématiques Blaise Pascal) ainsi que M. Lengagne (Polytech et Institut Pascal) pour le temps qu'ils ont consacré pour nous tout au long de ce projet , ainsi que leur aide et le partage de leurs connaissances.
Nous remercions aussi toutes les personnes ayant contribué d'une façon ou d'une autre à ce projet et qui nous ont aidé ou conseillé.

Conclusion

En conclusion, on s'est intéressé à travers ce projet à des problèmes de robotique, consistant à trouver la solution d'un problème d'optimisation sous contraintes.
La méthode qu'on a étudié est celle de la bissection qui consiste à borner la solution optimale du problème en découpant le domaine d'entrée en plusieurs sous domaine. D'autre part, on a abordé les fonctions B-splines qui, avec leurs propriétés uniques, permettent d'évaluer les extremas des fonctions sur un sous domaine. L'objectif de ce projet était d'étudier d'autres fonctions de base amenant à de meilleurs performances, c'est la raison pour laquelle on a examiné également les fonctions MINVO et les fonctions de Bernstein
Finalement, une comparaison est faite entre les trois fonctions de base, pour conclure que les fonctions MINVO sont les plus performantes et les plus adaptées à ce genre de problèmes.

Bibliographie

[1] Rawan Kalawoun, *Motion planning of multi-robot system for airplane stripping.*

[2] Jesus Tordesillas, Jonathan P. How, *MINVO Basis : Finding Simplexes with Minimum Volume Enclosing PolynomialCurves.*

[3] Luc Jaulin, *Analyse par intervalles.*

[4] Michel Kieffer *Estimation ensembliste par analyse par intervalles Application à la localisation d'un véhicule.*

[5] Wikipédia, *B-splines, https ://fr.wikipedia.org/wiki/B-spline.*