Analyse par Intervalles pour la Caractérisation de l'Espace des Configurations

Planification de Trajectoires sans Collision pour un Robot Polygonal

Rapport de Stage

ENSTA Bretagne

2025

Contents

1 Introduction

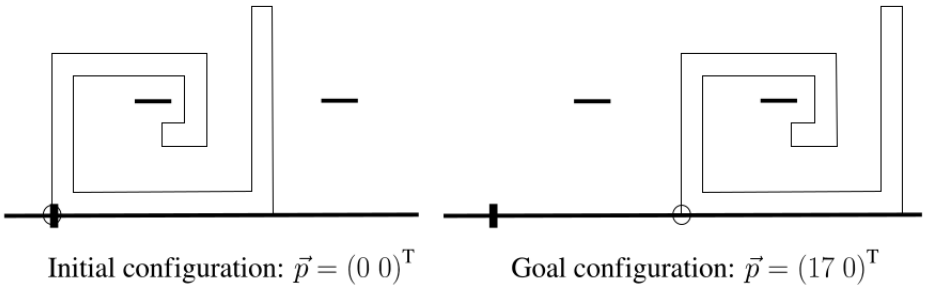

Figure 1: Configurations initiale et finale du robot polygonal

La planification de trajectoires pour des robots mobiles dans des environnements encombrés constitue un problème fondamental en robotique. L'objectif est de trouver un chemin reliant une configuration initiale à une configuration finale sans que le robot n'entre en collision avec les obstacles présents dans l'environnement. Cette problématique, bien qu'intuitive dans sa formulation, se révèle particulièrement complexe lorsque l'on cherche des solutions garanties.

Ce stage s'inscrit dans la continuité des travaux de Luc Jaulin [1] sur l'utilisation de l'analyse par intervalles pour la planification de trajectoires. Le papier de référence présente une approche novatrice combinant l'analyse par intervalles avec des algorithmes de graphes pour caractériser l'espace des configurations admissibles $\mathcal{S}$ d'un objet polygonal non convexe devant naviguer parmi des obstacles segments.

L'espace des configurations (C-space) permet de représenter l'état complet du robot par un point dans un espace de dimension égale au nombre de degrés de liberté [4]. Dans le cas étudié, un polygone à 14 sommets se déplace dans un plan avec deux degrés de liberté : la position p_1 de son premier sommet sur l'axe horizontal et son angle d'orientation p_2. L'espace des configurations est donc bidimensionnel, ce qui permet une visualisation directe des zones admissibles et interdites.

Les travaux antérieurs proposent deux algorithmes, FeasiblePath1 et FeasiblePath2, qui caractérisent l'ensemble $\mathcal{S}$ par des pavages (unions de boîtes) en utilisant l'algorithme SIVIA (Set Inversion Via Interval Analysis) [2]. Ces méthodes, bien que garanties, présentent une complexité en $O(\text{Aire}/\varepsilon^2)$ où ε représente la précision souhaitée.

L'objectif de ce stage est d'explorer différentes approches pour améliorer la caractérisation de la frontière de l'espace des configurations. Nous étudions successivement :

- Les séparateurs, qui permettent de distinguer l'intérieur et l'extérieur d'un ensemble
- Les contracteurs unitaires, qui réduisent efficacement les intervalles
- Le calcul direct de la frontière, qui exploite la structure géométrique du problème

Ces différentes approches sont comparées en termes de temps de calcul et de précision pour un paramètre $\varepsilon = 0.09$.

2 Introduction à la Théorie des Intervalles

2.1 Objectifs et Motivations

L'analyse par intervalles est une branche des mathématiques numériques qui permet d'effectuer des calculs garantis en présence d'incertitudes [3]. Contrairement à l'arithmétique flottante classique qui peut accumuler des erreurs d'arrondi, l'arithmétique d'intervalles fournit des encadrements certains des résultats.

Le but principal de cette théorie est de permettre :

- La propagation rigoureuse des incertitudes à travers les calculs
- La résolution garantie d'équations et d'inéquations non linéaires
- L'inversion ensembliste de fonctions
- La preuve formelle d'inclusion ou d'exclusion de solutions

2.2 Définitions Fondamentales

Définition 2.1 (Intervalle)**.** *Un intervalle* $[x]$ *de* $\mathbb{R}$ *est un sous-ensemble connexe et fermé défini par :*

$$[x] = [x^-, x^+] = \{x \in \mathbb{R} \mid x^- \leq x \leq x^+\}$$

où x^- *est la borne inférieure et* x^+ *la borne supérieure.*

L'ensemble de tous les intervalles de $\mathbb{R}$ est noté $\mathbb{IR}$. La largeur d'un intervalle est définie par $\text{width}([x]) = x^+ - x^-$. Le milieu est $\text{mid}([x]) = (x^- + x^+)/2$.

Définition 2.2 (Boîte)**.** *Une boîte* $[\mathbf{p}]$ *de* $\mathbb{R}^n$ *est le produit cartésien de* n *intervalles :*

$$[\mathbf{p}] = [p_1^-, p_1^+] \times \cdots \times [p_n^-, p_n^+]$$

Dans notre problème, l'espace des configurations étant bidimensionnel, nous travaillons avec des boîtes de $\mathbb{R}^2$: $[\mathbf{p}] = [p_1] \times [p_2]$.

2.3 Opérations sur les Intervalles

L'arithmétique d'intervalles étend les opérations classiques. Pour deux intervalles $[x]$ et $[y]$:

Addition :

$$[x] + [y] = [x^- + y^-, x^+ + y^+]$$

Soustraction :

$$[x] - [y] = [x^- - y^+, x^+ - y^-]$$

Multiplication :

$$[x] \cdot [y] = [\min(x^-y^-, x^-y^+, x^+y^-, x^+y^+), \max(x^-y^-, x^-y^+, x^+y^-, x^+y^+)]$$

Division : (pour $0 \notin [y]$)

$$[x]/[y] = [x] \cdot [1/y^+, 1/y^-]$$

Les fonctions élémentaires (sin, cos, exp, etc.) sont également étendues aux intervalles en calculant l'image de l'intervalle d'entrée.

2.4 Contracteurs

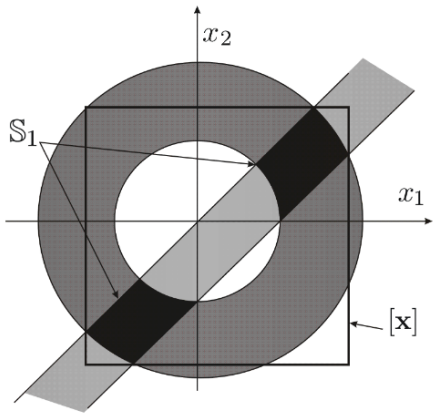

Figure 2: Illustration d'un contracteur : intersection d'une boîte avec un ensemble $\mathbb{S}_1$

Définition 2.3 (Contracteur). *Un contracteur $\mathcal{C}$ pour un ensemble $\mathcal{S} \subseteq \mathbb{R}^n$ est un opérateur qui associe à toute boîte $[\mathbf{p}]$ une boîte contractée $\mathcal{C}([\mathbf{p}])$ telle que :*

1. $\mathcal{C}([\mathbf{p}]) \subseteq [\mathbf{p}]$ *(contractance)*
2. $[\mathbf{p}] \cap \mathcal{S} \subseteq \mathcal{C}([\mathbf{p}])$ *(complétude)*

Un contracteur réduit une boîte tout en conservant toutes les solutions. Un contracteur est dit *minimal* si $\mathcal{C}([\mathbf{p}]) = \text{Hull}([\mathbf{p}] \cap \mathcal{S})$, c'est-à-dire la plus petite boîte contenant l'intersection.

2.5 Séparateurs

Définition 2.4 (Séparateur). *Un séparateur $\mathcal{S}ep$ pour un ensemble $\mathcal{S}$ est un opérateur qui, étant donnée une boîte $[\mathbf{p}]$, retourne deux sous-boîtes :*

- $[\mathbf{p}]^{in}$ *: la partie potentiellement à l'intérieur de $\mathcal{S}$*
- $[\mathbf{p}]^{out}$ *: la partie potentiellement à l'extérieur de $\mathcal{S}$*

Les séparateurs peuvent être combinés par des opérations booléennes [5]:

- Intersection ($\cap$) : combine deux séparateurs pour représenter $\mathcal{S}_1 \cap \mathcal{S}_2$
- Union ($\cup$) : combine deux séparateurs pour représenter $\mathcal{S}_1 \cup \mathcal{S}_2$
- Complémentation ($\neg$) : inverse les rôles de in et out

2.6 Pavages

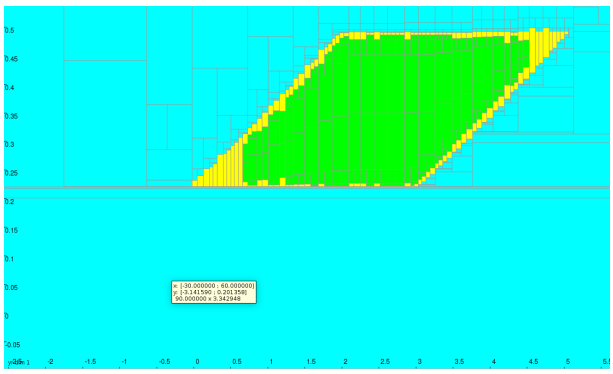

Figure 3: Illustration d'un pavage avec séparation intérieur/extérieur

Définition 2.5 (Pavage). *Un pavage $\mathcal{P}$ d'une boîte $[\mathbf{p}_0]$ est un ensemble de boîtes non chevauchantes dont l'union est égale à $[\mathbf{p}_0]$.*

Définition 2.6 (Sous-pavage). *Un sous-pavage de $\mathcal{P}$ est un sous-ensemble de boîtes de $\mathcal{P}$.*

L'algorithme SIVIA (Set Inversion Via Interval Analysis) construit un pavage en bisectant récursivement les boîtes jusqu'à atteindre une précision ε. Il produit trois sous-pavages :

- $\mathcal{P}^-$: boîtes prouvées à l'intérieur de $\mathcal{S}$
- $\mathcal{P}^+$: boîtes prouvées à l'extérieur de $\mathcal{S}$
- $\Delta\mathcal{P}$: boîtes indéterminées (frontière)

3 Le Problème du Bras Caméléon

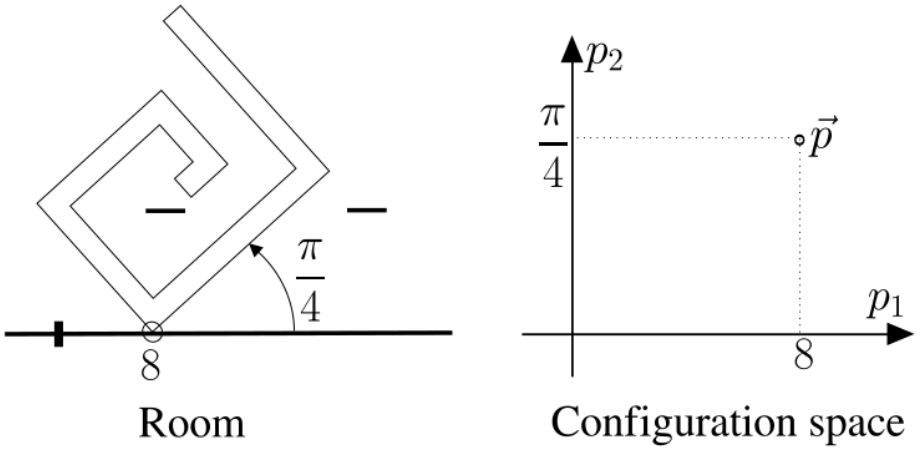

Figure 4: Correspondance entre l'espace physique et l'espace des configurations

3.1 Description du Problème

Le problème considéré est celui d'un robot polygonal non convexe devant naviguer dans un environnement contenant des obstacles segments. Le robot possède 14 sommets dont les coordonnées dans le repère objet sont données par :

Coordonnées x des sommets :

$$s_x = (0, 0, 14, 14, 10, 10, 12, 12, 2, 2, 18, 18, 20, 20)$$

Coordonnées y des sommets :

$$s_y = (0, 14, 14, 6, 6, 8, 8, 12, 12, 2, 2, 18, 18, 0)$$

3.2 Variables de Configuration

L'espace des configurations est paramétré par :

- $p_1 \in \mathbb{R}$: abscisse du premier sommet $\mathbf{s}_1$ dans le repère monde
- $p_2 \in [-\pi, \pi]$: angle d'orientation du robot par rapport à l'horizontale

La position d'un sommet $\mathbf{s}_i = (s_{xi}, s_{yi})$ dans le repère monde est donnée par la transformation :

$$x_i = p_1 + s_{xi}\cos(p_2) - s_{yi}\sin(p_2) \tag{1}$$
$$y_i = s_{xi}\sin(p_2) + s_{yi}\cos(p_2) \tag{2}$$

3.3 Définition des Obstacles

Deux segments obstacles sont considérés :

$$\text{Obstacle 1 : } \mathbf{a}_1 = (8, 10), \quad \mathbf{b}_1 = (11, 10) \tag{3}$$
$$\text{Obstacle 2 : } \mathbf{a}_2 = (25, 10), \quad \mathbf{b}_2 = (28, 10) \tag{4}$$

Ces segments sont horizontaux et situés à la hauteur $y = 10$.

3.4 Domaine de Recherche

Le domaine de recherche dans l'espace des configurations est :

$$[\mathbf{p}_0] = [-28, 57] \times [-1.4, 2.7]$$

Ce domaine a été choisi pour contenir l'intégralité du chemin solution.

3.5 Conditions de Collision

Une configuration $\mathbf{p} = (p_1, p_2)$ est admissible si et seulement si :

1. Aucune arête du robot n'intersecte un segment obstacle
2. Aucune extrémité d'obstacle n'est à l'intérieur du polygone robot

Ces deux conditions définissent naturellement deux cas de contact à analyser.

4 Analyse Géométrique : Distinction des Cas

La frontière de l'espace des configurations $\mathcal{S}$ correspond aux configurations où le robot est en contact avec un obstacle sans le traverser. Cette frontière est composée de deux types de contacts :

4.1 Cas 1 : Coin de l'Obstacle sur Arête du Robot

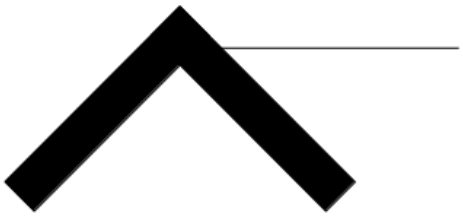

Figure 5: Cas 1 : un coin de l'obstacle touche une arête du robot

Dans ce cas, une extrémité (x_c, y_c) d'un segment obstacle se trouve exactement sur une arête $[\mathbf{s}_i, \mathbf{s}_{i+1}]$ du robot.

4.1.1 Formulation Mathématique

Soit (x_c, y_c) un coin de l'obstacle (par exemple $(8, 10)$). Le point appartient à l'arête $[\mathbf{s}_i, \mathbf{s}_{i+1}]$ du robot si et seulement s'il existe $t \in [0, 1]$ tel que :

$$x_c = p_1 + (s_{xi} + t \cdot d_{xi}) \cos(p_2) - (s_{yi} + t \cdot d_{yi}) \sin(p_2) \tag{5}$$
$$y_c = (s_{xi} + t \cdot d_{xi}) \sin(p_2) + (s_{yi} + t \cdot d_{yi}) \cos(p_2) \tag{6}$$

où $d_{xi} = s_{x(i+1)} - s_{xi}$ et $d_{yi} = s_{y(i+1)} - s_{yi}$.

4.1.2 Reformulation Polaire

En posant $u(t) = s_{xi} + t \cdot d_{xi}$ et $v(t) = s_{yi} + t \cdot d_{yi}$, on peut écrire la contrainte sur y sous forme polaire :

$$y_c = r(t) \cos(p_2 - \varphi(t))$$

où :

$$r(t) = \sqrt{u(t)^2 + v(t)^2} \tag{7}$$
$$\varphi(t) = \arctan 2(u(t), v(t)) \tag{8}$$

Cette formulation est dite *minimale* car p_2 n'apparaît qu'une seule fois dans l'équation, ce qui améliore l'efficacité de la contraction.

4.1.3 Résolution

La solution en p_2 est :

$$p_2 = \varphi(t) \pm \arccos\left(\frac{y_c}{r(t)}\right)$$

Cette équation n'a de solution que si $|y_c| \leq r(t)$. Une fois p_2 et t déterminés, p_1 s'obtient par :

$$p_1 = x_c - u(t)\cos(p_2) + v(t)\sin(p_2)$$

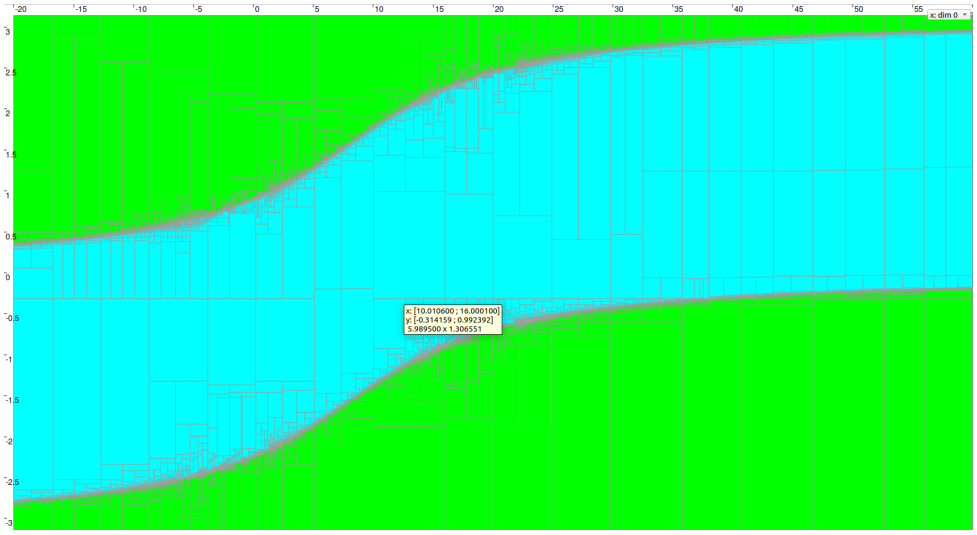

Figure 6: Pavage obtenu pour le Cas 1 avec un séparateur

4.2 Cas 2 : Sommet du Robot sur Segment Obstacle

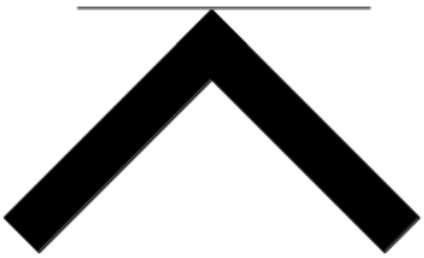

Figure 7: Cas 2 : un sommet du robot touche le segment obstacle

Dans ce cas, un sommet $\mathbf{s}_i$ du robot se trouve exactement sur un segment obstacle $[\mathbf{a}, \mathbf{b}]$.

4.2.1 Formulation Mathématique

Pour les obstacles horizontaux à hauteur $y = 10$, la contrainte est :

$$y_i = s_{xi}\sin(p_2) + s_{yi}\cos(p_2) = 10$$

4.2.2 Reformulation Polaire

En posant $r_i = \sqrt{s_{xi}^2 + s_{yi}^2}$ et $\varphi_i = \arctan 2(s_{xi}, s_{yi})$, cette équation devient :

$$r_i \cos(p_2 - \varphi_i) = 10$$

4.2.3 Résolution

Les solutions sont :

$$p_2 = \varphi_i \pm \arccos\left(\frac{10}{r_i}\right)$$

Cette équation n'a de solution que si $r_i \geq 10$. La contrainte supplémentaire est que l'abscisse x_i du sommet doit être dans l'intervalle $[x_a, x_b]$ du segment obstacle :

$$x_a \leq p_1 + s_{xi}\cos(p_2) - s_{yi}\sin(p_2) \leq x_b$$

Ce qui donne un intervalle pour p_1 une fois p_2 fixé.

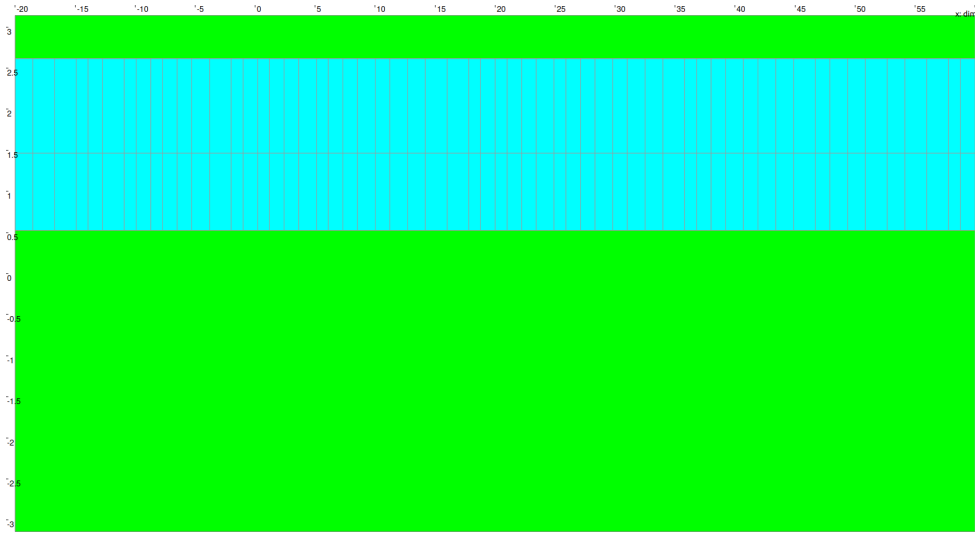

Figure 8: Pavage obtenu pour le Cas 2 avec un séparateur

5 Approche par Séparateurs

5.1 Principe Théorique

L'approche par séparateurs consiste à décomposer le problème en sous-problèmes élémentaires, puis à les combiner par des opérations booléennes. Chaque condition de collision est représentée par un séparateur.

Pour notre problème, nous définissons :

- $f_i(p_1, p_2)$: fonction donnant la hauteur du sommet i normalisée par 10
- $g_i(p_1, p_2)$: fonction pour l'intersection arête-coin de l'obstacle 1 (extrémité gauche)
- $h_i(p_1, p_2)$: fonction pour l'intersection arête-coin de l'obstacle 1 (extrémité droite)
- $i_k(p_1, p_2), j_k(p_1, p_2)$: fonctions similaires pour l'obstacle 2

5.2 Construction des Fonctions

Pour le Cas 2 (sommet sur segment), les fonctions f_i sont de la forme :

$$f_i(p_1, p_2) = \frac{1}{10}(s_{xi}\sin(p_2) + s_{yi}\cos(p_2)) - 1$$

Un sommet est à hauteur $y = 10$ si et seulement si $f_i(p_1, p_2) = 0$.

Pour le Cas 1 (coin sur arête), les fonctions sont plus complexes et dépendent des coordonnées du coin et des paramètres de l'arête. Par exemple, pour l'extrémité $(8, 10)$ de l'obstacle 1 :

$$g_k(p_1, p_2) = \alpha_k \cos(p_2) + \beta_k(8 - p_1)\sin(p_2) + \gamma_k$$

où les coefficients α_k, β_k, γ_k dépendent de l'arête k considérée.

5.3 Combinaison des Séparateurs

Pour chaque segment du robot (arête entre deux sommets consécutifs), on construit un séparateur combiné. Par exemple, pour le premier segment :

$$\text{sep\_seg1} = \neg f_1 \wedge ((\neg g_1 \wedge h_1) \vee (\neg i_1 \wedge j_1))$$

Cette expression logique signifie :

- Le sommet 1 n'est pas sur le segment obstacle ($\neg f_1$)
- ET soit le coin gauche est sur l'arête pour l'obstacle 1 ($\neg g_1 \wedge h_1$)
- OU le coin gauche est sur l'arête pour l'obstacle 2 ($\neg i_1 \wedge j_1$)

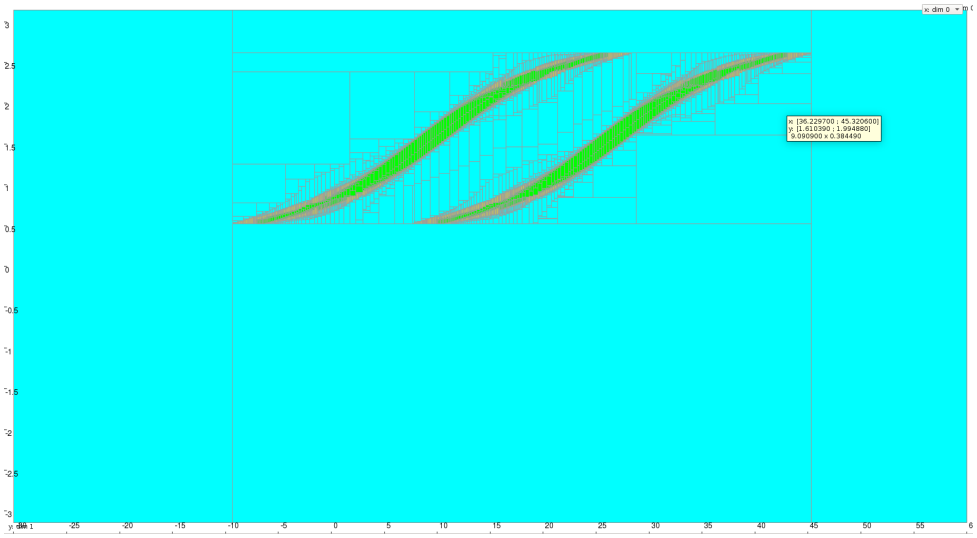

Figure 9: Pavage pour un seul segment du robot

5.4 Implémentation

L'implémentation utilise la bibliothèque `codac` [6] qui fournit les classes `SepFunction` pour créer des séparateurs à partir de fonctions, et les opérateurs `&` (intersection), `|` (union), et `~` (complémentation).

```
f1 = Function('p1', 'p2', '1/10 * (20*sin(p2)) - 1')
f2 = Function('p1', 'p2', '1/10 * (20*sin(p2) + 18*cos(p2)) - 1')
# ... autres fonctions f

sep_f1 = SepFunction(f1, [-oo, 0])
```

```
sep_f2 = SepFunction(f2, [-oo, 0])
```

Listing 1: Construction des séparateurs pour les fonctions f

La combinaison finale des séparateurs est réalisée par :

```
sep_seg1 = ~sep_f1 & ((~sep_g1 & sep_h1) | (~sep_i1 & sep_j1))
# ... autres segments

sep_final = sep_seg1 | sep_seg2 | ... | sep_seg14
```

Listing 2: Combinaison des séparateurs

L'algorithme SIVIA est ensuite appliqué avec ce séparateur combiné pour paver l'espace des configurations.

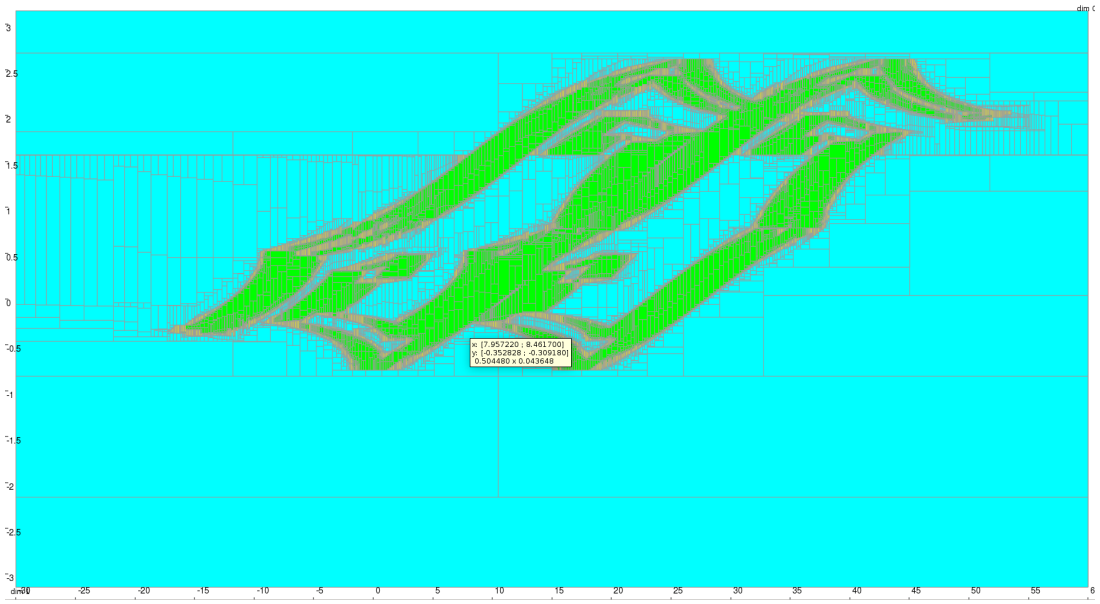

Figure 10: Pavage final obtenu avec la combinaison de tous les séparateurs

6 Approche par Contracteurs Unitaires

6.1 Motivation

L'approche par séparateurs, bien que correcte, peut manquer d'efficacité car elle ne propage pas les contraintes de manière optimale. Les contracteurs unitaires permettent une contraction plus fine en traitant chaque contrainte élémentaire séparément.

L'idée centrale est de concevoir des contracteurs où chaque variable n'apparaît qu'une seule fois dans l'expression de la contrainte. Cela garantit une contraction optimale sans surestimation due à la dépendance des variables.

6.2 Contracteur pour le Cas 1 : Formulation avec Paramètre

6.2.1 Variables Auxiliaires

Pour le Cas 1, nous introduisons le paramètre $t \in [0, 1]$ indiquant la position sur l'arête du robot. Les contraintes deviennent :

$$y_c = (s_{xi} + t \cdot d_{xi}) \sin(p_2) + (s_{yi} + t \cdot d_{yi}) \cos(p_2) \tag{9}$$

$$x_c = p_1 + (s_{xi} + t \cdot d_{xi}) \cos(p_2) - (s_{yi} + t \cdot d_{yi}) \sin(p_2) \tag{10}$$

6.2.2 Implémentation

```
ctc_y = CtcFunction(Function("p2", "t",
    f"({sxi} + t*{dxi})*sin(p2) + ({syi} + t*{dyi})*cos(p2) - {yc}"
       ))

ctc_x = CtcFunction(Function("p1", "p2", "t",
    f"p1 + ({sxi} + t*{dxi})*cos(p2) - ({syi} + t*{dyi})*sin(p2) -
       {xc}"))
```

Listing 3: Contracteurs pour le Cas 1 avec paramètre t

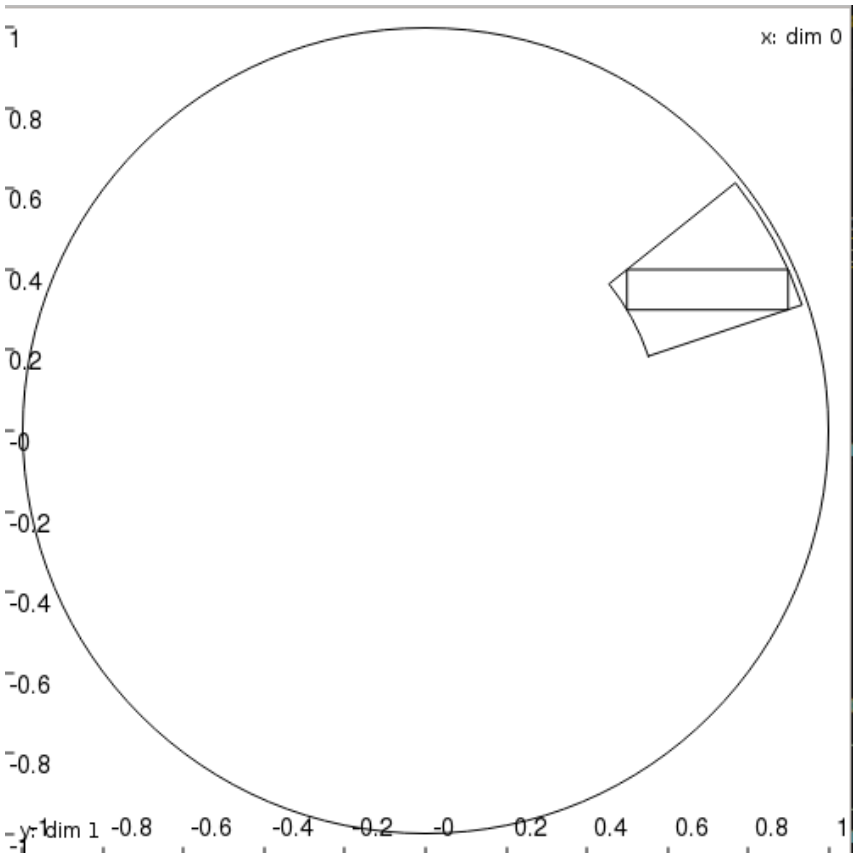

Figure 11: Illustration du contracteur pour le Cas 1

6.3 Contracteur pour le Cas 2 : Cercle et Bande

6.3.1 Interprétation Géométrique

La contrainte $r\cos(p_2 - \varphi) = y_a$ peut s'interpréter géométriquement :

- Le point $(\cos(p_2), \sin(p_2))$ décrit le cercle unité
- La contrainte définit une bande horizontale de largeur infinitésimale à hauteur y_a/r
- L'intersection cercle-bande donne les valeurs admissibles de p_2

6.3.2 Formulation du Contracteur

Soit un sommet avec $r = \sqrt{s_x^2 + s_y^2}$ et $\varphi = \arctan 2(s_x, s_y)$. Le contracteur pour la contrainte est :

$$\mathcal{C}_{cas2} : [p_2] \mapsto [p_2] \cap \{\theta : r\cos(\theta - \varphi) = y_a\}$$

```
ctc_band_circle = CtcFunction(
    Function("theta", "y", f"{r} * cos(theta - {phi}) - y")
)
```

Listing 4: Implémentation du contracteur Cas 2

L'intersection avec une bande (intervalle pour y) permet de contracter efficacement $[p_2]$.

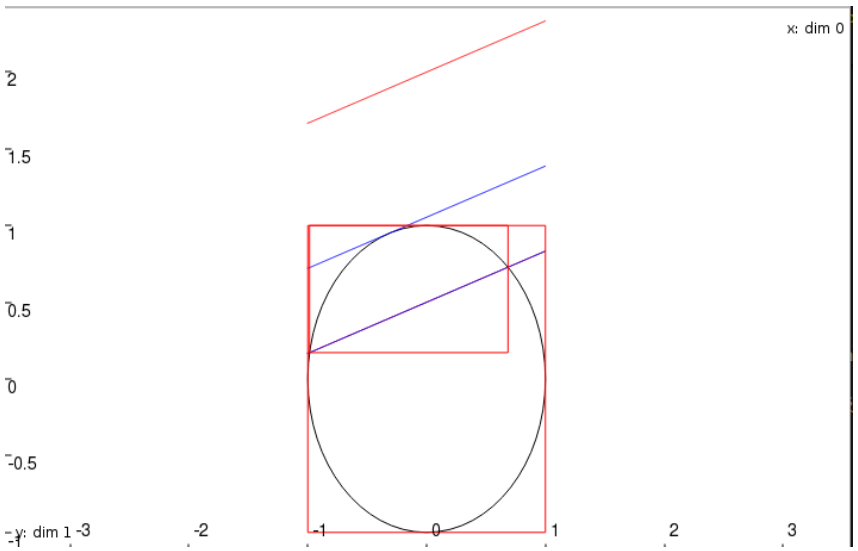

Figure 12: Intersection cercle-bande pour le Cas 2

6.4 Réseau de Contracteurs (Contractor Network)

Pour améliorer la propagation des contraintes, nous utilisons un réseau de contracteurs. L'idée est d'introduire des variables auxiliaires pour que chaque variable n'apparaisse qu'une fois par contrainte.

6.4.1 Variables Auxiliaires

- $c = \cos(p_2)$, $s = \sin(p_2)$
- d_1, d_2 : coordonnées relatives du coin par rapport au sommet
- ρ, θ : coordonnées polaires avec $\rho \in [0, L]$ (longueur de l'arête)

6.4.2 Système de Contraintes

$$\cos(p_2) - c = 0 \tag{11}$$
$$\sin(p_2) - s = 0 \tag{12}$$
$$x_c - p_1 - s_{xi} \cdot c + s_{yi} \cdot s - d_1 = 0 \tag{13}$$
$$y_c - s_{xi} \cdot s - s_{yi} \cdot c - d_2 = 0 \tag{14}$$
$$p_2 + \varphi - \theta = 0 \tag{15}$$
$$\text{CtcPolar}(d_1, d_2, \rho, \theta) \quad \text{avec } \rho \in [0, L] \tag{16}$$

6.4.3 Implémentation avec ContractorNetwork

```
cn = ContractorNetwork()
cn.add(ctc_cos, [p2, c])       # c = cos(p2)
cn.add(ctc_sin, [p2, s])       # s = sin(p2)
cn.add(ctc_d1, [p1, c, s, d1])  # contrainte sur d1
cn.add(ctc_d2, [c, s, d2])      # contrainte sur d2
cn.add(ctc_theta, [p2, theta])  # theta = p2 + phi
cn.add(ctc_polar, [d1, d2, rho, theta])  # CtcPolar
cn.contract()
```

Listing 5: Réseau de contracteurs pour le Cas 1

Le réseau de contracteurs propage automatiquement les contraintes de manière optimale jusqu'à convergence.

6.5 Structure du Contracteur Global

Le contracteur global combine les cas 1 et 2 pour tous les sommets, arêtes et obstacles. Pour chaque boîte $[\mathbf{p}]$:

1. Initialiser l'union des résultats comme ensemble vide
2. Pour chaque configuration de contact (cas 1 ou cas 2) :
   (a) Contracter la boîte avec les contraintes correspondantes
   (b) Si le résultat n'est pas vide, l'ajouter à l'union
3. Retourner l'union des boîtes contractées

Cette approche garantit que toutes les solutions potentielles sont conservées tout en réduisant au maximum l'espace de recherche.

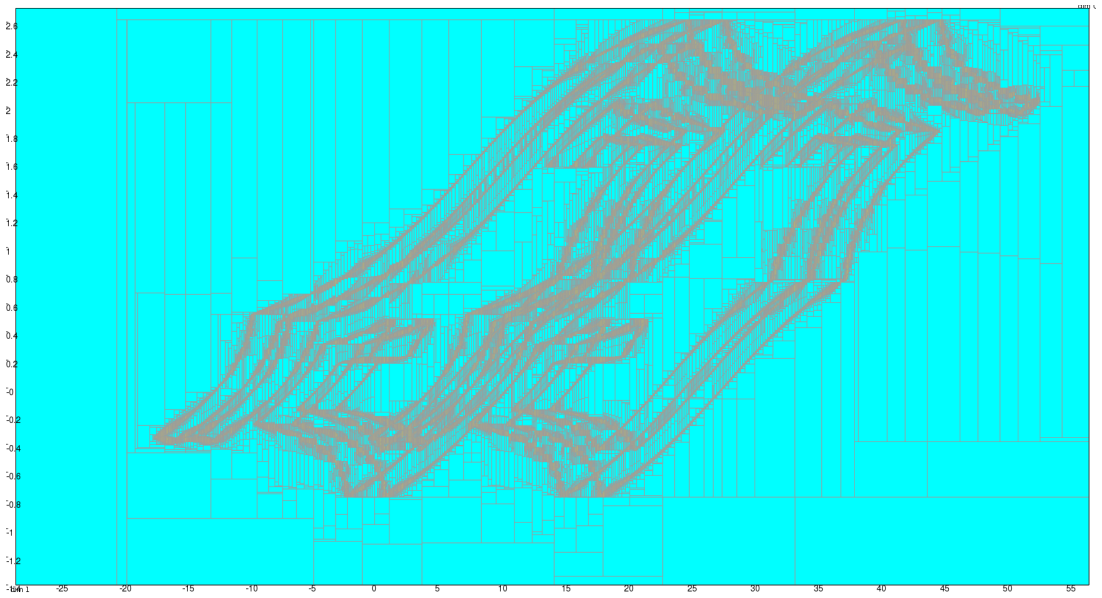

Figure 13: Résultat du contracteur global combinant tous les cas

7 Comparaison des Approches

7.1 Méthodologie de Comparaison

Pour comparer les différentes approches, nous utilisons le volume incertain comme métrique de précision. Le volume incertain correspond à l'aire totale des boîtes indéterminées (frontière) dans le pavage final.

7.1.1 Pourquoi le Volume Incertain ?

Le volume incertain est une mesure pertinente car :

- Il quantifie directement la précision de la caractérisation de la frontière
- Un volume incertain plus faible indique une meilleure approximation
- Il est comparable entre différentes méthodes à ε égal

7.1.2 Calcul du Volume

Pour un pavage $\mathcal{P}$, le volume incertain est :

$$V_{incertain} = \sum_{[\mathbf{p}] \in \Delta\mathcal{P}} \text{width}([p_1]) \times \text{width}([p_2])$$

7.2 Implémentation de SIVIA Modifié

Pour calculer le volume incertain, nous avons modifié l'algorithme SIVIA pour retourner les listes de boîtes :

```
1 def sivia_volume(X0, ctc, eps):
2     stack = [X0]
```

```
    inner_boxes = []
    boundary_boxes = []

    while stack:
        box = stack.pop()
        contracted = ctc.contract(box.copy())

        if contracted.is_empty():
            continue  # boite exterieure
        elif contracted == box and box.max_diam() < eps:
            boundary_boxes.append(box)
        elif contracted.max_diam() < eps:
            inner_boxes.append(contracted)
        else:
            # Bisection
            mid = box.max_diam_index()
            box1, box2 = box.bisect(mid)
            stack.extend([box1, box2])

    return inner_boxes, boundary_boxes
```

Listing 6: SIVIA modifié pour le calcul du volume

7.3 Résultats Comparatifs

Les tests ont été réalisés avec $\varepsilon = 0.09$ sur le domaine $[-28, 57] \times [-1.4, 2.7]$.

Table 1: Comparaison des différentes approches ($\varepsilon = 0.09$)

Méthode	**Temps (s)**	**Volume incertain**	**Précision**
Séparateurs	19	35	94
Contracteurs unitaires	36	46	92
Contracteur + CN	178	46	92

7.4 Analyse des Résultats

7.4.1 Temps de Calcul

Les séparateurs sont les plus rapides car :

- L'implémentation dans `codac` est très optimisée
- La combinaison booléenne est efficace pour ce type de problème

Les contracteurs unitaires sont plus lents car :

- Chaque contraction nécessite plusieurs passes de propagation
- Le réseau de contracteurs ajoute une surcharge computationnelle

Le contracteur avec ContractorNetwork est le plus lent car la propagation dans le réseau nécessite de nombreuses itérations.

7.4.2 Précision

Le volume incertain est plus faible pour les séparateurs dans ce cas particulier, ce qui peut s'expliquer par :

- Les séparateurs produisent des coupes plus nettes
- Les contracteurs peuvent introduire de la surestimation lors de l'union

8 Calcul Direct de la Frontière

8.1 Motivation

Les méthodes précédentes ont une complexité en $O(\text{Aire}/\varepsilon^2)$ car elles pavent l'ensemble du domaine. L'observation clé est que la frontière de collision est une courbe (dimension 1) dans l'espace des configurations (dimension 2).

En calculant directement les points de la frontière, nous pouvons atteindre une complexité en $O(\text{Longueur}/\varepsilon)$, ce qui représente un gain significatif.

8.2 Principe de la Méthode

8.2.1 Cas 1 : Courbes Paramétriques

Pour le Cas 1, la frontière est une courbe paramétrique $(p_1(t), p_2(t))$ avec $t \in [0, 1]$. En discrétisant t en sous-intervalles et en utilisant l'arithmétique d'intervalles, nous obtenons des boîtes garanties encadrant la courbe.

8.2.2 Cas 2 : Segments Horizontaux

Pour le Cas 2, la frontière dans l'espace (p_1, p_2) est composée de segments horizontaux (à p_2 constant). En effet, pour chaque sommet i avec $r_i \geq 10$, l'équation :

$$r_i \cos(p_2 - \varphi_i) = 10$$

a au plus deux solutions exactes en p_2 :

$$p_2 = \varphi_i \pm \arccos\left(\frac{10}{r_i}\right)$$

Pour chaque valeur de p_2, l'intervalle de p_1 est déterminé par la contrainte $x_i \in [x_a, x_b]$.

8.3 Algorithme

```
def compute_boundary_direct(eps):
    boxes = []

    # CAS 2: Segments horizontaux (exact)
    for obstacle in obstacles:
        for i in range(n_vertices):
            r = sqrt(sx[i]**2 + sy[i]**2)
            if r < ya:
```

```
                continue
            phi = atan2(sx[i], sy[i])
            delta = acos(ya / r)

            for p2_exact in [phi + delta, phi - delta]:
                # Calculer intervalle p1
                p1_interval = compute_p1_interval(p2_exact,
                    obstacle)
                boxes.append(create_box(p1_interval, p2_exact, eps)
                    )

    # CAS 1: Courbes parametriques
    for corner in obstacle_corners:
        for edge in robot_edges:
            for t_interval in subdivide([0, 1], n_samples):
                p1, p2 = compute_parametric(t_interval, corner,
                    edge)
                if not p1.is_empty() and not p2.is_empty():
                    boxes.append([p1, p2])

    return merge_boxes(boxes)
```

Listing 7: Calcul direct de la frontière

8.4 Garantie des Résultats

L'utilisation de l'arithmétique d'intervalles garantit que :

- Les boîtes produites contiennent effectivement la frontière
- Aucun point de la frontière n'est manqué
- Le recouvrement est conservatif (surestimation possible mais pas de sous-estimation)

8.5 Fusion des Boîtes

Pour réduire le nombre de boîtes, une étape de fusion est appliquée :

```
def merge_overlapping_boxes(boxes, tolerance=0.01):
    merged = []
    current = boxes[0]

    for box in boxes[1:]:
        if boxes_overlap(current, box, tolerance):
            current = hull(current, box)
        else:
            merged.append(current)
            current = box

    merged.append(current)
    return merged
```

Listing 8: Fusion des boîtes adjacentes

8.6 SIVIA Optimisé pour la Frontière

En combinant le calcul direct avec SIVIA, nous pouvons cibler uniquement les régions contenant la frontière :

```
def sivia_optimized(X0, ctc, eps):
    # Calcul direct de la frontiere approximative
    boundary_approx = compute_boundary_direct(eps * 10)

    # SIVIA uniquement sur les regions proches de la frontiere
    stack = expand_boxes(boundary_approx, margin=eps)

    # ... reste de SIVIA classique
```

Listing 9: SIVIA optimisé avec initialisation par calcul direct

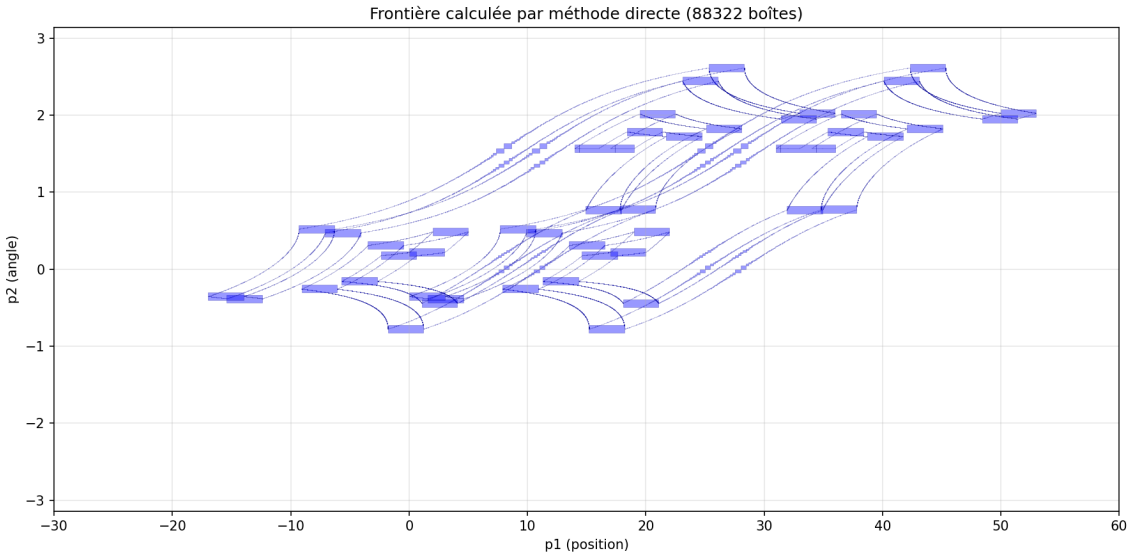

Figure 14: Frontière calculée par la méthode directe

8.7 Comparaison avec les Autres Méthodes

Table 2: Comparaison incluant le calcul direct ($\varepsilon = 0.09$)

Méthode	**Temps (s)**	**Volume incertain**	**Complexité**
Séparateurs	18	35	$O(\text{Aire}/\varepsilon^2)$
Contracteurs	178	46	$O(\text{Aire}/\varepsilon^2)$
Calcul direct	2.3	16	$O(\text{Long}/\varepsilon)$

8.8 Discussion sur les Grosses Boîtes

Certaines configurations peuvent produire des boîtes anormalement grandes. Ces cas se produisent lorsque :

1. **Tangence** : Le cercle unité est tangent à la bande de contrainte, créant une sensibilité élevée

2. **Dégénérescence** : Les contraintes sont presque parallèles, limitant la contraction
3. **Points de rebroussement** : La courbe paramétrique change de direction

Ces situations sont intrinsèques au problème et nécessitent un traitement spécial ou une précision accrue localement.

9 Conclusion

Ce stage a permis d'explorer différentes approches pour la caractérisation de l'espace des configurations dans un problème de planification de trajectoires sans collision. Les principales contributions sont :

Approche par séparateurs : Méthode systématique et modulaire permettant de combiner facilement les différentes contraintes par des opérations booléennes. Bien que correcte, cette approche peut souffrir d'une surestimation due à la propagation non optimale des contraintes.

Contracteurs unitaires : L'utilisation de formulations minimales où chaque variable n'apparaît qu'une fois permet une contraction plus efficace. Le réseau de contracteurs améliore encore la propagation en partageant les variables auxiliaires.

Calcul direct : En exploitant la structure géométrique du problème (la frontière est une courbe 1D), nous avons développé une méthode de complexité réduite $O(\text{Longueur}/\varepsilon)$ au lieu de $O(\text{Aire}/\varepsilon^2)$.

Les résultats montrent que le calcul direct est environ 8 fois plus rapide que les séparateurs et produit une approximation plus précise de la frontière. Cette amélioration est particulièrement significative pour les problèmes de dimension supérieure où la réduction de complexité devient critique.

Perspectives : Ces travaux ouvrent plusieurs pistes :

- Extension à des robots avec plus de degrés de liberté
- Traitement des obstacles non segments (arcs, polygones)
- Intégration dans un système de planification temps réel
- Parallélisation des calculs pour les applications embarquées

L'analyse par intervalles s'affirme comme un outil puissant pour la robotique, offrant des garanties formelles essentielles pour les applications critiques où la sécurité est primordiale.

References

[1] L. Jaulin, *Path planning using intervals and graphs*, Reliable Computing, 1999. `https://www.ensta-bretagne.fr/jaulin/paper_cameleon.pdf`

[2] L. Jaulin and E. Walter, *Set inversion via interval analysis for nonlinear bounded-error estimation*, Automatica, vol. 29, no. 4, pp. 1053-1064, 1993.

[3] R. E. Moore, *Methods and Applications of Interval Analysis*, SIAM, Philadelphia, 1979.

[4] T. Lozano-Pérez, *Spatial planning: a configuration space approach*, IEEE Transactions on Computers, vol. 32, no. 2, 1983.

[5] L. Jaulin and B. Desrochers, *Introduction to the algebra of separators with application to path planning*, Engineering Applications of Artificial Intelligence, vol. 33, pp. 141-147, 2014.

[6] S. Rohou, B. Desrochers, L. Jaulin, *The Codac library: Constraint-propagation for robotics*, 2022. `http://codac.io`

[7] E. W. Dijkstra, *A note on two problems in connexion with graphs*, Numerische Mathematik, vol. 1, pp. 269-271, 1959.

[8] O. Khatib, *Real-time obstacle avoidance for manipulators and mobile robots*, International Journal of Robotics Research, vol. 5, no. 1, pp. 90-98, 1986.