Stage de M2

-

Algèbres géométriques appliquées à la robotique en Coq

-

sous la direction de Julien Narboux

Jonathan Graff

Février - Juillet 2021

Sommaire

Remerciements

Je tiens tout particulièrement à remercier mon tuteur de stage, Julien Narboux, pour sa sympathie et pour avoir pris le temps de m'expliquer tout ce domaine que je découvrais. Ses explications ont toujours été très claires, et je ne compte plus le temps passé ensemble à débugger les preuves en Coq.

Je remercie également Laurent Théry pour sa disponibilité et ses explications quant à sa bibliothèque que nous avons réutilisée, ainsi que pour le temps qu'il a passé à l'adapter à notre problème.

Je remercie aussi Olivier Piccin pour ses explications sur les algèbres géométriques et sur la robotique en général. En particulier, son modèle réduit de robot 6R m'aura bien été utile.

Enfin, je remercie en particulier ma femme pour son soutien indéfectible pendant toute cette année.

I) Introduction

I.1) Présentation de la structure

Mon stage s'est déroulé au laboratoire ICube, situé à Illkirch. Ce laboratoire de recherche a été créé en 2013 par l'Université de Strasbourg, le SNRS, l'INSA et l'ENGEES. Ce laboratoire regroupe des chercheurs dans "le domaine des sciences de l'ingénieur et de l'informatique avec l'imagerie comme thème fédérateur. Les champs d'application privilégiés sont l'ingénierie pour la santé, l'environnement et le développement durable."[1]

Ce laboratoire regroupe plus de 650 membres. Il est divisé en quatre équipes de recherche :

- le Département Imagerie, Robotique, Télédétection et Santé (D-IRTS)
- le Département Électronique du Solide, Systèmes et Photonique (D-ESSP)
- le Département Mécanique (D-M)
- le département Informatique Recherche (D-IR), au sein duquel j'ai effectué mon stage.

Il y a au total 14 équipes de recherche réparties dans ces quatre départements. J'ai effectué mon stage dans l'équipe "Informatique Géométrique et Graphique" (IGG).

I.2) Contexte

Algèbres géométriques

Les algèbres géométriques sont un formalisme mathématique permettant de décrire de façon plus uniforme la géométrie 3D. Dans cet espace, les points, les vecteurs, les plans, les sphères, les cercles, les droites, ainsi que d'autres objets géométriques, mais aussi les transformations usuelles, sont tous représentés par le même objet : **un multivecteur**. Cela simplifie grandement les notations puisque tous les calculs pourront se faire dans cette algèbre. Un autre avantage est que les cas particuliers sont inexistants, par exemple le calcul permettant d'obtenir un plan à partir de deux vecteurs renverra 0 si les deux vecteurs sont colinéaires, il n'y a donc pas besoin de vérifier avant de construire notre plan que les vecteurs sont bien dans la configuration souhaitée. Egalement, ces algèbres incluent les nombres complexes de manière naturelle, mais aussi l'ensemble des quaternions.

Tous ces avantages font qu'en pratique ces algèbres sont très utilisées dans plusieurs domaines : surtout en physique (mécanique classique et quantique, théories électromagnétiques et de la

relativité...), mais aussi en biomécanique, pour la dynamique des vols spatiaux, la vision par ordinateur, et dans le domaine qui va nous intéresser ici : la robotique et plus précisément la robotique médicale.

Assistant de preuve

Les algorithmes implémentés dans des machines peuvent être vérifiés ou non. Par exemple, dans une voiture, ils ne le sont pas. Il n'est pas rare donc de trouver plusieurs centaines de millions de lignes de code pour un logiciel embarqué. Par contre, pas question de fonctionner ainsi dans un avion par exemple. Ici, chaque ligne de code doit être prouvée, il faut démontrer qu'aucun bug ne peut apparaître et faire crasher l'avion. Pour ce faire, on utilise un logiciel d'assistant de preuve qui va vérifier chaque ligne. Prouver un algorithme est en général bien plus long et fastidieux que de le créer, et donc par conséquent le nombre de lignes de code est beaucoup plus restreint, de l'ordre de cent mille pour un avion ou une fusée [2]. Mais cela a une utilité réelle. Par exemple, le 4 juin 1996, le vol 501 de la fusée Ariane s'est écrasé seulement 36.7 secondes après son décollage. Le problème est venu d'un bug informatique de dépassement d'entier dans les registres mémoire des calculateurs électroniques utilisés par le pilote automatique. La fusée transportait quatre satellites d'une valeur de 370 millions de dollars, heureusement aucun humain n'a péri [3]. Mais cela a montré l'importance de la preuve en informatique dans certains domaines. Si les codes avaient été prouvés à l'aide d'un logiciel assistant de preuve à ce moment-là, l'incident aurait très probablement été évité.

I.3) Objectifs

En robotique médicale, aujourd'hui, il n'existe pas d'obligation de prouver les algorithmes régissant la mécanique des robots. Pourtant ceux-ci peuvent exécuter des opérations très complexes, et il serait donc souhaitable que ces algorithmes soient prouvés afin d'être sûr que le robot ne va pas faire quelque chose de non désiré. On peut aisément imaginer les catastrophes que cela engendrerait.

L'objectif de mon stage est donc dans un premier temps de me familiariser avec le logiciel Coq, afin de comprendre comment celui-ci fonctionne puis dans un second temps, de me familiariser avec le concept des algèbres géométriques, concept complètement nouveau pour moi. Enfin, l'objectif est de réutiliser les bibliothèques d'algèbres géométriques de Laurent Théry, afin de les spécialiser pour faire de l'algèbre géométrique conforme, de prouver quelques théorèmes simples,

et de prouver en utilisant certains robots connus que le bras de robot se trouve bien à la fin là où il est souhaitable qu'il aille, et pas ailleurs !

II) Les algèbres géométriques

II.1) Présentation générale

Définitions

L'algèbre géométrique $\mathscr{G}^n$ peut être vue comme une extension de l'espace vectoriel $\mathbb{R}^n$. Les objets de cet algèbre sont appelés **multivecteurs**. L'algèbre possède donc une somme, prolongeant la somme classique des vecteurs, mais elle possède également un produit, appelé **produit géométrique** et vérifiant les axiomes d'une algèbre unitaire associative. Le produit de deux multivecteurs a et b se note ab. Ce produit possède un élément neutre, noté $\mathbf{1}$ et possède la propriété que tout carré est un nombre réel.

On peut déjà remarquer qu'on a alors :

- $\mathbb{R}^n \subset \mathscr{G}^n$
- $\mathbb{R} \subset \mathscr{G}^n$

On définit le **produit scalaire** (ou produit intérieur) par la formule :

$$\boxed{a \cdot b = \frac{1}{2}(ab + ba)}$$

Ce produit scalaire définit une norme avec $||a||^2 = a \cdot a$.

On dira que deux multivecteurs sont **orthogonaux** si leur produit scalaire est nul.

On définit également le **produit extérieur** par la formule

$$\boxed{a \wedge b = \frac{1}{2}(ab - ba)}$$

Ainsi, le produit extérieur est anti-commutatif : $a \wedge b = -b \wedge a$.

Avec ces formules, on remarque qu'on a :

Formule fondamentale :	$ab = a \cdot b + a \wedge b$
Anticommutativité :	Si a et b sont orthogonaux, alors $ab = -ba$.

Dans toute la suite, nous noterons en gras et en lettre minuscule les vecteurs.

Si $\{\boldsymbol{e_1}, \cdots, \boldsymbol{e_n}\}$ est une base orthonormée de $\mathbb{R}^n$, on a les propriétés suivantes :

• $\wedge$ est linéaire et associatif.

$$\bullet\ \boldsymbol{e_i} \wedge \boldsymbol{e_j} = \begin{cases} 0 & \text{si } i = j \\ \boldsymbol{e_i e_j} & \text{si } i \neq j \end{cases} \tag{1}$$

$$\bullet\ \boldsymbol{e_i e_j} = \begin{cases} 1 & \text{si } i = j \\ \boldsymbol{e_i} \wedge \boldsymbol{e_j} & \text{si } i \neq j \end{cases} \tag{2}$$

Les nouveaux éléments construits $\boldsymbol{e_i e_j}$ appartiennent donc à $\mathscr{G}^n$, mais ne sont pas des vecteurs de $\mathbb{R}^n$.

On peut également voir que $\mathscr{G}^n$ est un espace de dimension 2^n, une base étant formée par 1 et l'ensemble des produits de vecteurs distincts : $\{1, \boldsymbol{e_1}, \cdots, \boldsymbol{e_n}, \boldsymbol{e_1 e_2}, \boldsymbol{e_1 e_3}, \cdots \boldsymbol{e_{n-1} e_n}, \boldsymbol{e_1 e_2 e_3}, \cdots, \boldsymbol{e_1 \cdots e_n}\}$.

Pour simplifier les notations, l'élément $\boldsymbol{e_i} \wedge \boldsymbol{e_j} = \boldsymbol{e_i e_j}$ sera noté par la suite e_{ij}.

Bivecteurs - Interprétation géométrique

Prenons deux vecteurs $\boldsymbol{a}$ et $\boldsymbol{b}$. Si les deux vecteurs sont colinéaires, alors le produit extérieur de ces deux vecteurs sera nul. En effet, si $\boldsymbol{a} = \sum_{i=1}^{n} \lambda_i \boldsymbol{e_i}$ et $\boldsymbol{b} = C \sum_{j=1}^{n} \lambda_j \boldsymbol{e_j}$, alors :

$$\boldsymbol{a} \wedge \boldsymbol{b} = C \left(\left(\sum_{i=1}^{n} \lambda_i \boldsymbol{e_i} \right) \wedge \left(\sum_{j=1}^{n} \lambda_j \boldsymbol{e_j} \right) \right) = C \left(\sum_{i=1}^{n} \sum_{j=1}^{n} \lambda_i \lambda_j (\boldsymbol{e_i} \wedge \boldsymbol{e_j}) \right) \text{ par linéarité}$$

$$= C \sum_{i=1}^{n} \sum_{j=1}^{i-1} \lambda_i \lambda_j (\boldsymbol{e_i e_j} + \boldsymbol{e_j e_i}) \text{ d'après (1)}$$

$= 0$ par anti-commutativité.

Les éléments de l'algèbre géométrique $\mathscr{G}^n$ formés par le produit extérieur entre deux vecteurs non colinéaires sont appelés des **bivecteurs**. C'est un cas particulier de multivecteurs.

Géométriquement, dans un espace vectoriel euclidien, un vecteur définit une droite et sa norme est la longueur d'un segment de cette droite. Pour un bivecteur, c'est similaire : on peut imaginer un bivecteur $\boldsymbol{u} \wedge \boldsymbol{v}$ comme définissant le plan vectoriel engendré par $\boldsymbol{u}$ et $\boldsymbol{v}$, et sa norme est une partie de l'aire de ce plan : c'est exactement l'aire du parallélogramme engendré par $\boldsymbol{u}$ et $\boldsymbol{v}$. Un bivecteur possède également une orientation : celle de $\boldsymbol{u}$ vers $\boldsymbol{v}$.

Au passage, on peut aussi noter la formule du parallélogramme

$$||\boldsymbol{u} \wedge \boldsymbol{v}|| = ||\boldsymbol{u}|| \cdot ||\boldsymbol{v}|| \cdot \sin(\boldsymbol{u}; \boldsymbol{v})$$

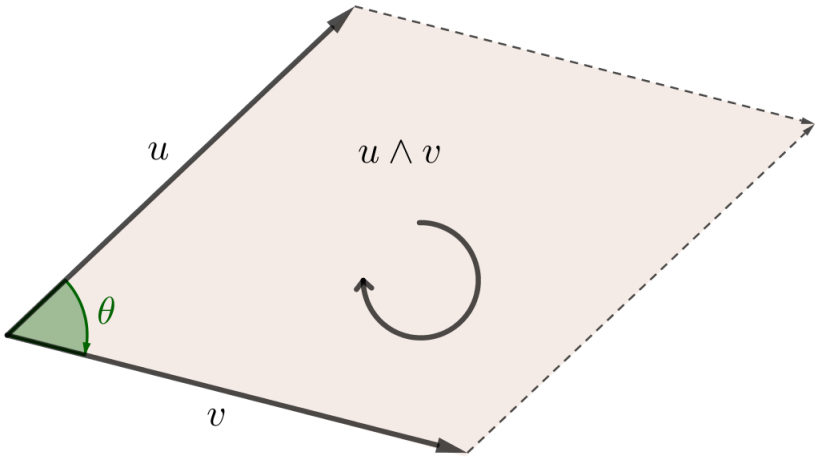


Figure 1: Un bivecteur

Egalité de bivecteurs

Deux vecteurs sont égaux s'ils ont même direction, même sens et même norme, pour des bivecteurs c'est pareil : deux bivecteurs $\boldsymbol{a} \wedge \boldsymbol{b}$ et $\boldsymbol{c} \wedge \boldsymbol{d}$ sont égaux si et seulement si ils définissent le même plan, ont la même orientation et ont la même norme. Au final, pour les représenter géométriquement, on doit tracer une partie du plan formé par les deux vecteurs, de même aire que leur parallélogramme et de même orientation. Ainsi, ces bivecteurs-là sont par exemple égaux :

Somme de bivecteurs

Pour additionner deux bivecteurs $M_1 = \boldsymbol{a} \wedge \boldsymbol{b}$ et $M_2 = \boldsymbol{c} \wedge \boldsymbol{d}$ dont les plans induits ont une intersection, on procède comme suit :

- on modifie si nécessaire un ou deux bivecteurs de façon à ce que deux vecteurs soient identiques (c'est toujours possible si les plans s'intersectent), on a donc deux bivecteurs $\boldsymbol{u} \wedge \boldsymbol{w}$ et $\boldsymbol{w} \wedge \boldsymbol{v}$
- Le résultat sera le bivecteur $(\boldsymbol{u} + \boldsymbol{v}) \wedge \boldsymbol{w}$

Lames et multivecteurs

On peut continuer le même genre de constructions pour construire des trivecteurs $\boldsymbol{a} \wedge \boldsymbol{b} \wedge \boldsymbol{c}$, où les trois vecteurs $\boldsymbol{a}$, $\boldsymbol{b}$ et $\boldsymbol{c}$ ne sont pas coplanaires. Dans ce cas, le trivecteur représentera un espace de dimension 3 avec une orientation et une norme qui correspondra au volume du parallélépipède engendré par les trois vecteurs, comme sur la figure ci-dessous :

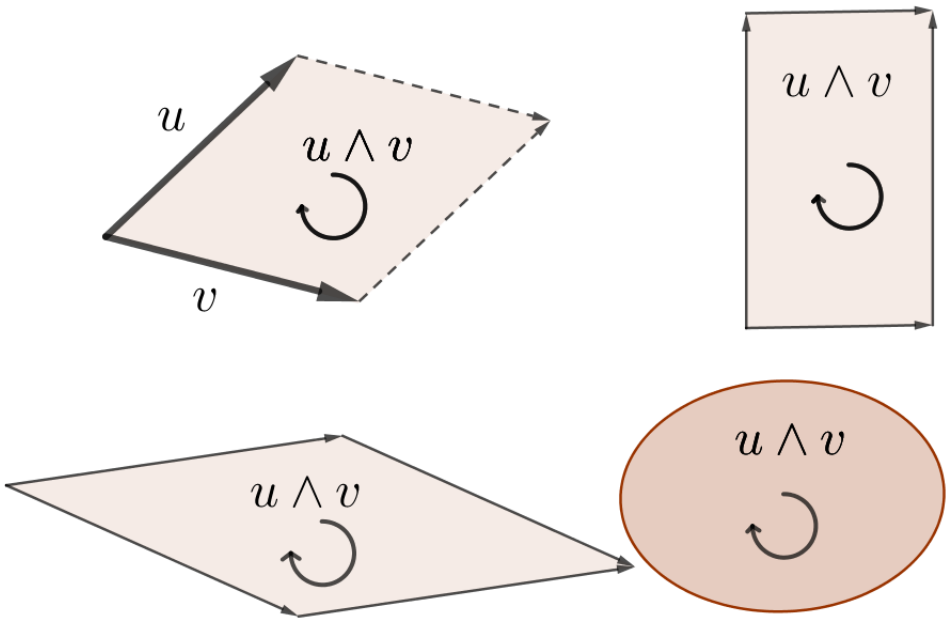


Figure 2: Egalité de bivecteurs

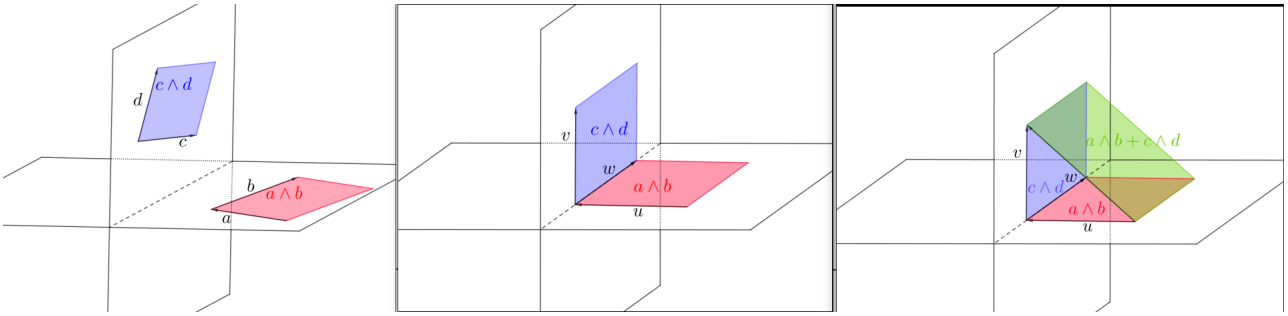


Figure 3: L'addition de deux bivecteurs

Plus généralement, ces objets sont appelés des lames : Une **k-lame** (ou lame de **grade** k) est un multivecteur $\boldsymbol{A}$ de la forme

$$\boldsymbol{A} = \boldsymbol{a_1} \wedge \boldsymbol{a_2} \wedge \cdots \wedge \boldsymbol{a_k}$$

où les vecteurs $\boldsymbol{a_1}, \cdots \boldsymbol{a_k}$ sont linéairement indépendants.

Les 0-lames sont les constantes réelles, les 1-lames les vecteurs, les bivecteurs sont les 2-lames, les trivecteurs les 3-lames...

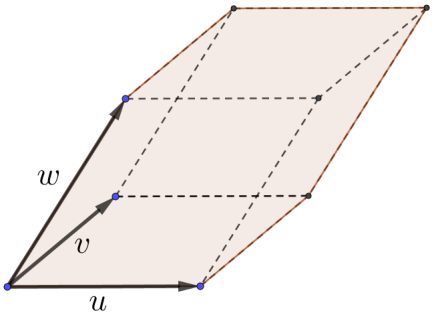


Figure 4: Un trivecteur

Le grade maximal est n : il n'y a qu'une n-lame (à une constante près) : $\boldsymbol{e_1} \wedge \cdots \wedge \boldsymbol{e_n}$. Cette lame est notée I et est appelée **pseudo-scalaire**.

Un multivecteur sous sa forme générale est donc une somme de lames de grades différents. Par exemple, dans $\mathscr{G}^3$, $2 + \boldsymbol{e_1} - 3\boldsymbol{e_3} + 2e_{12} - 4e_{123}$ est un multivecteur.

Les lames et multivecteurs seront notés avec une lettre majuscule et les lames seront notées en gras.

Réversion et inverse

Pour une k-lame $\boldsymbol{A} = \boldsymbol{a_1} \wedge \cdots \wedge \boldsymbol{a_k}$, on appelle réversion de $\boldsymbol{A}$, et on note $\tilde{\boldsymbol{A}}$ la lame $\tilde{\boldsymbol{A}} = \boldsymbol{a_k} \wedge \cdots \wedge \boldsymbol{a_1}$.

Pour une lame non nulle $\tilde{\boldsymbol{A}}$, le calcul de $\tilde{\boldsymbol{A}}$ donne : $\tilde{\boldsymbol{A}} = \boldsymbol{a_k} \wedge \cdots \wedge \boldsymbol{a_1} = (-1)^{\frac{k(k-1)}{2}} \boldsymbol{A}$ car le produit extérieur est anti-commutatif sur les vecteurs de la base et on effectue $1+2+\cdots+(k-1)$ inversions pour passer de $\tilde{\boldsymbol{A}}$ à $\boldsymbol{A}$.

Donc on trouve : $\boldsymbol{A}\tilde{\boldsymbol{A}} = (-1)^{\frac{k(k-1)}{2}} \boldsymbol{A}^2 = (-1)^{\frac{k(k-1)}{2}} ||\boldsymbol{A}||^2$, d'où les lames sont inversibles et

$$\boldsymbol{A}^{-1} = \frac{(-1)^{\frac{k(k-1)}{2}}}{||\boldsymbol{A}||^2} \tilde{\boldsymbol{A}}$$

En particulier, les vecteurs ont donc un inverse et on a : $\boldsymbol{u}^{-1} = \dfrac{\boldsymbol{u}}{||\boldsymbol{u}||^2}$.

Mais les multivecteurs en général n'ont pas d'inverse, l'algèbre n'est donc pas un corps.

Dual d'un multivecteur

Rappelons la définition du pseudo-scalaire : $\boldsymbol{I} = \boldsymbol{e_1} \wedge \cdots \wedge \boldsymbol{e_n}$. Donc $\boldsymbol{I}$ est inversible et $\boldsymbol{I}^{-1} = \pm \boldsymbol{I}$ selon la valeur de n.

Le **dual** d'un multivecteur A est le multivecteur noté A^* et égal à $A^* = A\boldsymbol{I^{-1}}$.

Géométriquement, on peut imaginer le dual d'un multivecteur comme tout l'espace n'étant pas engendré par ce multivecteur. Par exemple, dans $\mathscr{G}^3$, on a $\boldsymbol{I^{-1}} = -\boldsymbol{e_1} \wedge \boldsymbol{e_2} \wedge \boldsymbol{e_3}$, et donc on a : $\boldsymbol{e_1^*} = -\boldsymbol{e_1} \wedge \boldsymbol{e_1} \wedge \boldsymbol{e_2} \wedge \boldsymbol{e_3} = -\boldsymbol{e_2} \wedge \boldsymbol{e_3}$.

Une lame de grade k aura donc un dual de grade $n - k$.

II.2) Algèbre géométrique conforme (CGA)

Présentation

Dans tout ce que nous venons de voir précédemment, nous étions dans l'espace vectoriel euclidien $\mathbb{R}^3$, tous les objets comme les droites et les plans passaient par l'origine. Pour pouvoir traiter tous les objets de l'espace affine, ainsi que d'autres comme les sphères et les cercles, nous allons nous placer dans une algèbre géométrique bien particulière : **l'algèbre géométrique conforme**, abrégée **CGA** (pour Conformal Geometric Algebra).

Pour construire une telle algèbre, notée $\mathscr{G}^{4,1}$, nous allons prendre un espace de dimension 5. Une base orthogonale des vecteurs de $\mathscr{G}^{4,1}$ est constituée des trois premiers vecteurs de base de $\mathbb{R}^3$: $\boldsymbol{e_1}$, $\boldsymbol{e_2}$, et $\boldsymbol{e_3}$, ainsi que de deux vecteurs supplémentaires, notés $\boldsymbol{e_+}$ et $\boldsymbol{e_-}$ avec les conventions suivantes :

$$||\boldsymbol{e_+}|| = 1 \text{ et } ||\boldsymbol{e_-}|| = -1$$

En pratique, on utilisera plutôt les vecteurs $\boldsymbol{e_0}$ et $\boldsymbol{e_\infty}$ définis par :

$$\boldsymbol{e_0} = \frac{1}{2}(\boldsymbol{e_-} - \boldsymbol{e_+}) \text{ et } \boldsymbol{e_\infty} = \boldsymbol{e_+} + \boldsymbol{e_-}$$

de telle sorte que :

- $||\boldsymbol{e_0}|| = ||\boldsymbol{e_\infty}|| = 0$ (3)
- $\boldsymbol{e_0} \wedge \boldsymbol{e_\infty} = -\boldsymbol{e_+} \wedge \boldsymbol{e_-}$ (4)
- $\boldsymbol{e_0} \cdot \boldsymbol{e_\infty} = \boldsymbol{e_\infty} \cdot \boldsymbol{e_0} = -1$ (5)
- Pour $k \in \{1, 2, 3\}$, $\boldsymbol{e_0} \cdot \boldsymbol{e_k} = \boldsymbol{e_\infty} \cdot \boldsymbol{e_k} = 0$ (6)

Image d'un point en CGA

Dans toute la suite, on notera en gras $\boldsymbol{x}$ le vecteur de $\mathbb{R}^3$ et x le même vecteur dans le système de coordonnées CGA.

A $\boldsymbol{x}$ de coordonnées (a, b, c) dans une base donnée de $\mathbb{R}^3$, on associe le vecteur CGA par la formule : $x = a\boldsymbol{e_1} + b\boldsymbol{e_2} + c\boldsymbol{e_3} + \boldsymbol{e_0} + \frac{1}{2}||\boldsymbol{x}||^2\boldsymbol{e_\infty}$, ce qui s'écrit aussi :

$$x = \boldsymbol{x} + \boldsymbol{e_0} + \frac{1}{2}||\boldsymbol{x}||^2 \boldsymbol{e_\infty}$$

De cette manière-là, pour un autre point y, on obtient :

$$\begin{aligned} x \cdot y &= \left(\boldsymbol{x} + \boldsymbol{e_0} + \frac{1}{2}||\boldsymbol{x}||^2 \boldsymbol{e_\infty}\right) \cdot \left(\boldsymbol{y} + \boldsymbol{e_0} + \frac{1}{2}||\boldsymbol{y}||^2 \boldsymbol{e_\infty}\right) \\ &= \boldsymbol{x} \cdot \boldsymbol{y} + \frac{1}{2}||\boldsymbol{y}||^2 \boldsymbol{e_0} \cdot \boldsymbol{e_\infty} + \frac{1}{2}||\boldsymbol{x}||^2 \boldsymbol{e_\infty} \cdot \boldsymbol{e_0} \text{ car les autres termes s'annulent d'après (3) et (6)} \\ &= \boldsymbol{x} \cdot \boldsymbol{y} - \frac{1}{2}\boldsymbol{y}^2 - \frac{1}{2}\boldsymbol{x}^2 \text{ d'après (5)} \\ &= -\frac{1}{2}\left(\boldsymbol{x}^2 - 2\boldsymbol{x} \cdot \boldsymbol{y} + \boldsymbol{y}^2\right) \end{aligned}$$

et finalement :

$$x \cdot y = -\frac{1}{2}||\boldsymbol{x} - \boldsymbol{y}||^2$$

On remarque alors que le produit scalaire entre deux points est proportionnel à la distance au carré entre ces deux points, ce qui nous permet tout de suite de définir les points en CGA comme étant les vecteurs de norme nulle.

Normalisation

A un point CGA $x = \boldsymbol{x} + a\boldsymbol{e_0} + b\boldsymbol{e_\infty}$, on remarque que $-\boldsymbol{e_\infty} \cdot x = a$ d'après les propriétés précédentes. On obtient ainsi un procédé de "normalisation", similaire à ce qui se fait en géométrie projective :

Si $a \neq 0$, le vecteur $\dfrac{x}{-\boldsymbol{e_\infty} \cdot x}$ a pour coefficient 1 devant $\boldsymbol{e_0}$, et donc les coordonnées du point correspondant se lisent devant $\boldsymbol{e_1}$, $\boldsymbol{e_2}$ et $\boldsymbol{e_3}$.

Un point CGA possède donc en prime un "poids", égal à $-\boldsymbol{e_\infty} \cdot x$.

On remarque également que l'image de l'origine est $\boldsymbol{e_0}$, ce vecteur représente donc l'origine du repère en CGA (d'où la notation).

Aussi, le vecteur $\boldsymbol{e_\infty}$ peut être vu comme un point à l'infini : si on divise $\boldsymbol{x}$ par la constante $-\boldsymbol{x} \cdot \boldsymbol{e_0} = \dfrac{1}{2}\boldsymbol{x}^2$, (donc le même point, au poids près) on trouve :

$$\begin{aligned} \frac{\boldsymbol{x}}{-\boldsymbol{x} \cdot \boldsymbol{e_0}} &= \frac{2}{\boldsymbol{x}^2}\left(\boldsymbol{x} + \boldsymbol{e_0} + \frac{1}{2}\boldsymbol{x}^2 \boldsymbol{e_\infty}\right) \\ &= 2\left(\frac{1}{\boldsymbol{x}} + \frac{1}{2}\boldsymbol{e_\infty} + \frac{\boldsymbol{e_0}}{\boldsymbol{x}^2}\right) \xrightarrow[\boldsymbol{x} \to \infty]{} \boldsymbol{e_\infty} \end{aligned}$$

Objets géométriques en CGA - Représentations IPNS et OPNS

La force des algèbres géométriques est qu'avec elles, on peut représenter non seulement les points, mais aussi d'autres lieux géométriques de $\mathbb{R}^3$ comme des sphères, des plans, des droites, et des cercles toujours avec des multivecteurs. En fait, il existe même deux représentations différentes pour chaque objet E : l'IPNS et l'OPNS (Inner et Outer Product Null Space).

- l'IPNS consiste à dire que le multivecteur x correspond à notre espace E si

$$\forall y \text{ vecteur de } \mathscr{G}^{4,1}, y \in E \Leftrightarrow x \cdot y = 0.$$

- l'OPNS consiste à dire que le multivecteur x correspond à notre espace E si

$$\forall y \text{ vecteur de } \mathscr{G}^{4,1}, y \in E \Leftrightarrow x \wedge y = 0.$$

Ces deux représentations sont duales l'une de l'autre. En effet, on a la formule (qui ne sera pas démontrée ici)

$$a \cdot b = (a \wedge b^*)^*$$

Donc $a \cdot b = 0 \Longleftrightarrow a \wedge b^* = 0$.

On peut donc facilement passer d'une représentation à une autre.

On obtient le tableau suivant :

Objets	Caractéristiques	Représentation IPNS	Grade	Représentation OPNS	Grade
Point x	$\boldsymbol{x}$: coordonnées dans $\mathbb{R}^3$ s_1, s_2, s_3, s_4 : sphères contenant x	$x = \boldsymbol{x} + \boldsymbol{e_0} + \frac{1}{2}\|\|\boldsymbol{x}\|\|^2\boldsymbol{e_\infty}$	1	$x^* = s_1 \wedge s_2 \wedge s_3 \wedge s_4$	4
Sphère s	$\boldsymbol{c}$: coordonnées du centre r : rayon a, b, c, d : points sur s	$s = \boldsymbol{c} + \frac{1}{2}(\boldsymbol{c}^2 - r^2)\boldsymbol{e_\infty} + \boldsymbol{e_0}$	1	$s^* = a \wedge b \wedge c \wedge d$	4
Plan P	$\boldsymbol{n}$: vecteur normal à P d : distance de P à l'origine a, b, c : points sur P	$P = \boldsymbol{n} + d\boldsymbol{e_\infty}$	1	$P^* = a \wedge b \wedge c \wedge \boldsymbol{e_\infty}$	4
Droite D	P_1, P_2 : plans contenant D a, b : points sur D	$D = P_1 \wedge P_2$	2	$D^* = a \wedge b \wedge \boldsymbol{e_\infty}$	3
Cercle c	s_1, s_2 : sphères contenant c a, b, c : points sur P	$c = s_1 \wedge s_2$	2	$c^* = a \wedge b \wedge c$	3
Paire de points P_p	s_1, s_2, s_3 : sphères contenant P_p a, b : points de P_p	$P_p = s_1 \wedge s_2 \wedge s_3$	3	$P_p^* = a \wedge b$	2

Table 1: Tableau des représentation IPNS et OPNS des objets géométriques classiques

Une paire de points représente, en un multivecteur, deux points.

Transformations : réflexions, rotations et translations

Soient $\boldsymbol{a}$ et $\boldsymbol{x}$ deux vecteurs. Comme vu auparavant, $\boldsymbol{a}$ est donc inversible et $\boldsymbol{a}^{-1} = \frac{\boldsymbol{a}}{\boldsymbol{a^2}}$. Donc :

$$\boldsymbol{x} = \boldsymbol{xaa}^{-1} = (\boldsymbol{xa})\boldsymbol{a}^{-1} = (\boldsymbol{x} \cdot \boldsymbol{a} + \boldsymbol{x} \wedge \boldsymbol{a})\boldsymbol{a}^{-1} = (\boldsymbol{x} \cdot \boldsymbol{a})\boldsymbol{a}^{-1} + (\boldsymbol{x} \wedge \boldsymbol{a})\boldsymbol{a}^{-1}$$

On va à présent montrer que l'égalité

$$\boldsymbol{x} = (\boldsymbol{x} \cdot \boldsymbol{a})\boldsymbol{a}^{-1} + (\boldsymbol{x} \wedge \boldsymbol{a})\boldsymbol{a}^{-1} \tag{7}$$

représente la décomposition orthogonale de $\boldsymbol{x}$ selon le vecteur $\boldsymbol{a}$:

On remarque déjà que $(\boldsymbol{x} \cdot \boldsymbol{a})\boldsymbol{a}^{-1} = \frac{\boldsymbol{x} \cdot \boldsymbol{a}}{\boldsymbol{a}^2}\boldsymbol{a}$, donc est bien colinéaire à $\boldsymbol{a}$.

Montrons à présent que $(\boldsymbol{x} \wedge \boldsymbol{a})\boldsymbol{a}^{-1}$ est orthogonal à $\boldsymbol{a}$, c'est-à-dire que $(\boldsymbol{x} \wedge \boldsymbol{a})\boldsymbol{a}^{-1} \cdot \boldsymbol{a} = 0$:

$$
\begin{aligned}
(\boldsymbol{x} \wedge \boldsymbol{a})\boldsymbol{a}^{-1} \cdot \boldsymbol{a} &= ((\boldsymbol{x}\boldsymbol{a} - \boldsymbol{x} \cdot \boldsymbol{a})\boldsymbol{a}^{-1}) \cdot \boldsymbol{a} \text{ d'après la formule fondamentale} \\
&= (\boldsymbol{x}\boldsymbol{a}\boldsymbol{a}^{-1}) \cdot \boldsymbol{a} - ((\boldsymbol{x} \cdot \boldsymbol{a})\boldsymbol{a}^{-1}) \cdot \boldsymbol{a} \text{ en développant} \\
&= \boldsymbol{x} \cdot \boldsymbol{a} - (\boldsymbol{x} \cdot \boldsymbol{a})(\boldsymbol{a}^{-1} \cdot \boldsymbol{a}) \text{ car } \boldsymbol{x} \cdot \boldsymbol{a} \text{ est une constante} \\
&= \boldsymbol{x} \cdot \boldsymbol{a} - (\boldsymbol{x} \cdot \boldsymbol{a})(\frac{\boldsymbol{a}}{\boldsymbol{a}^2} \cdot \boldsymbol{a}) \\
&= \boldsymbol{x} \cdot \boldsymbol{a} - \boldsymbol{x} \cdot \boldsymbol{a} \\
&= 0
\end{aligned}
$$

On a donc bien démontré que la décomposition de $\boldsymbol{x}$ donnée en (7) est la décomposition orthogonale selon $\boldsymbol{a}$.

A présent, calculons $\boldsymbol{a}\boldsymbol{x}\boldsymbol{a}^{-1}$:

$$\boldsymbol{a}\boldsymbol{x}\boldsymbol{a}^{-1} = (\boldsymbol{a}\boldsymbol{x})\boldsymbol{a}^{-1} = (\boldsymbol{a} \cdot \boldsymbol{x})\boldsymbol{a}^{-1} + (\boldsymbol{a} \wedge \boldsymbol{x})\boldsymbol{a}^{-1} = (\boldsymbol{x} \cdot \boldsymbol{a})\boldsymbol{a}^{-1} - (\boldsymbol{x} \wedge \boldsymbol{a})\boldsymbol{a}^{-1}$$

On remarque qu'il s'agit de la même décomposition qu'en (7), mais avec le signe qui a changé. Le vecteur $\boldsymbol{a}\boldsymbol{x}\boldsymbol{a}^{-1}$ représente donc la réflexion de $\boldsymbol{x}$ par rapport au vecteur $\boldsymbol{a}$, comme illustré sur le dessin ci-dessous :

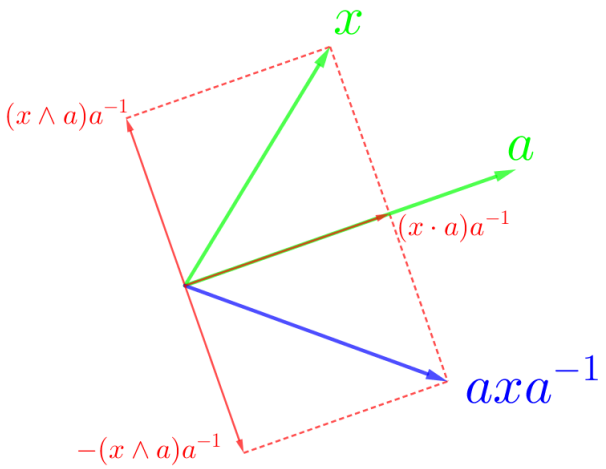


Figure 5: Réflexion de $\boldsymbol{x}$ par rapport au vecteur $\boldsymbol{a}$

Cette opération (appelée le **sandwich**) est assez pratique pour faire les calculs : par exemple pour composer deux réflexions (selon $\boldsymbol{a}$ et $\boldsymbol{b}$), il suffit de "sandwicher" deux fois : $\boldsymbol{b}\boldsymbol{a}\boldsymbol{x}\boldsymbol{a}^{-1}\boldsymbol{b}^{-1}$.

Mais comme ce vecteur est aussi égal à $(\boldsymbol{ba})\boldsymbol{x}(\boldsymbol{ba})^{-1}$, on peut représenter cette transformation par un seul multivecteur, $\boldsymbol{ba}$.

En pratique, on va également choisir les vecteurs $\boldsymbol{a}$ et $\boldsymbol{b}$ unitaires de sorte que $\boldsymbol{a}^{-1} = \boldsymbol{a}$ et $\boldsymbol{b}^{-1} = \boldsymbol{b}$. Ainsi l'inverse de $\boldsymbol{ba}$ sera $\tilde{\boldsymbol{ba}} = \boldsymbol{ab}$

Comme toute rotation et toute translation est la composée de deux réflexions, on peut ainsi définir des multivecteurs représentant ces transformations et les "sandwicher" au multivecteur qu'on souhaite transformer.

En pratique, les formules (non démontrées ici, se référer à [4]) sont :

- Pour une rotation d'angle θ de plan ayant $\boldsymbol{n}$ comme vecteur normal, on note

$$R = cos\left(\frac{\theta}{2}\right) - sin\left(\frac{\theta}{2}\right)\boldsymbol{n}$$

(appelé **rotor**). L'image du multivecteur X par cette rotation est $RX\tilde{R}$.

- Pour une translation de vecteur $\boldsymbol{t}$, on note

$$T = 1 - \frac{1}{2}\boldsymbol{te_\infty}$$

L'image du multivecteur X par cette translation est $TX\tilde{T}$.

III) L'assistant de preuve Coq

III.1) Présentation

Coq est un logiciel assistant de preuve développé par l'INRIA et basé sur le langage Gallina. Un assistant de preuve est un logiciel permettant de faire des démonstrations mathématiques afin de prouver des assertions. Pour cela, il faut d'abord formuler le théorème à démontrer, puis le démontrer à l'aide de "tactiques" qui sont des instructions de modification de la preuve.

Un des plus grands tours de force de ce logiciel a été de démontrer formellement le théorème de Feit-Thompson (tout groupe fini d'ordre impair est résoluble) dont la démonstration initiale faisait 250 pages. Il a fallu cependant une équipe d'une dizaine de chercheurs qui ont travaillé à temps plein pendant 6 ans afin d'aboutir à ce résultat. On voit donc que la mécanisation de la démonstration d'un théorème peut être très longue. De même, vérifier la validité d'un algorithme prend en général bien plus de temps que de trouver l'algorithme lui-même.

III.2) Exemple

Le plus simple est de visualiser un exemple. Prouvons la formule bien connue :

$$\forall n \in \mathbb{N}, \sum_{k=0}^{n} = \frac{n(n+1)}{2}$$

Premièrement, il nous faut définir la somme des n premiers entiers. Coq utilise un langage de programmation fonctionnel, c'est-à-dire que tout ce que nous manipulons est fonction. Les définitions se font donc de façon récursive. Voici la définition :

```
Fixpoint sum_integers (n:nat) :=
    match n with
    0 => 0
    | S p =>  p + 1 + (sum_integers p)
    end.
```

Figure 6: Définition de la somme des n premiers entiers

Le mot-clé "Fixpoint" permet de définir des objets de façon récursive, on précise que n est de type entier naturel (nat).

En Coq, les entiers sont également codés de façon récursive de la façon suivante : tout entier naturel n est soit 0, soit il existe un entier naturel noté p tel que $n = S\ p$, où S est la fonction successeur. Avec cette notation, on doit donc séparer deux cas : soit n vaut 0, et alors la réponse est 0, soit n s'écrit $S\ p$ et dans ce cas-là, la somme des n premiers entiers est celle jusqu'à p et $p+1$.

On vérifie que notre fonction calcule bien la somme en testant la somme des cinq premiers entiers :

```
Eval compute in (sum_integers 5).
```

```
sum_integers 5
     : nat
     = 15
     : nat
```

Le résultat a donc l'air correct. On peut maintenant définir notre théorème :

```
Lemma sum_integers_explicit : forall n: nat, 2*(sum_integers n) = n * (n+1).
```

Figure 7: Théorème à démontrer

On évite, pour des raisons de divisions dans les entiers naturels d'exprimer directement le théorème. En effet, celui-ci implique que la valeur $n(n+1)$ est divisible par 2, ce qui est vrai, mais non prouvé, et donc à faire en Coq. On voit alors dans une fenêtre à droite qu'il reste une proposition à démontrer (le theorème lui-même) :

```
(1/1)
forall n : nat, 2 * sum_integers n = n * (n + 1)
```

Pour le démontrer, nous allons appliquer différentes tactiques, la plupart du temps similaire à ce que nous ferions sur une feuille de papier. Ici, la proposition commence par un $\forall$, donc on aimerait écrire "Soit $n \in \mathbb{N}$", ce qui se fait avec la tactique "intros" :

```
Proof.
    intros.
```

On voit alors que la fenêtre des propriétés à démontrer est modifiée : en-haut de la barre, les hypothèses (n est entier naturel) et en-dessous, ce qu'il reste à démontrer :

```
n :  nat

(1/1)
2 * sum_integers n = n * (n + 1)
```

On veut raisonner par récurrence sur n, ce qui s'écrit :

```
    induction n.
```

La fenêtre de démonstration nous dit cette fois-ci qu'il y a deux propositions à démontrer. On commence par le théorème pour $n = 0$, puis le théorème pour $n+1$ sachant celui-ci vrai pour n.

L'initialisation est très simple à montrer, il ne s'agit que d'un simple calcul, on peut utiliser pour cela la tactique "intuition". La fenêtre de démonstration devient :

```
n :  nat
IHn :  2 * sum_integers n = n * (n + 1)

(1/1)
2 * sum_integers (S n) = S n * (S n + 1)
```

Il ne reste plus qu'une proposition à démontrer, et pour celle-ci, les hypothèses sont que n est entier naturel et que la propriété est vraie au rang n. On utilise la tactique "simpl" pour simplifier la proposition, c'est-à-dire que Coq va remplacer "sum_integers (S n)" par sa définition : "n+1+sum_integers n" :

```
n :  nat
IHn :  2 * sum_integers n = n * (n + 1)

(1/1)
n + 1 + sum_integers n + (n + 1 + sum_integers n + 0) =
S (n + 1 + n * S (n + 1))
```

Ensuite, les trois tactiques suivantes vont chercher des propriétés déjà connues par la bibliothèque de Coq :

```
repeat rewrite Plus.plus_assoc.
rewrite <- plus_n_O.
repeat rewrite plus_n_Sm.
```

Ici, l'associativité de l'addition, le fait que $n+0 = n$, et une formule sur $S(n+m)$. La tactique "repeat" permet d'utiliser plusieurs fois la tactique qui suit. On obtient :

```
n :  nat
IHn :  2 * sum_integers n = n * (n + 1)

(1/1)
n + 1 + sum_integers n + n + 1 + sum_integers n = n + 1 + S (n * (n + 2))
```

On simplifie ce qui est trouvé à l'aide de la tactique "ring_simplify", et on obtient :

```
n :  nat
IHn :  2 * sum_integers n = n * (n + 1)
```

```
(1/1)
2 * n + 2 * sum_integers n + 2 = n * n + 3 * n + 2
```

On peut maintenant appliquer l'hypothèse de récurrence grâce à la tactique

```
rewrite IHn.
```

qui donne

```
n :  nat
IHn :  2 * sum_integers n = n * (n + 1)
```

```
(1/1)
2 * n + n * (n + 1) + 2 = n * n + 3 * n + 2
```

Enfin, il ne reste plus qu'un calcul, qu'on vérifie grâce à la tactique "ring", et à la fin est affiché :

```
No more subgoals.
```

Le théorème est donc entièrement prouvé :

```
Lemma sum_integers_explicit : forall n: nat, 2*(sum_integers n) = n * (n+1).
Proof.
    intros.
    induction n.
    intuition.
    simpl.
    repeat rewrite Plus.plus_assoc.
    rewrite <- plus_n_O.
    repeat rewrite plus_n_Sm.
    ring_simplify.
    rewrite IHn.
    ring.
Qed.
```

Figure 8: Démonstration complète de la formule en Coq

On pourra donc le réutiliser par la suite dans d'autres démonstrations.

IV) Implémentation de mouvements de robots à l'aide des algèbres géométriques en Coq

Dans cette partie, nous allons implémenter en Coq les différentes formules de CGA vues au deuxième paragraphe (les différentes représentations IPNS et OPNS), nous prouverons que cela donne bien ce qui est attendu (la formule d'un cercle donne bien un cercle...). Nous constaterons ensuite sur le logiciel CLUCalc (qui est un outil de visualisation 3D des algèbres géométriques) que ce qui a été implémenté en Coq donne bien ce qui est attendu graphiquement pour trois robots donnés.

IV.1) Algèbres Géométriques Conformes

Tout d'abord, nous réutiliserons la bibliothèque de Laurent Théry [5], qui a déjà implémenté les algèbres géométriques en Coq. Nous spécialisons cette librairie pour n'utiliser que des algèbres géométriques. On donne donc la dimension égale à 5. La classe *vect* contiendra les multivecteurs en général et la classe *point* contiendra les points de $\mathbb{R}^3$.

```
Definition p := Qparams' 5.

SubClass point := Kn (Qparams' 3).
SubClass vect := Vect p.
```

Après plusieurs vérifications de propriétés de base sur le produit extérieur, le produit scalaire et le produit géométrique, on implémente la formule d'un point en CGA, ainsi que les différents objets géométriques du tableau (1).

```
(** to pass from a 3D point to a CGA point *)
Definition Point2CGA (x:point) : vect := x + ((1#2) * (x *s x)) .* e_inf + e_0.

(** Sphere definition from its center and its square radius and from 4 points*)
Definition Sphere_c_r (center:point) (sq_radius: Qc) : IOvect := IPNS (Point2CGA center - ((1#2) * sq_radius) .* e_inf).
Definition Sphere_4p (p1 p2 p3 p4 : point ) : IOvect := OPNS (Point2CGA p1 v Point2CGA p2 v Point2CGA p3 v Point2CGA p4).

(** Plane definition from a normal vector and its distance to 0 and from 3 points*)
Definition Plane_n_d (n: point) (d:Qc) := IPNS(n+d.*e_inf).
Definition Plane_3p (p1 p2 p3 : point ) := OPNS(Point2CGA p1 v Point2CGA p2 v Point2CGA p3 v e_inf).

(** Midplane IPNS definition from 2 points *)
Definition IPNSMidplane (a b : point) := IPNS(Point2CGA a - Point2CGA b).

(** Circle definition from 2 spheres and from 3 points*)
Definition Circle_2S (S1 S2 : vect ) := IPNS(S1 v S2).
Definition Circle_3p (p1 p2 p3 : point ) := OPNS(Point2CGA p1 v Point2CGA p2 v Point2CGA p3).

(** Line definition from 2 points *)
Definition LineIPNS (Plane1 Plane2 : vect) := IPNS(Plane1 v Plane2).
Definition LineOPNS (p1 p2 : point) := OPNS(Point2CGA p1 v Point2CGA p2 v e_inf).

(** PointPair IPNS definition from 3 spheres and from 2 points (TODO : Check planes ??)*)
Definition IPNSPointPair (S1 S2 S3 : vect) := S1 v S2 v S3.
Definition OPNSPointPair (P1 P2 : vect ) := P1 v P2.
```

Figure 9: Implémentation du tableau (1) en Coq

On a choisi de donner un "flag" IPNS ou OPNS à chaque multivecteur. En effet, ce typage aidera à savoir si l'objet géométrique représenté doit être considéré comme une représentation IPNS ou OPNS. Ensuite, on définit la fonction "is_on_object" qui dit si oui ou non un point est sur l'objet considéré :

```
Definition is_on_object (V : IOvect) := match V with
IPNS v => fun x:point => v *s (Point2CGA x) = 0
|OPNS v => fun x:point => v v (Point2CGA x) = 0%v
end.
```

Et ensuite, on démontre plusieurs lemmes, comme par exemple celui-ci :

```
Lemma OPNSLine_compat : forall (a b x : point), is_on_object (LineOPNS a b) x <-> Aligned a b x.
```

On y dit que pour tout point a, b et x, il est équivalent de dire que x est sur l'objet défini par $LineOPNS\ a\ b$ (donc que $x \wedge (LineOPNS\ a\ b) = 0$) et que a, b et x sont alignés (la fonction *Aligned* étant définie auparavant).

Ces démonstrations étant en général peu importantes dans les livres, elles n'y figurent pas toutes. Reconstruire la preuve en Coq permet donc de bien vérifier que ces théorèmes sont valides. L'idée générale de ces démonstrations est de remplacer les vecteurs par leur décomposition selon la base, de développer les calculs avec le produit extérieur, et de simplifier. Une démonstration à la main impliquerait peut-être une méthode différente, plus élégante, mais le fait d'utiliser un

logiciel comme Coq permet de finalement simplifier la logique des démonstrations, ou tout du moins de laisser les calculs compliqués à la machine.

IV.2) Cinématique directe

Dans cette section-là, nous allons calculer la cinématique directe de différents robots. On ne considérera que des robots série, c'est-à-dire constitué d'une seule chaîne d'éléments ayant entre eux des liaisons soit glissière (translations) soit pivotantes (des rotations). La cinématique directe consiste à trouver la position et la direction du bout du robot (appelé **effecteur**) étant donnés tous les paramètres nécessaires pour le calculer. En pratique, on se donne une position de référence, une longueur pour chaque liaison glissière et un angle pour chaque rotation.

On a vu dans le chapitre (II.2)) que pour appliquer une translation de vecteur t, il suffit de calculer $M = 1 - \frac{1}{2} t \boldsymbol{e_\infty}$, dans le cas d'une rotation de plan de vecteur normal $\boldsymbol{n}$ et d'angle θ, de calculer $M = cos\left(\frac{\theta}{2}\right) - sin\left(\frac{\theta}{2}\right)\boldsymbol{n}$ et enfin, en notant x le point auquel on applique la transformation, de calculer $Mx\tilde{M}$.

On obtient donc les lignes suivantes :

```
Definition angle :Type := Qc*Qc.
Definition translator (t:point) : vect := Vadd p C[1%Qc] ((-1/2) .* t) *g e_inf.
Definition rotor (theta : angle) (plan_norm : vect) :=
  let (c,s) := theta in
  Vadd  p C[c] (Vopp (s .* plan_norm)).

(** Transformations using sandwiching: *)
Definition transform (T:vect) (x:vect) := T *g x *g ('R T).
```

Figure 10: Définitions des angles, des translations et des rotations

Les réels étant assez compliqués à manipuler en Coq (car mathématiquement cet ensemble est compliqué, puisque c'est un ensemble de limites de suites de Cauchy), on a choisi ici de travailler dans le corps $\mathbb{Q}$. Un angle θ est donc représenté par le couple $(\cos(\theta), \sin(\theta))$ et les seuls angles considérés seront donc ceux dont ces valeurs seront rationnelles. En pratique, cela n'est pas très limitant car on peut montrer que tout angle réel peut être limite d'une suite d'angles ayant des cosinus et des sinus rationnels. Les robots étant des objets concrets, une approximation de chaque angle sera donc amplement suffisante.

Robot RRP

Le robot RRP est le robot possédant les liaisons schématiques suivantes :

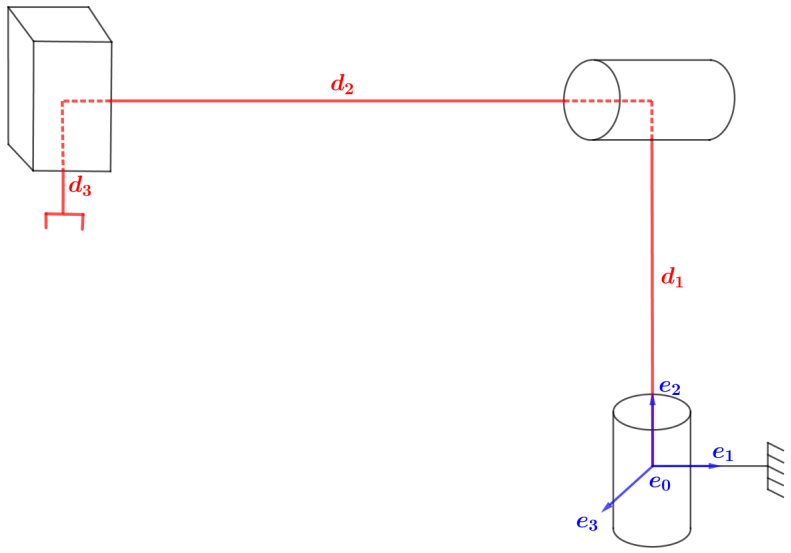


Figure 11: Schéma du robot RRP en configuration initiale

Il y a donc deux rotations, d'axe $\boldsymbol{e_2}$ et d'axe $\boldsymbol{e_1}$, et une translation d'axe $\boldsymbol{e_2}$.

En notant d_i la longueur du bras i, pour $i \in \{1, 2, 3\}$, on obtient en Coq les définitions suivantes :

```
(**    Robot RRP    *)

(* All the parameters *)
Definition d1:Qc := 3/1.
Definition d2:Qc := 5/1.
Definition d3:Qc := 2/1.

(* Creation of the motor M *)
Definition R1 ht1 := rotor ht1 (OPNS2IPNS (LineOPNS origin  (v2p e_2))).
Definition R2 ht2 := rotor ht2 (OPNS2IPNS (LineOPNS origin (v2p (Vopp e_1)))).
Definition T1 := translator (v2p(d1 .* e_2)).
Definition T2 := translator (v2p(Vopp (d2 .* e_1))).
Definition T3 d3 := translator (v2p(Vopp (d3 .* e_2))).
```

Figure 12: Paramètres du robot RRP

$R1$ et $R2$ représentent les deux rotations, $T1$ et $T2$ les deux longueurs des deux premiers bras et $T3$ la longueur de la dernière translation (on remarque que $T3$ a un argument).

Le robot RRP est donc créé ainsi :

En rentrant les trois paramètres variables du robot, on obtient ainsi la position finale de l'effecteur. Testons ceci avec $d_3 = 2$, $\theta_1 \approx 106.26°$ et $\theta_2 \approx 134.76°$. Ces angles ont été choisis car

```
Definition robotRRP ht1 ht2 d3 := (R1 ht1) *g T1 *g (R2 ht2) *g T2 *g (T3 d3).
```

Figure 13: Définition du robot RRP

$\left(\cos\left(\frac{\theta_1}{2}\right), \sin\left(\frac{\theta_1}{2}\right)\right) = \left(\frac{3}{5}, \frac{4}{5}\right)$ et $\left(\cos\left(\frac{\theta_2}{2}\right), \sin\left(\frac{\theta_2}{2}\right)\right) = \left(\frac{5}{13}, \frac{12}{13}\right)$.

Voici le résultat obtenu (la fonction *effector* renvoie l'image de l'origine par les transformations successives, donc la position finale de l'effecteur) :

```
Definition robotRRPtest := robotRRP half_theta_1 half_theta_2 d3.
Compute effector robotRRPtest.
```

Figure 14: Test du robot RRP

```
= ({| this := 467 # 169; canon := Qred_involutive (315692 # 114244) |},
    ({| this := 745 # 169; canon := Qred_involutive (3147625 # 714025) |},
    ({| this := 744 # 169; canon := Qred_involutive (125736 # 28561) |},
    tt)))
  : point
```

Figure 15: Position de l'effecteur du robot RRP pour notre test

Ceci signifie que l'effecteur a une position de coordonnées $\left(\frac{467}{169}, \frac{745}{169}, \frac{744}{169}\right) \approx (2.76, 4.41, 4.40)$.

Vérifions cela. Nous allons utiliser le logiciel CLUCalc, avec lequel nous allons construire notre robot. Puis nous allons placer les angles des rotations et la longueur de la translation considérés et vérifier les résultats obtenus. Voici le robot obtenu :

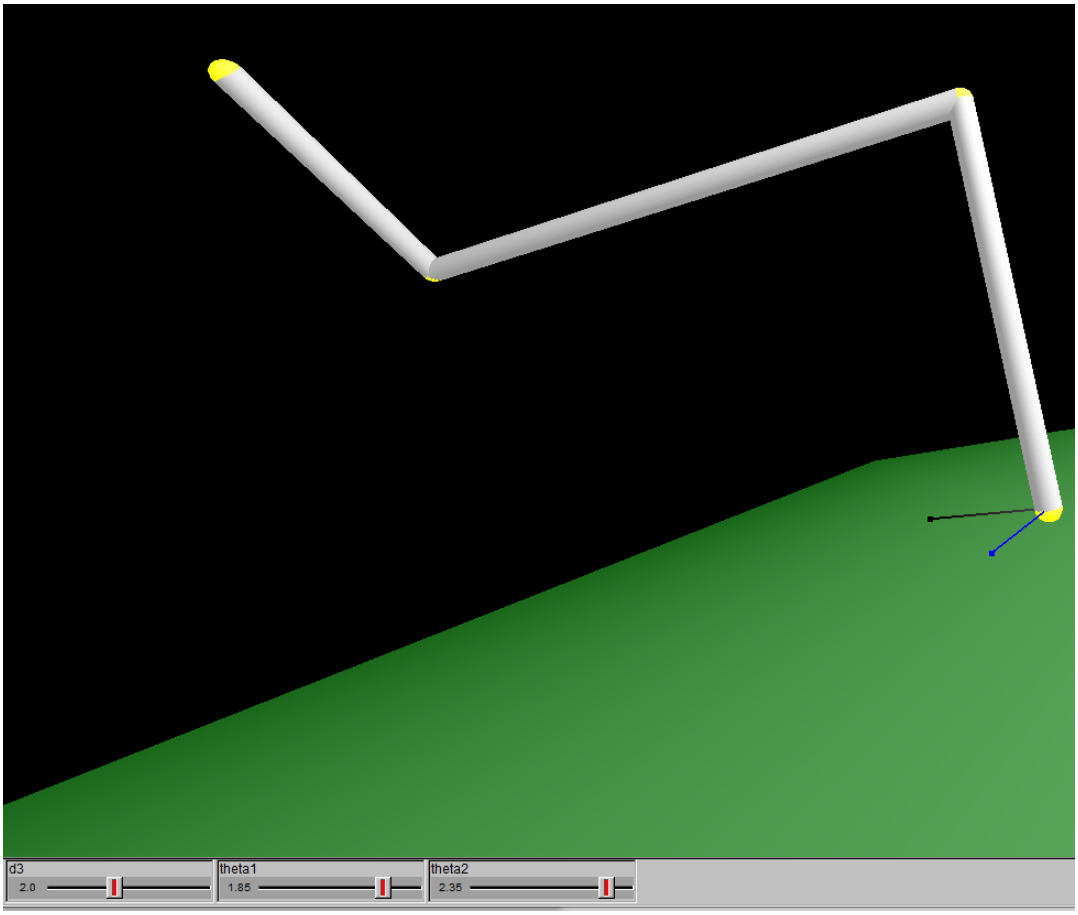


Figure 16: robot RRP construit en CLUCalc

En faisant afficher les coordonnées de la sphère finale (et de toutes les sphères intermédiaires), on peut en lire le centre en regardant les trois premières composantes. On s'aperçoit qu'il s'agit bien, aux arrondis près, des coordonnées trouvées en Coq :

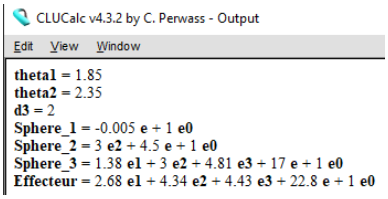


Figure 17: Position de l'effecteur du RRP en CLUCalc

Robot SCARA

Voici un exemple de robot SCARA, ainsi que ses liaisons schématiques :

Figure 18: Exemple de robot SCARA

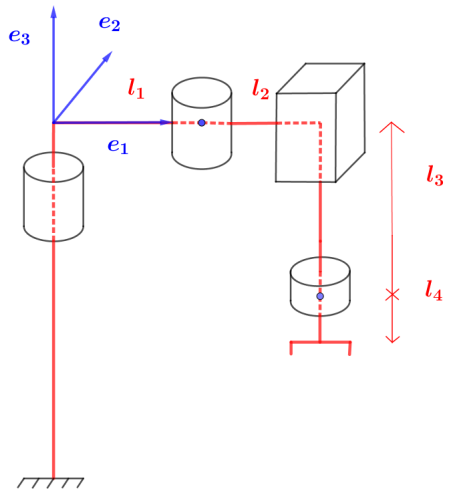


Figure 19: Schéma du robot SCARA en configuration initiale

Il y a donc trois bras, trois rotations de même axe $\boldsymbol{e_3}$ et une translation d'axe $\boldsymbol{e_3}$.

En notant l_i la longueur du bras i, pour $i \in \{1, 2, 3, 4\}$, on obtient en Coq les définitions suivantes :

```
(**    Robot SCARA    *)

(* All the parameters *)
Definition l1:Qc := 4/1.
Definition l2:Qc := 3/1.
Definition l3:Qc := 2/1.
Definition l4:Qc := 1/1.

(* Creation of the motor M *)
Definition RS1 ht1 := rotor ht1 (OPNS2IPNS (LineOPNS origin (v2p e_3))).
Definition RS2 ht2 := rotor ht2 (OPNS2IPNS (LineOPNS origin (v2p e_3))).
Definition RS3 ht3 := rotor ht3 (OPNS2IPNS (LineOPNS origin (v2p e_3))).
Definition TS1 := translator (v2p(l1 .* e_1)).
Definition TS2 := translator (v2p(l2 .* e_1)).
Definition TS3 l3 := translator (v2p(Vopp (l3 .* e_3))).
Definition TS4 := translator (v2p(Vopp (l4 .* e_3))).
```

Figure 20: Paramètres du robot SCARA

$RS1$, $RS2$ et $RS3$ représentent les trois rotations, $TS1$, $TS2$ et $TS4$ les trois longueurs des

bras et $TS3$ la longueur de la translation.

Le robot SCARA est donc créé et testé comme précédemment, avec des angles de $\theta_1 \approx 106.26°$ et $\theta_2 \approx 134.76°$. On rajoutera $\theta_3 \approx 123.86°$ $\left(\text{car } \left(\cos\left(\frac{\theta_3}{2}\right), \sin\left(\frac{\theta_3}{2}\right)\right) = \left(\frac{8}{17}, \frac{15}{17}\right)\right)$. L'angle θ_4 n'influe pas sur la position finale de l'effecteur (mais sur sa direction, ce qui est important si celui-ci n'est pas vu comme un point, mais comme un objet, par exemple une pince), on ne lui donne donc pas de valeur.

```
Definition robotSCARA ht1 ht2 ht3 l3 := (RS1 ht1) *g TS1 *g (RS2 ht2) *g TS2 *g (TS3 l3) *g (RS3 ht3) *g TS4.

Definition robotSCARAtest := robotSCARA half_theta_1 half_theta_2 half_theta_3 l3.
Compute effector robotSCARAtest.
```

Figure 21: Définition et test du robot SCARA

On trouve :

```
= ({|
    this := -10873 # 4225;
    canon := Qred_involutive (-183753700 # 71402500) |},
   ({|
    this := 5136 # 4225;
    canon := Qred_involutive (1812372291600 # 1490902050625) |},
   ({|
    this := -3;
    canon := Qred_involutive (-4472706151875 # 1490902050625) |}, tt)))
  : point
```

Figure 22: Résultat du test du robot SCARA

La position de l'effecteur est donc approximativement de $(-2.57, 1.22, -3)$.

Testons encore sur CLUCalc. Voici le robot avec la position testée :

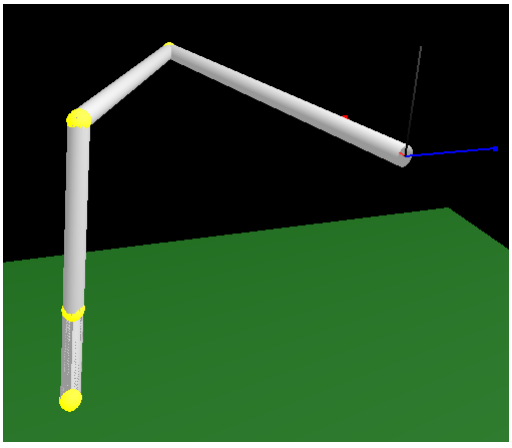


Figure 23: robot SCARA construit en CLUCalc

Et la position finale de l'effecteur :

theta1 = 1.83
theta2 = 2.34
theta3 = 2.16
d3 = 2
effecteur = -2.57 **e1** + 1.3 **e2** + -2.9 **e3** + 8.35 **e** + 1 **e0**

Figure 24: Position de l'effecteur du SCARA en CLUCalc

On voit que la position de l'effecteur $(-2.57, 1.3, -2.9)$ est toujours correcte, aux arrondis près.

Robot 6R

Un robot 6R est un robot ayant 6 axes de rotation. Il est constitué d'une base, ayant trois axes de rotation, et d'une partie plus petite appelée "poignet" avec trois autres axes de rotation, à l'extrémité du robot. En voici un exemple, avec les liaisons schématiques :

Figure 25: Un robot 6R

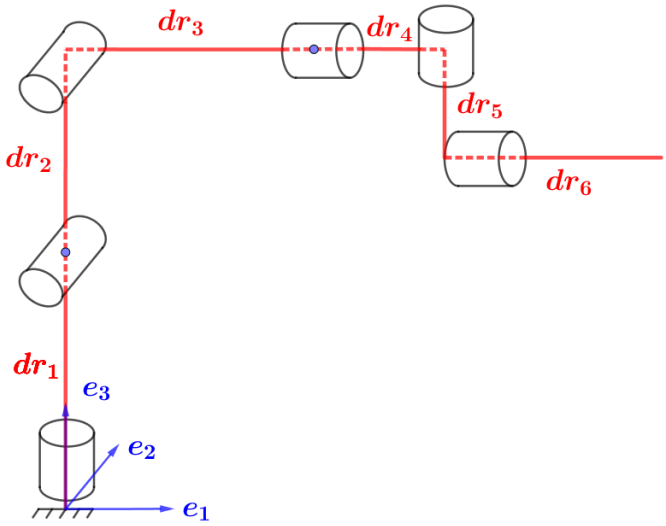


Figure 26: Schéma du robot 6R en configuration initiale

Il y a donc six bras, avec 6 rotations d'axe (dans l'ordre) $\boldsymbol{e_3}$, $\boldsymbol{e_2}$, $\boldsymbol{e_2}$, $\boldsymbol{e_1}$, $\boldsymbol{e_3}$ et $\boldsymbol{e_1}$.

En notant dr_i la longueur du bras i, pour $i \in \{1, 2, 3, 4, 5, 6\}$, on obtient en Coq les définitions suivantes :

```
(**    Robot 6R    *)

(* All the parameters *)
Definition dr1:Qc := 1/1. (* base-waist*)
Definition dr2:Qc := 2/1. (* upper arm *)
Definition dr3:Qc := 3/2. (* lower arm *)
Definition dr4:Qc := 1/2. (* wrist *)
Definition dr5:Qc := 1/2. (* wrist2 *)
Definition dr6:Qc := 1/2. (* effector *)

(* Creation of the motor M *)
Definition RR1 ht1 := rotor ht1 (OPNS2IPNS (LineOPNS origin (v2p (e_3)))).
Definition RR2 ht2 := rotor ht2 (OPNS2IPNS (LineOPNS origin (v2p (Vopp e_2)))).
Definition RR3 ht3 := rotor ht3 (OPNS2IPNS (LineOPNS origin (v2p (Vopp e_2)))).
Definition RR4 ht4 := rotor ht4 (OPNS2IPNS (LineOPNS origin (v2p (Vopp e_1)))).
Definition RR5 ht5 := rotor ht5 (OPNS2IPNS (LineOPNS origin (v2p (e_3)))).
Definition RR6 ht6 := rotor ht6 (OPNS2IPNS (LineOPNS origin (v2p e_1))).
Definition TR1 := translator (v2p(dr1 .* e_3)).
Definition TR2 := translator (v2p(dr2 .* e_3)).
Definition TR3 := translator (v2p(dr3 .* e_1)).
Definition TR4 := translator (v2p(dr4 .* e_1)).
Definition TR5 := translator (v2p(Vopp (dr5 .* e_3))).
Definition TR6 := translator (v2p(dr6 .* e_1)).
```

Figure 27: Paramètres du robot 6R

Le robot 6R est créé et testé comme précédemment, avec des angles de $\theta_1 \approx 106.26°$, $\theta_2 \approx 134.76°$, $\theta_3 \approx 123.86°$, $\theta_4 \approx 147.48°$, $\theta_5 \approx 92.79°$, $\theta_6 \approx 142.15°$ (toujours parce que les demi-angles ont un cosinus et un sinus rationnels). L'angle θ_6 n'influe pas sur la position finale de l'effecteur, mais à nouveau sur sa direction.

```
Definition robot6R ht1 ht2 ht3 ht4 ht5 ht6 := (RR1 ht1) *g TR1 *g (RR2 ht2) *g TR2 *g (RR3 ht3)*g TR3 *g (RR4 ht4)
                                              *g TR4 *g (RR5 ht5) *g TR5 *g (RR6 ht6) *g TR6.
Definition robot6Rtest := robot6R half_theta_1 half_theta_2 half_theta_3 half_theta_4 half_theta_5 half_theta_6.
Compute effector robot6Rtest.
```

Figure 28: Définition et test du robot 6R

On trouve :

```
= ({|
    this := 1446748742509 # 1283602531250;
    canon := Qred_involutive
                (742820139186922750362500 # 659054183292562890625000) |},
   ({|
    this := -898791223944 # 641801265625;
    canon := Qred_involutive
                (-576845345059902004125000 # 411908864557851806640625) |},
   ({|
    this := -4205366693 # 1770486250;
    canon := Qred_involutive
                (-19567820078388501003906 25 # 82381772911570361328125 0) |},
   tt)))
  : point
```

Figure 29: Résultat du test du robot 6R

La position de l'effecteur est donc approximativement de $(1.13, -1.40, -2.38)$.

Testons encore sur CLUCalc. Voici le robot avec la position testée :

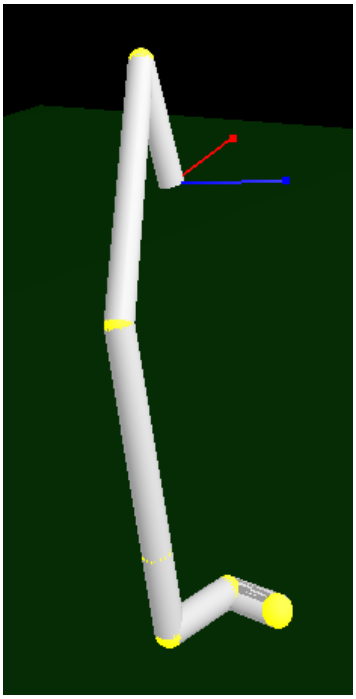


Figure 30: robot 6R construit en CLUCalc

Et la position finale de l'effecteur (et des points de liaison intermédiaires) :

theta1 = 1.83
theta2 = 2.34
theta3 = 2.16
theta4 = 2.58
theta5 = 1.64
theta6 = 2.49
Sphere_1 = 1 **e3** + 0.495 **e** + 1 **e0**
Sphere_2 = 0.368 **e1** + -1.39 **e2** + -0.391 **e3** + 1.1 **e** + 1 **e0**
Sphere_3 = 0.449 **e1** + -1.69 **e2** + -1.86 **e3** + 3.26 **e** + 1 **e0**
Sphere_4 = 0.476 **e1** + -1.8 **e2** + -2.35 **e3** + 4.47 **e** + 1 **e0**
Sphere_5 = 0.628 **e1** + -1.33 **e2** + -2.44 **e3** + 4.04 **e** + 1 **e0**
Effecteur = 1.1 **e1** + -1.46 **e2** + -2.35 **e3** + 4.42 **e** + 1 **e0**

Figure 31: Position de l'effecteur du 6R en CLUCalc

On voit que la position est toujours correcte, aux arrondis près.

Conclusion

On a donc dans cette partie créé en Coq les robots considérés et vérifié que pour ces trois robots, le résultat renvoyé par Coq est bien celui renvoyé par CLUCalc. Tout ceci a uniquement été fait en utilisant les algèbres géométriques conformes. Cela aurait bien sûr aussi pu être fait à partir de la géométrie classique, mais on peut observer ici la simplicité des opérations utilisées.

IV.3) Cinématique inverse

Dans cette partie-là, nous allons nous intéresser au problème inverse. Nous allons nous donner une position finale à atteindre, et trouver quels sont les angles et les longueurs des translations à mettre à nos différents paramètres pour atteindre ce point. Lors de ce problème, plusieurs difficultés peuvent apparaître :

- Le point considéré est inatteignable. Pour considérer un point atteignable, on peut se référer à la documentation du robot qui devrait donner le rayon d'action de ce robot, ou calculer celui-ci nous-mêmes. Dans le cas du robot SCARA par exemple, on peut voir que l'ensemble des points atteignables est l'ensemble des points de coordonnées

$$(x, y, z) \text{ avec } |l_2 - l_1|^2 \leq x^2 + y^2 \leq (l_1 + l_2)^2 \text{ et } z \in [-l_{3,max} - l_4, -l_4]$$

  où $l_{3,max}$ est la longueur maximale du vérin effectuant la translation. En effet, pour la cote, c'est évident, et pour les deux premières coordonnées, on peut se référer au dessin ci-dessous, représentant le robot SCARA vu de dessus :

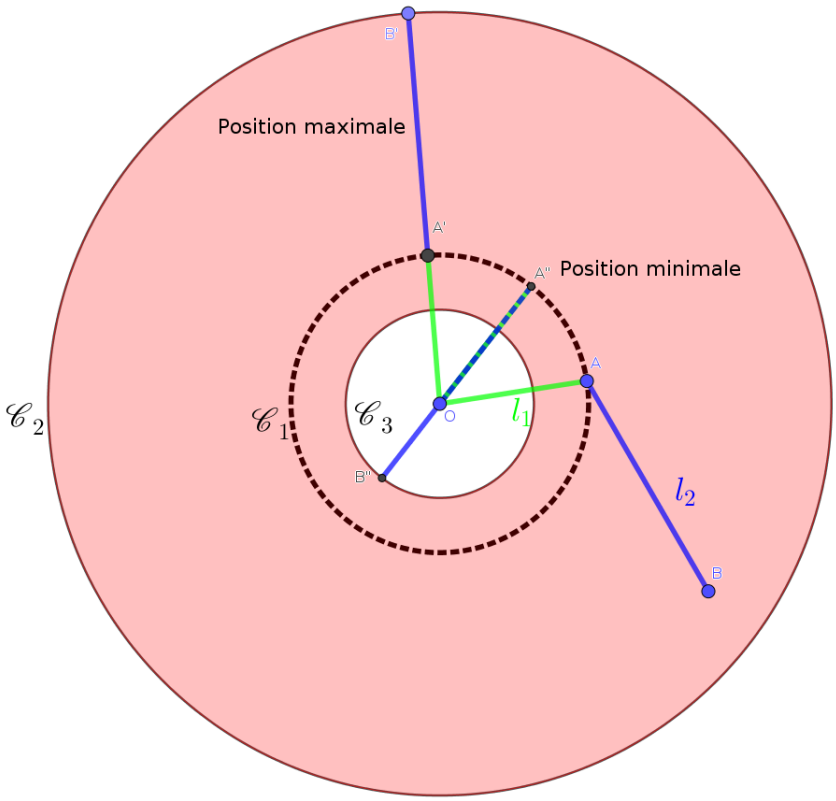


Figure 32: Robot SCARA vu de dessus.

Les bras $[OA]$ et $[AB]$ sont en position quelconque. Le point A peut décrire le cercle $\mathscr{C}_1$. Au maximum (cas représenté sur la figure par les points O, A' et B'), le deuxième bras est parallèle au premier et de même sens. Le point B décrit dans ce cas le cercle $\mathscr{C}_2$.

Au minimum (cas représenté sur la figure par les points O, A'' et B''), le deuxième bras est parallèle au premier et de sens contraire. Le point B décrit dans ce cas le cercle $\mathscr{C}_3$.

L'anneau rouge correspond donc à l'ensemble des points atteignables par le robot SCARA.

- Il peut y avoir plusieurs solutions, même une infinité. Par exemple, pour le robot SCARA, si un point est atteignable et strictement compris dans l'anneau rouge au-dessus, il y aura toujours deux façons d'atteindre le point considéré. Le point A se situe à l'une des deux intersections du cercle de centre O et de rayon l_1, et de celui de centre B et de rayon l_2 :

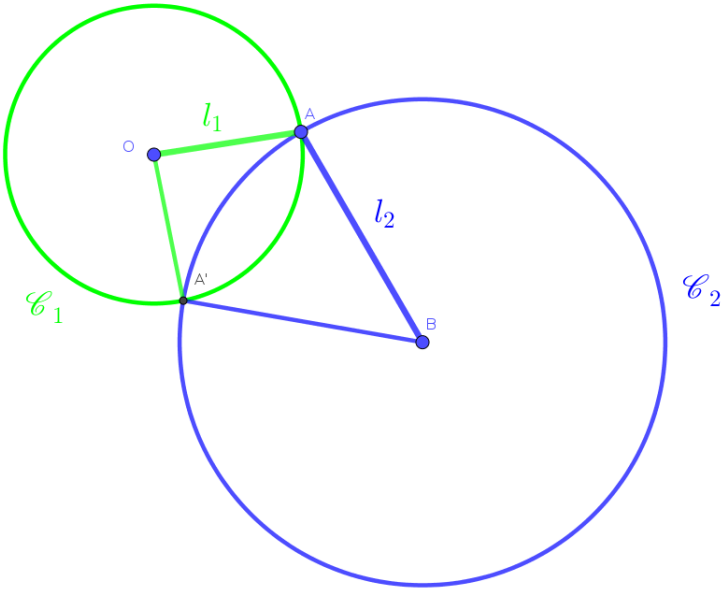


Figure 33: Les deux points A et A' permettent d'atteindre le point B.

Pour le robot 6R, ayant 6 degrés de liberté, il y a une infinité de façons d'atteindre un point atteignable donné. Mais ici, également la direction finale de l'effecteur va nous intéresser, ainsi le problème de la cinématique inverse pour ce robot sera en fait de trouver les angles à donner aux différents bras pour avoir l'effecteur à la position et dans la direction voulue. Ce problème aura aussi plusieurs solutions, à cause de phénomènes similaires à celui du robot SCARA.

Robot SCARA

Nous avons déjà vu que l'ensemble des points atteignables est donné par

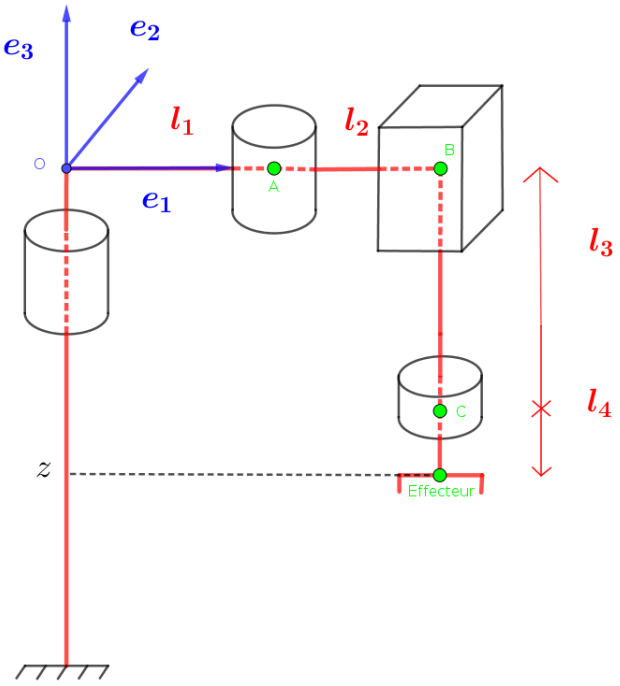
$$E = \left\{(x, y, z) \in \mathbb{R}^3 : |l_2 - l_1|^2 \leq x^2 + y^2 \leq (l_1 + l_2)^2 \text{ et } z \in [-l_{3,max} - l_4, l_4]\right\}$$

Soit donc $(x, y, z) \in E$. On cherche donc deux angles θ_1 et θ_2 dans $]-\pi, \pi]$ et une longueur $l_3 \in [0, l_{3,max}]$ tels que l'effecteur soit en position (x, y, z).

On voit de façon évidente que les angles ne vont pas influer sur la côte, et donc comme $z = -l_3 - l_4$, on peut déjà en déduire que $l_3 = -z - l_4$.

Il reste à trouver les deux angles. Pour cela, on va d'abord trouver les coordonnées du point A :

- On sait que ce point est sur le plan P d'équation $z = 0$.



- Sa distance à O est de l_1, donc il est situé sur la sphère S_1 de centre O de rayon l_1.
- Le point B a pour coordonnées $(x, y, 0)$, et A est situé à une distance de l_2 de B, donc A est sur la sphère S_2 de centre B de rayon l_2.

Au final, pour avoir l'emplacement du point A, il suffit avec les algèbres géométriques de calculer

$$P \wedge S_1 \wedge S_2$$

où $P = \boldsymbol{e_3}$, $S_1 = -\frac{1}{2}l_1^2\boldsymbol{e_\infty} + \boldsymbol{e_0}$ et $S_2 = x\boldsymbol{e_1} + y\boldsymbol{e_2} + \frac{1}{2}(x^2 + y^2 - l_2^2)\boldsymbol{e_\infty} + \boldsymbol{e_0}$ d'après les formules du tableau (1).

Ceci renverra une paire de points (notée Pp) car il y a deux solutions, comme vu au-dessus. Pour en choisir un, il faut calculer $\frac{Pp^* \pm \sqrt{Pp^* \cdot Pp^*}}{\boldsymbol{e_\infty} \cdot Pp^*}$ (voir [6] pour plus de détails), le signe choisi donnant un point ou l'autre.

Voici les calculs effectué en CLUCalc ainsi que les résultats obtenus :

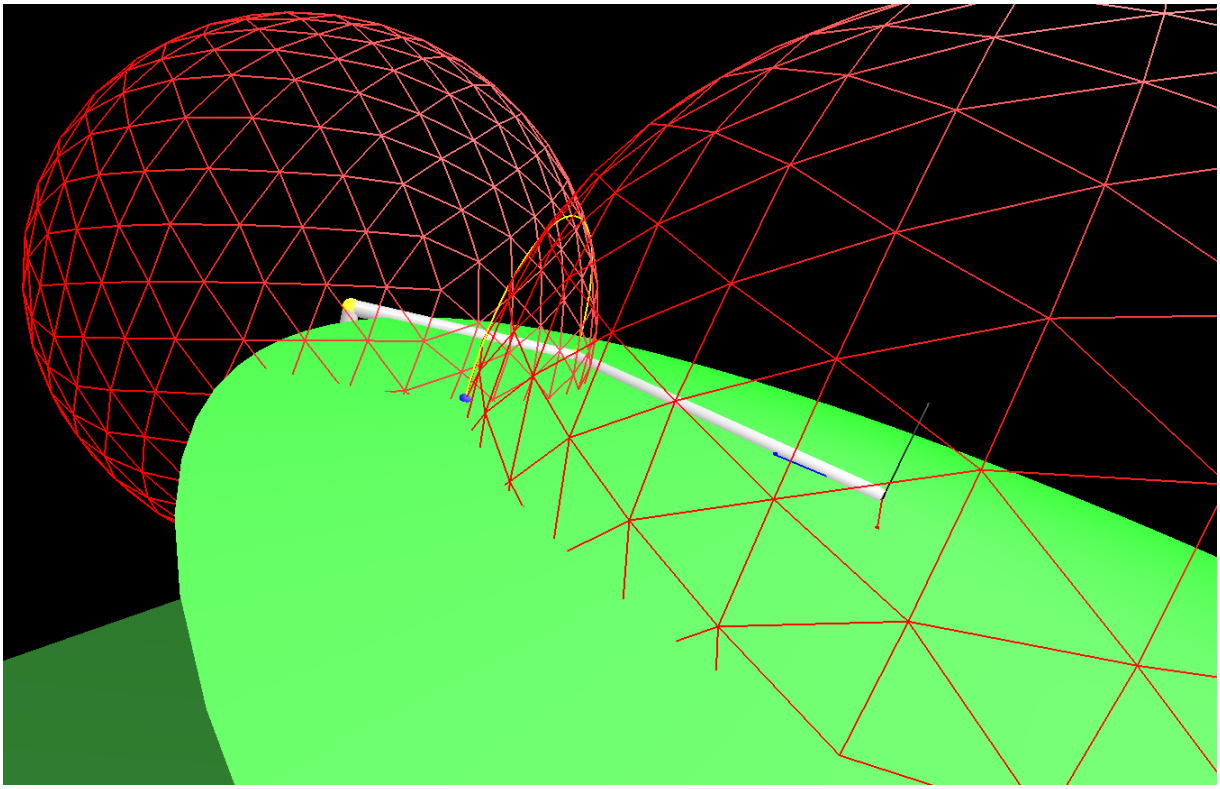


Figure 34: L'intersection des deux sphères (le cercle jaune) et du plan $z = 0$ (en vert clair) donne deux points : un dessiné en bleu, le bras est sur l'autre.

```
S1 = -1/2*d1*d1*e+e0;
S2 = Pt+1/2*(Pt.Pt-d2*d2)*e+e0;
Circle = S1^S2;
Plane = e3;
Pp = Circle^Plane;
?point0 = (-*Pp+sqrt(*Pp . *Pp))/(e. (*Pp))
?point1 = (-*Pp-sqrt(*Pp . *Pp))/(e. (*Pp));
?PointA= -(*Cc0);
```

Figure 35: Les calculs

effecteur = 5.89 **e1** + 1.52 **e2** + -2 **e3** + 20.5 **e** + 1 **e0**
point0 = 3.93 **e1** + -0.755 **e2** + 8 **e** + 1 **e0**
point1 = 3.07 **e1** + 2.56 **e2** + 8 **e** + 1 **e0**
PointA = 3.93 **e1** + -0.755 **e2** + 8 **e** + 1 **e0**

Figure 36: Les résultats

On remarque qu'effectivement un des deux points de la paire de points a les mêmes coordonnées que le point A.

Sur le dessin de la figure (33), la longueur AB mesure $\sqrt{x^2 + y^2}$, et on a $\theta_1 = (\overrightarrow{OA}, \overrightarrow{OB})$ et $\theta_2 = (\overrightarrow{AO}, \overrightarrow{AB})$.

Pour obtenir les angles correspondants, il reste à utiliser les formules d'Al-Kashi :

$$\theta_1 = arccos\left(\frac{l_1^2 + AB^2 - l_2^2}{2l_1AB}\right) \text{ et } \theta_2 = arccos\left(\frac{l_1^2 + l_2^2 - AB^2}{2l_1l_2}\right)$$

V) Conclusion

V.1) Synthèse

Nous avons donc vu dans ce mémoire comment les algèbres géométriques ont permis par des raisonnements relativement simples (une fois le concept compris) de calculer l'emplacement de différents bras de robots à partir des paramètres (cinématique directe) et comment à partir de l'emplacement final déterminer les paramètres de départ (cinématique indirecte).

Le logiciel assistant de preuve Coq a permis de démontrer certains résultats sur ces algèbres géométriques et de faire le lien entre cette géométrie et la géométrie "classique". Ce logiciel a également permis de "créer" virtuellement ces robots pour calculer la cinématique directe ou indirecte à partir des donnés initiales.

Tout ce qui a été expliqué dans ce mémoire a été soit démontré en Coq, soit vérifié empiriquement sur le logiciel CLUCalc.

V.2) Difficultés rencontrées - Perspectives

Comme vu plus haut, la manipulation des réels en Coq est extrêmement compliquée, surtout lorsqu'il s'agit de faire des calculs avec. Dans le cas de la cinématique inverse, pour choisir un point à partir d'une paire de points, il nous faut une formule contenant une racine carrée. Il n'est donc plus suffisant dans ce cas de travailler sur les rationnels, car la partie sous la racine ne sera pas dans le cas général un carré de rationnel.

Nous avons implémenté le même code adapté pour les réels en Coq, mais ceci ne permet pas de faire les calculs avec, juste de retrouver les résultats théoriques. En effet, la bibliothèque des réels en coq n'est pas prévue pour faire des calculs (par exemple les nombres 0 et 1 sont définis, mais pas 2 !).

Nous avons également essayé avec les bibliothèques Interval et C-CoRN de Coq de faire calculer des intervalles qui contiendraient la bonne valeur, mais cela n'a pas été concluant non plus.

Une fois ce problème surmonté, il serait intéressant de démontrer qu'appliquer la cinématique indirecte puis la cinématique directe aboutit bien à la situation de départ.

Une autre perspective serait de trouver un chemin pour l'effecteur d'un point à un autre sans passer par une région qui serait interdite (même toutes les parties du robot, pas seulement l'effecteur, ne pourraient pas passer par cette région).

Bibliographie

[1] Icube. Présentation d'icube. https://icube.unistra.fr/.

[2] Gabriel Dabi-Schwebel. Combien de lignes de code par application ? https://www.1min30.com/developpement-web/combien-de-lignes-de-code-par-application-infographie-7627, 18 Décembre 2013.

[3] Wikipédia. Vol 501 d'ariane 5. https://fr.wikipedia.org/wiki/Vol_501_d

[4] Leo Dorst. Geometric Algebra. 2000.

[5] Laurent Fuchs and Laurent Théry. A formalization of grassmann-cayley algebra in coq and its application to theorem proving in projective geometry. Janvier 2012.

[6] Dietmar Hildenbrand. Foundations of Geometric Algebra Computing. 2013.

[7] Wikipédia. Algèbre géométrique (structure). https://fr.wikipedia.org/wiki/Alg

[8] Eduardo Bayro-Corrochano and Julio Zamora-Esquivel. Differential and inverse kinematics of robot devices using conformal geometric algebra. Août 2006.

[9] A. Kleppe and O. Egeland. Inverse kinematics for industrial robots using conformal geometric algebra. 2016.

[10] Li-Ming Li, Zhi-Ping Shi, and Yong-Dong Li. Formalization of geometric algebra in hol light. Novembre 2018.

[11] Reynald Affeldt and Cyril Cohen. Formal foundations of 3d geometry to model robot manipulators. 2017.