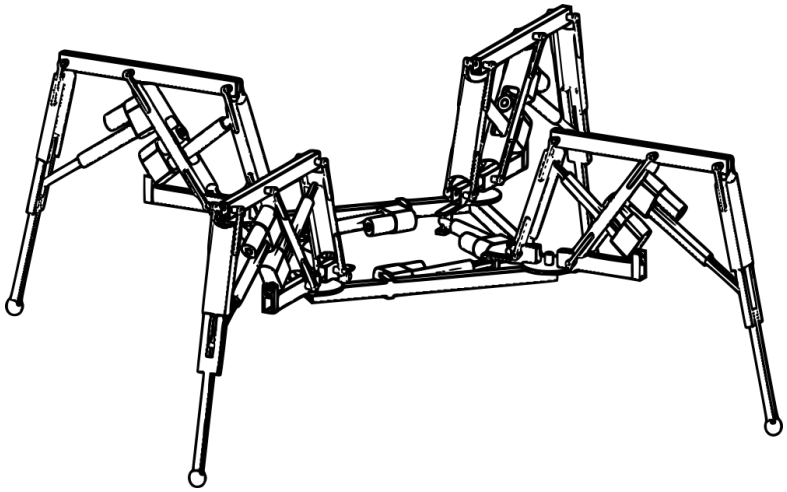
Stage d'application - Rapport Final

Megabot
contrôle, planification et simulation

Stage d'une durée de 13 semaines effectué au sein du service Eirlab de l'ENSEIRB-MATMECA du 31/05/21 au 23/07/21 puis du 23/08/21 au 24/09/21

Etudiant :
Marc Duclusaud

Maître de stage :
Julien Allali



EirLab

Bordeaux INP
ENSEIRB
MATMECA

17 Octobre 2021

Table des matières

Introduction **1**

1 Présentation du Megabot **2**

2 Contrôle **3**
2.1 Existant 3
2.2 Solutions apportées 5
2.2.1 Commande sur la position des pattes 6
2.2.2 Commande sur les positions de la patte flottante et le centre de gravité 6

3 Planification **7**
3.1 Existant 7
3.2 Solutions apportées 8
3.2.1 Elaboration des trajectoires 8
3.2.2 Algorithme de marche 11

4 Simulation **11**
4.1 Existant 12
4.2 Solutions apportées 12
4.2.1 Réalisation de l'URDF 13
4.2.2 Simulation sur PyBullet 14

5 Conclusion **15**

Bilan **15**

Introduction

Eirlab est un atelier de fabrication numérique dont l'objectif est de permettre aussi bien aux étudiants qu'aux particuliers de concevoir, prototyper et réaliser des projets technologiques innovants en mettant à leur disposition un espace de travail ainsi que du matériel difficilement accessible dans le cadre privé. Hébergé au sein des locaux de l'ENSEIRB-MATMECA à Talence, il tient le rôle de Fablab associatif du groupe Bordeaux INP.

Les 400 m^2 constituant Eirlab se découpent en un large espace de travail propice à la conception et à la discussion autour de projets, une salle des machines permettant l'usinage des pièces nécessaires à leur réalisation, un atelier mécanique en permettant l'assemblage et enfin un atelier électronique permettant au besoin de réaliser les circuits nécessaires à leur automatisation.

La présence régulière de personnes qualifiées dans les différents domaines nécessaires à la réalisation d'un projet permet à chacun d'y être formé et guidé sur les différentes problématiques qu'il peut rencontrer : modélisation assistée par ordinateur, algorithmique, électronique, mécanique, etc.

Tout cela en fait un lieu avant tout de partage, où chacun peut apprendre à utiliser une multitude d'outils logiciels puissants et de machines-outils complexes permettant entre autre l'impression 3D, le découpage au laser ou encore l'usinage via machine CNC.

1 Présentation du Megabot

L'objectif de ce stage d'application au sein d'Eirlab a été de travailler sur l'algorithme de marche du Megabot, un robot quadrupède à vérins électriques de grande envergure réalisé par M. Allali.

FIGURE 1 – Megabot

Dans un premier temps, il a fallut revoir le contrôle du robot en améliorant son modèle cinématique. Il s'agit de l'ensemble des algorithmes effectuant la traduction d'un ordre de position en une commande d'élongation de vérins. Après cela, nous avons du reprendre la planification du mouvement en modifiant l'algorithme de marche afin que le robot puisse suivre un déplacement courbe. Enfin, afin de vérifier la validité des algorithmes implémentés indépendemment des contraintes réelles telles que la déformation de la structure sous son poids, nous nous sommes chargés de modéliser le Megabot et de l'intégrer à un simulateur (PyBullet).

2 Contrôle

Le contrôle a pour objectif l'établissement de la cinématique inverse du Megabot, c'est à dire l'implémentation d'un ensemble de méthodes permettant de déterminer les élongations des 12 vérins nécessaires à l'obtention d'une position donnée.

2.1 Existant

Afin d'établir la cinématique inverse du Megabot, sa cinématique directe a tout d'abord été implémentée. Il s'agit de l'algorithme permettant de déterminer pour des élongations de vérins données la position du robot. Dans un premier temps la cinématique directe était réalisée dans le plan de la patte en déterminant de proche en proche la position de chacune de ses articulations en fonction de v_1 et v_2 à l'aide des longueurs des membres et des positions des articulations connues.

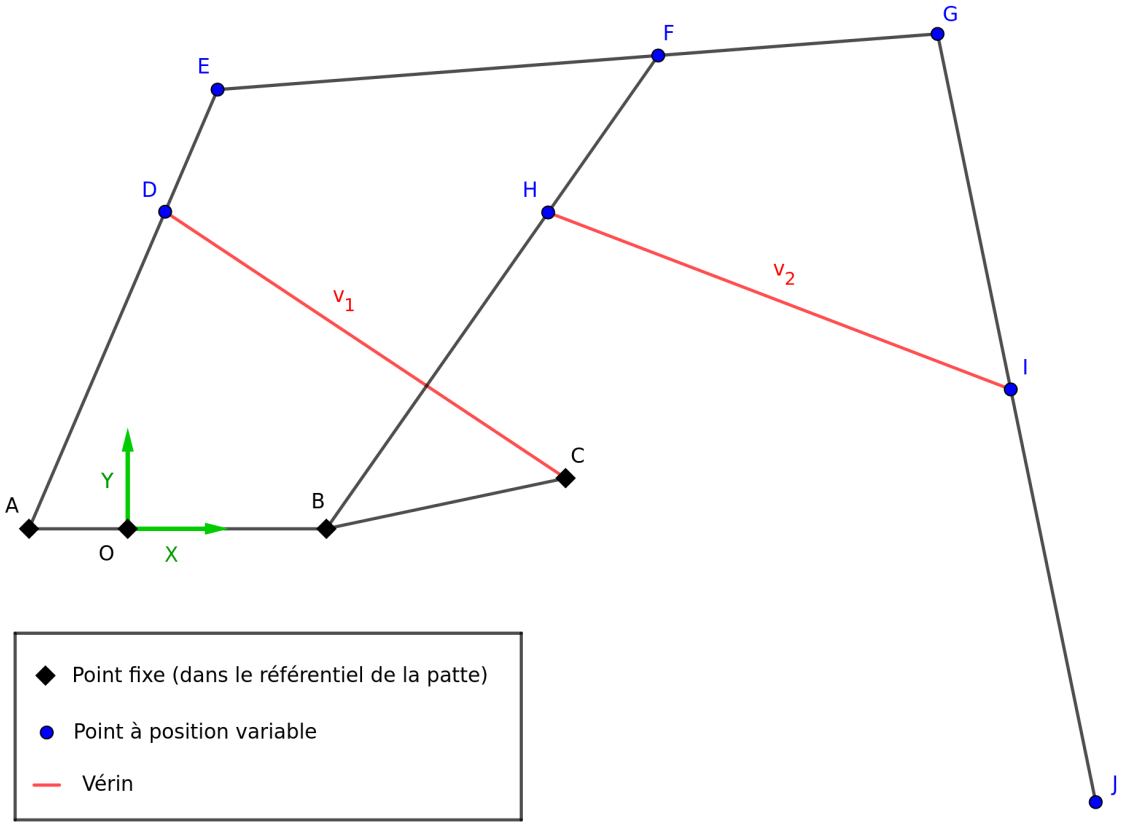


FIGURE 2 – Structure d'une patte du Megabot

Une fois la position du bout de la patte J exprimée dans le référentiel $(\vec{X}, \vec{Y})$ de la patte (cf. Figure 2), il devenait facile connaissant l'angle de la patte au châssis de l'exprimer dans le référentiel $(\vec{x}, \vec{y}, \vec{z})$ du robot en fonction de v_1, v_2 et v_3 (cf. Figure 3).

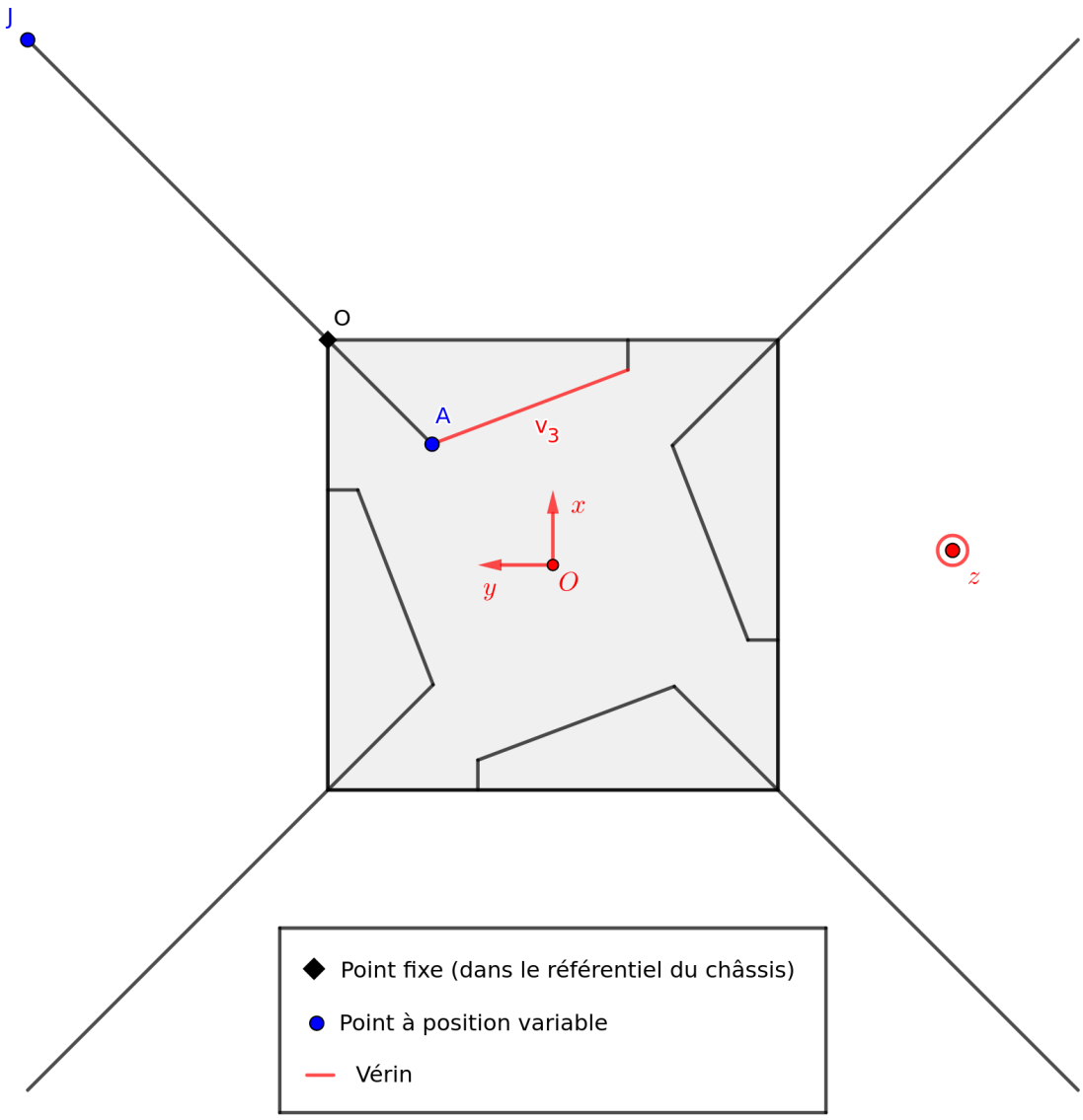


FIGURE 3 – Châssis du Megabot vu de dessus

La cinématique inverse était ensuite réalisée à l'aide d'un solveur qui, prenant en entrée la cinématique directe et la position (x, y, z) souhaitée dans le plan de la patte, renvoyait avec une précision choisie l'élongation des vérins v_1, v_2 et v_3 permettant d'atteindre cette position.

Le principal défaut de cette méthode est qu'elle a recours à une "boîte-noire" qui ne se base pas sur l'expression mathématique de notre cinématique directe mais sur des algorithmes d'optimisation généraux. Cela amène généralement à un temps d'exécution plus élevé qu'en se basant sur une expression connue et offre moins d'adaptabilité à notre modèle. L'ajout de contraintes sur nos entrées et sorties par exemple ne peut pas être intégré à notre contrôle en utilisant cette méthode. Un autre point critiquable de cette méthode est qu'elle aboutit à un contrôle de position dans le référentiel du robot et non dans un repère fixe : nous ne pouvons pas savoir quelles pattes sont au sol ni avoir un suivi de la position du Megabot dans l'espace au cours d'un déplacement.

2.2 Solutions apportées

Afin d'améliorer le contrôle du robot, nous avons décidé d'expliciter les matrices jacobiennes associées au modèle cinématique direct du Megabot. En effet, en utilisant une forme matricielle la question du contrôle se ramène à la résolution d'un problème d'optimisation quadratique de la forme :

$$\underset{\Delta V}{minimize} \quad \frac{1}{2}\Delta V^T (J^T J)\Delta V - (J\Delta X)^T \Delta V$$

$$\begin{aligned} subject\ to \qquad & G\Delta V \leq h \\ & A\Delta V = b \\ & lb \leq \Delta V \leq ub \end{aligned}$$

Ce problème, où ΔV est le vecteur des variations des élongations des vérins à déterminer, J la jacobienne de la cinématique directe et ΔX le vecteur des variations de position, peut être résolu en minimisant l'erreur quadratique sur ΔX à l'aide de solveurs tels que `qpsolvers.minimize` en Python. Cette modélisation du problème permet également l'introduction de limites (lb, ub) et de contraintes d'égalités et d'inégalités (G, h, A, b). Ces contraintes sont un atout non négligeable pour l'élaboration du contrôle car elles permettent tout d'abord de limiter les élongations des vérins à des distances effectivement réalisables, mais surtout de gérer la position du centre de gravité du robot.

En effet, une problématique importante que nous avons cherché à résoudre dès la phase de contrôle a été le maintien du projeté du centre de gravité du robot dans son polygone de sustentation, c'est à dire le polygone convexe formé par les points de contact entre le sol et le robot. Dans le cas où le centre de gravité du Megabot venait à ne plus être au dessus de cette zone, alors il basculerait, ce qui modifierait ses pattes au sol et fausserait toute planification de mouvement.

Deux principaux modes de contrôle ont été implémentés, chacun ayant des avantages et inconvénients. La partie la plus complexe de ce travail de modélisation a été l'élaboration des jacobiennes associées à la cinématique du Megabot, dont le détail des calculs est explicité en annexe.

2.2.1 Commande sur la position des pattes

Le premier mode de contrôle prend en entrée les nouvelles positions absolues des extrémités des 4 pattes du Megabot et renvoie les variations d'élongation des vérins, de la position du centre du robot O et de son orientation Ω. Il contraint les élongations des vérins dans leurs limites physiques, le centre de gravité au dessus du triangle de sustentation formé par les 3 pattes au sol et permet de contraindre les déplacements et l'inclinaison du centre du robot, ce qui a pour objectif d'améliorer le confort du passager.

L'avantage de ce modèle est son exactitude. Pour une entrée réalisable physiquement, le modèle retourne des élongations de vérins conformes à celles attendues, c'est-à-dire que l'application de la cinématique directe à ces élongations aboutit à notre entrée de position. Cependant, il possède un défaut : contraindre le centre de gravité du robot n'est pas suffisant.

Dans l'optique de la planification qui va suivre la phase de contrôle, si le centre de gravité est laissé libre dans son triangle de sustentation, nous risquons lors du changement de patte en déplacement de nous retrouver dans le cas où le centre de gravité est à l'extérieur du triangle de sustentation dès le début du mouvement, ce qui se traduira par un soulèvement de la patte opposée à celle souhaitée et à une impossibilité pour le solveur de trouver une solution à notre problème d'optimisation. C'est pourquoi un second mode de contrôle a été développé.

2.2.2 Commande sur les positions de la patte flottante et le centre de gravité

Cette fois-ci les entrées ont été réduites à uniquement la nouvelle position de la patte en déplacement (les 3 autres restant au sol, elles ne doivent pas bouger) et celle du projeté du centre de gravité, tandis que les sorties sont restées inchangées.

Avec ce nouveau modèle, il est devenu possible d'effectuer une commande de position à la fois sur la patte et le centre de gravité, ce qui assure un contrôle du non basculement du robot. Néanmoins, un de ses inconvénients est que dans le cas de déplacements du la patte et du centre de gravité incompatibles, la sortie peut s'écarter de celle attendue.

3 Planification

La phase de planification consiste en l'élaboration du mouvement indépendemment des réflexions liées au contrôle. Il s'agit de savoir ce que va faire le robot et non comment il va le réaliser et se traduit par l'implémentation d'algorithmes fournissant des positions successives pour les pattes et le centre de gravité du Megabot.

3.1 Existant

La marche du Megabot était initialement uniquement rectiligne. Pour une entrée de direction, les positions extrêmes atteignables le long de cette direction par chacune des pattes étaient calculées et 9 points distincts étaient déterminés : la position initiale de la patte S_i, le point extrême avant F_i, le point extrême arrière B_i et 6 points placés au-dessus de ces derniers (cf. Figure 4).

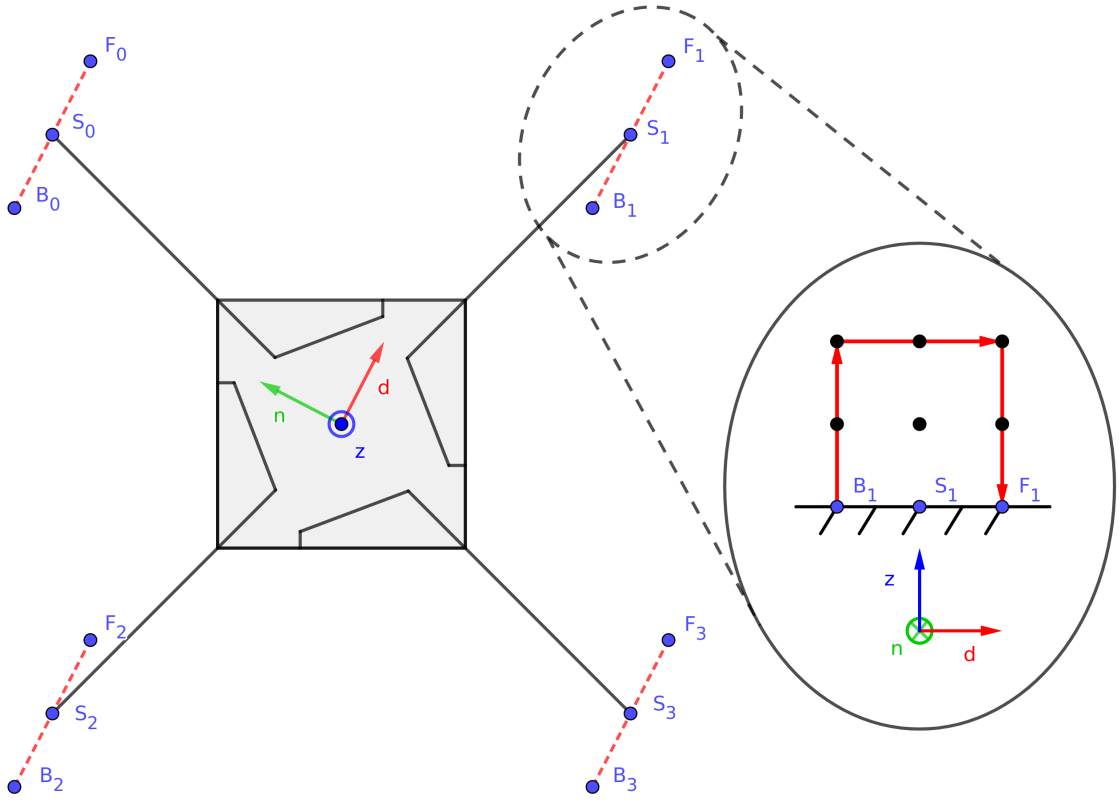


FIGURE 4 – Planification initiale du déplacement

La marche s'initiait par un abaissement du Megabot sur le côté gauche afin d'avancer les pattes arrière et avant droites, puis effectuait l'opération inverse afin d'avancer les pattes arrière et avant gauches. Ainsi, le centre de gravité du robot était supposé maintenu au dessus du triangle de sustentation.

Lors de l'avancée d'une patte du Megabot, différents points de passage parmi les 9 précédemment calculés étaient choisis et parcourus un à un. Ceux retenus dans la version classique de la marche était ceux détaillé sur la Figure 4. Le fait de lever la patte avant d'initier sa translation permettait d'éviter tout frottement sur le sol, car les déformations de la structure du Megabot sous son poids impliquait un écart entre le parcours planifié et le parcours réel de la patte. Lorsqu'une patte avançait de B à F, son antagoniste reculait en ligne droite de F à B sur le sol afin de faire avancer le corps du robot.

Le premier inconvénient de cette implémentation de la marche était le fait qu'elle n'autorisait pas de déplacement courbe. Une autre critique qui pouvait y être apportée était qu'elle ne garantissait pas le parcours de la trajectoire exacte de la part des pattes du fait que les élongations de vérins nécessaires pour effectuer le trajet d'un point de passage à l'autre était importantes. Enfin il est important de préciser que cette planification n'assure aucun contrôle précis du centre de gravité du Megabot.

3.2 Solutions apportées

3.2.1 Elaboration des trajectoires

Nous avions pour objectif de permettre au Megabot un déplacement courbe. De ce fait nous avons tout d'abord cherché à traduire une entrée de direction (2 dimensions) et de rotation (1 dimension) en une trajectoire pour le Megabot. Cette modélisation a pour objectif de pouvoir commander le robot à l'aide d'une manette à 2 joysticks.

La direction permet de définir le vecteur $\vec{d}$ représenté sur la Figure 5 tandis que la valeur de rotation permet de définir la position du centre de rotation R sur l'axe perpendiculaire à la direction : plus la valeur est proche de 0, moins la courbe est importante et plus R est éloigné, à droite pour une rotation positive et à gauche pour une rotation négative. Les trajectoires des 4 pattes sont ensuite déterminées en traçant les cercles de centre R passant par leurs extrémités.

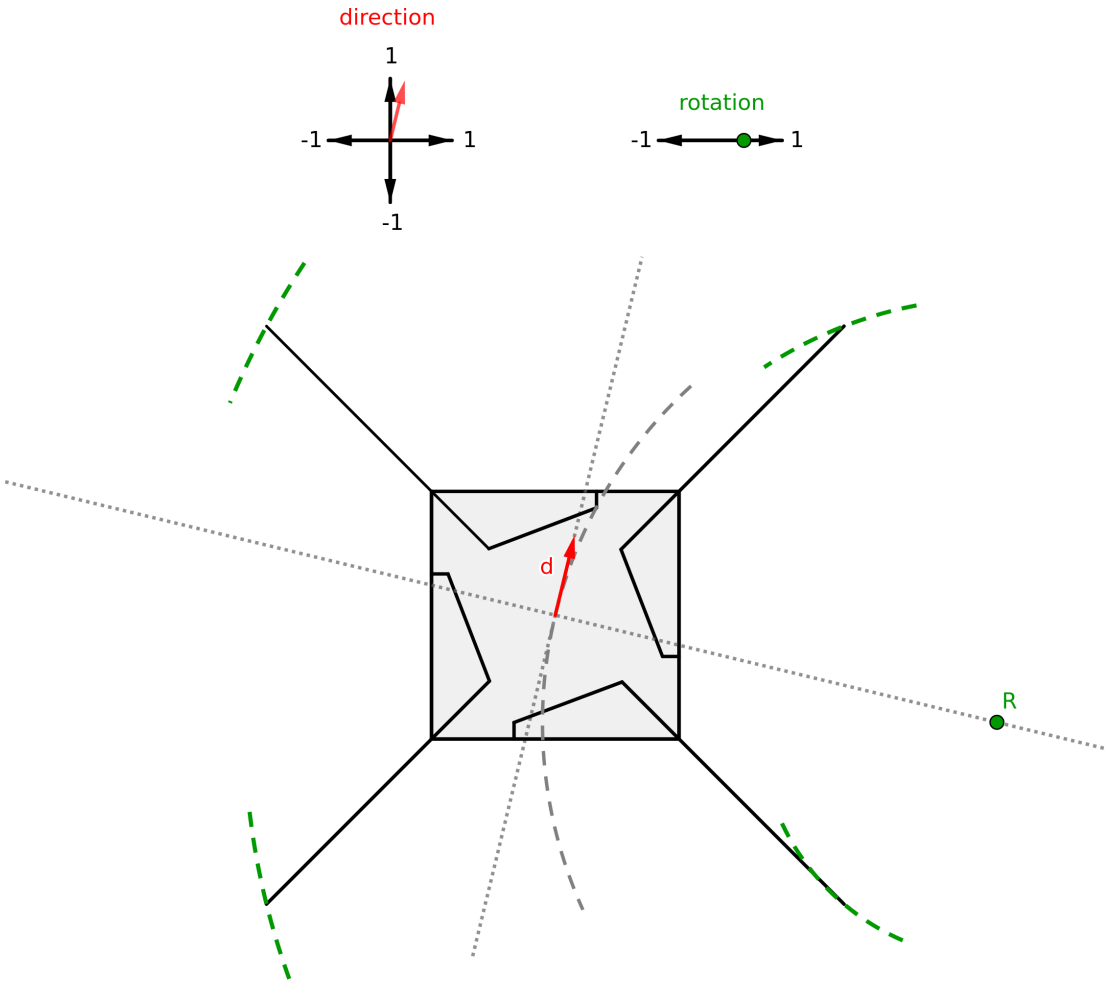


FIGURE 5 – Elaboration de la trajectoire

Dans le cas d'une direction nulle et d'une rotation, R est placé au centre du Megabot. Dans le cas d'une rotation nulle et d'une direction, les trajectoires des pattes sont simplement des droites parallèles à l'axe de direction.

Une fois les trajectoires des 4 pattes déterminées, nous nous sommes questionnés sur la gestion du centre de gravité du robot. La solution retenue permettant de garantir que le centre de gravité soit constamment au dessus du triangle de sustentation a été de lui faire suivre une trajectoire circulaire autour de l'intersection des diagonales du polygone formé par les extrémités des 4 pattes P_0, P_1, P_2 et P_3 durant leur déplacement (cf. Figure 6). Nous noterons ce point C dans la suite des explications. Ce choix nous amène à un découpage de notre marche en 4 phases, précédées d'une phase d'initialisation :

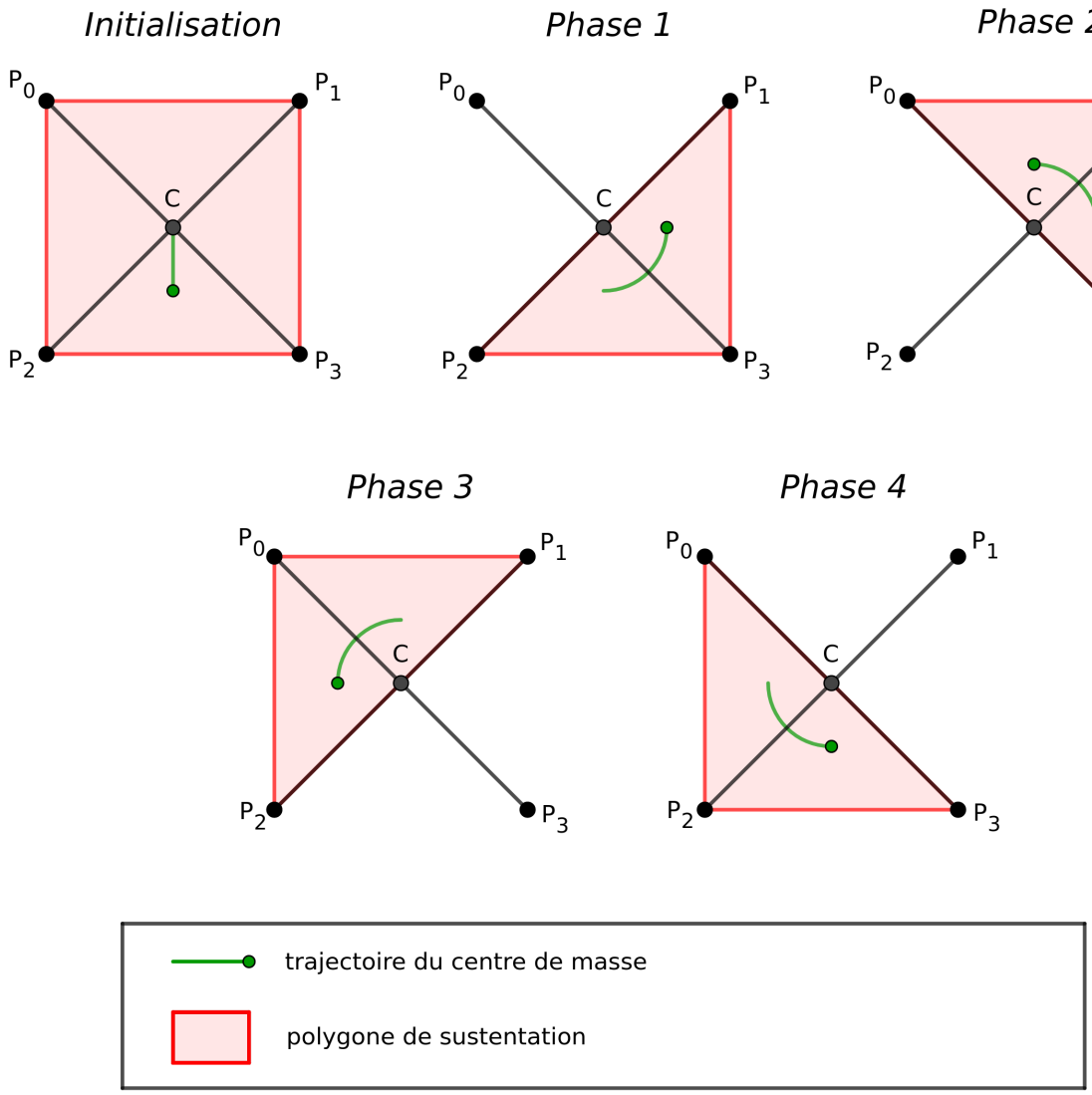


FIGURE 6 – Phases de la marche

3.2.2 Algorithme de marche

Une fois les trajectoires des extrémités des 4 pattes et du centre de gravité établies, nous avons pour chacune des phases détaillées à la Figure 6 procédé à une discrétisation du parcours des pattes et du centre de gravité. Le point C est mis à jour à chaque étape de cette discrétisation, ce qui assure un maintien du centre de gravité au dessus du polygone de sustentation.

Durant la phase d'initialisation, les 4 pattes sont fixes au sol et le centre de gravité est déplacé vers l'arrière du Megabot. Ensuite, lors de chacune des phases suivantes, une patte est levée, déplacée le long de sa trajectoire puis abaissée en même temps que le centre de gravité effectue un quart de tour autour de C. Cette stratégie implique que les pattes sont déplacées une à une dans le sens anti-horaire. L'algorithme de contrôle implémenté dans la partie 2 permet de traduire cette commande en élongations de vérins et d'effectuer la marche.

4 Simulation

Afin de vérifier notre algorithme de marche il est nécessaire de procéder à une simulation du Megabot. En effet, bien que nous ayons effectué des tests unitaires et d'intégration de nos fonctions au cours du développement, tester directement notre marche sur le Megabot réel serait trop précipité, car dans le cas où nous constaterions des écarts entre le résultat et le comportement attendu, rien ne nous garantirait que l'erreur proviendrait bien de notre partie algorithmique et non de notre électronique ou de notre mécanique.

4.1 Existant

Le simulateur initialement utilisé est un simulateur custom réalisé avec OpenGL. Il représente les segments principaux de la structure du Megabot et trace l'évolution des commandes et des positions des 12 vérins au cours de la marche.

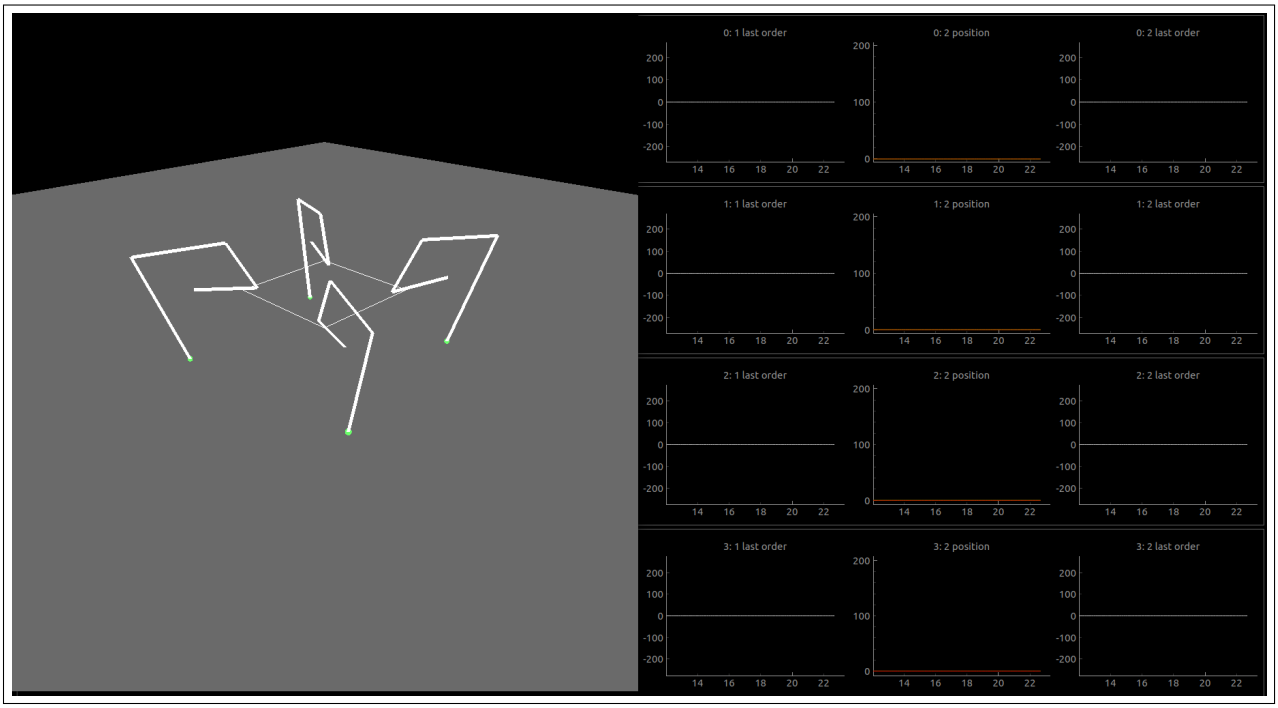


FIGURE 7 – Simulateur initialement utilisé

Le principal défaut de ce simulateur est qu'il ne simule pas la physique et, de ce fait, qu'il est impossible de voir si le robot marche réellement. En effet, le châssis du Megabot y est fixe et seules les pattes bougent ce qui ne permet pas de savoir quelles pattes sont au sol et ce qui écarte la problématique de basculement du Megabot.

4.2 Solutions apportées

Afin de simuler la physique lors des tests de marche du Megabot, nous avons choisis d'utiliser PyBullet. Ce choix implique tout d'abord la réalisation d'un fichier URDF contenant une modélisation du Megabot.

4.2.1 Réalisation de l'URDF

Le format URDF représente les robots sous forme d'un arbre de pièces reliées par des liaisons (pivot, glissière, etc). Cette modélisation très commune est adaptée à un robot articulé, mais empêche la formation de cycle et donc toute fermeture géométrique au sein de la structure du robot. Comme l'ensemble des mouvements du Megabot repose sur des boucles cinématiques au niveau des vérins, il a été nécessaire de trouver un moyen de contourner ce problème. La solution trouvée a été de réaliser un modèle cinématique incomplet et de rajouter des contraintes refermant les boucles a posteriori dans PyBullet.

Une première étape a été la modélisation du Megabot sur OnShape, en incluant le maximum de détail possible et en renseignant le poids et les matrices inertielles de chaque pièce. L'ensemble des liaisons entre les pièces ont été construites dans un assemblage, excepté pour celles amenant des fermetures géométriques, puis le modèle OnShape a été converti en URDF en utilisant l'outil `onshape-to-robot`.

FIGURE 8 – Modélisation du Megabot sur Onshape

4.2.2 Simulation sur PyBullet

Une fois l'URDF généré, les contraintes de fermetures géométriques ont été construites via PyBullet, et la simulation de la marche a pu être réalisée.

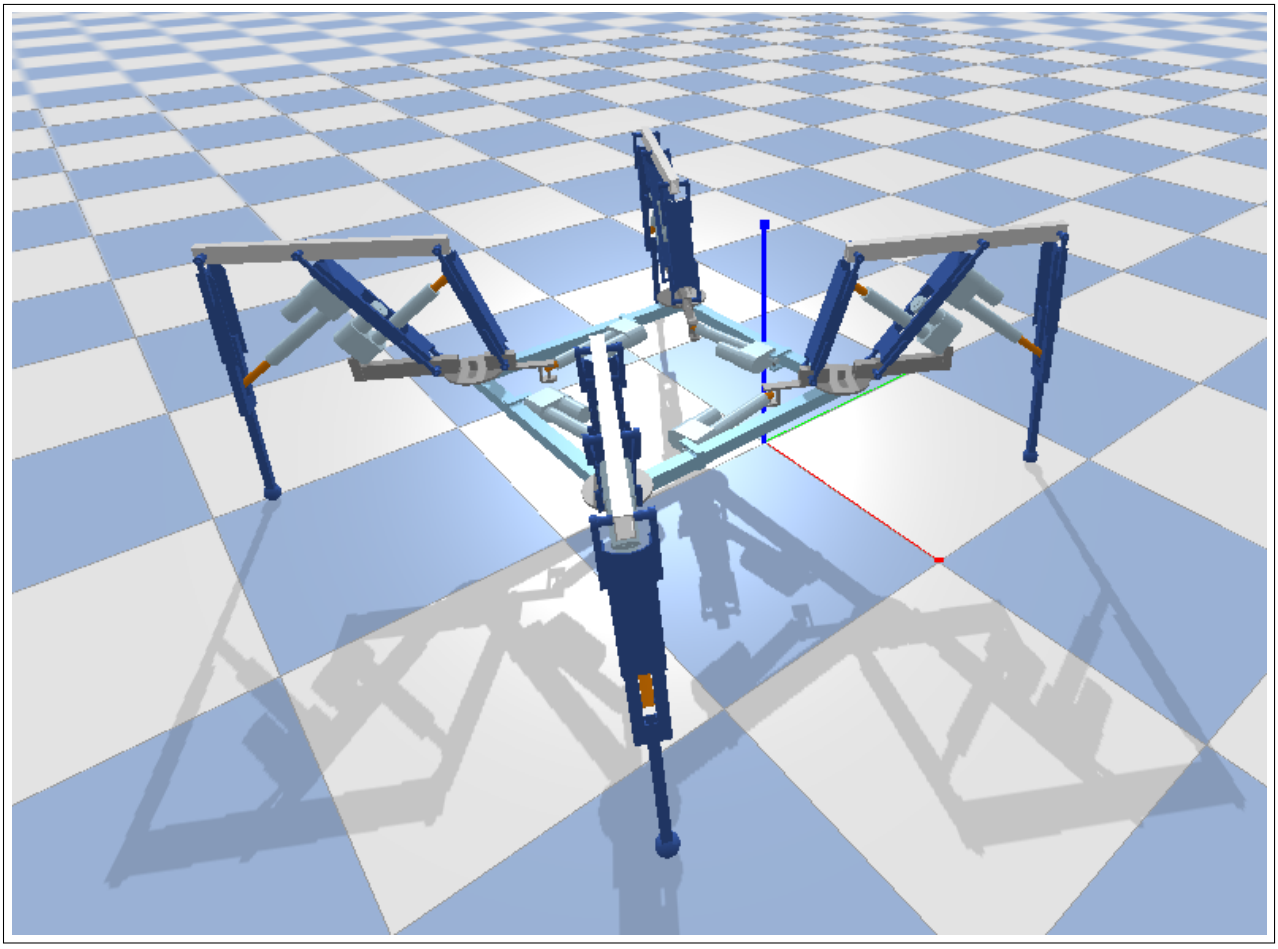


FIGURE 9 – Simulation du Megabot dans PyBullet

La comportement du Megabot dans le simulateur est proche de celui attendu, néanmoins il est possible de constater de légers écarts avec le résultat attendu. Cette différence s'explique de par une répartition de masse différente dans le modèle du robot de la partie algorithmique et dans celui simulé, en particulier au niveau des vérins qui sont considérés comme des segments de longueur variable au poids uniformément réparti dans l'algorithmique et comme deux pièces de poids fixes en translation dans la simulation. Une autre piste pouvant expliquer les écarts est également la prise en compte de la dynamique dans le simulateur, là où la cinématique du Megabot le considère comme statique dans l'algorithmique du fait de sa faible vitesse de déplacement.

5 Conclusion

L'objectif initial du stage s'articulait autour de 2 axes principaux : le contrôle du Megabot et la planification de son déplacement. Le contrôle a été entièrement revu et le modèle cinématique a été calculé avec exactitude, ce qui est vérifié par les tests mis en place systématiquement au cours du stage. Les deux modes de contrôle élaborés sont donc des outils robustes et pérennes qui pourront servir de base à des travaux plus avancés sans nécessiter de révision.

La planification du mouvement associé à une marche courbe correspond également aux attentes exprimées en début de stage. Il est néanmoins important d'avoir regard critique sur les décisions prises telles que le choix de contraindre le centre de gravité à une trajectoire circulaire. Bien que cette solution soit opérationnelle, il en existe d'autres et explorer de manière exhaustive plusieurs possibilités de planification dans le futur permettrait peut-être d'aboutir à une solution plus optimale, ou à minima aux avantages et inconvénients différents.

La simulation quant à elle est un objectif qui est apparu plus tardivement durant le stage. Des éléments indépendant de notre volonté ont empêché la réalisation de tests de marche sur le Megabot, ce qui a motivé la mise en place du simulateur physique. Néanmoins, il s'agit d'une bonne pratique permettant de faciliter la détection d'erreurs ainsi que l'identification de leur provenance, ce qui en fait un outil important pour le Megabot.

Bilan du stage

Ce stage à Eirlab m'a permis d'approfondir mes connaissances en robotique, domaine dans lequel je souhaite poursuivre mes études. Ma compréhension des mathématiques appliquées à la robotique en particulier s'est vue énormément améliorée, ce pour quoi je remercie Adrien Boussicault, maître de conférence à l'université de Bordeaux, qui m'a aiguillé lors des différents problèmes que j'ai pu rencontrer dans l'élaboration du modèle cinématique du Megabot.

Ce stage m'a également permis de gagner en autonomie ainsi qu'en travail d'équipe, une partie du stage ayant été réalisée accompagné de Victor Bregeon et de Guillaume Fornes, alors en stage de 1ère année au sein d'Eirlab, que je remercie pour leur implication sur le projet.

Enfin, ce stage a été pour moi l'occasion de me familiariser davantage avec le Fablab associatif de Bordeaux INP et ses membres ainsi que de travailler sur le Megabot, un de ses gros projets. Pour cela, je remercie Julien Allali, maître de conférence au sein de l'ENSEIRB-MATMECA, qui m'a donné la chance de réaliser ce stage.