Notes de cours GPA546

Ilian Bonev, ing. Yanick Noiseux, ing.

21 septembre 2014

Table des matières

Chapitre 1

Introduction à la robotique industrielle

1.1 Introduction

Selon la norme internationale ISO 8373 [1], un *robot industriel* (figure 1.1) est un « manipulateur multi-application reprogrammable commandé automatiquement, programmable sur trois axes ou plus, qui peut être fixé sur place ou mobile, destiné à être utilisé dans des applications d'automatisation industrielle ». Selon cette même norme, un manipulateur est une « machine dont le mécanisme est généralement composé d'une série de segments, articulés ou coulissants l'un par rapport à l'autre, ayant pour but de saisir et/ou de déplacer des objets (pièces ou outils) généralement suivant plusieurs degrés de liberté ». La partie extrême du manipulateur qui porte l'outil (préhenseur, pince de soudage, etc.) s'appelle l'*effecteur* du robot.

1.2 Degré de liberté et types de robots

Dans l'espace tridimensionnel, un corps rigide libre peut se déplacer selon six *degrés de libertés* (*ddl*) : trois translations et trois rotations. On utilise le terme *pose* pour désigner la localisation du corps par rapport à un référentiel. Une pose est composée d'une *position* et d'une *orientation*.

(a) (b)

FIGURE 1.1 – Robots anthropomorphiques à six et cinq ddl : (a) robot sériel à six articulations KR 1000 1300 TITAN PA de KUKA et (b) robot sériel à cinq articulations KR 470 PA de KUKA.

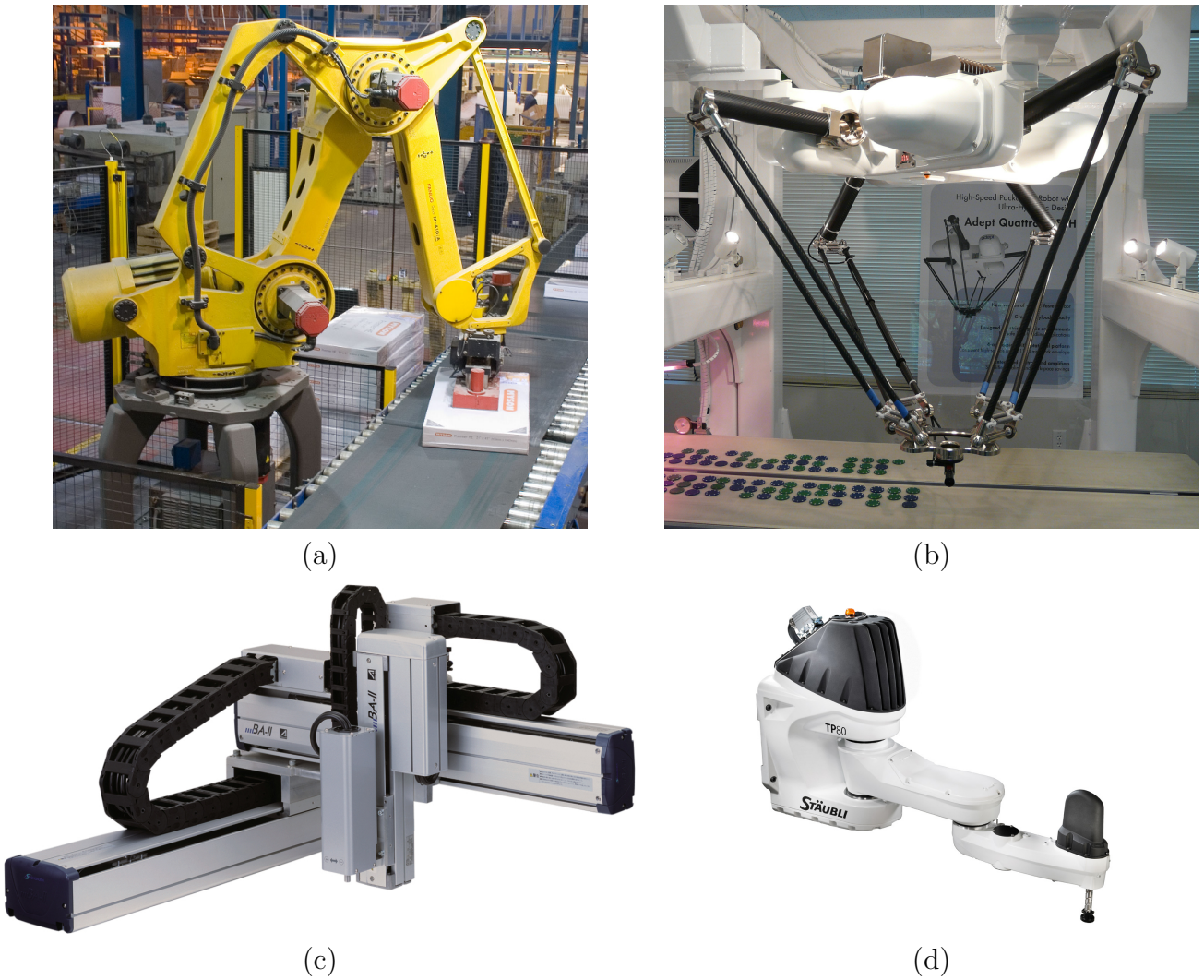
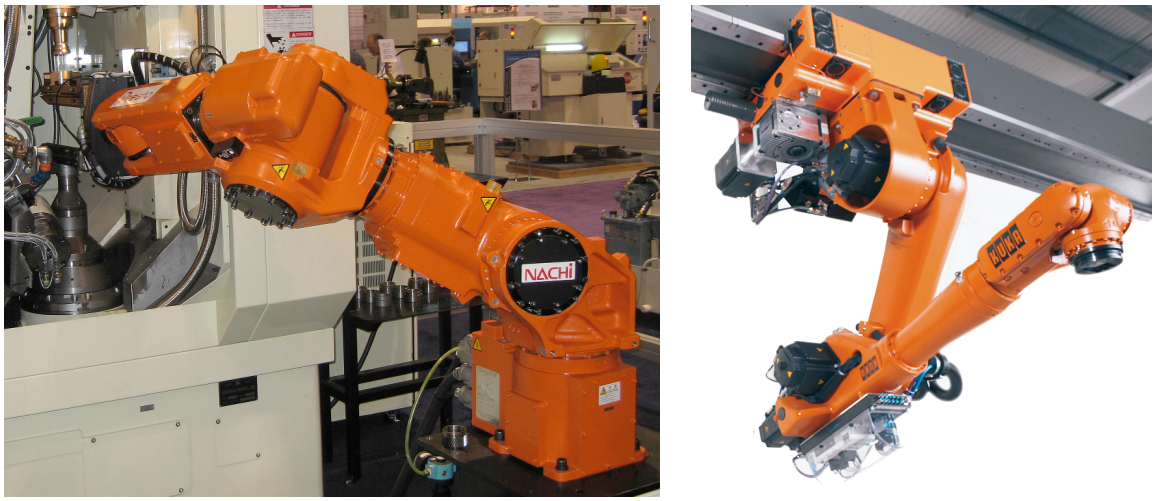


FIGURE 1.2 – Robots à quatre ddl : (a) robot de palettisation (de Fanuc), (b) robot parallèle (le Quattro d'Adept Technology), (c) robot cartésien (de Toshiba) et (d) robot SCARA (le TP80 de Stäubli).

Pour placer un corps rigide n'importe où dans l'espace tridimensionnel, on a besoin d'un robot avec minimum six articulations motorisées, c.à.d. d'un robot à six ddl (figure 1.1a). La grande majorité de robots industriels sont de type sériel. Un *robot sériel* est composé d'une série de segments reliés par des articulations motorisées *rotoïdes* (en rotation) ou *prismatiques* (en translation). Dans certaines applications, on n'a pas besoin de déplacer les objets selon six ddl mais seulement selon cinq ou même quatre ou trois ddl. Par exemple, la figure 1.1b illustre un robot sériel à cinq ddl utilisé pour la palettisation. Le robot de la figure 1.2a est lui aussi utilisé pour la palettisation, mais il a seulement quatre ddl. Deux mécanismes parallélogrammes servent à restreindre deux des trois ddl en rotation de l'effecteur. Ces trois premiers robots sériels sont de type *anthropomorphique*.

Dans un *robot parallèle*, l'effecteur est relié à la base via plusieurs « bras », et la plupart des articulations ne sont pas motorisées. Les robots parallèles peuvent eux aussi avoir six, cinq, quatre, trois ou même deux ddl. Les robots parallèles à six ddl les plus connus sont les hexapodes, comme ceux qui déplacent les cockpits des simulateurs de vol. Les robots parallèles sont généralement plus rigides et plus rapides que les robots sériels. En revanche, ils sont beaucoup plus difficiles à étudier et il en existe de milliers d'architectures différentes. Nous n'allons pas étudier ce type de robots, mais sachez que l'équipe du laboratoire CoRo de l'ÉTS a conçu et développé plusieurs robots parallèles. La figure 1.2b illustre un des robots parallèles les plus rapides. Nous avons un de ces robots Delta à quatre ddl au laboratoire A-0610. Une panoplie d'informations sur les robots parallèles est disponible sur le site Web ParalleMIC.



(a) (b)

FIGURE 1.3 – Robots redondants à sept ddl : (a) le robot MR20 de Nachi et (b) un robot à six articulations de KUKA monté sur un guide linéaire.

De retour aux robots sériels, le robot illustré à la figure 1.2c est un robot *cartésien* alors que le robot de la figure 1.2d est un robot *SCARA*. Ces trois derniers robots sont tous à quatre ddl (trois translations et une rotation autour d'un axe vertical) et servent à déplacer rapidement des petits objets.

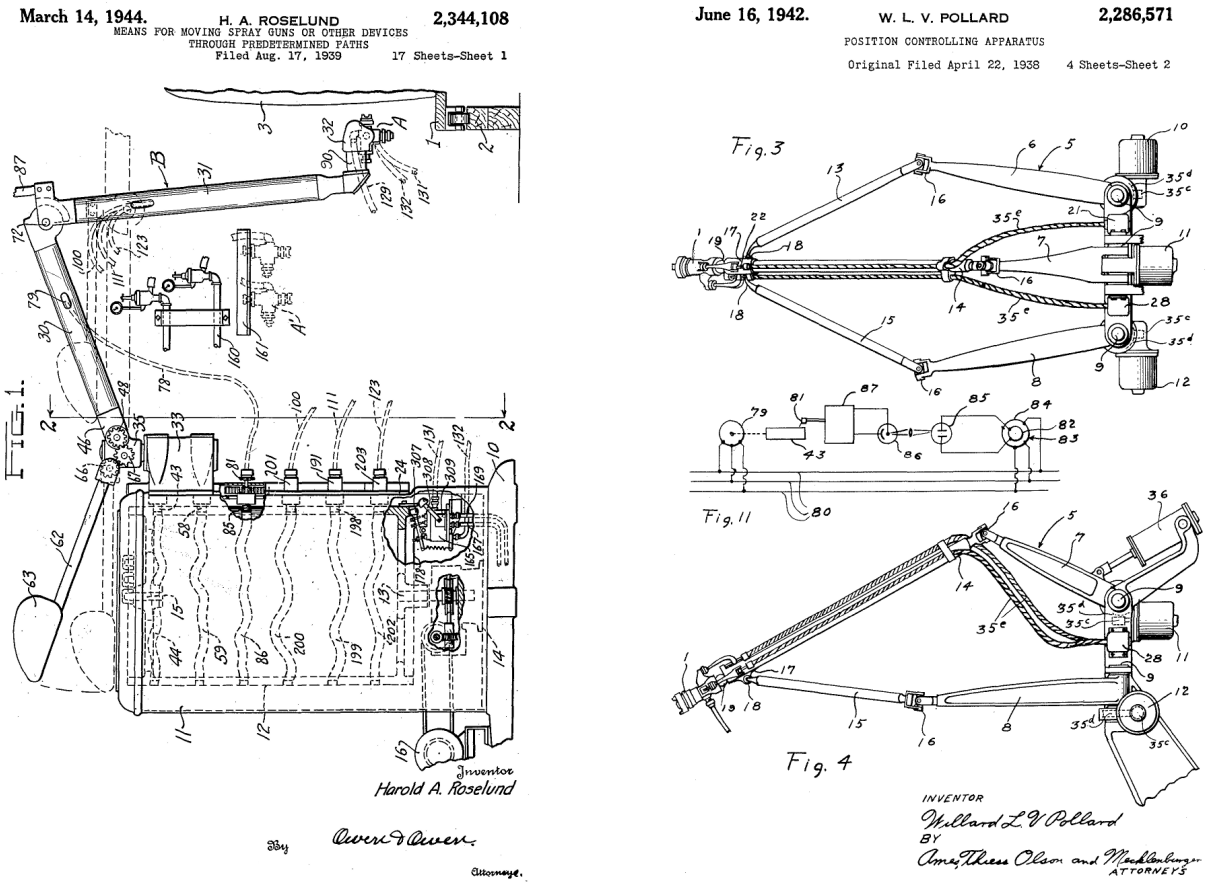
Enfin, il existe aussi des robots à sept ddl (c.à.d. avec sept articulations motorisées) comme le robot illustré à la figure 1.3a, mais ils sont rarement utilisés. L'avantage d'un tel robot redondant est l'existence d'une infinité de possibilités pour attendre une pose désirée, ce qui permet au robot de contourner des obstacles. Beaucoup plus souvent, on monte un robot à six articulations sur un guide linéaire (figure 1.3b)ou sur une table pivotante, ce qui résulte aussi en un système robotique redondant. Cependant, la raison principale n'est pas de contourner des obstacles mais d'augmenter l'*espace de travail* du robot (l'ensemble de poses que l'effecteur du robot peut atteindre).

1.3 Les origines des robots industriels

Un des premiers robots industriels a été conçu par l'américain Willard L. G. Pollard Jr. qui a fait une demande de brevet pour son invention en 1934 [2]. Il s'agit d'un robot parallèle à deux ddl destiné à l'application de peinture sur la carrosserie d'une automobile. Une licence de ce brevet a été vendue à la compagnie DeVilbiss en 1937. En 1941, DeVilbiss a fabriqué le premier robot industriel (un robot de peinture) sous la direction de Harold Roselund. Ce robot n'était pas le robot de Pollard, mais un robot sériel (figure 1.4a) inventé par M. Roselund lui-même [3].

Il est intéressant de mentionner, qu'entre-temps, le père de M. Pollard, M. Willard L. V. Pollard a lui aussi inventé un robot de peinture fort intéressant et a déposé une demande de brevet en 1938 [4]. Il s'agit d'un robot parallèle à cinq ddl (figure 1.4b) qui ressemble beaucoup au fameux robot Delta, inventé par le professeur suisse Raymond Clavel, à la fin des années 1980.

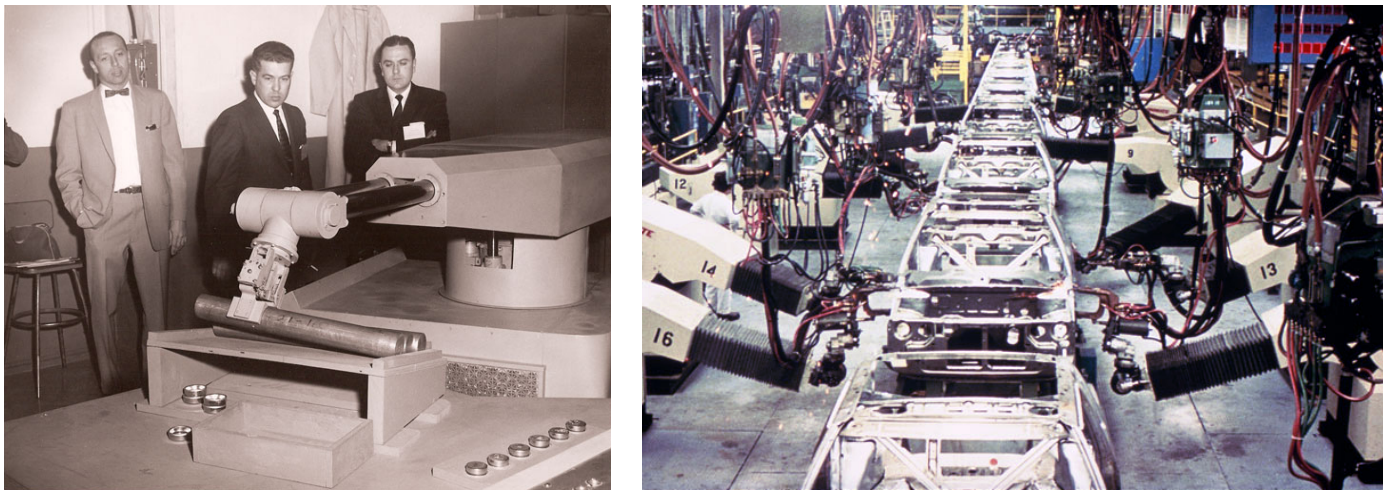
De toute évidence, les inventions des messieurs Pollard et de M. Roselund n'ont pas eu l'effet désiré. C'est beaucoup plus tard, en 1954 que l'américain George Devol fait la demande d'un brevet pour un robot de transfert qui va donner naissance à la robotique industrielle, telle qu'on la connait aujourd'hui [5]. C'était dans un cocktail au Connecticut en 1956 que M. Devol rencontre M. Joseph Engelberger [6], un jeune ingénieur qui travaillait dans l'industrie spatiale, et lui fait part de son invention. Engelberger devient rapidement passionné par l'invention de Devol et démarre la compagnie Unimation. Entre temps, la compagnie AMF (American Machine and Foundry) introduit le robot Versatran, le premier robot cylindrique. En 1962, six robots Versatran ont été installés dans une usine de Ford.



(a) (b)

FIGURE 1.4 – Deux des premiers brevets de robots industriels.

C'est en 1961 qu'Unimation développe leur premier prototype, l'Unimate, et le vend (à forte perte) à General Motors (figure 1.5a). Ce robot a été utilisé pour assister une machine à coulée par injection. En 1964, General Motors commande 66 autres robots Unimate, mais la demande pour ces robots reste toujours très faible. En 1969, Unimation installe 26 robots soudeurs sur une ligne d'assemblage de Chevrolet Vega, chez General Motors (figure 1.5b).



(a) (b)

FIGURE 1.5 – Les premiers robots industriels : (a) le premier robot Unimate juste avant son entrée en fonction en 1961 et (b) la première ligne robotisée qui consiste en 26 robots soudeurs Unimate.

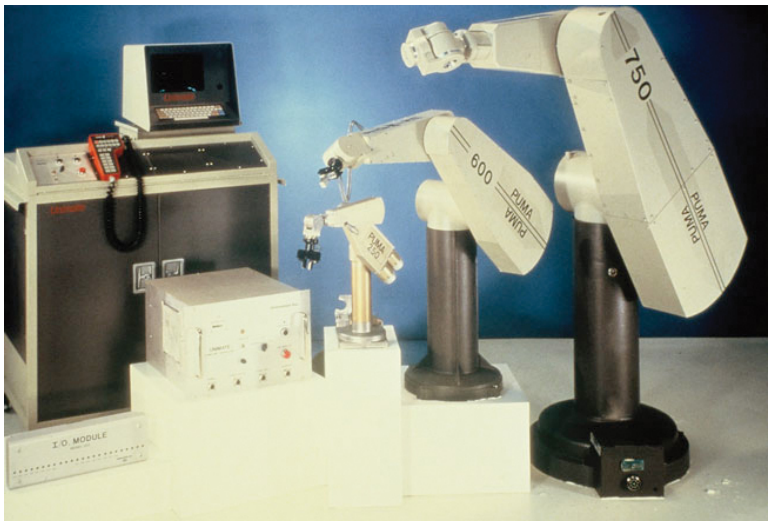


FIGURE 1.6 – Les trois premiers modèles du robot PUMA commercialisés par Unimation.

En 1969, Victor Scheinman développe le Stanford Arm à l'Université Standford. Il s'agit de l'architecture qui est aujourd'hui utilisée par presque tous les robots sériels à six ddl. En 1973, Scheinman démarre sa propre entreprise et commercialise son design sous le nom Vicarm. À ce moment, il y a déjà environ 3000 robots industriels en opération à travers le monde, construit par quelques 71 compagnies. Environ un tiers de ces robots avait été construit par Unimation.

Unimation achète le Vicarm en 1977, l'améliore, et le renomme PUMA pour Programmable Universal Machine for Assembly (figure 1.6). Le premier robot PUMA est installé en 1979, dans une usine de General Motors. Unimation finit par être vendu à l'entreprise suisse Stäubli en 1989.

Aux débuts des années 1970, plusieurs grandes entreprises se lancent dans la fabrication de robots industriels [7]. Asea (aujourd'hui ABB), après avoir installé quelques 25 robots Unimate, développe aon propre robot à six ddl en 1973, l'IRB 6, le premier robot à six ddl avec un parallélogramme (figure 1.7a) . Le parallélogramme sert à placer le troisième moteur plus proche de la base et ainsi diminuer l'inertie du robot. La même année, le fabricant allemand KUKA développe son propre robot à six ddl, le FAMULUS.

(a) (b) (c)

FIGURE 1.7 – Les premiers robots de la nouvelle ère de la robotique industrielle : (a) le premier robot de Asea (aujourd'hui ABB), fabriqué en 1973 ; (b) le premier robot de Yaskawa, fabriqué en 1977 ; et (c) le premier robot SCARA, fabriquée en 1980.

En 1968, Unimation vend une licence à Kawasaki Heavy Industries, qui développe le premier robot industriel au Japon en 1969, le Kawasaki-Unimate 2000. Tout de suite après la sortie du robot IRB 6, plusieurs « copies » apparaissent au Japon, dont une fabriquée en 1977 par Yaskawa, le Motoman L10 (figure 1.7b). En 1974, Fujitsu-Fanuc, un grand fabricant de machines à commandes numériques, produit son premier robot industriel. En 1980, 19 000 robots industriels ont déjà été fabriqués au Japon par quelques 150 fabricants, dont Kawasaki, Yaskawa, Kitachi, Mitsubishi Heavy Industries, Fanuc et Nachi. Le Japon devient le plus grand fabricant et utilisateur de robots industriels.

En 1978, Hiroshi Makino, professeur à l'université de Yamanashi, invente le robot SCARA (Selective Compliance Assembly Robot Arm) [8]. Sankyo Seiki et Nitto Seiko sont les premiers fabricants à produire des versions commerciales de ce robot, en 1981 (figure 1.7c).

1.4 Les robots industriels d'aujourd'hui

Après une année record, quelque 159 346 robots industriels ont été installés à travers le monde en 2012 [9]. En cinquante ans, plus de 2 470 000 robots industriels ont été installés, dont au moins la moitié sont encore en service (la durée de vie utile d'un robot industriel est environ douze ans).

Le Japon continue d'être le plus gros marché pour la robotique industrielle (avec 28 680 robots installés en 2012), suivi par la Chine (22 987 nouveaux robots) et la Corée du Sud (19 424 nouveaux robots). En Europe, c'est l'Allemagne qui est le pays qui a installé le plus grand nombre de robots industriels en 2012 (17 528), suivi de loin par l'Italie (4 402), la France (2 956), la Grande-Bretagne (2 943) et l'Espagne (2 005). Quant aux Amériques, c'est toujours les États-Unis qui mènent le palmarès en 2012 (avec 22 414 nouvelles installations), suivies par le Mexique (2 106), le Canada (1 749) et le Brésil (1 600). L'Afrique reste le continent le moins robotisé avec seulement 393 nouvelles installations en 2012.

Environ deux tiers des robots industriels installés en 2012 étaient de type anthropomorphique. Cette même année, environ 20 % des nouveaux robots étaient de de type cartésien et 11 % de type SCARA. Les robots parallèles, majoritairement de type Delta, constituaient environ 1 % des ventes.

Le club des fabricants de robots a peu changé en quarante ans (depuis l'entrée en jeu d'ABB, KUKA, Fanuc et Yaskawa). L'entreprise japonaise Fanuc a toujours été le plus important fabricant de robots industriels avec plus de 260 000 robots vendus. Elle est probablement déjà dépassée par l'entreprise japonaise Yaskawa (dont la division robotique se nomme Motoman), le plus gros fabricant de variateurs de vitesse et de servomoteurs, avec plus de 270 000 robots vendus. Les numéros trois et quatre dans ce palmarès sont le groupe ABB (dont la division robotique est basée en Suède) avec plus de 200 000 unités vendues et l'entreprise allemande KUKA avec quelque 150 000 unités vendues. Il existe plusieurs autres fabricants de robots industriels, dont les plus importants sont Kawasaki (au Japon, avec plus de 100 000 robots vendus), Denso (au Japon, avec plus de 60 000 installations), Mitsubishi (au Japon), Nachi (au Japon), Epson (au Japon), Stäubli (en Suisse) et Adept Technology (aux États-Unis).

En quarante ans, la robotique industrielle a évolué de façon relativement lente. Certes, à part quelques exceptions, les ventes de robots progressent d'année en année, mais de façon linéaire et le prix d'un robot industriel reste relativement élevé (environ 30 000 $ pour un petit robot). À part quelques innovations remarquables qui restent très niche, telles que les robots à sept ddl ou les robots parallèles, les robots industriels d'aujourd'hui ressemblent énormément aux robots d'il y a quatre décennies (figure 1.7).

Bien évidemment, les robots d'aujourd'hui sont nettement plus performants. À titre d'exemple, le plus récent robot SCARA de la compagnie Stäubli (figure 1.2d) peut effectuer 200 aller-retour par minute avec une charge de 0,1 kg (25 mm en haut, 300 mm en horizontal, 25 mm vers le bas) 📹. Le robot M-2000iA/1200 de la compagnie Fanuc peut manipuler des charges de 1200 kg 📹. Plusieurs modèles de robots peuvent être contrôlés en force (plutôt que seulement en position). Les avances en vision robotique sont aussi spectaculaires. Enfin, de plus en plus la programmation des robots se fait à l'aide de logiciels

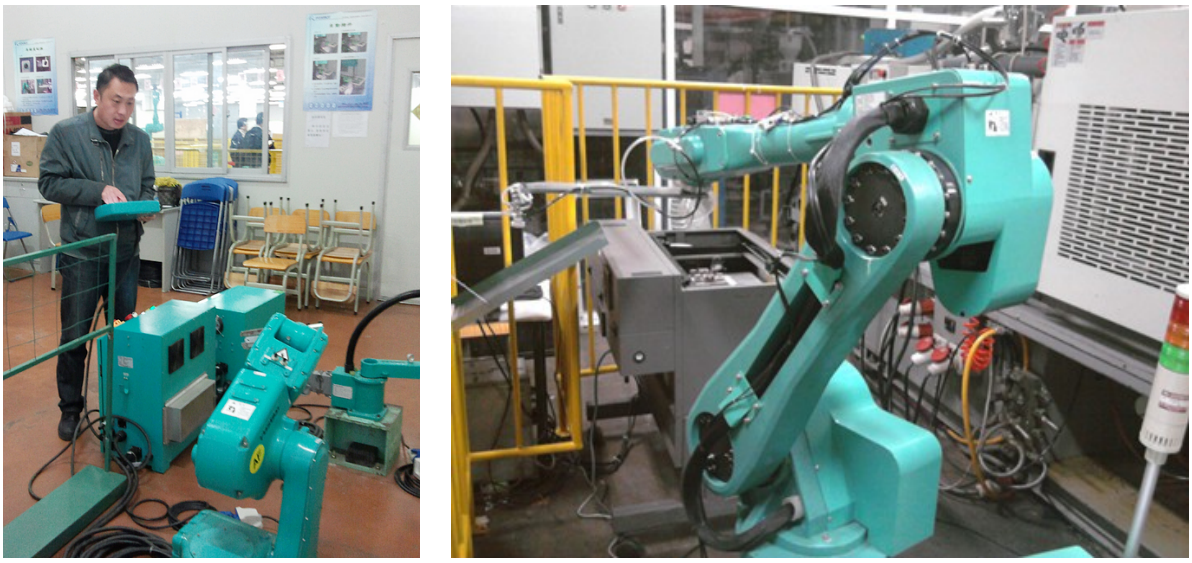


FIGURE 1.8 – Deux modèles de la série de robots industriels Foxbot fabriqués par le géant taïwanais Foxconn.

de simulation très avancés. Par contre, il n'existe toujours pas de langage de programmation commun et la programmation d'un robot industriel reste assez difficile. De plus, même le plus petit robot industriel reste assez dangereux pour l'humain et nécessite l'installation de dispositifs de sécurité dispendieux.

Cependant, depuis environ trois ans, on assiste à un revirement important dans le monde de la robotique industrielle. Premièrement, à cause du prix montant de la main-d'œuvre en Chine, la demande dans ce pays pour les robots industriels a récemment explosé et la Chine est déjà le plus gros marché. Le pays a maintenant ses propres fabricants de robots industriels, tels qu'Estun Robotics, dont la division robotique est dirigée par un ancien étudiant gradué de l'Université Laval, et plusieurs des gros fabricants tels que Fanuc, ABB, KUKA et Yaskawa, y ont des unités de production. En 2011, Foxconn, un géant manufacturier taïwanais spécialisé dans la fabrication de produits électroniques et principalement implanté en Chine continentale où il emploie 1.5 millions de travailleurs, a annoncé son intention de fabriquer un million de robots industriels en trois ans. La date buttoir a été repoussée de quelques années en juin 2012, mais Foxconn a quand même installé 30 000 de ces robots en 2012, dont le prix varie entre 20 000 $ et 25 000 $ (figure 1.8).

Deuxièmement, la compagnie danoise Universal Robots a lancé en 2009 le premier robot industriel convivial, destiné aux PME (figure 1.9a). L'entreprise offre aujourd'hui deux modèles à six ddl, UR5 et UR10. Le poids, la charge utile et le prix du robot UR5 (que nous avons à l'ÉTS) sont respectivement 18 kg, 5 kg, et 28 000 $. Ces robots sont très faciles à programmer et peuvent être mis en service sans protection particulière. En mode collaboration, dès que le robot entre en contact avec un objet et qu'une force d'au moins 150 newtons est exercée, le robot s'arrête automatiquement. Cette compagnie d'environ 100 employés a déjà vendu plus de 2500 robots. Il s'agit sans aucun doute d'une entreprise qui va enfin révolutionner le domaine de la robotique industrielle.

En septembre 2012, Rethink Robotics, une entreprise fondée par un ancien professeur du MIT, Rodney Brooks, a lancé Baxter (que nous avons aussi à l'ÉTS), un autre robot destiné aux PME et qui se vaut encore plus conviviale (figure 1.9b). À seulement 22 000 $, Baxter offre non un, mais deux bras à sept articulations chaque, une charge utile de 2 kg par bras, un système de vision, et un système intégré de sécurité. Ce robot ne nécessite aucune programmation. Les mouvements à exécuter lui sont enseignés à l'aide d'une interface graphique et en manipulant directement les bras.

(a) (b)

FIGURE 1.9 – Deux robots susceptibles de révolutionner la robotique industrielle : (a) UR10 de l'entreprise danoise Universal Robots et (b) Baxter de l'entreprise américaine Rethink Robotics.

1.5 Applications

L'industrie automobile représente toujours le plus grand utilisateur de robots industriels, avec 40 % des nouvelles installations en 2012 [9]. L'industrie de l'électronique a quant à elle acquit 21 % des nouveaux robots en 2012. Étonnamment, l'industrie alimentaire reste un des plus petits marchés pour la robotique industrielle, avec une part de seulement 3 % en 2011 (soit 4 900 robots).

Parmi les applications les plus communes, ABB recense le soudage à l'arc, le soudage par point, la manutention, le chargement/déchargement de machines outils, la mise en peinture, le transfert de pièces (palettisation, emboîtage, etc.), l'assemblage, l'enlèvement de matière (ébavurage, polissage, etc.), et l'application de colle ou de scellant. Les robots industriels sont aussi utilisés dans des endroits inhabituels tels que les parcs d'amusement, les hôpitaux, et même les restaurants. Des dizaines d'exemples d'applications sont disponibles sur la chaîne YouTube du cours GPA546.

(a) (b)

FIGURE 1.10 – La robotique industrielle dans les hôpitaux : (a) un système développé par KUKA et Siemens Healthcare ; (b) un système pour le remplissage de seringues développé par Health Robotics.

Chapitre 2

Le robot industriel IRB 1600 de la compagnie ABB

Le robot industriel que vous allez utiliser dans ce cours est le modèle IRB 1600 du fabricant ABB. Nous en avons quatre au local A-0610. Le système est composé d'un manipulateur sériel à six ddl et d'un contrôleur IRC5.

2.1 Le manipulateur

Le modèle 3D du manipulateur est montré à la figure 2.1. Il s'agit d'un manipulateur à six articulations rotoïdes (celle à la base désignée par le numéro 1) actionnées par des moteurs AC sans brosse. La lecture de l'angle de rotation de chaque articulation est fournie par un résolveur directement monté sur l'arbre de chacun des moteurs. La rotation de chaque articulation est connue de façon absolue sur un tour de moteur, mais un système électronique mémorise le nombre de tours. Ce système est situé dans la base du robot et protégé par une batterie lors de la mise hors tension du système.

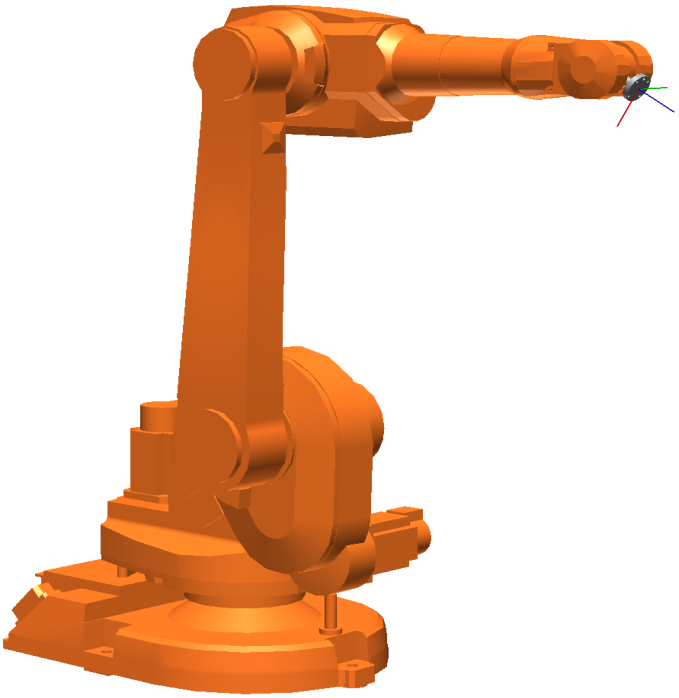
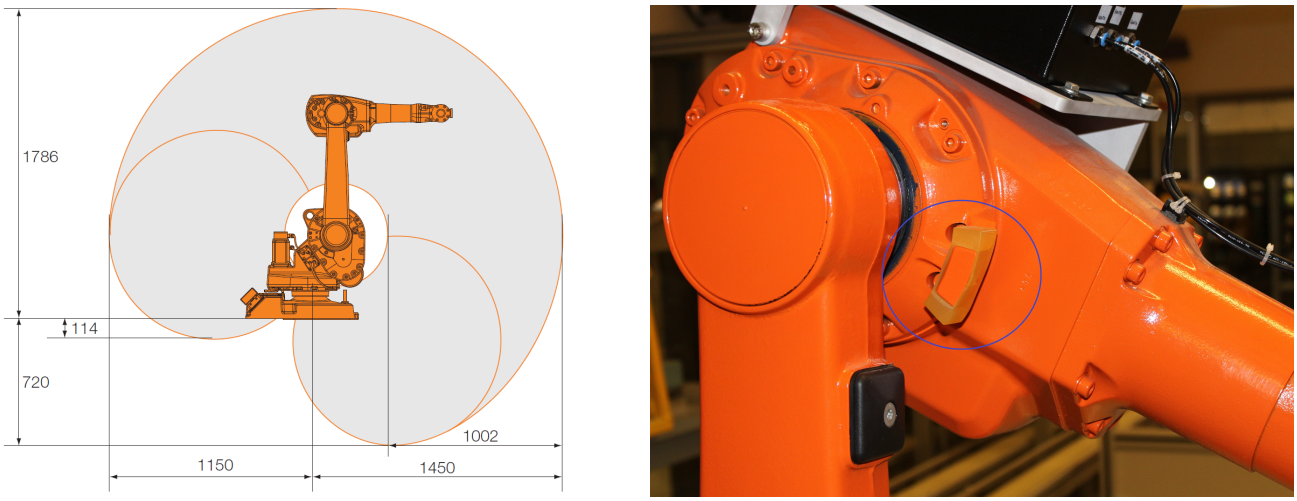


FIGURE 2.1 – Modèle 3D du robot IRB 1600.



(a) (b)

FIGURE 2.2 – (a) Section verticale de l'enveloppe de travail du robot IRB 1600 [10] et (b) une butée mécanique sur ce même robot.

2.1.1 Espace de travail et performance du robot

Nous allons appeler l'ensemble de poses que l'effecteur du robot peut atteindre l'*espace de travail*. L'enveloppe de travail d'un robot sériel avec des articulations rotoïdes est définie par les longueurs des segments, la disposition des axes des articulations, les limites des articulations et les interférences mécaniques entre les différents segments. La plage de déplacement et la vitesse maximale pour chaque articulation sont montrées au tableau 2.1 :

TABLEAU 2.1 – Capacités des articulations du robot IRB 1600.

Articulation	Plage (par défaut)	Plage (au local A-0610)	Vitesse
1	±180°	±180°	150°/s
2	−90° à 150°	−48° à 150°	160°/s
3	−245° à 65°	−135° à 65°	170°/s
4	±200°	±200°	320°/s
5	±115°	±112°	400°/s
6	±400°	110° à −290°	460°/s

La figure 2.2(a) montre une section verticale de ce qu'on appelle souvent l'*enveloppe de travail* du robot IRB 1600. Il s'agit simplement de la zone atteignable par le centre du *poignet* du robot (le point d'intersection des axes des trois dernières articulations). L'espace de travail du robot est quant à lui une entité géométrique à six dimensions fort complexe et ne peut pas être représenté graphiquement. De plus, l'espace de travail du robot dépend de l'outil utilisé.

Les articulations des robots au local A-0610 ont des plages de travail plus restrictives que celles fournies par le fabricant afin de réduire le risque de collisions. Il s'agit simplement de butées programmées dans le contrôleur. Cependant, afin d'être conforme aux normes et éviter les bris, les robots possèdent sur leurs articulations des butées mécaniques, comme illustré à la figure 2.2(b). Le robot IRB 1600 a de telles butées mécaniques sur les articulations 1, 2, 3 et 5.

Malgré la présence de réducteurs (ayant un ratio entre 44 et 130) entre les moteurs et les articulations, ces dernières peuvent atteindre des vitesses impressionnantes comme le montre le tableau 2.1. La combinaison des mouvements articulaires permet à l'effecteur du robot IRB 1600 d'atteindre des vitesses linéaires de l'ordre de 5 m/s.

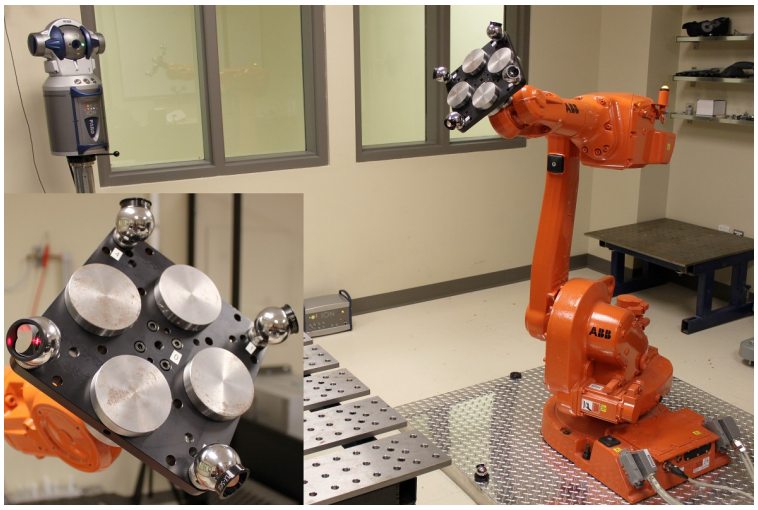


FIGURE 2.3 – Étalonnage d'un robot IRB 1600 et validation de sa précision au laboratoire CoRo.

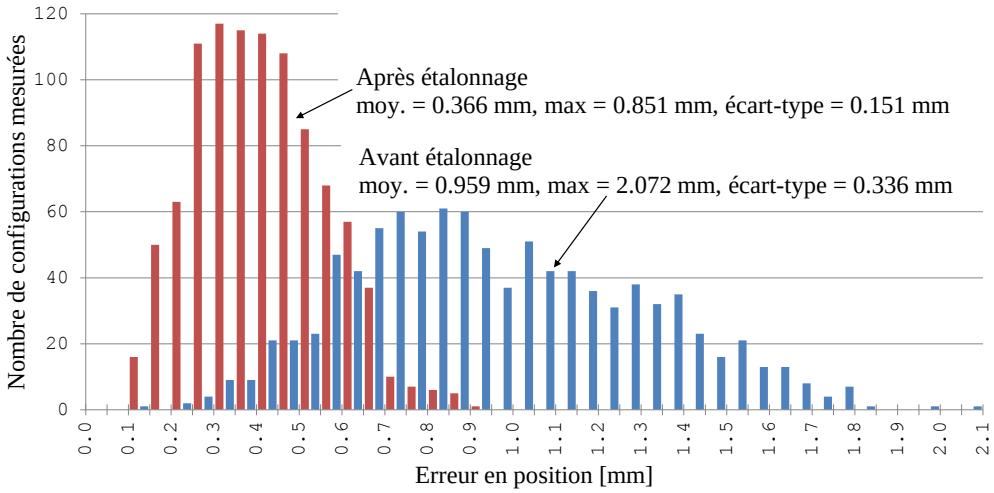
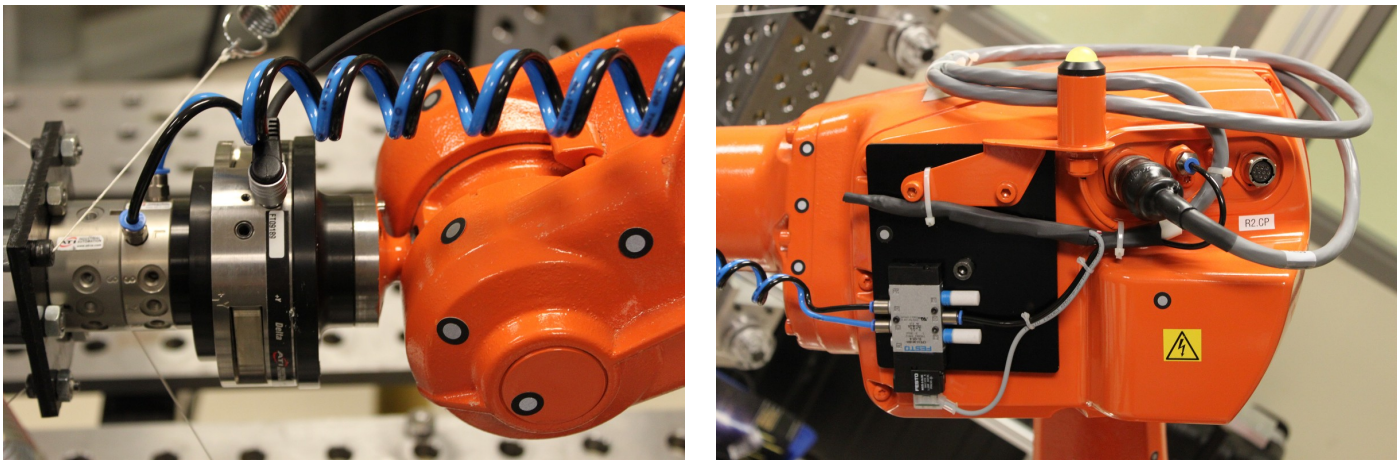


FIGURE 2.4 – Histogramme d'erreurs en position d'un robot IRB 1600 dans 1000 configurations arbitraires, avant et après étalonnage [12].

Les robots industriels sont capables de répéter un mouvement avec très peu de variation. Lorsqu'un robot essaie d'amener son effecteur à une même pose plusieurs fois, il existe une formule statistique pour calculer ce qu'on appelle *la répétabilité*. Dans le cas de la répétabilité en position, on peut dire de façon simpliste que la répétabilité est l'erreur maximale en position lorsque le robot va essayer d'amener son effecteur à une pose, pour une deuxième fois. Si le chemin emprunté est toujours le même, on parle de la répétabilité unidirectionnelle. Sinon, on parle de la répétabilité multi-directionnelle.

ABB spécifie que la répétabilité du robot IRB 1600 est de ± 0.050 mm. Au fait, il s'agit de la répétabilité unidirectionnelle. Selon des tests effectués au laboratoire de recherche CoRo à l'aide d'un système d'interférométrie laser [11], la répétabilité bidirectionnelle du robot est environ ± 0.150 mm.

Malheureusement, la *précision* du robot est loin d'être aussi bonne que la répétabilité. Il s'agit de l'erreur entre la pose désirée et la pose atteinte par l'effecteur, lors d'un seul déplacement. La précision est primordiale lorsqu'on programme un robot avec des coordonnées calculées ou provenant d'un modèle CAO. Selon des tests effectués au laboratoire de recherche CoRo à l'aide d'un laser de poursuite [11], l'erreur maximale en position du robot mesuré sur un des réflecteurs sphériques montrés à la figure 2.3 est environ 2 mm. À l'aide d'un processus complexe qu'on appelle *étalonnage*, il est possible de réduire cette erreur à environ 0.7 mm (figure 2.4) [12]. L'équipe du laboratoire CoRo travaille sur le développement de méthodes plus performantes pour améliorer la précision d'un robot industriel.



(a) (b)

FIGURE 2.5 – L'effecteur du robot IRB 1600 : (a) la bride du robot sur laquelle une plaque de montage est fixée, puis un capteur de forces/torques de ATI, une autre plaque de montage, un changeur d'outil d'ATI, et un outil (partiellement caché) ; (b) les connecteurs sur l'avant-bras.

2.1.2 Description des composants principaux

La bride de montage Par définition, un robot industriel est une machine multi-fonction et est rarement vendu avec de l'outillage. Par conséquent, l'extrémité de chaque robot industriel est munie d'une bride de montage sur laquelle on peut fixer un outil tel qu'un préhenseur ou une pince de soudage par point. Généralement, une bride de montage consiste en une plaque circulaire avec au moins quatre trous filetés et des trous de localisation. Lorsque le fabricant d'un robot industriel parle de la *charge utile* du robot (5 kg dans le cas du robot IRB 1600), il s'agit de la masse maximale de tout ce qui est fixé à sa bride (par exemple, une plaque de montage, un préhenseur et une pièce saisie) que le robot peut manipuler, mais à condition que le centre de gravité du tout soit très proche du centre de la bride. Dans la documentation technique détaillée, le fabricant d'un robot industriel va alors spécifier la charge maximale en fonction de la position du centre de gravité de l'effecteur par rapport à la bride de montage. Il est par conséquent important de connaître les caractéristiques physiques des objecs manipulés comme le poids, le centre de gravité et les moments d'inertie.

Enfin, il faut noter que dans plusieurs applications, un changeur d'outil est monté sur la bride du robot pour permettre le changement d'outils automatisé. Les changeurs d'outil sont généralement actionnés par deux conduits pneumatiques. Un des plus importants fabricants de changeurs d'outil est la compagnie américaine ATI. Nos robots IRB 1600 ne sont pas équipés d'un tel changeur d'outil, mais le robot IRB 120 utilisé en classe l'est. La figure 2.5(a) montre un autre exemple de changeur d'outil monté sur le robot IRB 1600 du laboratoire de recherche CoRo.

Les conduits pour l'effecteur Le plus souvent, l'outil installé sur la bride de montage aura besoin de conduits électriques et/ou pneumatiques. Ainsi, la plupart des robots industriels auront quelques conduits électriques et pneumatiques qui partent de la base du robot, passent à l'intérieur des membrures et ressortant vers l'extrémité du bras. Dans le cas du robot IRB 1600, il s'agit d'un conduit pneumatique (maximum 8 bars) et de 28 conduits électriques (maximum 50 mA par conduit) comme montré à la figure 2.5(b). Par la suite, il en revient au concepteur de la cellule robotique de faire le lien entre les connecteurs de raccord et l'outil. La plus grande difficulté ici est de mettre en place un conduit flexible répondant à tous les besoins de mouvement sans toutefois risquer de coincer et/ou se faire arracher durant un mouvement.

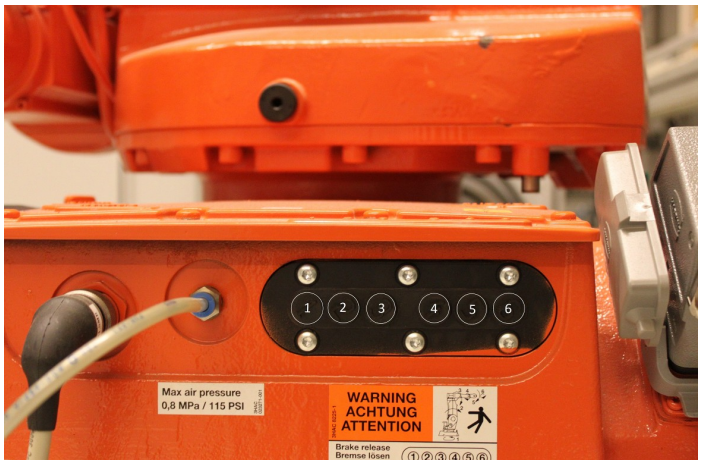


FIGURE 2.6 – Boutons pour relâcher les freins à NE JAMAIS ACTIONNER.

La base La base du robot doit être fixée sur une surface très rigide. Il est important de s'assurer de répondre aux normes du fabricant pour la méthode de fixation, afin d'assurer la sécurité. Le robot génère des forces très élevées lors d'un arrêt d'urgence. Le torque généré peut être plusieurs fois plus élevé que le torque produit durant le fonctionnement normal. Les quatre robots au laboratoire A-0610 offrent les mêmes comportements et possibilités, malgré qu'ils ne soient pas exactement ancrés avec la même orientation dans le laboratoire.

Freins électriques Pour des raisons de sécurité, il est strictement interdit d'actionner les boutons de relâchement mécanique des freins (figure 2.6) des robots IRB 1600 au local A-0610. Notez que le poids des deux bras est d'environ 50 kg. En plus de danger de blessures graves, il est également possible de briser le robot, son effecteur et les équipements autour de lui (glissoire, etc.).

2.2 Le contrôleur IRC5 et le FlexPendant

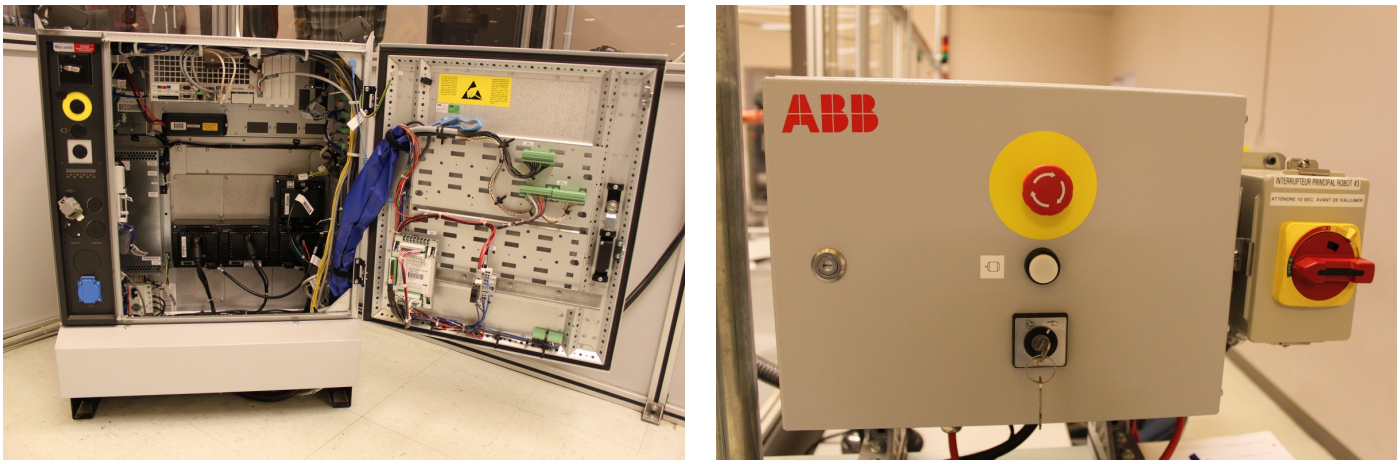
2.2.1 Le contrôleur IRC5

Le contrôleur IRC5 est le plus récent contrôleur de robot de la compagnie ABB. Il peut contrôler jusqu'à quatre robots en mode synchrone ou non. ABB utilise un seul contrôleur pour tous ses modèles de robots. Le format du boîtier est offert en plusieurs modèles et les variateurs dans le contrôleur sont évidemment différents d'un robot à l'autre. Le contrôleur utilise un système d'opération en temps réel appelé RobotWare qui pourra interpréter et exécuter les programmes implantés. Au laboratoire A-0610, les quatre robots IRB 1600 sont indépendants. Cependant, ils sont connectés dans un réseau local.

Le contrôleur est principalement composé des items suivants (figure 2.7a) :

- un ordinateur industriel de type Pentium 4 cadencé à 1.4 GHz, protégé par une carte CompactFlash de 256 Mo. Cette carte s'assure de préserver la mémoire lors d'une perte de tension dans le système. Ceci permet au robot de ne perdre aucune information importante [1], et ce même s'il était en pleine exécution d'un mouvement.
- une carte de contrôle des axes (Motorola PowerPC 250 MHz) ;
- un groupe puissance permettant le contrôle des moteurs (adapté au type de robot) ;
- une carte d'entrées/sorties de type TOR 24 Vdc permettant l'interaction entre les équipements sur la table, l'effecteur et le robot.

1. Il est important de comprendre que la protection du système prend quelques secondes lors de la mise hors tension. Pour cette raison, il ne faut jamais rallumer le robot sans attendre au moins 10 secondes.



(a) (b)

FIGURE 2.7 – (a) Contrôleur IRC5 standard de la compagnie ABB et (b) panneau opérateur externe d'un des quatre robots au laboratoire A-0610.

FIGURE 2.8 – Boîtier de commande FlexPendant de ABB.

Dans le cas de chaque contrôleur IRC5 des robots IRB 1600 au laboratoire A-0610, le bouton d'arrêt d'urgence, le bouton lampe de mise sous puissance des moteurs, et le sélecteur du mode à clé sont déportés sur un panneau opérateur externe (figure 2.7b). Un sectionneur général y est aussi présent.

2.2.2 Le boîtier de commande

Le boîtier de commande appelé FlexPendant par ABB permet à l'opérateur d'interagir avec le contrôleur du robot (figure 2.8). Le FlexPendant est en fait un ordinateur avec écran tactile de sept pouces avec une manette 3D et des boutons . Il communique avec le contrôleur par un réseau Ethernet privé et fonctionne sous un système d'exploitation WindowsCE. Ceci a deux avantages : il est possible de personnaliser les écrans en fonction du projet avec l'option ScreenMaker et voire même ne pas avoir le boîtier relié en tout temps avec le robot.

Voici quelques règles et informations concernant le FlexPendant :

- Le boîtier de commande est relié au contrôleur par un câble multi-brin relativement fragile. Il est important de ne pas le forcer, le coincer, l'écraser sous les souliers ou encore tirer dessus.
- Il est interdit d'utiliser des objets pour toucher l'écran tactile. Il faut plutôt utiliser les doigts. L'explication des différents menus et fonctionnalités est faite en classe. Le tout est expliqué dans le document IRC5 avec FlexPendant.

- Dans le coin du boîtier de commande, nous retrouvons un arrêt d'urgence de type bouton « pousser-tourner ». C'est ce bouton qui garantit la complète sécurité du programmeur quand il entre dans une cellule, à moins d'avoir un système équipé d'un dispositif de cadenassage.
- Sur le côté du boîtier de commande, nous retrouvons une gâchette « homme-mort ». Cette gâchette est composée de deux interrupteurs, à trois positions, en parallèle. La position du milieu active le robot.
- Pour piloter le robot en mode manuel, il faut utiliser la manette 3D.
- Enfin, nous retrouvons des boutons de contrôle sur le devant du boîtier : départ, arrêt, avancement pas-à-pas ou recul pas-à-pas. Il est important de comprendre que l'arrêt est un contrôlé par le système et fait en douceur, alors que l'arrêt d'urgence et la gâchette « homme-mort » arrêtent le système de manière brusque en activant immédiatement les freins, ce qui use le robot.

2.3 Programmation et manipulation du robot

2.3.1 Programmation hors ligne

La programmation hors ligne se fait de deux manières distinctes. La première est de modéliser une cellule virtuelle à l'aide du logiciel RobotStudio (qui est installé dans plusieurs laboratoires du Département) et de faire toute la programmation à l'aide de l'environnement virtuel de RobotStudio. Pour faciliter la réalisation des laboratoires, les quatre cellules du laboratoire A-0610 sont déjà modélisées et disponibles dans la section Laboratoires du site Web du cours (figure 2.9). L'utilisation de ces cellules vous serait montrée à la deuxième séance de laboratoire. Pour plus d'information sur le logiciel RobotStudio, vous pouvez visiter la chaîne RobotStudio sur YouTube et visionner les nombreux tutoriels ou lire le Manuel d'utilisation de RobotStudio, disponible sur le site Web du cours.

La deuxième manière est d'éditer vos modules à l'aide d'un logiciel texte de type ASCII. Vous pouvez donc utiliser un éditeur de texte pour MS Windows tel que Bloc-notes, UltraEdit, WinEdt, ou EditPlus. Il est fortement déconseillé de copier-coller à partir d'un fichier autre d'un fichier en format ASCII (par exemple, un fichier PDF ou MS Word).

2.3.2 Programmation en ligne

Pour interagir avec un des contrôleurs de robot au local A-0610, il faut s'y trouver physiquement. L'application RobotStudio est un logiciel de la compagnie ABB permettant d'éditer un programme et de configurer le contrôleur du robot. La démarche pour mettre en communication votre application RobotStudio avec votre robot attitré, à travers le réseau Ethernet du laboratoire, est expliquée dans les documents sur le site Web du cours et montré en classe : Procédures au laboratoire A-0610 et Manuel d'utilisation – RobotStudio 5.13.

⚠ Si vous n'êtes pas encore approuvé ou si vous êtes seul, vous n'avez pas le droit de bouger le robot. Cependant, vous pouvez quand même interagir avec son contrôleur pour la programmation.

2.3.3 Essai d'un programme et pilotage manuel

Pour piloter, bouger ou encore déverminer un programme, vous devez utiliser le FlexPendant (par exemple, afin d'enseigner des configurations). Il faut tourner le sélecteur de modes à clé (figure 2.7b) vers l'icône avec une main. Ceci met le robot en mode manuel, et la vitesse linéaire de l'effecteur est limitée à 250 mm/s. Lorsque le robot est sous tension, c'est uniquement en mode manuel que l'opérateur peut se trouver à l'intérieur de la cellule robotique avec le FlexPendant. En mode manuel, l'opérateur doit garder la gâchette active pour opérer le robot. L'ordre d'activation est le suivant :

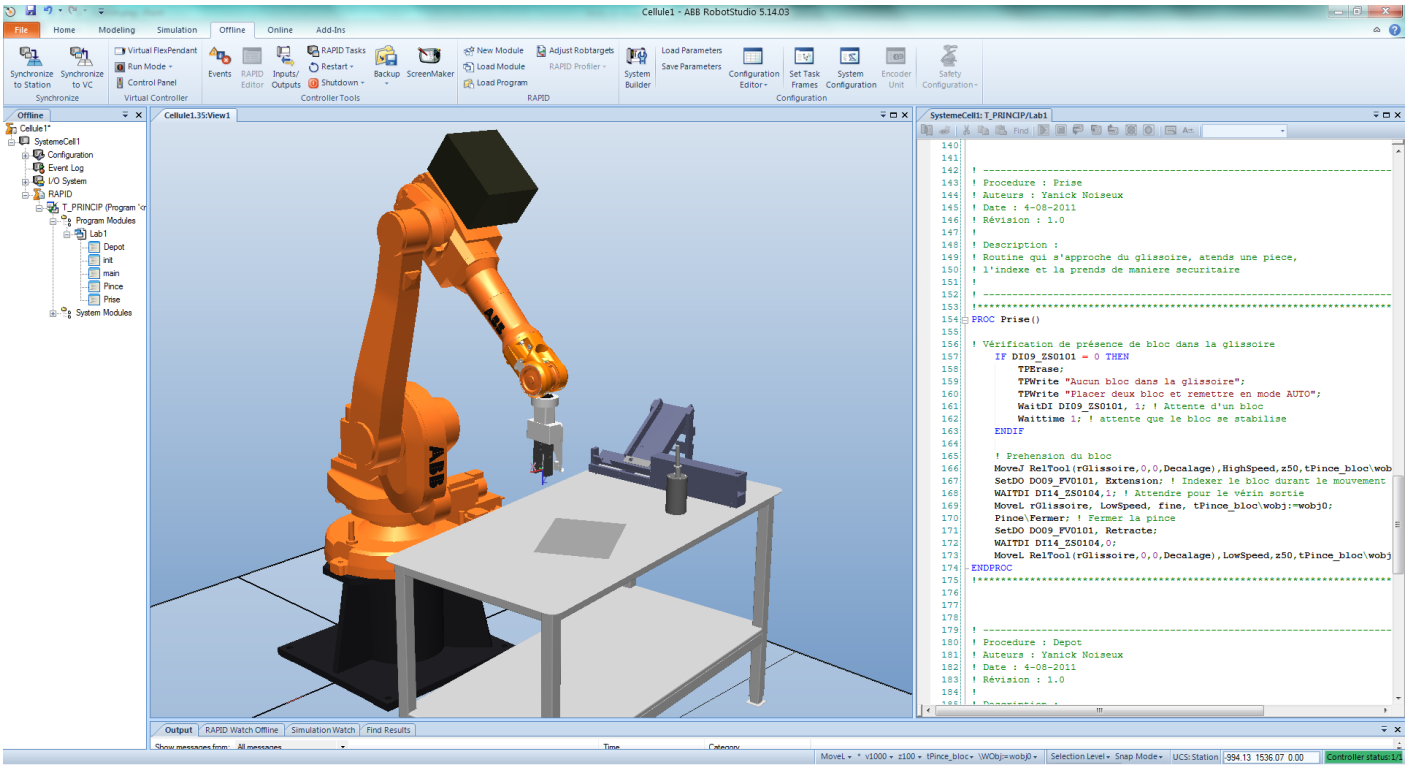


FIGURE 2.9 – Programmation hors ligne dans RobotStudio avec une cellule virtuelle.

1. L'opérateur doit réactiver les moteurs (bouton blanc au-dessus du sélecteur de modes à clé) suite à un arrêt d'urgence ou à un retour du mode automatique.
2. Ensuite, l'opérateur doit enfoncer la gâchette à moitié jusqu'à ce que les moteurs soient activés (voyant blanc allumé constant) et la maintenir dans cet état.
3. Enfin, l'opérateur peut piloter le robot ou exécuter un programme.

La démarche est expliquée en détail dans les deux références suivantes disponibles sur le site Web du cours : Procédures au laboratoire A-0610 et les chapitres 5 et 6 du manuel IRC5 avec FlexPendant d'ABB.

Pour exécuter un programme à pleine vitesse en mode automatique, le sélecteur de modes à clé doit être en position « automatique ». En mode automatique, l'opérateur doit activer le robot en réalisant les actions suivantes afin d'exécuter son programme :

1. sécuriser la zone ;
2. activer les moteurs (bouton blanc au-dessus du sélecteur de modes à clé) ;
3. lancer l'exécution du programme.

Chapitre 3

La sécurité

Malheureusement, la plupart de robots industriels n'obéissent pas aux trois lois de la robotique formulées pas l'écrivain Isaac Asimov. Voici alors trois maximes à ne pas oublier :

- si le robot ne bouge pas, ne pas assumer qu'il ne bougera pas ;
- si le robot répète une séquence, ne pas assumer qu'il va continuer à le faire ;
- maintenir du respect pour ce qu'est un robot et ce qu'il peut faire.

3.1 Les normes de sécurité

L'utilisation et la mise en service d'un robot ou d'un environnement robotisée sont régies par une norme canadienne s'assurant d'un minimum de sécurité pour les humains ayant à interagir avec cette installation. La norme ayant force de loi et à laquelle nous devons répondre au Québec est la norme CSA-Z434-03 [14]. Celle-ci s'applique pour tout robot industriel, neuf ou usagé. Cette norme intervient lors de la conception, l'installation, la programmation et l'utilisation de robots industriels. Il est donc important pour un ingénieur de la comprendre et de s'assurer du respect de celle-ci.

3.2 La sécurité au laboratoire A-0610

Dans le cadre du laboratoire A-0610, il est important de respecter les équipements et d'avoir un comportement sécuritaire. Pour s'assurer que tout se déroule correctement, le Département a mis en place un guide de sécurité afin de réduire au minimum les risques d'accident. Il faut comprendre que le laboratoire n'est pas toujours sous la supervision d'un chargé de TP. Pour s'assurer du respect du guide de sécurité, celui-ci prévoit des sanctions associées au non-respect des différentes règles. Plusieurs personnes (professeurs, techniciens, gardiens, étudiants, etc.) peuvent aviser l'enseignant du cours en cas de non-respect du guide. L'enseignant prendra alors les mesures nécessaires pour sanctionner l'équipe ou l'étudiant fautifs.

Un comportement sécuritaire passe par une formation adéquate. Toute personne ayant à interagir avec une cellule robotisée devrait avoir formation suffisante. Plus de 60 % des accidents avec les robots industriels arrivent lors de la programmation et la mise en service. Il est donc primordial que vous ayez compris les Règles de sécurités et que vous les respectiez.

3.2.1 Dispositifs d'urgence

Le laboratoire répond aux critères de sécurité de la norme CSA-Z434-03 ce qui signifie que plusieurs dispositifs de sécurité sont déployés, plus spécifiquement des boutons d'arrêt d'urgence, des portes sécurisées, et un rideau de lumière. Ces dispositifs de sécurité interviennent à différents niveaux.

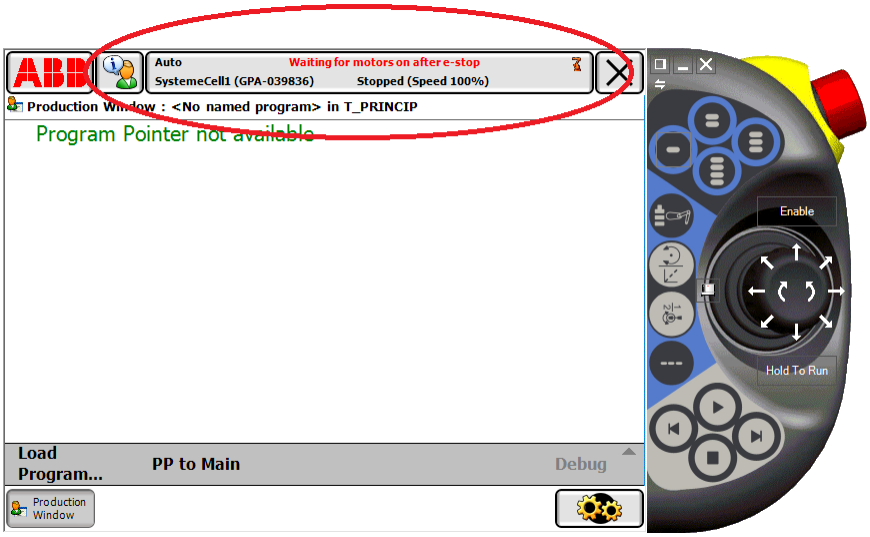


FIGURE 3.1 – Attente du réarmement des moteurs.

Les boutons d'arrêt d'urgence du laboratoire Les boutons d'arrêt d'urgence interviennent en tout temps sur un robot, qu'il soit en mode manuel ou automatique. Les deux arrêts d'urgence situés sur le mur en plâtre provoquent l'arrêt de tous les équipements du laboratoire. À l'inverse, l'arrêt d'urgence sur le mur en Plexiglas devant chaque robot arrête uniquement le robot correspondant. Quand un arrêt d'urgence a été activé, pour ramener l'opération normale des équipements concernés, on doit enlever l'alarme (tirer le bouton d'arrêt d'urgence). Une fois l'alarme retirée, on doit réarmer la cellule robotisée. Dans le cas d'une alarme de laboratoire, on réarme le laboratoire (bouton vert sur le mur en plâtre), et ensuite on réarme chaque cellule (bouton jaune sous l'arrêt d'urgence situé sur les murs en Plexiglas).

Rideau de lumière Le rideau de lumière permet de séparer la cellule 1 et 2. Il crée un arrêt d'urgence sur les robots 1 et/ou 2, seulement si les robots sont en mode automatique. Quand le rideau est coupé, un gyrophare clignote afin de mettre en évidence le déclenchement du système de sécurité. Pour le réarmer, il faut appuyer sur un des deux boutons verts à l'entrée des cellules 1 et 2.

Porte de cellule L'interrupteurs d'urgence sur chaque porte de cellule est actif que si le robot est en mode automatique. L'ouverture de la porte crée un arrêt d'urgence sur le robot. Il n'y a aucun réarmement à faire pour ce type d'arrêt. Il faut noter que la cellule 4 a deux portes avec interrupteurs d'urgence. Il faut alors fermer les deux portes pour pouvoir utiliser le robot 4 en mode automatique.

Les boutons d'arrêt d'urgence des robots Les boutons d'urgences sur chaque panneau opérateur externe de contrôleur d'un robot et sur son boîtier de commande interviennent en tout temps sur le robot, qu'il soit en mode manuel ou automatique. Quand un arrêt d'urgence a été activé, on doit enlever l'alarme (tourner le bouton d'arrêt d'urgence). Une fois l'alarme retirée, on doit acquitter la faute sur le robot avec le bouton « Acknowledge » à l'écran du FlexPendant, et ensuite réarmer les moteurs (figure 3.1) avec le bouton blanc situé sur le panneau opérateur externe (figure 2.7).

Chapitre 4

Programmation

4.1 Structure d'un projet

Le langage de programmation des robots ABB s'appelle RAPID. Un programme en RAPID est un ou plusieurs fichiers de type texte (ASCII), non compilés. Le contrôleur du robot interprété le code, tout en validant la syntaxe du programme en entier. Un programme contenant des erreurs de syntaxe peut être chargé dans le contrôleur, mais ne peut pas être exécuté.

Il existe deux types de fichiers possibles. Le premier type est le plus important : les *modules* (.mod) qui contiennent le programme. Cela ne fait aucune différence pour le contrôleur du robot si le programme est écrit dans un seul ou dans plusieurs modules. L'utilisation de plusieurs modules se justifie uniquement dans la plus grande facilité de saisi et de réutilisation des modules par le programmeur. Par contre, il ne peut y avoir qu'un seul programme actif dans le contrôleur du robot ; c'est-à-dire qu'un seul des modules peut contenir une procédure appelée « main ».

Lors du chargement dans le contrôleur des différents modules, le contrôleur assumera automatiquement l'ensemble de modules comme un programme. Les modules peuvent être chargés séparément, un par un, ou en utilisant le deuxième type de fichier : le *programme* (.pgf), qui est simplement un fichier de type XML qui spécifie la liste des modules à charger (exemple 4.1). Le contrôleur amorce le programme en partant par la procédure « main ». Celle-ci doit être au début d'un module, juste après la déclaration des variables globales à la tâche (exemple 4.2).

Exemple 4.1 – Fichier du type programme pour contrôleur ABB

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<Program>
 <Module>ModuleA.mod</Module>
 <Module>MainModule.mod</Module>
</Program>
```

Exemple 4.2 – Module en langage RAPID

```
MODULE NomDuModule()
    !VARIABLES et/ou CONSTANTES et/ou PERSISTANTES GLOBALES
    CONST   num length:=10;
    VAR     num width;
    PERS    num area:=0;

    PROC main()
        width := 5;
        calc;
        TPWrite "L'aire du rectangle est " \Num:=area;
    ENDPROC

    PROC calc()
        area := length * width;
    ENDPROC
ENDMODULE
```

4.2 Syntaxe

Le contrôleur n'est pas sensible à la case, ce qui signifie par exemple que dans un module, les variables « p1 » et « P1 » seront considérées les mêmes. Le langage RAPID n'est pas à structure imposée, non plus. Il est donc possible d'utiliser plusieurs espaces, tabulations et retours de chariot, afin de faire une belle mise en page (exemple 4.3). Enfin, c'est le point d'exclamation qui est utilisé pour les commentaires.

Exemple 4.3 – Mise en page possible

```
MODULE Mouvement()
    CONST robtarget P1:=[[300,0,500],[0,1,0,0],[0,-1,0,0],[9E9,9E9,9E9,9E9,9E9,9E9]];
    CONST robtarget P2:=[[0  ,0,500],[0,1,0,0],[0,-1,0,0],[9E9,9E9,9E9,9E9,9E9,9E9]];
    PERS robtarget Parray{2}:=[[[0,0,500],[0,1,0,0],[0,-1,0,0],[9E9,9E9,9E9,9E9,9E9,9E9]],
                               [[0,0,500],[0,1,0,0],[0,-1,0,0],[9E9,9E9,9E9,9E9,9E9,9E9]]];
    !Commentaire
    PROC main()
        ! Ligne de mouvement classique
        MoveJ p1,vMax,z5,tool0;
        ! Ligne de mouvement classique, mais avec une mise en page
        ! souhaitee par le programmeur
        MoveL p2,
              v1000,
              z5,
              tool0;
    ENDPROC
ENDMODULE
```

La plupart des instructions exigent un point-virgule à fin. Les deux procédures dans l'exemple 4.4 sont interprétées de la même manière par le contrôleur. Le programmeur a donc toute la lassitude nécessaire afin de rendre son programme lisible, bien documenté, et facile à déverminer et à corriger.

Exemple 4.4 – Délimitateur d'instruction

```
    PROC ConfigsOn()
        ! Activation des configurations du robot
        ConfJ\On;
        ConfL\On;
    ENDPROC

    PROC ConfigsOn()
        ! Activation des configurations du robot
        ConfJ\On; ConfL\On;
    ENDPROC
```

Les procédures et les fonctions acceptent le passage de paramètres. Il est possible lors de la déclaration de définir la méthode ou de forcer des valeurs. Le passage de valeurs peut se faire de plusieurs façons :

- Par valeur. Une copie de la valeur est transmise à la routine.
  Exemple : PROC routine1(num nom_parametre)
- Par référence. Le pointeur du registre mémoire est transmis (exemple 4.5).
  Exemple : PROC routine1(INOUT num nom_parametre)
- Par déclaration d'une variable. Il faut s'assurer que le passage est de type variable.
  Exemple : PROC routine1(VAR num nom_variable)
- Par déclaration d'une persistante. Il faut s'assurer que le passage est de type persistante.
  Exemple : PROC routine1(PERS wobjdata nom_persistante)

Exemple 4.5 – Passage de paramètre par référence

```
    PROC main()
        VAR w := 4;
        multpar10 w; !la valeur de w deviendra 40
    ENDPROC
    PROC multpar10(INOUT num p)
        p := 10*p;
    ENDPROC
```

Il est possible de rendre le passage de paramètre optionnel en ajoutant une barre oblique (« \ ») devant la déclaration. Si cela est fait, il est important que l'on vérifie la présence de la variable avant de l'utiliser, à l'aide de la fonction PRESENT, comme montrée à l'exemple 4.6.

Exemple 4.6 – Syntaxe d'un passage de paramètre optionnel

```
FUNC bool ProcheDe(robtarget rTmp,\num dist)
    VAR bool Resultat:=FALSE;
    VAR robtarget pos1;
    VAR num d;

    ! teste la distance
    pos1:=CRobT(\Tool:=tpince_bout \Wobj:=wobj0);
    d := distance( pos1.trans, rTmp.trans);

    ! verifie le resultat avec parametre ou valeurs par defaut
    IF PRESENT(dist) Then
      Resultat:= (d<dist);
    ELSE
      Resultat:= (d<50);
    ENDIF

    ! retourne le resultat
    RETURN Resultat;
ENDFUNC
```

Une fonction peut retourner une valeur avec l'instruction RETURN comme le montre l'exemple 4.7.

Exemple 4.7 – Syntaxe d'une fonction

```
FUNC num veclen(pos vector)
    RETURN SQRT(POW(vector.x,2)+POW(vector.y,2)+POW(vector.z,2));
ENDFUNC
```

Il est important de noter que le nom d'une procédure, d'une fonction ou d'une interruption ne peut pas avoir plus de 32 caractères, et ne peut par commencer par un caractère numérique ou par un caractère spécial. Cependant, noter que le tiret bas (« _ ») est toléré comme montré à l'exemple 4.8.

Exemple 4.8 – Syntaxe de déclaration

```
VAR signaldo  _17FV0601;
```

Il est également important de noter que les noms des modules, procédures, fonctions et registres mémoires doivent être uniques. Exemple, le contrôleur ne sera pas utiliser un module portant le même nom d'une routine ou encore comprendre l'assignation si une fonction porte le même nom d'une variable.

4.3 Opérateurs arithmétiques et booléens

Les tableaux 4.1 et 4.2 énumèrent les différentes opérations arithmétiques et booléennes.

Tableau 4.1 – Opérations arithmétiques.

Instruction	Fonctionnement	Opérande	Type de résultat
+	addition	num + num	num[3]
+	addition	dnum + dnum	dnum[3]
+	opération unaire plus ; maintien du signe	+num ou +dnum ou +pos	même[1,3]
+	addition vecteur	pos + pos	pos
−	soustraction	num − num	num[3]
−	soustraction	dnum − dnum	dnum[3]
−	opération unaire moins ; changement du signe	−num or −pos	même[1,3]
−	opération unaire moins ; changement du signe	−num ou −dnum ou −pos	même[1,3]
−	soustraction vecteur	pos − pos	pos
*	multiplication	num * num	num[3]
*	multiplication	dnum * dnum	dnum[3]
*	multiplication vecteur scalaire	num * pos ou pos * num	pos
*	produit vectoriel	pos * pos	pos
*	enchaînement des rotations	orient * orient	orient
/	division	num / num	num
/	division	dnum / dnum	dnum
DIV[2]	division d'un nombre entier	num DIV num	num
DIV[2]	division d'un nombre entier	dnum DIV dnum	dnum
MOD[2]	module d'un nombre entier ; reste	num MOD num	num
MOD[2]	module d'un nombre entier ; reste	dnum MOD dnum	dnum

4.4 Type de variables de base

Le langage RAPID offre la possibilité au programmeur de se créer des registres mémoires afin de stocker des informations pertinentes au déroulement d'un projet. Son approche est similaire à un langage de programmation informatique. La déclaration est structurée et les registres que l'on désire accessibles de partout doivent être déclarés dans le début du module avant toutes instructions. Ce que l'on désire

1. Le résultat reçoit le même type que l'opérande. Si l'opérande a un type de données pseudonyme, le résultat reçoit le type de base du pseudonyme (num, dnum ou pos).
2. Opérations de nombres entiers, par exemple 14 DIV 4 = 3, 14 MOD 4 = 2 (les opérandes qui ne sont pas des nombres entiers ne sont pas autorisés).
3. Conserve la représentation du nombre entier (exact) tant que les opérandes et le résultat sont maintenus dans le sous-domaine du type numérique.
4. any : Les opérandes doivent avoir des types compatibles.

Tableau 4.2 – Opérations booléennes.

Instruction	Fonctionnement	Opérande	Type de résultat
<	inférieur à	num < num	bool
<	inférieur à	dnum < dnum	bool
<=	inférieur ou égal à	num <= num	bool
<=	inférieur ou égal à	dnum <= dnum	bool
=	égal à	any[4] = any[4]	bool
>=	supérieur ou égal à	num >= num	bool
>=	supérieur ou égal à	dnum >= dnum	bool
>	supérieur à	num > num	bool
>	supérieur à	dnum > dnum	bool
<>	différent de	any[4] <> any[4]	bool
AND	ET	bool AND bool	bool
XOR	OU exclusif	bool XOR bool	bool
OR	OU	bool OR bool	bool
NOT	négation	NOT bool	bool

local à une routine doit-être dans son en-tête, avant toute instruction comme montré à l'exemple 4.9. Il est impossible de déclarer une variable à mi-programme. L'ordre de classement des registres dans la déclaration n'est pas imposée. Il est possible de les regrouper par type, besoin, équipement, etc.

Exemple 4.9 – Déclaration de variables

```
MODULE MainModule
    PERS    ! declaration des variables globales protegees
    CONST   ! declaration des constantes globales
    VAR     ! declaration des variables globales

    PROC main() ! routine principale
        ! Instructions executables;
        ...
    ENDPROC

    ! Declaration de fonctions et de procedures
    PROC nom_routine()
        CONST ! declaration de constantes globales
        VAR ! declaration de variables globales
        ! instructions executables
        ...
    ENDPROC

ENDMODULE
```

La définition d'un registre mémoire peut être de trois types :

1. Constante (CONST) : Une constante représente une valeur statique lors de l'exécution du programme. Elle peut seulement recevoir une nouvelle valeur manuellement que ce soit lors d'une programmation en ligne ou hors-ligne.
2. Variable (VAR) : Une variable représente une valeur dynamique que le programme peut modifier lors de son exécution. Elle peut recevoir une nouvelle valeur en tout temps. Celle-ci n'est réinitialisée que lorsqu'on demande au contrôleur de réinitialiser un programme (opération appelée « PP to Main »).
3. Persistante (PERS) : Une persistante peut être décrite comme étant une variable « persistante » dans le temps. Lorsqu'on modifie sa valeur, sa définition est automatiquement mise à jour par le contrôleur afin de retenir toujours la dernière valeur. Il est important de noter que pour faire cette rétention, la variable doit-être déclarée avec une initialisation comme montré à l'exemple 4.10.

Les persistantes ne peuvent être déclarées qu'au niveau d'un module, et non à l'intérieur d'une procédure.

Exemple 4.10 – Déclaration de persistantes

```
PERS ToolData tActif; ! persistante dont la valeur ne sera pas sauvegardee
PERS wobjdata wobj1:=[FALSE,TRUE,"", [[405.478,-499.583,61.8855],[0.564191,0.11975,0.215742,0.787911]],
    [[0,0,0],[1,0,0,0]]]; ! persistante dont la valeur sera sauvegardee en modifiant automatiquement cette
         declaration du sauvagarde
```

Le langage RAPID possède tous les types de données atomiques classiques : booléen, numérique, chaîne de caractères, etc. Ceux-ci ne seront pas expliqués dans le présent document, puisqu'ils le sont dans la section Type de données du manuel de référence de RAPID. Il est intéressant de noter seulement que les valeurs de type entier ou nombre décimal sont stockées en utilisant le type de donnée num.

Le langage RAPID possède également le type *enregistrement*. Un enregistrement est composé d'un mélange de données atomiques et/ou d'enregistrements. Chaque composante d'un enregistrement a un nom et peut être adressée en utilisant le nom de l'enregistrement suivi par le séparateur « . » suivi par le nom de la composante, etc. (par exemple, l'instruction bloc1.trans.x := 10 changera la coordonnée x de la pose bloc1).

Les enregistrements les plus importants sont les poses (pose), les référentiels (wobjdata), la pose de l'effecteur du robot combinée à la configuration du robot (robtarget) et les définitions d'outil (tooldata).

pose Une pose est une représentation mathématique d'un référentiel par rapport à un autre. Dans RAPID, un enregistrement de type pose est composé d'un enregistrement de type position (pos) et d'un enregistrement de type orientation (orient), par exemple $[[0,0,0],[1,0,0,0]]$. La représentation d'une orientation est un concept difficile que vous êtes supposés d'avoir vu dans le cours GPA445 (Conception assistée par ordinateur). Les représentations à l'aide de trois angles d'Euler, de quatre quaternions, et d'une matrice de rotation vous seront expliquées en détail dans le chapitre 6. Le langage RAPID utilise la représentation par quaternions.

wobjdata L'enregistrement wobjdata (« work object data ») permet de définir un référentiel de travail. Il est très rare, voire quasi impossible, d'avoir un référentiel de robot parfaitement aligné avec son milieu de travail. Il est donc souvent nécessaire de se définir un référentiel par rapport à un objet et/ou une surface de travail. De plus, les cellules étant parfois complexes, il n'est pas rare d'avoir plus d'un lieu de travail. Dans le cas d'un robot ABB, les enregistrements de type wobjdata doivent toujours être déclarés globaux et persistantes (PERS). L'enregistrement wobjdata est une structure complexe composée des éléments suivants :

- robhold (bool) : Sert à définir si le robot tient l'outil, ou si l'outil est fixe. Dans le cas du local A-0610, cette composante sera toujours égale à false, car chaque outil est fixé à la bride du robot.
- ufprog (bool) : Sert à définir si le référentiel se déplace dans l'espace ou s'il est fixe. Dans le cas des cellules du local A-0610, cette composante sera toujours égale à true, car aucun convoyeur n'est doté d'encodeurs permettant de suivre le mouvement.
- ufmec (string) : Sert à définir le mécanisme à suivre si ufprog est faux. Dans le cas des cellules du local A-0610, cette composante sera toujours égale à une chaîne de caractères vide.
- uframe (pose) : Sert à définir la pose du référentiel du lieu de travail (« user frame ») par rapport au référentiel de l'atelier wobj0 (figure 4.1). La valeur de cette composante provient le plus souvent d'un enseignement manuel à l'aide du FlexPendant ou de la fonction DefFrame.
- oframe (pose) : Sert à définir la pose de l'objet sur lequel on va travailler (« object frame ») par rapport au uframe (figure 4.1). Souvent, cette composante est laissée à sa valeur par sans effet, soit $[[0,0,0],[1,0,0,0]]$ (c'est-à-dire que le référentiel oframe coïncide avec le référentiel uframe). La valeur de cette composante, si modifiée, provient le plus souvent d'un enseignement manuel à l'aide du FlexPendant ou de la fonction DefFrame.

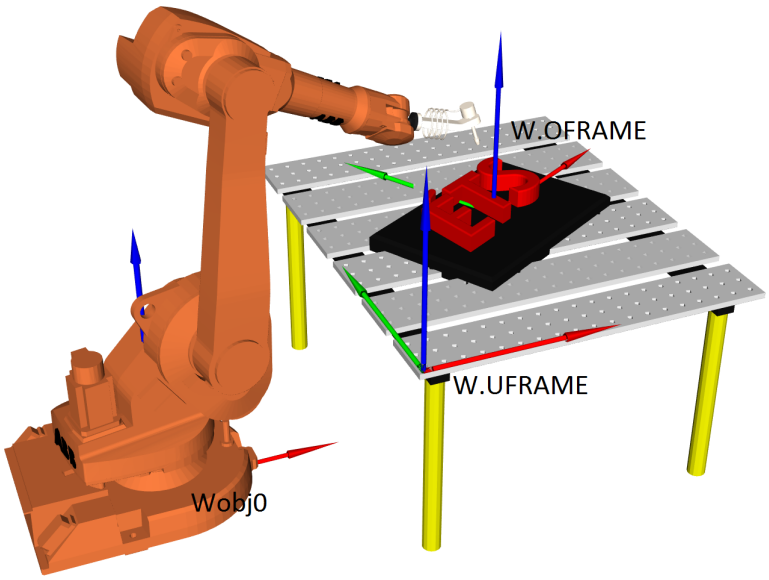


FIGURE 4.1 – Référentiels faisant partie d'un enregistrement de type workoject, dont le nom est W.

robtarget L'enregistrement de type robtarget est utilisé pour représenter une pose de l'outil du robot par rapport à un référentiel (wobjdata) non spécifié, ainsi que la configuration du robot et les positions des moteurs supplémentaires (tel qu'un guide linéaire ou une table pivotante). La structure de l'enregistrement robtarget se définit comme suit :

- trans (pos) : Contient les coordonnées x, y et z (en mm) d'un référentiel d'outil non spécifié par rapport à un référentiel de travail (wobjdata) non spécifié.
- rot (orient) : Contient le quaternion exprimant l'orientation du même référentiel d'outil par rapport au même référentiel de travail (wobjdata).
- robconf (confdata) : Permet de définir la valeur du cadran des articulations 1, 4 et 6, ainsi que le numéro de configuration (de 0 à 7, dans le cas des robots sériels à 6 ddl). Ce concept est assez difficile et vous sera expliqué en classe à l'aide du robot IRB 120, ainsi que plus tard dans ce manuel.
- extax (extjoint) : Permets d'interagir avec les articulations externes du robot. Nous n'en possédons aucune au local A-0610. Donc, les valeurs doivent rester à [9E9, 9E9, 9E9, 9E9, 9E9, 9E9] afin que le contrôleur n'utilise pas ces valeurs.

tooldata L'enregistrement de type tooldata permet de définir un référentiel sur un outil ainsi que les caractéristiques physiques de l'outil. Dans le cadre du laboratoire A-0610, l'outil de chaque robot est fixé à la bride du robot. En plus de contenir un référentiel physique, l'enregistrement va également contenir le poids, le centre de gravité et les moments d'inertie de l'outil. Il est important de bien définir ceux-ci si on veut optimiser la précision et la vitesse du robot. Les caractéristiques de la charge manipulée ne sont pas définies ici. L'enregistrement de type tooldata est une structure composée principalement des trois niveaux suivants :

- robhold (bool) : Sert à définir si le robot tient l'outil, ou si l'outil est fixe. Dans le cas des cellules du local A-0610, cette valeur sera toujours égale à false, car robot tient l'outil.
- tframe (pose) : Permet de définir la pose du référentiel de l'outil par rapport au référentiel de la bride, appelé tool0.
- tload (loaddata) : Permet de définir la charge physique de l'outil et ainsi permettre au robot de connaître les forces physiques générées par l'outil lors de son déplacement et ainsi optimiser sa vitesse et sa trajectoire.

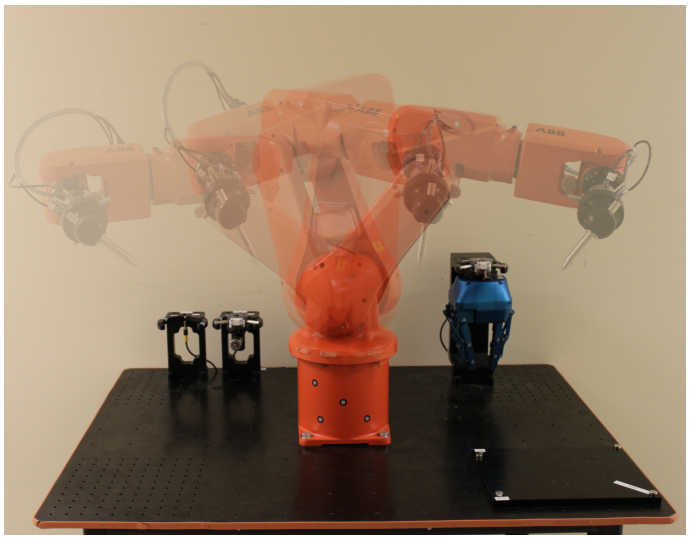


FIGURE 4.2 – Déplacement linéaire de l'outil (commande MoveL).

Dans le cadre des laboratoires de GPA546, deux outils sont déjà définis pour vous sur les robots : tPince_Bout et tPince_Bloc. La définition de ces outils est présentée à la section Documentation du site du cours GPA546.

4.5 Commandes de déplacement

Le but premier d'une commande de déplacement est de déplacer l'outil du robot d'une manière spécifique. Dans les commandes de déplacements les plus utilisées, on doit toujours spécifier un robtarget comme cible. Il existe trois types de déplacements dont la cible est un robtarget.

4.5.1 Types de déplacement

Déplacement linéaire Dans un déplacement linéaire, l'origine du référentiel de l'outil spécifié suit une trajectoire linéaire (figure 4.2). La réorientation de l'effecteur, si nécessaire, se fait de façon automatique (l'opérateur ne peut ne pas influencer cette réorientation). Ce type de déplacement est très contraignant, car il oblige souvent le robot à réaliser des mouvements complexes et parfois brusques.

Un déplacement linéaire peut également être limité par ce qu'on appelle des singularités. Quand un programmeur rencontre un singularité dans un mouvement linéaire, il est fréquent que la seule solution soit un changement physique de la cellule de travail afin que le robot ne repasse plus sur cette singularité. L'explication de ce qu'est une singularité sera abordée ultérieurement dans le cours. Les singularités ne sont pas liées à une marque de robot, mais bien au design mécanique ce qui signifie que certains robots en possèdent plus que d'autres et que toutes les marques de robot ont ce problème avec certains de leurs modèles. Par contre, chaque fabricant du robot a des limitations spécifiques additionnelles lors d'un déplacement linéaire. Dans le cas d'un robot ABB, la somme des mouvements pour chaque paire d'articulations 1 et 4, 1 et 6, 3 et 4 ou 3 et 6, ne doit pas dépasser 180°.

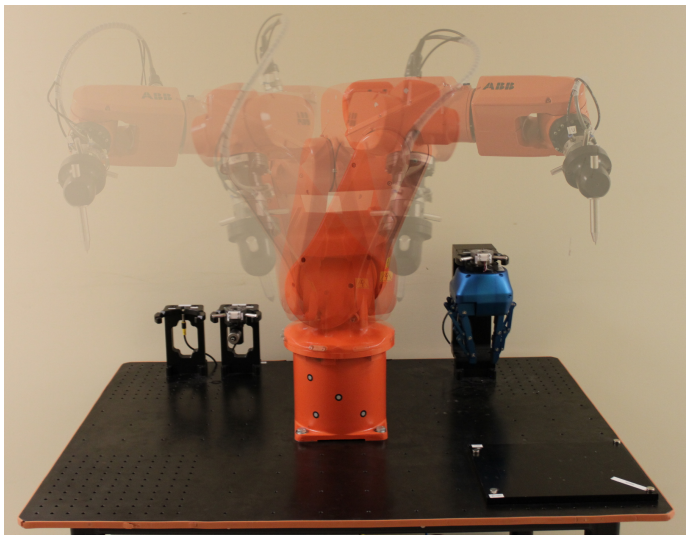


FIGURE 4.3 – Déplacement articulaire de l'outil (commande MoveJ).

Les paramètres de base d'une commande de linéaire sont comme suit :

```
MoveL robtarget, speeddata, zonedata, tooldata, \wobj:=wobjdata;
```

1. MoveL : Commande pour un déplacement linéaire de l'endroit où il se trouve vers le robtarget spécifié.
2. robtarget : Valeur venant d'une variable, fonction ou autre qui indique la destination de l'outil.
3. speeddata : Vitesse de croisière maximale que l'outil peut d'atteindre durant le déplacement. Nous allons expliquer en détail ce paramètre dans les sections suivantes.
4. zonedata : Précision d'arrêt ou d'approche de l'origine du référentiel de l'outil sur un robtarget à atteindre. Nous allons expliquer en détail ce paramètre dans les sections suivantes.
5. tooldata : Le référentiel de l'outil et les caractéristiques physiques de l'outil.
6. wobjdata : Paramètre optionnel qui défini par rapport à quel référentiel de travail on veut positionner le référentiel de l'outil à la pose définie dans le robtarget. Si ce paramètre est omis, le référentiel par défaut est celui de l'atelier (appelée wobj0).

Exemple 4.11 – Lignes de commande d'un déplacement linéaire

```
MoveL RelTool(robPrise,0,0,-300), v2000, z40, tPince_bout\wobj:=fixture;

MoveL [[337.69,-859.05,774.51],[0.000762984,0.852406,-0.522879,0.000750765],
       [-1,-1,-2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]], v2000 \T:=5, z40, tPince_bout;

MoveL tStart, v2000, z40, tPince_bout;
```

Déplacement articulaire Dans un déplacement articulaire, la trajectoire suivie par l'effecteur du robot est difficilement prévisible (figure 4.3). Ce type de déplacement offre une grande liberté de mouvement sans risque de singularité durant le déplacement. Le robot réalise un déplacement articulaire en définissant les valeurs des articulations à la configuration où il se trouve, et ensuite en calculant les valeurs de celles-ci à la destination finale. Une fois le déplacement nécessaire de chaque articulation connu, le robot démarre toutes les articulations en même temps, à différentes vitesses, afin de toutes les arrêter en même temps, à la destination finale (définie par un robtaget).

Les paramètres de base d'une commande de déplacement articulaire sont comme suit :

```
MoveJ robtarget, speeddata, zonedata, tooldata, \wobj:=wobjdata;
```

1. MoveJ : Commande indiquant un déplacement articulaire de l'endroit où il se trouve vers le robtarget spécifié.
2. robtarget : Valeurs venant d'une variable, fonction ou autre qui indique la destination de l'outil.
3. speeddata : Vitesse de croisière maximale que l'outil peut d'atteindre durant le déplacement. Nous allons expliquer en détail ce paramètre dans les sections suivantes.
4. zonedata : Précision d'arrêt ou d'approche de l'origine du référentiel de l'outil sur un robtarget à atteindre. Nous allons expliquer en détail ce paramètre dans les sections suivantes.
5. tooldata : Le référentiel de l'outil et les caractéristiques physiques de l'outil.
6. wobjdata : Paramètre optionnel qui défini par rapport à quel référentiel de travail on veut positionner le référentiel de l'outil à la pose définie dans le robtarget. Si ce paramètre est omis, le référentiel par défaut est celui de l'atelier (appelée wobj0).

Exemple 4.12 – Lignes de commande d'un déplacement articulaire

```
MoveJ RelTool(robPrise,0,0,-300), v2000, z40, tPince_bout\wobj:=fixture;
MoveJ tStart, v2000, z40, tPince_bout;
```

Déplacement circulaire Dans un déplacement circulaire, l'origine du référentiel de l'outil spécifié suit une trajectoire circulaire passante par trois robtargets (le robtarget actuel, un intermédiaire et un final). Un déplacement circulaire a les mêmes contraintes qu'un déplacement linéaire puisque l'on force le trajet. Il est donc à risque pour ce qui concerne les singularités.

Les paramètres de base d'une commande de déplacement articulaire sont comme suit :

```
MoveC robtarget, robtarget, speeddata, zonedata, tooldata, \wobj:=wobjdata;
```

1. MoveC : Commande indiquant un déplacement circulaire par calcul de trois points (position courante, une valeur intermédiaire et une valeur finale).
2. robtarget (1^{er}) : Valeur venant d'une variable, fonction ou autre qui indique le robtarget intermédiaire par lequel l'outil doit passer.
3. robtarget (2^{e}) : Valeur venant d'une variable, fonction ou autre qui indique la destination finale de l'outil.
4. speeddata : Vitesse de croisière maximale que l'outil peut d'atteindre durant le déplacement. Nous allons expliquer en détail ce paramètre dans les sections suivantes.
5. zonedata : Précision d'arrêt ou d'approche de l'origine du référentiel de l'outil sur un robtarget à atteindre. Nous allons expliquer en détail ce paramètre dans les sections suivantes.
6. tooldata : Le référentiel de l'outil et les caractéristiques physiques de l'outil.
7. wobjdata : Paramètre optionnel qui défini par rapport à quel référentiel de travail on veut positionner le référentiel de l'outil à la pose définie dans le robtarget. Si ce paramètre est omis, le référentiel par défaut est celui de l'atelier (appelée wobj0).

Exemple 4.13 – Lignes de commande d'un déplacement circulaire

```
MoveL p1, v2000, z40, tPince_bout\wobj:=fixture;
MoveC p2,p3, v2000, z40, tPince_bout\wobj:=fixture;
```

Configuration mécanique imposée ou non Quand un robot déplace sont outil vers la pose d'un robtarget, il doit normalement respecter la configuration physique (configuration moteur) imposée par ce robtarget pour atteindre l'objectif. Il est possible d'avoir un problème à définir la bonne configuration physique avec des robtargets calculés ou même pour un point décalé à partir d'un autre. Il est donc possible de spécifier au robot de ne plus tenir compte de cette imposition pour solutionner la pose d'un robtarget, ce qui signifie que le robot choisi lui-même sa configuration mécanique pour atteindre la pose. Afin d'éviter les mouvements imprévus, l'approche abordée dans le contrôleur IRC5 est de faire le plus petit déplacement possible. Ce qui ne laisse pas le robot choisir son sens de rotation des articulations, donc si l'une d'entre elles atteint une limite durant le déplacement, le robot tombe en faute.

Les paramètres de base d'une commande de configuration moteur sur les mouvements de type articulaire :

```
ConfJ \On | \Off;
```

1. ConfJ : Commande définissant si le robot respecte ou non les configurations moteurs imposées durant les mouvements joints.
2. \On | \Off : On, active l'imposition des configurations moteurs. Off, désactive l'imposition des configurations moteurs.

Les paramètres de base d'une commande de configuration moteur sur les mouvements de type linéaire et circulaire :

```
ConfL \On | \Off;
```

1. ConfL : Commande définissant si le robot respecte ou non les configurations moteurs imposées durant les mouvements linéaires et circulaires.
2. \On | \Off : On, active l'imposition des configurations moteurs. Off, désactive l'imposition des configurations moteurs.

À noter qu'il est possible d'éviter les problèmes pour ConfL\On et \Off lors de grand changement de valeurs sur les articulations (exemple une rotation de 270deg sur l'outil) en intégrant des points intermédiaires pour rendre le mouvement de chaque paire d'axes (1+4), (1+6), (3+4) or (3+6) inférieur à 180 degrés lors des déplacements.

4.5.2 Vitesse et précision du déplacement

Vitesse La vitesse est un enregistrement de type speeddata qui s'évalue toujours au niveau du référentiel de l'outil. Il existe dans le contrôleur plusieurs constantes de type speeddata déjà définies (v1, v10, v20, v50, v100, etc.). L'enregistrement de type speeddata est une structure composée des éléments suivants :

1. v_tcp : vitesse linéaire de l'origine du référentiel de l'outil en mm/s ;
2. v_ori : vitesse angulaire du référentiel de l'outil, exprimée en degrés/s ;
3. v_leax : vitesse des articulations prismatiques externes en mm/s ;
4. v_reax : vitesse des articulations rotoïdes externes en mm/s.

Exemple 4.14 – Définir des vitesses

```
! Vitesse pour ependre la colle sans degats.
CONST speeddata vColle:=[56.7, 120, 1000, 500];

MoveL p1, v2000, z40, tPince_bout\wobj:=fixture;
MoveC p2,p3, vColle, z40, tPince_bout\wobj:=fixture;
```

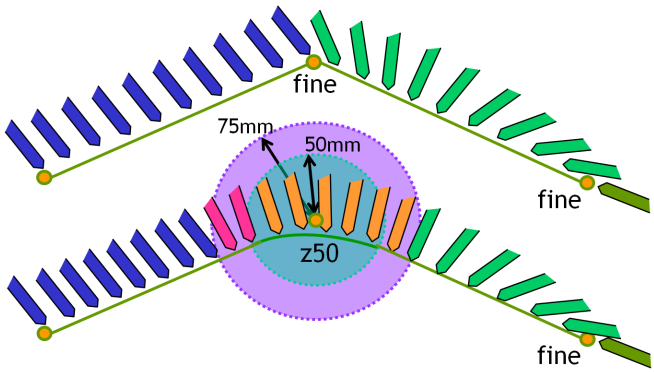


FIGURE 4.4 – Interpolation entre deux déplacements linéaires.

L'interpolation L'interpolation est un enregistrement de type zonedata et définit la tolérance en position à respecter par rapport au robtarget spécifié, dans un déplacement, lorsqu'il y a un autre déplacement tout de suite après. Cette fonctionnalité a pour but d'assurer la fluidité des parcours et d'optimisation les temps de cycles. On peut faire le parallèle avec la conduite d'automobile. En effet, quand un conducteur doit changer de direction dans un milieu piéton, il va décélérer, s'immobiliser complètement, puis tourner. À l'opposé, lorsqu'il prend une sortie d'autoroute, il suivra un arc de cercle lui permettant de ne pas top ralentir.

L'enregistrement de type zonedata est composé de plusieurs valeurs qui définissent le comportement de l'outil lors de l'arrivée dans la zone d'interpolation. Les trois premiers paramètres nous sont utiles au laboratoire(voir la figure 4.4), et les quatre derniers peuvent contenir une valeur arbitraire.

1. finep : Variable booléenne définissant si le robot doit s'arrêter ;
2. pzone_tcp : rayon en mm de la sphère où l'origine du référentiel de l'outil peut commencer l'interpolation vers le prochain trajet ;
3. pzone_ori : rayon en mm de la sphère où l'origine du référentiel doit se trouver pour que l'orientation de l'outil puisse commencer l'interpolation vers le prochain trajet. Ce rayon doit être égal ou plus grand que le rayon précédant ;
4. pzone_eax : à ignorer au laboratoire A-0610 ;
5. zone_ori : à ignorer au laboratoire A-0610 ;
6. zone_leax : à ignorer au laboratoire A-0610.
7. zone_reax : à ignorer au laboratoire A-0610.

Voici enfin un point important en ce qui concerne l'interpolation sur un robot ABB. Quand vous faites faire un déplacement, le paramètre finep égale à false. Le comportement sera le suivant. L'ordinateur calculera le mouvement, puis le transmettra à la carte de mouvement et immédiatement la transmission complétée, il continuera l'exécution des lignes de programme suivantes. Cela signifie pour le programmeur qu'une activation d'une sortie ou une prise de mesure ne se fera pas toujours à l'endroit prévu. Ceci n'est pas vraiment un problème dans la plupart des cas, puisque souvent nous immobilisons le robot lors d'actions sur les sorties. Cependant, si une telle activation doit être faite lors d'un mouvement, il y a des instructions permettant de pallier à ce désagrément.

4.5.3 Fonctions de calculs de robtargets et de poses

Souvent, nous avons besoin de calculer un nouveau robtarget à partir d'un robtaget donné. Il existe deux fonctions à cet effet, dont la première est la plus utilisée.

RelTool Cette fonction retourne un robtarget en faisant une transformation (par exemple, une rotation) par rapport au référentiel d'un robtarget donné. Cette fonction vous sera expliquée en classe, ainsi que plus tard dans ce manuel.

```
r1:=RelTool(robtarget, Dx, Dy, Dz, \Rx, \Ry, \Rz);
```

1. RelTool : Fonction qui retourne un robtarget calculé à partir du système d'axes d'un robtarget donné, mais exprimé dans le référentiel du robtarget donné ;
2. robtarget : valeur à partir duquel on trouve un nouveau robtarget ;
3. Dx : déplacement en mm dans l'axe x du robtarget donné ;
4. Dy : déplacement en mm dans l'axe y du robtarget donné ;
5. Dz : déplacement en mm dans l'axe z du robtarget donné ;
6. \Rx : rotation en degré autour de l'axe x du robtarget donné.
7. \Ry : rotation en degré autour de l'axe y du robtarget donné.
8. \Rz : rotation en degré autour de l'axe z du robtarget donné.

Si deux ou trois rotations sont spécifiées en même temps, elles sont effectuées tout d'abord autour de l'axe x, puis autour du nouvel axe y et ensuite autour du nouvel axe z.

Offs Cette fonction retourne un robtarget décalé dans le repère du robtarget donné. Cette fonction vous sera expliquée en classe, ainsi que plus tard dans ce manuel.

```
r1:=Offs(robtarget, Dx, Dy, Dz);
```

1. Offs : Fonction qui retourne un robtarget décalé dans le référentiel du robtarget donné ;
2. robtarget : valeur à partir duquel on trouve un nouveau robtarget ;
3. Dx : déplacement en mm dans l'axe x du repère ;
4. Dy : déplacement en mm dans l'axe y du repère ;
5. Dz : déplacement en mm dans l'axe z du repère ;

DefFrame Cette fonction retourne une pose à partir de l'origine des trois robtargets. Il faut comprendre que lors de l'enseignement de robtargets, l'humain arrive à avoir une erreur relativement faible au niveau du positionnement en translation, mais que l'orientation est souvent de piètre qualité. Donc la fonction « DefFrame » définit une pose avec avec orientation contrôlée à partir de la translation des robtargets.

```
r1:=DefFrame(robtarget (1), robtarget (2), robtarget (3),\Origin:=[1,2 ou 3]);
```

1. Offs : Fonction qui retourne une pose à partir de trois robtargets ;
2. robtarget (1) : premier robtarget ;
3. robtarget (2) : deuxième robtarget ;
4. robtarget (3) : troisième robtarget ;
5. \Origin : paramètre optionnel définissant la méthode utilisée pour solutionner la pose résultante ;
   (a) \Origin :=1 ou paramètre omis : Le premier robtarget défini l'origine de nouveau système d'axe, l'origine du robtarget 2 est nécessairement sur le vecteur x de la pose calculée et le troisième robtarget positionne le plan XY (Y positif) ;

(b) \Origin :=2 : Le premier robtarget est sur l'axe x négatif de la pose calculée, l'origine du robtarget 2 est l'origine de la pose calculée et le troisième robtarget positionne le plan XY (Y positif) ;

(c) \Origin :=3 : (méthode utilisée lors de l'enseignement avec la méthode de 3 points dans le boîtier de commande) Le premier robtarget est sur l'axe x proche de l'origine de la pose calculée, l'origine du robtarget 2 est sur l'axe x de la pose calculée, mais loin de l'origine et le troisième robtarget positionne le vecteur Y positif qui sera perpendiculaire à l'axe x, fixant par le même fait l'origine de la pose calculée.

4.6 Interaction avec les entrées/sorties

4.6.1 Lecture d'états

Il existe plusieurs fonctions permettant de lire l'état d'une entrée ou d'une sortie. Nous allons présenter ici celles qui peuvent aider dans la réalisation des projets du cours. Pour une entrée digitale (active ou non) il y a deux possibilités :

- la fonction TestDI : Cette fonction vous retourne true ou false.
- L'opérateur = : L'opérateur ne fonctionne pas partout (loin d'être homogène), il permet de vérifier si une entrée digitale est égale à 1 ou à 0.

Pour une sortie digitale (active ou non) il y a une possibilité.

- La fonction Doutput : La fonction retourne une valeur de type dionum et celle-ci est compatible pour une comparaison avec une valeur numérique.

4.6.2 Assignation d'états

Il existe plusieurs instructions pour assigner une sortie digitale.

- L'instruction SET : L'instruction active la sortie spécifiée en paramètre.
- L'instruction RESET : L'instruction désactive la sortie spécifiée en paramètre.
- L'instruction SetDO : L'instruction assigne la valeur spécifiée en second paramètre dans la sortie spécifiée au premier paramètre. Cette instruction offre des paramètres optionnels permettant un meilleur contrôle de l'assignation programmée.
- L'instruction PulseDO : L'instruction active la sortie pour un certain temps avant de la remettre inactive. Le système n'attend pas la fin du délai pour continuer le déroulement du programme.

4.7 Attente d'évènements

Puisque le robot doit continuellement interagir avec le monde extérieur, il normal d'avoir à attendre des informations avant de continuer le déroulement du code. La plupart des instructions d'attente offrent une option de temps maximum permettant au programmeur de gérer les cas anormales.

- L'instruction WaitTime : L'instruction arrête le déroulement du programme pour un certain temps défini en secondes.
- L'instruction WaitDI : L'instruction arrête le déroulement du programme tant qu'un capteur d'entrée n'est pas à l'état souhaité (second paramètre de l'instruction).
- L'instruction WaitDO : L'instruction arrête le déroulement du programme tant que l'actionneur n'est pas dans l'action souhaité (second paramètre de l'instruction).
- L'instruction WaitUntil : L'instruction arrête le déroulement du programme tant que le résultat de l'équation booléenne n'est pas vrai.

4.8 Message à l'opérateur

TPWrite Cette instruction permet d'afficher une chaîne de caractères d'un maximum de 80 caractères. Il est important de noter qu'un boîtier de commande affiche un maximum de 40 caractères par ligne et qu'il ne fait aucun formatage. Donc, afin d'éviter de couper des mots assurez-vous de manuellement faire votre mise-en-page. Le retour de chariot est automatiquement fait après 40 caractères.

TPErase Cette instruction permet d'effacer l'écran usager.

TPReadFK Cette instruction permet de poser une question dont le choix de réponse est limité à cinq boutons dont le texte est configurable. La valeur retournée est une valeur de 1 à 5 représentant la position du bouton choisi.

TPReadNum Cette instruction permet de poser une question dont le choix de réponse est une valeur numérique.

4.9 Liste des instructions et fonctions potentiellement utiles en laboratoire

Les tableaux 4.3 et 4.4 donnent respectivement les instructions et les fonctions potentiellement utiles dans le cadre de ce cours. Les noms des instructions et fonctions les plus importantes sont en gras.

Tableau 4.3 – Liste d'instructions utiles.

Instruction	Description
AccSet	Réduction de l'accélération
Add	Ajout d'une valeur numérique
AliasIO	Définition d'un signal d'entrée/sortie avec un nom d'alias
:=	Affecte une valeur
BitClear	Suppression d'un bit donné dans une donnée byte
BitSet	Définition d'un bit donné dans une donnée byte
BookErrNo	Réservation d'un numéro d'erreur du système RAPID
Break	Arrêt de l'exécution du programme
CallByVar	Appel de procédure par une variable
CirPathMode	Réorientation de l'outil pour le chemin circulaire
Clear	Suppression de la valeur
ClkReset	Réinitialisation d'une horloge utilisée pour le minutage
ClkStart	Démarrage d'une horloge utilisée pour le minutage
ClkStop	Arrêt d'une horloge utilisée pour le minutage
Comment	Commentaire
Compact IF	Si une condition est remplie, alors... (une seule instruction)
ConfJ	Contrôle de la configuration au cours des mouvements articulaires subséquents
ConfL	Contrôle de la configuration au cours des mouvements linéaires subséquents
CONNECT	Association d'une interruption à une routine d'interruption
Decr	Décrémentation de 1

suite à la page suivante...

Instruction *suite*	**Description** *suite*
ErrLog	Écriture d'un message d'erreur
ErrRaise	Écriture d'un avertissement et appel d'un gestionnaire d'erreurs
ErrWrite	Écriture d'un message d'erreur
EXIT	Met fin l'exécution du programme
ExitCycle	Arrêt du cycle actuel et démarrage du suivant
FOR	Répète un nombre donné de fois
GetDataVal	Obtention de la valeur d'un objet de données
GetSysData	Obtention des données système
GetTrapdata	Obtention de données d'interruption pour la routine TRAP actuelle
GOTO	Passage à une nouvelle instruction
IDelete	Annulation d'une interruption
IDisable	Désactivation des interruptions
IEnable	Activation des interruptions
IError	Demande une interruption en cas d'erreur
IF	Si une condition est remplie, alors, ... sinon, ...
Incr	Incrémentation de 1
IndAMove	Mouvement de position absolue indépendant
IndCMove	Mouvement continu indépendant
IndDMove	Mouvement de position delta indépendant
IndReset	Réinitialisation indépendante
IndRMove	Mouvement indépendant des positions relatives
InvertDO	Inversement de la valeur d'un signal de sortie numérique
IPers	Interruption lors de la modification d'une variable persistante
ISignalDI	Commande d'interruptions à partir d'un signal d'entrée numérique
ISignalDO	Interruption à partir d'un signal de sortie numérique
ISignalGI	Commande d'une interruption à partir d'un groupe de signaux d'entrée numériques
IsignalGO	Commande d'une interruption à partir d'un groupe de signaux de sortie numériques
ISleep	Désactivation d'une interruption
ITimer	Commande d'une interruption chronométrée
IVarValue	Commande d'interruption d'une valeur variable
IWatch	Activation d'une interruption
Label	Nom de ligne
MoveAbsJ	Déplacement du robot vers une position articulaire absolue
MoveC	Déplacement circulaire du robot
MoveCDO	Mouvement circulaire du robot et définition d'une sortie numérique dans le raccordement
MoveCSync	Déplacement circulaire du robot et exécution d'une procédure RAPID
MoveJ	Déplacement du robot selon un mouvement articulaire
MoveJDO	Mouvement du robot par mouvement articulaire et définition d'une sortie numérique dans le raccordement
MoveJSync	Déplacement du robot selon un mouvement articulaire et exécution d'une procédure RAPID

suite à la page suivante...

Instruction *suite*	**Description** *suite*
MoveL	Mouvement linéaire du robot
MoveLDO	Mouvement linéaire du robot et définition d'une sortie numérique dans le raccordement
MoveLSync	Mouvement linéaire du robot et exécution d'une procédure RAPID
PathRecStart	Lancement de l'enregistreur de trajectoire
PathRecStop	Arrêt de l'enregistreur de trajectoire
PathResol	changement de la résolution de la trajectoire
PDispOff	désactivation du déplacement d'un programme
PDispOn	Activation du déplacement de programme
PDispSet	Activation du déplacement d'un programme en utilisant un référentiel connu
ProcCall	Appel d'une nouvelle procédure
PulseDO	Génération d'une impulsion sur un signal de sortie numérique
RAISE	Appel d'un gestionnaire d'erreurs
RaiseToUser	Transmission d'une erreur au niveau utilisateur
Reset	Réinitialisation d'un signal de sortie numérique
ResetRetryCount	Réinitialise le nombre de nouvelles tentatives
RestoPath	Rétablissement de la trajectoire après une interruption
RETRY	Redémarrage après une erreur
RETURN	Fin de l'exécution d'une routine
SearchC	Recherche en cercle en utilisant le robot
SearchL	Recherche linéaire à l'aide du robot
Set	Configuration d'un signal de sortie numérique
SetDO	Modifie la valeur d'un signal de sortie numérique
SetGO	Modifie la valeur d'un groupe de signaux de sortie numériques
SetSysData	Paramétrage des données système
SingArea	Définition de l'interpolation autour de chaque point
SpeedRefresh	Mettre à jour la vitesse générale en cours de déplacement
SpyStart	Démarrage de l'enregistrement des données de temps d'exécution
SpyStop	Arrêt de l'enregistrement des données temporelles d'exécution
StartMove	Relance du mouvement du robot
StartMoveRetry	Relance du mouvement du robot et de l'exécution
StepBwdPath	Recul d'un pas sur la trajectoire
Stop	Arrêt de l'exécution du programme
StopMove	Arrêt du mouvement du robot
StopMoveReset	Réinitialisation de l'état d'arrêt des mouvements système
StorePath	Enregistrement de la trajectoire lorsqu'une interruption survient
TEST	En fonction de la valeur d'une expression ...
TestSignDefine	Définition d'un signal de test
TestSignReset	Remise à zéro de toutes les définitions des signaux de test
TextTabInstall	Installation d'un tableau de texte
TPErase	Effacement du texte imprimé sur le FlexPendant
TPReadDnum	Lit un numéro sur le FlexPendant
TPReadFK	Lecture des touches de fonction
TPReadNum	Lecture d'un numéro sur le FlexPendant
TPShow	Fenêtre de commutation sur le FlexPendant

suite à la page suivante...

Instruction *suite*	**Description** *suite*
TPWrite	Écriture sur le FlexPendant
TriggC	Mouvement circulaire du robot avec événement
TriggIO	Définition d'un événement d'entrée/sortie de durée ou de position fixe à proximité d'un point d'arrêt
TriggJ	Déplacements du robot en mode articulaire avec événement
TriggL	Mouvement linéaire du robot avec événement

Tableau 4.4 – Liste de fonctions utiles.

Fonction	**Description**
Abs	Calcule de la valeur absolue
ACos	Calcule la valeur d'arc cosinus
ASin	Calcule la valeur d'arc sinus
ATan	Calcule la valeur d'arc tangente
ATan2	Calcule la valeur d'arc tangente2
BitAnd	Opération logique AND au niveau du bit sur les données d'octet
BitCheck	Vérifie si un bit spécifié dans une donnée byte est défini
BitLSh	Opération logique LEFT SHIFT au niveau du bit sur l'octet
BitNeg	Opération logique NEGATION au niveau du bit sur les données d'octet
BitOr	Opération logique OR au niveau du bit sur les données d'octet
BitRSh	Opération logique RIGHT SHIFT au niveau du bit sur les données d'octet
BitXOr	Opération logique XOR au niveau du bit sur les données d'octet
ByteToStr	Conversion d'un octet en données de chaîne de caractères
CalcJointT	Calcule les valeurs articulaires (un jointtarget) à partir d'un robtarget
CalcRobT	Calcule un robtarget à partir d'un jointtarget
CDate	Lecture de la date en cours sous forme de chaîne de caractères
CJointT	Lecture des angles articulaires actuels
ClkRead	Lit une horloge utilisée pour le minutage
Cos	Calcule la valeur de cosinus
CPos	Lecture des données de position actuelle d'un outil par rapport à un workobject
CRobT	Lecture du robtarget actuel d'un outil par rapport à un workobject
CTime	Lecture de l'heure en cours sous forme de chaîne de caractères
CTool	Lecture des données de l'outil actuel
CWObj	Lecture des données de l'objet de travail actuel
DefFrame	Définition d'un référentiel
Dim	Obtention de la taille d'un tableau
Distance	Distance entre deux points
DnumToNum	Convertit dnum en num
DotProd	Produit scalaire de deux vecteurs de type pos
DOutput	Lecture de la valeur d'un signal de sortie numérique
EulerZYX	Obtention des angles d'Euler à partir d'un quaternion
Exp	Calcul de la valeur exponentielle

suite à la page suivante...

Fonction *suite*	**Description** *suite*
GetNextSym	Obtention du symbole correspondant suivant
GetSysInfo	Obtention des informations concernant le système
GetTaskName	Obtention du nom et du numéro de la tâche actuelle
GetTime	Lecture de l'heure actuelle comme une valeur numérique
IsPers	Persistant
IsSysId	Test de l'identificateur système
IsVar	Variable
MirPos	Réflexion d'une position
NOrient	Normalisation d'un quaternion
NumToDnum	Conversion d'un num en un dnum
NumToStr	Conversion de la valeur numérique en chaîne de caractères
Offs	Modification de la partie position d'un robtarget
OpMode	Lecture du mode de fonctionnement
OrientZYX	Création d'une orientation à partir des angles d'Euler
PoseInv	Inversion des données de pose
PoseMult	multiplication des données de pose
PoseVect	Application d'une transformation à un vecteur
Pow	Calcule la fonction puissance d'une valeur
Present	Test d'utilisation d'un paramètre facultatif
ReadMotor	Lecture des angles du moteur actuel
ReadNum	Lecture d'un nombre à partir d'un fichier ou d'un canal en série
ReadStr	Lecture d'une chaîne de caractères à partir d'un fichier ou d'un canal en série
ReadStrBin	Lecture d'une chaîne de caractères à partir d'un fichier ou d'un canal en série binaire
ReadVar	Lecture d'une variable à partir d'un dispositif
RelTool	Réalisation d'un déplacement par rapport à l'outil
RemainingRetries	Nombre de nouvelles tentatives qu'il est encore possible d'effectuer
RobName	Obtention du nom du robot TCP
Round	Arrondi d'une valeur numérique
RunMode	Lecture du mode de fonctionnement
Sin	Calcule la valeur de sinus
Sqrt	Calcule la valeur de la racine carrée
StrDigCalc	Opérations arithmétiques avec type de données stringdig
StrDigCmp	Comparaison de deux chaînes de caractères ne comportant que des chiffres
StrFind	Recherche d'un caractère dans une chaîne de caractères
StrLen	Obtention de la longueur de la chaîne de caractères
StrMap	Conversion d'une chaîne de caractères
StrMatch	Recherche d'une séquence dans la chaîne de caractères
StrMemb	Vérification d'appartenance d'un caractère à une séquence
StrOrder	Vérification si les chaînes de caractères sont ordonnées
StrPart	Recherche d'une partie d'une chaîne de caractères
StrToByte	Conversion d'une chaîne de caractères en données d'octet
StrToVal	Conversion d'une chaîne de caractères en valeur
Tan	Calcule la valeur de la tangente

suite à la page suivante...

Fonction *suite*	**Description** *suite*
TestAndSet	Test de la variable et configuration en cas de besoin
TestDI	Test de définition d'une entrée numérique
TestSignRead	Lecture de la valeur du signal de test
Trunc	Valeur numérique tronquée
Type	Renvoie le nom du type de donnée pour une variable
UIAlphaEntry	Entrée alpha interaction utilisateur
IClientExist	Client interaction utilisateur existant
UIDnumEntry	Entrée numérique utilisateur
UIDnumTune	Ajustement de la valeur numérique interaction utilisateur
UIListView	Vue liste des interactions utilisateur
UIMessageBox	Type avancé de boîte de message utilisateur
UINumEntry	Entrée numérique utilisateur
UINumTune	Ajustement de la valeur numérique interaction utilisateur
ValToStr	Conversion d'une valeur en chaîne de caractères
VectMagn	Longueur d'un vecteur

4.10 Exercices

4.1. Faire un programme qui simule le chargement d'une barquette de transport. La prise sera à l'endroit du porte-crayon et le dépôt se fera dans la grille de trous.
- Le robot devra partir d'un robtarget de repos et y retourner à la fin.
- L'outil tCrayon est correctement défini.
- Les vitesses de déplacement générales sont à 5 m/s, les approches et retraits à 500 mm/s.
- Vu que la barquette n'est pas nécessairement toujours au même endroit, il est important que le système s'ajuste rapidement sans avoir à ré-enseigner les 64 robtarget de dépôt. Le technicien doit pouvoir redémarrer le système avec un minimum de trois robtarget à enseignér.

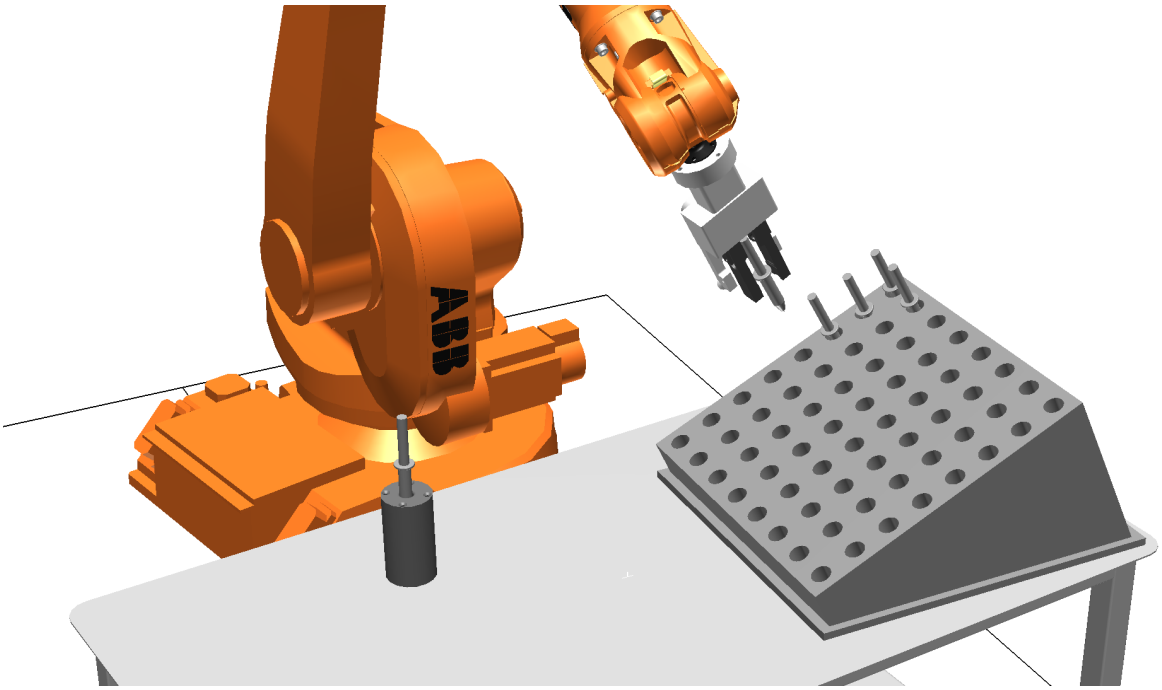


FIGURE 4.5 – Exercice 4.1 .

4.2. Ajouter une ou des routines au programme de l'exercice précédent, afin de permettre au technicien de rapidement repartir le système quand la grille est déplacée. Un menu demandera au début du programme si l'on exécute la séquence de production ou si on ajuste le référentiel. Quand l'on choisit d'ajuster, un écran proposera un menu qui guidera l'opérateur, le programme ira placer le robot près du premier robtarget définissant le référentiel (légèrement au-dessus) et arrêtera le programme. Le technicien pourra alors piloter le robot manuellement afin d'ajuster le robot au bon endroit. Par la suite, il pourra relancer le programme. Le programme mettra alors à jour le robtarget en capturant la nouvelle valeur. Ensuite, le programme déplacera le robot au 2e point et/ou 3e point pour refaire l'exercice de mise à jour. Quand tous les points seront mis à jour, le système recalculera le référentiel et se mettra en production.

4.3. Faire un programme qui permettra à un robot de suivre le contour d'un carré (figure 4.6) sur la table (à noter que l'on considère la surface de la table parallèle à la base du robot). Afin de définir la longueur et l'orientation du bloc deux robtarget rob1 et rob2 sont enseignés à l'extrémité de l'arrête la plus proche de l'axe x du référentielle du Wobj0. (Les notions vues dans le chapitre 5 faciliteront la solutions)
- Le robot devra partir d'un robtarget de repos et y retourner à la fin.
- L'outil tCrayon est correctement défini comme sur la figure.
- Les vitesses de déplacement sont à 1 m/s et le parcours sur la figure ainsi que les approches et retraits à 100 mm/s.

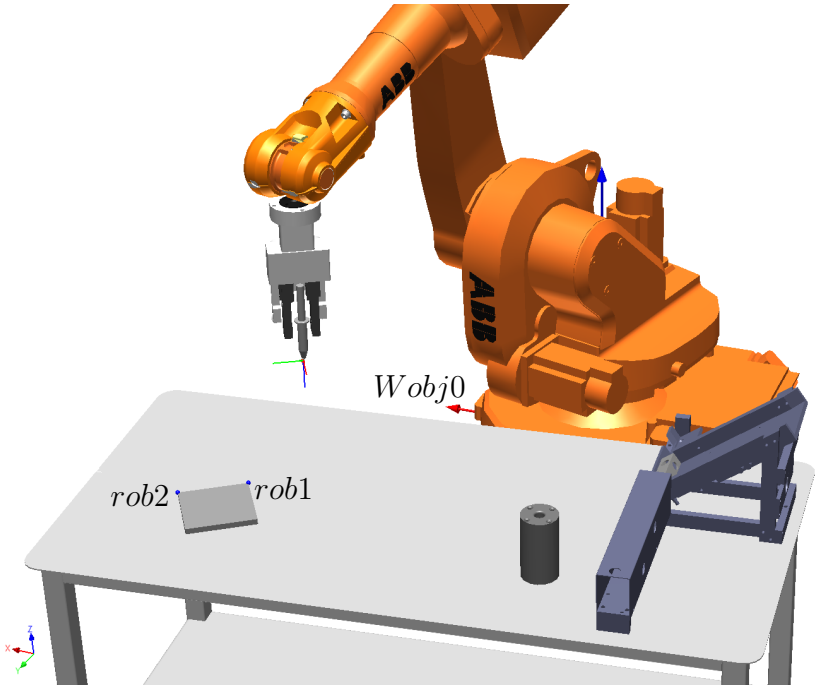


FIGURE 4.6 – Exercice 4.3.

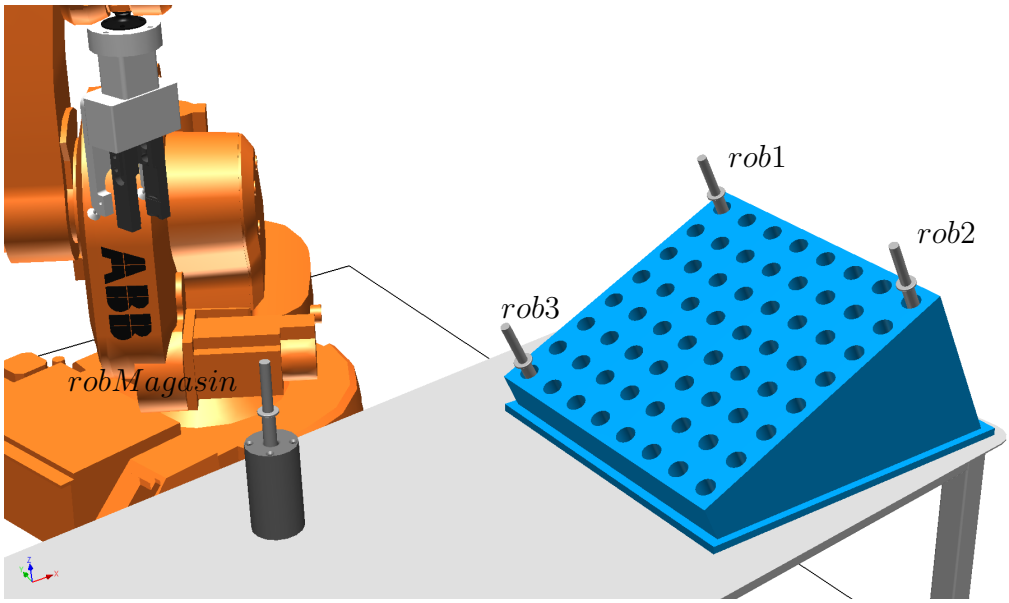


FIGURE 4.7 – Exercice 4.1.

4.11 Réponses aux exercices

4.1.

Exemple 4.15 – Code RAPID pour l'exercice 4.1 - Solution 1 (à l'aide de RelTool)

```
MODULE Ex3_1_solution

 ! trois robtarget pour definir un robtarget correctement orientee sur le plan
 PERS robtarget rob1 := [[256.87,-834.24,522.49],[0.140914,-0.610945,0.758012,-0.179745],[-1,0,-1,0],
                         [9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
 PERS robtarget rob2 := [[179.85,-1173.64,523.25],[0.140914,-0.610945,0.758012,-0.179745],[-1,0,-1,0],
                         [9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
 PERS robtarget rob3 := [[-53.02,-767.35,371.78],[0.140914,-0.610945,0.758012,-0.179745],[-2,0,-2,0],
                         [9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
 ! robtarget correctement alignee
 PERS robtarget robTrouRectifie:=[[256.87,-834.24,522.49],[0.136496,-0.608866,0.762131,-0.172648],
                                  [-1,0,-1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
 ! Outil de travail
 PERS tooldata tCrayon := [TRUE,[[0,0,283.37],[0.707107,0,0,-0.707107]],[1.7,[12.2,0,158],[1,0,0,0],
                           0.009,0.003,0.012]];
 ! robtarget de repos
 CONST robtarget robRepos := [[-460.80,-506.08,629.91],[0.0057914,0.730498,-0.682877,0.0042632],
                              [-2,0,-2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
 ! robtarget de prise
 CONST robtarget robMagasin := [[-460.80,-706.08,429.91],[0.0057914,0.730498,-0.682877,0.0042632],
                                [-2,0,-2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
 ! Distance pour approche/retrait
 CONST num Decalage := -125;
 ! nombre de pas a faire sur une ligne ou colonne
 CONST num QteDePas := 7;
 !Distance des pas en X et Y
 PERS  num PasEnX := 49.7186;
 PERS  num PasEnY := 50.1466;

PROC main()
  Pince\Ouvert;
  Calcul;
  ! Changement de colonne
  FOR i FROM QteDePas to 0 DO
   ! Changement de ligne
   FOR j FROM 0 to QteDePas DO
 Prise;
 Depot RelTool(robTrouRectifie, i*PasEnX, j*PasEnY,0);
```

```
   ENDFOR
  ENDFOR
  ! retour a position de repos
  MoveJ robRepos, v1000, fine, tCrayon;
ENDPROC

PROC Calcul()
  VAR pose pTmp;

  ! Calcul le pas de deplacmeent en XY
  PasEnX:=distance(rob2.trans,rob1.trans) / QteDePas;
  PasEnY:=distance(rob3.trans,rob1.trans) / QteDePas;
  ! defini le premier point avec une orientation calculee
  pTmp:=DefFrame(rob1,rob2,rob3);
  ! recupere valeur confdata et ExternalAxe
  robTrouRectifie := rob1;
  ! assigne les valeurs de la matrice
  robTrouRectifie.trans := pTmp.trans;
  robTrouRectifie.rot := pTmp.rot;

ENDPROC

PROC Prise()
 ! Approche
 MoveJ RelTool(robMagasin,0,0,Decalage), v5000, z50, tCrayon;
 ! Position d prise
 MoveL robMagasin, v500, fine, tCrayon;
 !Fermer pince
 Pince\Fermer;
 !Degagement
 MoveL RelTool(robMagasin,0,0,Decalage), v5000, z50, tCrayon;

ENDPROC

PROC Depot(robtarget robDepot)
 ! Approche
 MoveJ RelTool(robDepot,0,0,Decalage), v5000, z50, tCrayon;
 ! Position Depot
 MoveL robDepot, v500, fine, tCrayon;
 !Ouvrir pince
 Pince\Ouvert;
 ! Degaement
 MoveL RelTool(robDepot,0,0,Decalage), v5000, z50, tCrayon;
ENDPROC

PROC Pince(\switch Ouvert | switch Fermer)
 ! Action sur la pince
 IF Present(Fermer) then
  ! verifie si la sortie doit changer d'etat
  IF Doutput(DO01_EE_PINCE01)=0 then
    Set DO01_EE_PINCE01; ! Ferme la pince
 WaitTime 0.1;  ! Attend que la pince bouge physiquement
  endif
 ELSE
  ! verifie si la sortie doit changer d'etat
  IF Doutput(DO01_EE_PINCE01)=1 then
    Reset DO01_EE_PINCE01; ! Ferme la pince
 WaitTime 0.1;    ! Attend que la pince bouge physiquement
  endif
 ENDIF
ENDPROC

ENDMODULE
```

Exemple 4.16 – Code RAPID pour l'exercice 4.1 - Solution 2 (à l'aide d'un workobject)

```
MODULE Ex3_2b_solution

 ! trois robtarget pour definir un robtarget correctement orientee sur le plan
 PERS robtarget rob1 := [[-53.02,-767.35,371.78],[0.140914,-0.610945,0.758012,-0.179745],[-2,0,-2,0],
                        [9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
 PERS robtarget rob2 := [[-127.34,-1106.92,373.87],[0.14091,-0.610946,0.758012,-0.179742],[-2,0,-2,0],
                        [9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
```

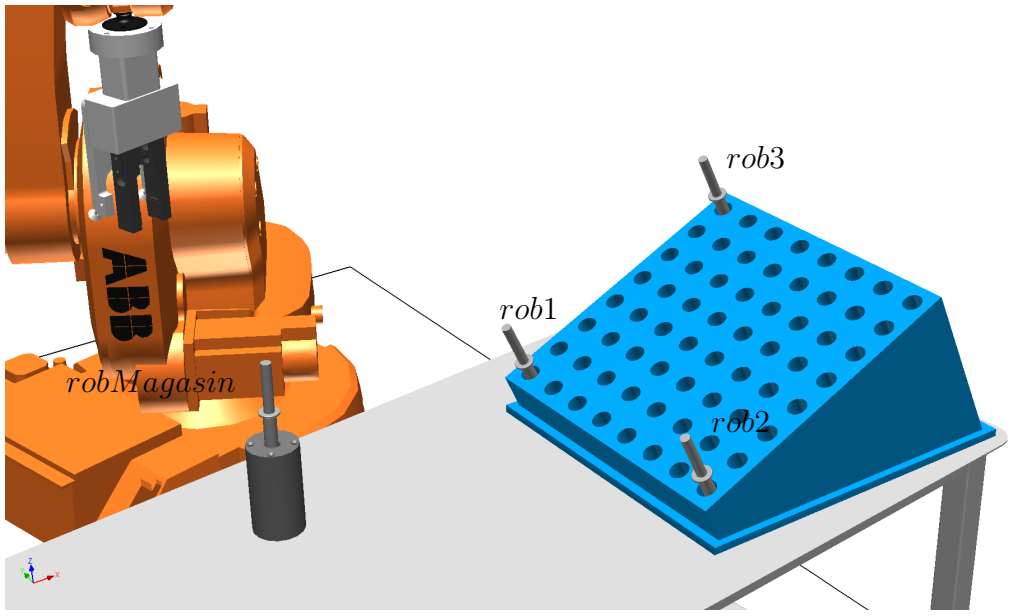


FIGURE 4.8 – Exercice 4.1.

```
PERS robtarget rob3 := [[256.87,-834.24,522.49],[0.140914,-0.610945,0.758012,-0.179745],[-1,0,-1,0],
                        [9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
PERS wobjdata wPlan := [FALSE, TRUE, "", [[-53.02,-767.35,371.78],
                        [0.612117,0.135693,-0.173281,-0.759522]],[[0, 0, 0],[1, 0, 0, 0]]];

! robtarget correctement alignee
PERS robtarget robTrouRectifie := [[0,0,0],[4.37114E-08,-1,0,0],[-2,0,-2,0],
                                   [9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
! Outil de travail
PERS tooldata tCrayon := [TRUE,[[0,0,283.37],[0.707107,0,0,-0.707107]],[1.7,[12.2,0,158],[1,0,0,0],
                          0.009,0.003,0.012]];
! robtarget de repos
CONST robtarget robRepos := [[-460.80,-506.08,629.91],[0.0057914,0.730498,-0.682877,0.0042632],
                             [-2,0,-2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
! robtarget de prise
CONST robtarget robMagasin := [[-460.80,-706.08,429.91],[0.0057914,0.730498,-0.682877,0.0042632],
                               [-2,0,-2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
! Distance pour approche/retrait
CONST num Decalage := -125;
! nombre de pas a faire sur une ligne ou colonne
CONST num QteDePas := 7;
!Distance des pas en X et Y
PERS  num PasEnX := 49.6592;
PERS  num PasEnY := 50.1466;

PROC main()
  Pince\Ouvert;
  Calcul;
  ! Changement de colonne
  FOR i FROM QteDePas to 0 DO
   ! Changement de ligne
   FOR j FROM QteDePas to 0 DO
 Prise;
 Depot Offs(robTrouRectifie, i*PasEnX, j*PasEnY,0);
   ENDFOR
  ENDFOR
  ! retour a position de repos
  MoveJ robRepos, v1000, fine, tCrayon;
ENDPROC

PROC Calcul()

  ! Calcul le pas de deplacement en XY
  PasEnX:=distance(rob2.trans,rob1.trans) / QteDePas;
  PasEnY:=distance(rob3.trans,rob1.trans) / QteDePas;
  ! definit le plan de travail
```

```
  wPlan.uframe:=DefFrame(rob1,rob2,rob3);
  ! recupere valeur confdata et ExternalAxe
  robTrouRectifie := rob1;
  ! assigne les valeurs de la matrice
  robTrouRectifie.trans := [0,0,0];
  robTrouRectifie.rot := OrientZYX(0,0,180);

ENDPROC

PROC Prise()
 ! Approche
 MoveJ RelTool(robMagasin,0,0,Decalage), v5000, z50, tCrayon;
 ! Position d prise
 MoveL robMagasin, v500, fine, tCrayon;
 !Fermer pince
 Pince\Fermer;
 !Degaement
 MoveL RelTool(robMagasin,0,0,Decalage), v5000, z50, tCrayon;
ENDPROC

PROC Depot(robtarget robDepot)
 ! Approche
 MoveJ RelTool(robDepot,0,0,Decalage), v5000, z50, tCrayon\wobj:=wplan;
 ! Position Depot
 MoveL robDepot, v500, fine, tCrayon\wobj:=wplan;
 !Ouvrir pince
 Pince\Ouvert;
 ! Degaement
 MoveL RelTool(robDepot,0,0,Decalage), v5000, z50, tCrayon\wobj:=wplan;
ENDPROC

PROC Pince(\switch Ouvert | switch Fermer)
 ! Action sur la pince
 IF Present(Fermer) then
  ! verifie si la sortie doit-etre change d'etat
  IF Doutput(DO01_EE_PINCE01)=0 then
    Set DO01_EE_PINCE01; ! Ferme la pince
 WaitTime 0.1;  ! Attend que la pince bouge physiquement
  endif
 ELSE
  ! verifie si la sortie doit-etre change d'etat
  IF Doutput(DO01_EE_PINCE01)=1 then
    Reset DO01_EE_PINCE01; ! Ferme la pince
 WaitTime 0.1;    ! Attend que la pince bouge physiquement
  endif
 ENDIF
ENDPROC

ENDMODULE
```

4.2.

Exemple 4.17 – Code RAPID pour l'exercice 4.2

```
MODULE Ex3_3_Solution

PROC Menu()
 VAR num rep;
 ! demande le choix a l'operateur
 TPErase;
 TPReadFK rep, "Le robot demarre dans quelle mode?","Production", "", "", "", "Calibration";
 ! passe en mode calibration sinon va directe en production
 IF (rep=5) then
    Calib;
 ENDIF
ENDPROC

PROC Calib()
 VAR num rep;
 TPErase;
 TPReadFK rep, "Enseignement avec crayon?","", "", "", "non", "oui";
 ! passe en mode calibration sinon va directe en production
 IF (rep=5) then
```

```
  pince\Ouvert;
    Prise;
 ENDIF

 RobEnseigne rob1,1;
 RobEnseigne rob2,2;
 RobEnseigne rob3,3;

 ! retour a position de repos
 MoveJ robRepos, v1000, fine, tCrayon;
 TPReadFK rep, "Ouverture de la pince","", "", "", "", "OK";
 Pince\Ouvert;

ENDPROC

PROC RobEnseigne(INOUT robtarget robTmp, num numero)
 VAR num rep;
 VAR string phrase;

 TPerase;
 phrase:="Deplacement au-dessus du point #" + numtostr(numero,0);
 TPReadFK rep, phrase,"", "", "", "", "OK";
 !Deplacement au point
 MoveJ RelTool(robTmp,0,0,-200), v100, z10, tCrayon\wobj:=wobj0;
 MoveL RelTool(robTmp,0,0,-50), v50, fine, tCrayon\wobj:=wobj0;
 ! arret du robot sans replacement automatique
 TPReadFK rep,"Appuyez sur OK et positionnez le robot." ,"", "", "", "", "OK";
 STOP\NoRegain;
 ! ici l'operateur Place le robot
 robTmp:=CRobt(\tool:=TCrayon,\wobj:=Wobj0);
 MoveL RelTool(robTmp,0,0,-200), v100, z10, tCrayon\wobj:=wobj0;

ENDPROC

PROC CALCUL()

ENDPROC

ENDMODULE
```

4.3.

Exemple 4.18 – Code RAPID pour l'exercice 4.3

```
MODULE Ex3_3_solution

 PERS tooldata tCrayon := [TRUE,[[0,0,283.37],[0.707107,0,0,-0.707107]],[1.7,[12.2,0,158],
                           [1,0,0,0],0.009,0.003,0.012]];
 CONST robtarget robRepos :=
     [[227.90,881.05,588.08],[0.0023277,0.489786,0.871809,0.00737201],[0,-1,-2,0],
                              [9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
 CONST robtarget rob1 := [[321.12,938.67,362.59],[0.00232984,0.489783,0.87181,0.00737314],[0,-1,-2,0],
                          [9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
 CONST robtarget rob2 := [[420.27,1006.41,363.33],[0.00234821,0.489748,0.87183,0.00738166],[0,-1,-2,0],
                          [9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];

 VAR  robtarget robTrv1;
 VAR  robtarget robTrv2;
 VAR  robtarget robTrv3;
 VAR  robtarget robTrv4;

  VAR num cote;
  VAR num angle;

PROC main()
  !procedure de calcul
  Calcul;
  ! approche
  MoveJ RelTool(rob1,0,0,-200), v1000, z50, tCrayon;
  ConfL\Off;
  ! Parcours
  MoveL robTrv1, v100, fine, tCrayon;
  MoveL robTrv2, v100, fine, tCrayon;
  MoveL robTrv3, v100, fine, tCrayon;
```

```
  MoveL robTrv4, v100, fine, tCrayon;
  MoveL robTrv1, v100, fine, tCrayon;
  ConfL\On;
  ! degagement
  MoveL RelTool(rob1,0,0,-200), v200, fine, tCrayon;
  ! retour a repos
  MoveJ robRepos , v1000, fine, tCrayon;
ENDPROC

PROC Calcul()
  ! valeurs necessaire pour calcul
  cote := distance(rob1.trans,rob2.trans);
  angle := atan2((rob2.trans.y-rob1.trans.y),(rob2.trans.x-rob1.trans.x));
  ! defini le premier point avec une orientation calculee
  robTrv1 := rob1;
  robTrv1.rot := OrientZYX(angle,0,180);
  ! relatif au premier point puisque l'orientation est controle
  robTrv2 := reltool(robTrv1,cote,0,0);
  robTrv3 := reltool(robTrv1,cote,-cote,0);
  robTrv4 := reltool(robTrv1,0,-cote,0);
ENDPROC

ENDMODULE
```

Chapitre 5

Transformations de coordonnées

Pour réussir le cours GPA546, vous devez maîtriser la trigonométrie ainsi que le calcul vectoriel. Ce chapitre fait une brève révision de ces deux concepts et introduit la notion de matrice de rotation et de transformations de coordonnées.

Dans ce cours, nous allons utiliser le terme *référentiel* comme synonyme des termes *système de coordonnées cartésiennes* et *repère cartésien*. Nous allons utiliser uniquement des *référentiels directs*, c'est-à-dire des référentiels qui respectent la règle de la main droite.

5.1 Trigonométrie et calcul vectoriel dans le plan

Plutôt que donner une liste de formules trigonométriques et de règles de calcul vectoriel, nous allons faire une révision de celles-ci en détaillant un exemple concret en robotique. Plus précisément, nous allons calculer la cinématique directe et inverse d'un robot sériel plan à 2 ddl.

Soit le référentiel $\mathcal{F}$ (avec origine O et axes x et y) et les vecteurs $\mathbf{v}_{OA}$ et $\mathbf{v}_{AB}$ qui servent à modéliser un robot sériel plan à 2 ddl, tel qu'illustré à la figure 5.1.

Les trois équations suivantes démontrent la notation que nous allons adopter dans ce cours ainsi que l'utilisation de la trigonométrie de base que nous allons faire :

$$\mathbf{v}_{OA} = \begin{bmatrix} x_{OA} \\ y_{OA} \end{bmatrix} = \begin{bmatrix} l_1 \cos\theta_1 \\ l_1 \sin\theta_1 \end{bmatrix}, \tag{5.1}$$

$$\mathbf{v}_{AB} = \begin{bmatrix} x_{AB} \\ y_{AB} \end{bmatrix} = \begin{bmatrix} l_2 \cos(\theta_1 + \theta_2) \\ l_2 \sin(\theta_1 + \theta_2) \end{bmatrix}, \tag{5.2}$$

$$\mathbf{v}_{OB} = \begin{bmatrix} x_B \\ y_B \end{bmatrix}, \tag{5.3}$$

où l_1 et l_2 sont les longueurs des segments OA et AB, respectivement. Les composantes x_{OA} et y_{OA} du vecteur $\mathbf{v}_{OA}$ sont les projections de ce vecteur sur les axes x et y, respectivement, du référentiel $\mathcal{F}$. Pareillement, les composantes x_{AB} et y_{AB} du vecteur $\mathbf{v}_{AB}$ sont les projections de ce vecteur sur les axes x et y, respectivement. Enfin, les composantes x_B et y_B du vecteur $\mathbf{v}_{OB}$ sont les projections de ce vecteur sur les axes x et y, respectivement, et aussi les coordonnées cartésiennes de l'effecteur, soit le point B. La position de l'effecteur peut aussi être représentée par l'équation suivante :

$$\begin{aligned} \mathbf{v}_{OB} = \mathbf{v}_{OA} + \mathbf{v}_{AB} &= \begin{bmatrix} x_{OA} \\ y_{OA} \end{bmatrix} + \begin{bmatrix} x_{AB} \\ y_{AB} \end{bmatrix} = \begin{bmatrix} x_{OA} + x_{AB} \\ y_{OA} + y_{AB} \end{bmatrix} \\ &= \begin{bmatrix} l_1 \cos\theta_1 + l_2 \cos(\theta_1 + \theta_2) \\ l_1 \sin\theta_1 + l_2 \sin(\theta_1 + \theta_2) \end{bmatrix} = \begin{bmatrix} l_1 \cos\theta_1 + l_2 \left(\cos\theta_1 \cos\theta_2 - \sin\theta_1 \sin\theta_2\right) \\ l_1 \sin\theta_1 + l_2 \left(\sin\theta_1 \cos\theta_2 + \cos\theta_1 \sin\theta_2\right) \end{bmatrix}. \end{aligned} \tag{5.4}$$

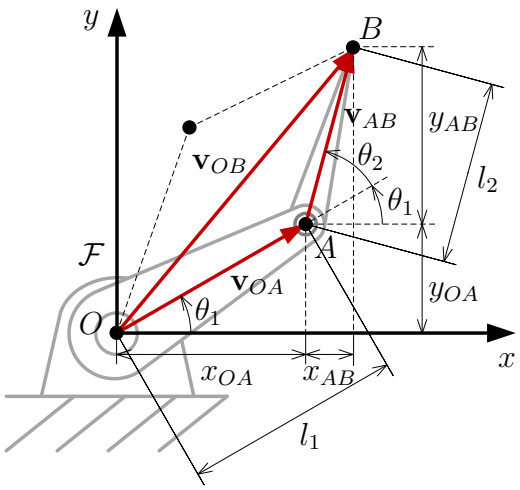


FIGURE 5.1 – Cinématique directe et inverse d'un robot sériel plan à 2 ddl.

L'équation (5.4) représente la solution de la *cinématique directe* du robot sériel plan à 2 ddl. Si on connait les rotations des deux moteurs, θ_1 et θ_2, ainsi que les longueurs des deux bras, cette équation nous permet de calculer la position de l'effecteur.

À l'opposé, le problème de la *cinématique inverse* consiste à calculer les *variables articulaires* θ_1 et θ_2, en connaissant la position de l'effecteur, soit les cordonnées x_B et y_B. Pour résoudre ce problème, nous allons calculer le produit vectoriel $\mathbf{v}_{OB}^T\mathbf{v}_{OB}$ (qui représente la longueur du vecteur $\mathbf{v}_{OB}$ au carré) en utilisant les expressions pour ce vecteur trouvées dans les équations (5.3) et (5.4) :

$$x_B^2+y_B^2 = (l_1\cos\theta_1 + l_2(\cos\theta_1\cos\theta_2 - \sin\theta_1\sin\theta_2))^2 + (l_1\sin\theta_1 + l_2(\sin\theta_1\cos\theta_2 + \cos\theta_1\sin\theta_2))^2\,. \quad (5.5)$$

En développant l'équation (5.5) et en utilisant l'identité trigonométrique $\sin^2\theta + \cos^2\theta = 1$, on obtient l'équation suivante :

$$x_B^2 + y_B^2 = l_1^2 + l_2^2 + 2l_1l_2\cos\theta_2. \quad (5.6)$$

Puisque $2l_1l_2 \neq 0$, les deux solutions de cette équation trigonométrique sont

$$\theta_2 = \pm\arccos\left(\frac{x_B^2 + y_B^2 - l_1^2 - l_2^2}{2l_1l_2}\right). \quad (5.7)$$

Ces deux solutions sont réelles uniquement si l'argument de la fonction $\arccos(\cdot)$ est dans l'intervalle $[-1, 1]$. Puisque $2l_1l_2 > 0$, il est facile de démontrer que cette dernière condition pour l'argument de la fonction $\arccos(\cdot)$ est équivalente à l'inégalité

$$(l_1 - l_2)^2 \leq x_B^2 + y_B^2 \leq (l_1 + l_2)^2. \quad (5.8)$$

Les deux solutions de l'équation (5.7) coïncident ($\theta_2 = 0$) lorsque le robot est complètement étendu ($x_B^2 + y_B^2 = l_1^2 + l_2^2$).

Une fois les deux solutions pour θ_2 trouvées, on peut calculer les deux solutions correspondantes pour l'angle θ_1, en substituant chaque solution pour θ_2 dans l'expression droite de l'équation vectorielle (5.4) et en égalant à l'expression droite de l'équation vectorielle (5.3) :

$$\begin{bmatrix}\cos\theta_1(l_1 + l_2\cos\theta_2) - l_2\sin\theta_1\sin\theta_2 \\ \sin\theta_1(l_1 + l_2\cos\theta_2) + l_2\cos\theta_1\sin\theta_2\end{bmatrix} = \begin{bmatrix}x_B \\ y_B\end{bmatrix}, \quad (5.9)$$

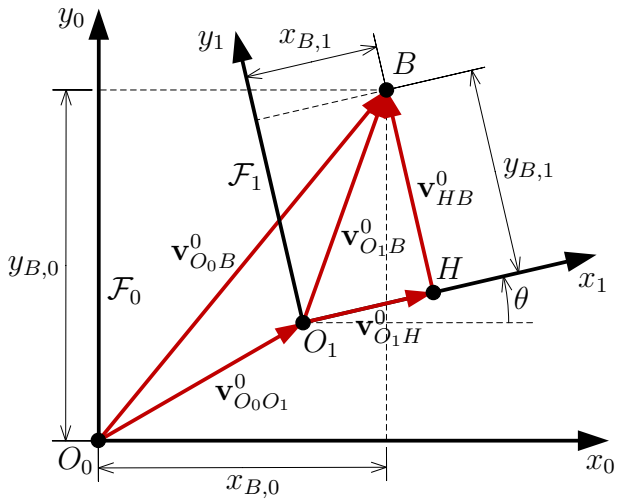


FIGURE 5.2 – Transformations de coordonnées dans le plan.

Cette équation vectorielle représente un système de deux équations linéaires en $\sin\theta_1$ et $\cos\theta_1$, dont la solution est

$$\cos\theta_1 = \frac{x_B l_1 + x_B l_2 \cos\theta_2 + y_B l_2 \sin\theta_2}{l_1^2 + l_2^2 + 2 l_1 l_2 \cos\theta_2}, \tag{5.10}$$

$$\sin\theta_1 = \frac{y_B l_1 + y_B l_2 \cos\theta_2 - x_B l_2 \sin\theta_2}{l_1^2 + l_2^2 + 2 l_1 l_2 \cos\theta_2}. \tag{5.11}$$

En utilisant l'équation (5.6), ce système peut être réécrit comme

$$\cos\theta_1 = \frac{x_B l_1 + x_B l_2 \cos\theta_2 + y_B l_2 \sin\theta_2}{x_B^2 + y_B^2}, \tag{5.12}$$

$$\sin\theta_1 = \frac{y_B l_1 + y_B l_2 \cos\theta_2 - x_B l_2 \sin\theta_2}{x_B^2 + y_B^2}. \tag{5.13}$$

Ce système a une seule solution pour l'angle θ_1 à l'exception du cas spécial $x_B^2 + y_B^2 = 0$ qui peut arriver uniquement lorsque $l_1 = l_2$. Cette solution est donnée par la fonction atan2$(\cdot,\cdot)$ (une variation à deux arguments de la fonction arctan$(\cdot)$) qui est très utilisée en robotique :

$$\theta_1 = \text{atan2}\left(y_B l_1 + y_B l_2 \cos\theta_2 - x_B l_2 \sin\theta_2, x_B l_1 + x_B l_2 \cos\theta_2 + y_B l_2 \sin\theta_2\right). \tag{5.14}$$

Enfin, le cas spécial $x_B^2 + y_B^2 = 0$ représente une singularité où l'angle θ_1 peut être arbitraire. Dans cette singularité, l'effecteur (le point B) reste immobile même si l'angle θ_1 varie.

5.2 Matrice de rotation et transformations de coordonnées dans le plan

Soit les deux référentiels $\mathcal{F}_0$ (avec origine O_0 et axes x_0 et y_0) et $\mathcal{F}_1$ (avec origine O_1 et axes x_1 et y_1) illustrés à la figure 5.2, où le vecteur $\mathbf{v}^0_{O_0O_1}$ représente les coordonnées de l'origine du référentiel $\mathcal{F}_1$ par rapport au référentiel $\mathcal{F}_0$ (d'où l'exposant « 0 ») et θ est l'angle l'entre l'axe x_0 et l'axe x_1 (l'angle de rotation). Admettons que les coordonnées d'un point B sont connues uniquement dans le référentiel $\mathcal{F}_1$:

$$\mathbf{v}^1_{O_1B} = \begin{bmatrix} x_{B,1} \\ y_{B,1} \end{bmatrix} \tag{5.15}$$

où l'exposant « 1 » sur le vecteur signifie que les composantes du vecteur sont exprimées par rapport au référentiel $\mathcal{F}_1$. On cherche maintenant les coordonnées du point B par rapport au référentiel $\mathcal{F}_0$. Celles-ci peuvent être obtenues de la manière suivante :

$$\begin{aligned}
\mathbf{v}^0_{O_0B} &= \mathbf{v}^0_{O_0O_1} + \mathbf{v}^0_{O_1B} = \mathbf{v}^0_{O_0O_1} + \mathbf{v}^0_{O_1H} + \mathbf{v}^0_{HB} \\
&= \mathbf{v}^0_{O_0O_1} + \begin{bmatrix} x_{B,1}\cos\theta \\ x_{B,1}\sin\theta \end{bmatrix} + \begin{bmatrix} y_{B,1}\cos(\theta+90°) \\ y_{B,1}\sin(\theta+90°) \end{bmatrix} \\
&= \mathbf{v}^0_{O_0O_1} + \begin{bmatrix} x_{B,1}\cos\theta \\ x_{B,1}\sin\theta \end{bmatrix} + \begin{bmatrix} -y_{B,1}\sin\theta \\ y_{B,1}\cos\theta \end{bmatrix} \\
&= \mathbf{v}^0_{O_0O_1} + \begin{bmatrix} x_{B,1}\cos\theta - y_{B,1}\sin\theta \\ x_{B,1}\sin\theta + y_{B,1}\cos\theta \end{bmatrix} \\
&= \mathbf{v}^0_{O_0O_1} + \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x_{B,1} \\ y_{B,1} \end{bmatrix} \\
&= \mathbf{v}^0_{O_0O_1} + \mathbf{R}^0_1\mathbf{v}^1_{O_1B},
\end{aligned} \tag{5.16}$$

où la matrice

$$\mathbf{R}^0_1 = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \tag{5.17}$$

est la *matrice de rotation* qui représente l'orientation du référentiel $\mathcal{F}_1$ par rapport au référentiel $\mathcal{F}_0$.

Il est très important de noter que la première colonne et la deuxième colonne de la matrice de rotation sont les vecteurs unitaires le long de l'axe x_1 et l'axe y_1, respectivement, exprimés dans le référentiel $\mathcal{F}_0$. Ainsi, la matrice de rotation est une *matrice orthogonale* qui respecte la relation suivante :

$$\left(\mathbf{R}^0_1\right)^T = \left(\mathbf{R}^0_1\right)^{-1}. \tag{5.18}$$

Enfin, il est possible de démontrer que $\left(\mathbf{R}^0_1\right)^T$ est en fait la matrice $\mathbf{R}^1_0$ dont la première colonne et la deuxième colonne sont les vecteurs unitaires le long des axes x_0 et y_0, respectivement, exprimés dans le référentiel $\mathcal{F}_1$. Ainsi, si nous connaissions les coordonnées du point B par rapport au référentiel $\mathcal{F}_0$, les coordonnées de ce point par rapport au référentiel $\mathcal{F}_1$ peuvent être trouvées avec l'équation suivante :

$$\mathbf{v}^1_{O_1B} = \mathbf{v}^1_{O_1O_0} + \mathbf{v}^1_{O_0B} = \mathbf{v}^1_{O_1O_0} + \mathbf{R}^1_0\mathbf{v}^0_{O_0B}. \tag{5.19}$$

Comme Albert Einstein l'a dit : « Tout est relatif! ».

5.3 Matrice de rotation et transformations de coordonnées en trois dimensions

Dans le cas de deux référentiels dans l'espace, on peut suivre exactement la même logique, mais le processus est un peu plus difficile à comprendre. Nous allons alors donner uniquement le résultat final. Soit les deux référentiels $\mathcal{F}_0$ et $\mathcal{F}_1$ illustrés à la figure 5.3[4], où le vecteur $\mathbf{v}^0_{O_0O_1}$ représente les coordonnées de l'origine du référentiel $\mathcal{F}_1$ par rapport au référentiel $\mathcal{F}_0$. Admettons que les coordonnées d'un point B sont connues uniquement dans le référentiel $\mathcal{F}_1$. Alors les coordonnées de ce point par rapport au référentiel $\mathcal{F}_0$ peuvent être trouvées à l'aide de l'équation suivante :

$$\mathbf{v}^0_{O_0B} = \mathbf{v}^0_{O_0O_1} + \mathbf{R}^0_1\mathbf{v}^1_{O_1B}, \tag{5.20}$$

4. Dans le cas des référentiels en trois dimensions, nous n'allons pas libeller leurs axes (« x », « y », « z ») mais plutôt dessiner les axes x en rouge, les axes y en vert, et les axes z en bleu, comme dans les logiciels de CAO.

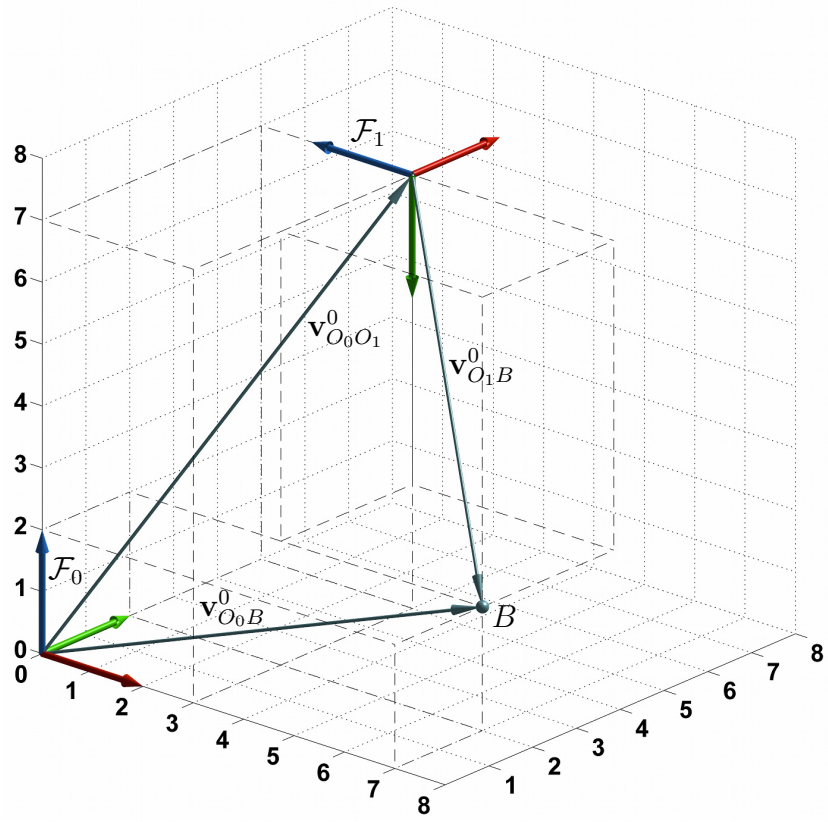


FIGURE 5.3 – Transformation de coordonnées dans l'espace.

où la matrice $\mathbf{R}^0_1$ est la matrice de rotation 3×3 dans l'espace.

Contrairement à la matrice de rotation dans le plan, il n'existe pas de formule unique pour la matrice de rotation dans l'espace, puisqu'il y a plusieurs façons de représenter l'orientation d'un référentiel par rapport à un autre dans l'espace (douze conventions d'angles d'Euler, les quaternions, l'axe équivalent de rotation, etc.). Par contre, cette matrice de rotation 3×3 a les même propriétés que la matrice de rotation dans le plan. Notamment, la première colonne, la deuxième colonne, et la troisième colonne de la matrice de rotation dans l'espace représentent les vecteurs unitaires le long de l'axe x_1, l'axe y_1, et l'axe z_1, respectivement, par rapport au référentiel $\mathcal{F}_0$. La matrice de rotation dans l'espace est elle aussi une matrice orthogonale et l'équation (5.19) s'applique dans le cas présent, telle quelle.

Enfin, nous allons résoudre l'exemple concret qui est représenté à la figure 5.3. La position du point B par rapport au référentiel $\mathcal{F}_1$ est :

$$\mathbf{v}^1_{O_1B} = \begin{bmatrix} -3 \\ 5 \\ -4 \end{bmatrix}, \tag{5.21}$$

alors que la position de l'origine du référentiel $\mathcal{F}_1$ par rapport au référentiel $\mathcal{F}_0$ est

$$\mathbf{v}^0_{O_0O_1} = \begin{bmatrix} 3 \\ 5 \\ 7 \end{bmatrix}. \tag{5.22}$$

La partie la plus difficile est de calculer la matrice de rotation $\mathbf{R}^0_1$. Il s'agit d'obtenir les expressions des vecteurs unitaires le long des axes du référentiel $\mathcal{F}_1$, par rapport au référentiel $\mathcal{F}_0$. Dans cet exemple, ce calcul est relativement simple, puisque les axes des deux référentiels sont parallèles ou orthogonaux,

mais souvent ce calcul est assez complexe. Nous avons alors

$$\mathbf{R}_1^0 = \begin{bmatrix} 0 & 0 & -1 \\ 1 & 0 & 0 \\ 0 & -1 & 0 \end{bmatrix}. \tag{5.23}$$

Ainsi, la position du point B par rapport au référentiel $\mathcal{F}_0$ est

$$\mathbf{v}_{O_0B}^0 = \begin{bmatrix} 3 \\ 5 \\ 7 \end{bmatrix} + \begin{bmatrix} 0 & 0 & -1 \\ 1 & 0 & 0 \\ 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} -3 \\ 5 \\ -4 \end{bmatrix} = \begin{bmatrix} 7 \\ 2 \\ 2 \end{bmatrix}. \tag{5.24}$$

Dans cet exemple, il est facile de valider notre réponse en inspectant le cuboïde carré en pointillé entre le référentiel $\mathcal{F}_0$ et le point B (figure 5.3).

Admettons, maintenant, que les coordonnées du point B sont connues uniquement dans le référentiel $\mathcal{F}_0$. Alors les coordonnées de ce point par rapport au référentiel $\mathcal{F}_1$ peuvent être trouvées à l'aide de l'équation suivante :

$$\mathbf{v}_{O_1B}^1 = \mathbf{v}_{O_1O_0}^1 + \mathbf{R}_0^1 \mathbf{v}_{O_0B}^0, \tag{5.25}$$

où la position du point B par rapport au référentiel $\mathcal{F}_0$ est :

$$\mathbf{v}_{O_0B}^0 = \begin{bmatrix} 7 \\ 2 \\ 2 \end{bmatrix}, \tag{5.26}$$

alors que la position de l'origine du référentiel $\mathcal{F}_0$ par rapport au référentiel $\mathcal{F}_1$ est

$$\mathbf{v}_{O_1O_0}^1 = \begin{bmatrix} -5 \\ 7 \\ 3 \end{bmatrix}. \tag{5.27}$$

La matrice $\mathbf{R}_0^1$ est la matrice de rotation qui représente l'orientation du référentiel $\mathcal{F}_0$ par rapport au référentiel $\mathcal{F}_1$. $\mathbf{R}_1^0$. Ces colonnes sont composées des vecteurs unitaires le long des axes du référentiel $\mathcal{F}_0$ exprimés par rapport au référentiel $\mathcal{F}_1$. Nous avons alors

$$\mathbf{R}_0^1 = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & -1 \\ -1 & 0 & 0 \end{bmatrix}. \tag{5.28}$$

Notez que la transposée de la matrice $\mathbf{R}_0^1$ est bel et bien la matrice $\mathbf{R}_0^1$ que nous avons obtenue dans l'équation (5.23).

Enfin, la position du point B par rapport au référentiel $\mathcal{F}_1$ est

$$\mathbf{v}_{O_1B}^1 = \begin{bmatrix} -5 \\ 7 \\ 3 \end{bmatrix} + \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & -1 \\ -1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 7 \\ 2 \\ 2 \end{bmatrix} = \begin{bmatrix} -3 \\ 5 \\ -4 \end{bmatrix}. \tag{5.29}$$

Pour valider cette réponse, il suffit d'inspecter le cuboïde carré en pointillé entre le référentiel $\mathcal{F}_1$ et le point B (figure 5.3).

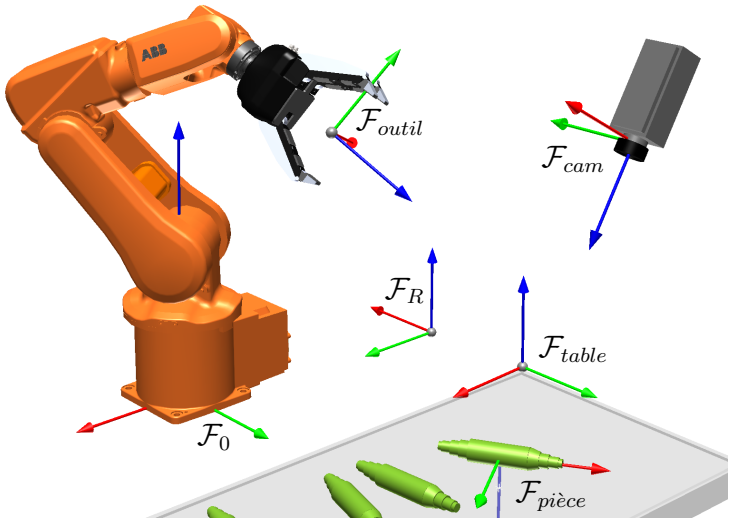


FIGURE 5.4 – Exemple de transformations de coordonnées.

5.4 Transformations de coordonnées successives

En robotique, nous avons souvent besoin de plusieurs référentiels : le référentiel de la base, le référentiel de l'atelier, le référentiel d'une caméra, le référentiel d'un objet, etc. Souvent, la pose d'un objet sera donnée par rapport à un référentiel, qui est défini par rapport à un autre référentiel, qui est défini par rapport au référentiel de la base du robot. La figure 5.4 illustre un exemple d'une telle situation. Le système de vision calcule la pose de la pièce, c'est-à-dire, de son référentiel $\mathcal{F}_{pièce}$) par rapport au référentiel de la caméra, $\mathcal{F}_{cam}$, et plus précisément le vecteur de position $\mathbf{v}^{cam}_{O_{cam}O_{pièce}}$ et la matrice de rotation $\mathbf{R}^{cam}_{pièce}$. Le référentiel de la caméra est défini par rapport au référentiel de l'atelier, $\mathcal{F}_R$, à l'aide du vecteur de position $\mathbf{v}^{R}_{O_R O_{cam}}$ et de la matrice de rotation $\mathbf{R}^{R}_{cam}$. Enfin, le référentiel de l'atelier est défini par rapport au référentiel de la base du robot, $\mathcal{F}_0$, à l'aide du vecteur de position $\mathbf{v}^{0}_{O_0 O_R}$ et de la matrice de rotation $\mathbf{R}^{0}_{R}$.

Voici alors le calcul que le contrôleur du robot doit effectuer afin de pouvoir saisir la pièce :

$$\mathbf{v}^{0}_{O_0 O_{pièce}} = \mathbf{v}^{0}_{O_0 O_R} + \mathbf{R}^{0}_{R}\mathbf{v}^{R}_{O_R O_{pièce}} = \mathbf{v}^{0}_{O_0 O_R} + \mathbf{R}^{0}_{R}\left(\mathbf{v}^{R}_{O_R O_{cam}} + \mathbf{R}^{R}_{cam}\mathbf{v}^{cam}_{O_{cam} O_{pièce}}\right), \tag{5.30}$$

pour la position de la pièce par rapport au référentiel de la base du robot et

$$\mathbf{R}^{0}_{pièce} = \mathbf{R}^{0}_{R}\mathbf{R}^{R}_{cam}\mathbf{R}^{cam}_{pièce}, \tag{5.31}$$

pour l'orientation de la pièce par rapport au référentiel de la base du robot.

Ce type de calcul n'est pas intuitif ni compact. Une méthode de calcul plus conviviale sera alors introduite dans le chapitre suivant.

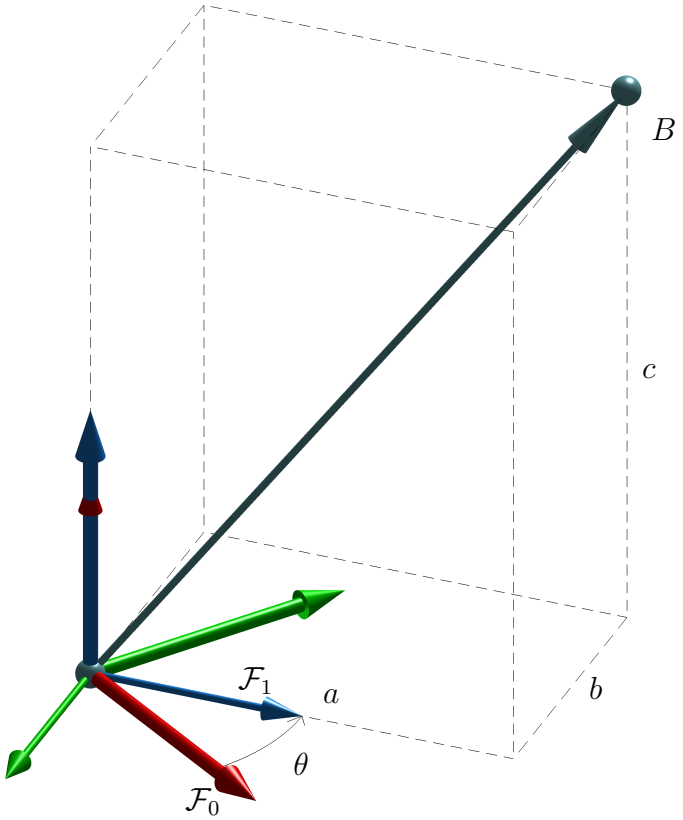


FIGURE 5.5 – Question 5.1.

5.5 Exercices

5.1. Répondre aux différentes questions en rapport avec la figure 5.5 :
   (a) Donner le vecteur de position $\mathbf{v}^0_{O_0O_1}$.
   (b) Donner la matrice de rotation $\mathbf{R}^0_1$.
   (c) Donner le vecteur de position $\mathbf{v}^1_{O_1B}$.
   (d) Donner le vecteur de position $\mathbf{v}^0_{O_0B}$.

5.2. Répondre aux différentes questions en rapport avec la figure 5.6 :
   (a) Donner le vecteur de position $\mathbf{v}^0_{O_0O_1}$.
   (b) Donner la matrice de rotation $\mathbf{R}^0_1$.

5.3. Répondre aux différentes questions en rapport avec la figure 5.7 :
   (a) Donner le vecteur de position $\mathbf{v}^0_{O_0O_1}$.
   (b) Donner la matrice de rotation $\mathbf{R}^0_1$.

5.4. Répondre aux différentes questions en rapport avec la figure 5.8 :
   (a) Donner le vecteur de position $\mathbf{v}^0_{O_0O_1}$.
   (b) Donner la matrice de rotation $\mathbf{R}^0_1$.

5.5. Répondre aux différentes questions en rapport avec la figure 5.9 :
   (a) Donner le vecteur de position $\mathbf{v}^0_{O_0O_1}$.
   (b) Donner la matrice de rotation $\mathbf{R}^0_1$.
   (c) Donner le vecteur de position $\mathbf{v}^1_{O_1B}$.
   (d) Donner le vecteur de position $\mathbf{v}^0_{O_0B}$.

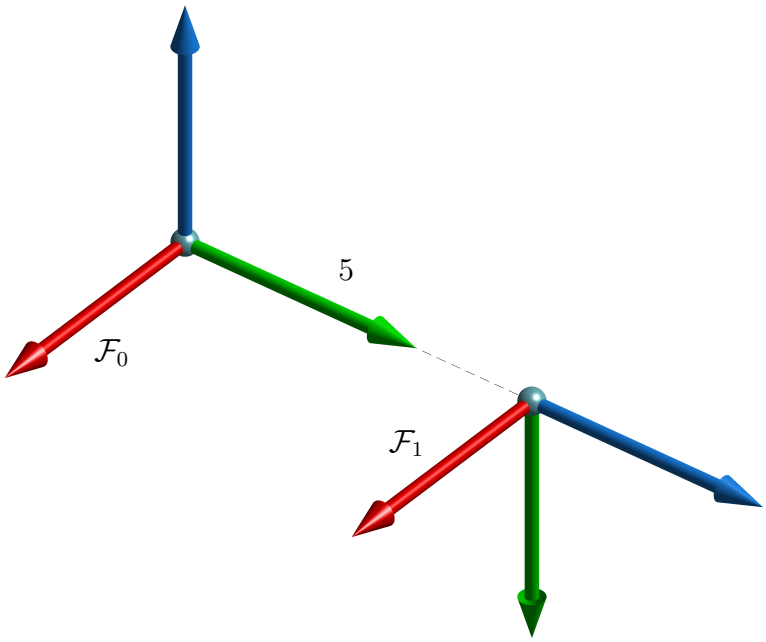


FIGURE 5.6 – Question 5.2.

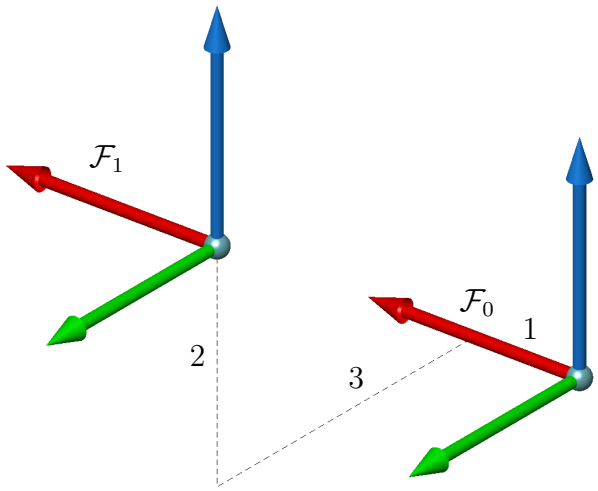


FIGURE 5.7 – Question 5.3.

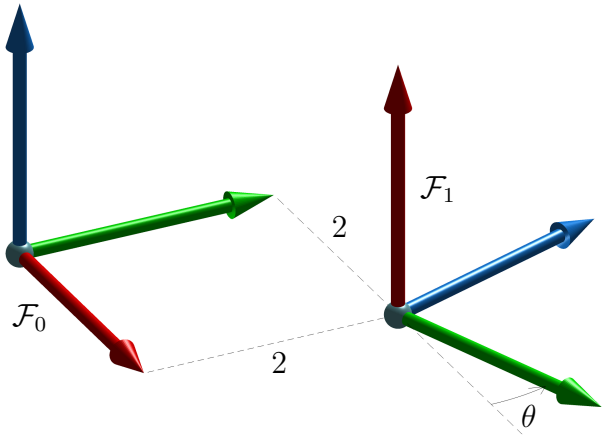


FIGURE 5.8 – Question 5.4.

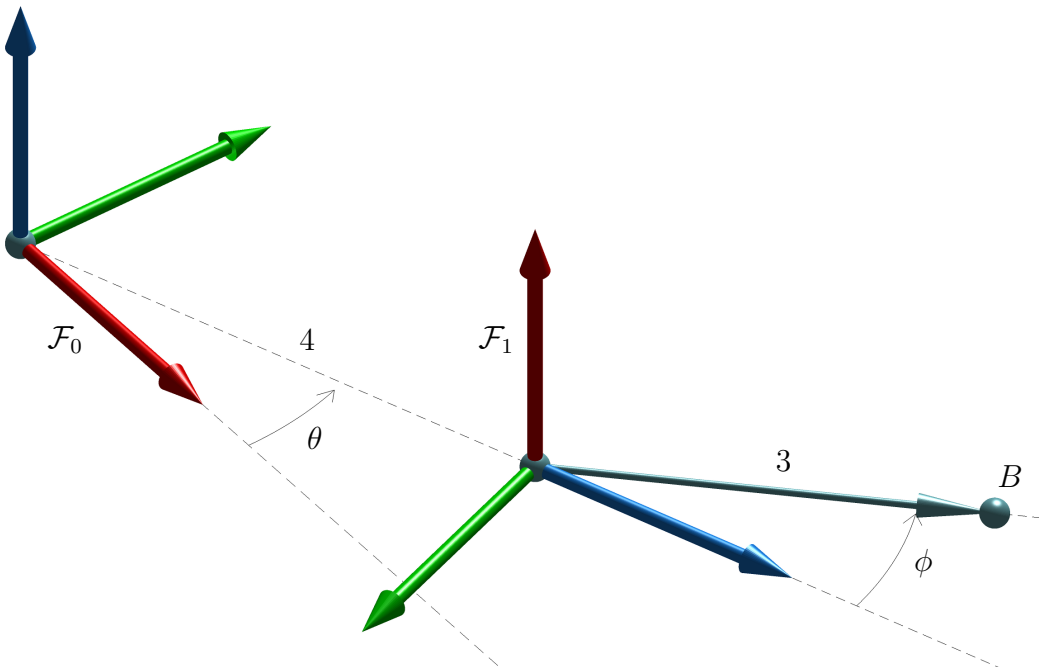


FIGURE 5.9 – Question 5.5.

5.6 Réponses aux exercices

5.1. $\mathbf{v}^0_{O_0O_1} = \begin{bmatrix}0\\0\\0\end{bmatrix}$, $\mathbf{R}^0_1 = \begin{bmatrix}0 & \sin\theta & \cos\theta\\0 & -\cos\theta & \sin\theta\\1 & 0 & 0\end{bmatrix}$, $\mathbf{v}^1_{O_1B} = \begin{bmatrix}c\\-b\\a\end{bmatrix}$, $\mathbf{v}^0_{O_0B} = \begin{bmatrix}-b\sin\theta + a\cos\theta\\b\cos\theta + a\sin\theta\\c\end{bmatrix}$.

5.2. $\mathbf{v}^0_{O_0O_1} = \begin{bmatrix}0\\5\\0\end{bmatrix}$, $\mathbf{R}^0_1 = \begin{bmatrix}1 & 0 & 0\\0 & 0 & 1\\0 & -1 & 0\end{bmatrix}$.

5.3. $\mathbf{v}^0_{O_0O_1} = \begin{bmatrix}1\\3\\2\end{bmatrix}$, $\mathbf{R}^0_1 = \begin{bmatrix}1 & 0 & 0\\0 & 1 & 0\\0 & 0 & 1\end{bmatrix}$.

5.4. $\mathbf{v}^0_{O_0O_1} = \begin{bmatrix}2\\2\\0\end{bmatrix}$, $\mathbf{R}^0_1 = \begin{bmatrix}0 & \cos\theta & -\sin\theta\\0 & \sin\theta & \cos\theta\\1 & 0 & 0\end{bmatrix}$.

5.5. $\mathbf{v}^0_{O_0O_1} = \begin{bmatrix}4\cos\theta\\4\sin\theta\\0\end{bmatrix}$, $\mathbf{R}^0_1 = \begin{bmatrix}0 & \sin\theta & \cos\theta\\0 & -\cos\theta & \sin\theta\\1 & 0 & 0\end{bmatrix}$, $\mathbf{v}^1_{O_1B} = \begin{bmatrix}0\\-3\sin\phi\\3\cos\phi\end{bmatrix}$,

$\mathbf{v}^0_{O_0B} = \begin{bmatrix}4\cos\theta - 3\sin\theta\sin\phi + 3\cos\theta\cos\phi\\4\sin\theta + 3\cos\theta\sin\phi + 3\sin\theta\cos\phi\\0\end{bmatrix}$.

Chapitre 6

Transformations homogènes

Nous avons vu dans la section 5.4 que lorsqu'il y a plusieurs transformations de coordonnées successives, les équations que nous avons présentées dans le chapitre précédant deviennent rapidement trop complexes. Dans ce chapitre, nous introduirons la notion de *matrice homogène* qui nous permettra essentiellement de travailler uniquement avec des matrices dites homogènes. Nous présenterons également les concepts de transformations homogènes consécutives. Enfin, nous étudierons la paramétrisation de l'orientation à l'aide des angles d'Euler et des quaternions.

6.1 Matrice homogène

À partir de maintenant, nous nous intéresserons uniquement à la pose d'un référentiel par rapport à un autre. Ainsi, les seuls vecteurs que nous utiliserons (indirectement) seront les vecteurs qui représentent la position de l'origine d'un référentiel $\mathcal{F}_b$ par rapport à un autre référentiel $\mathcal{F}_a$. Ainsi, par souci de simplicité et pour être conformes avec la notation que nous avons adopté pour les matrices de rotation, nous noterons un tel vecteur de position par $\mathbf{p}_b^a$, plutôt que par $\mathbf{v}_{O_aO_b}^a$.

Nous ne rentrerons pas dans les détails, mais il est relativement facile de démontrer que les équations (5.30) et (5.31) peuvent être combinées est réécrit comme une seule équation matricielle :

$$\mathbf{H}_{pièce}^0 = \mathbf{H}_R^0 \mathbf{H}_{cam}^R \mathbf{H}_{pièce}^{cam}, \tag{6.1}$$

où $\mathbf{H}_b^a$ est une matrice 4×4 dite homogène définie par

$$\mathbf{H}_b^a = \begin{bmatrix} & \mathbf{R}_b^a & & \mathbf{p}_b^a \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{6.2}$$

Dans ce contexte, la matrice homogène $\mathbf{H}_b^a$ représente la pose du référentiel $\mathcal{F}_b$ par rapport au référentiel $\mathcal{F}_a$. Comme dans le cas des matrices de rotations, $[\mathbf{H}_b^a]^{-1} = \mathbf{H}_a^b$. Cependant, $[\mathbf{H}_b^a]^{-1} \neq [\mathbf{H}_b^a]^T$ et nous avons plutôt

$$\mathbf{H}_a^b = [\mathbf{H}_b^a]^{-1} = \begin{bmatrix} & [\mathbf{R}_b^a]^T & & -[\mathbf{R}_b^a]^T \mathbf{p}_b^a \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} & \mathbf{R}_a^b & & -\mathbf{R}_a^b \mathbf{p}_b^a \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} & \mathbf{R}_a^b & & \mathbf{p}_a^b \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{6.3}$$

Ainsi, la matrice homogène peut être utilisée pour trouver la pose d'un référentiel par rapport à un autre. Considérons la figure 5.4 de nouveau. Admettons que nous connaissons la pose du référentiel

$\mathcal{F}_{table}$ par rapport au référentiel $\mathcal{F}_0$, $\mathbf{H}^0_{table}$, ainsi que la pose du référentiel $\mathcal{F}_R$ par rapport au référentiel $\mathcal{F}_0$, $\mathbf{H}^0_R$. La pose du référentiel $\mathcal{F}_{table}$ par rapport au référentiel $\mathcal{F}_R$ peut alors se calculer de la manière suivante :

$$\mathbf{H}^R_{table} = \mathbf{H}^R_0 \mathbf{H}^0_{table} = [\mathbf{H}^0_R]^{-1} \mathbf{H}^0_{table}. \tag{6.4}$$

6.2 Transformations homogènes consécutives

Dans la section précédente, nous avons démontré comment utiliser la matrice homogène afin de trouver la pose d'un référentiel $\mathcal{F}_b$ par rapport à un autre référentiel $\mathcal{F}_a$, quand les poses de ces référentiels sont données par rapport à d'autres référentiels. Essentiellement, on essaie de « se trouver un chemin » du référentiel $\mathcal{F}_a$ vers le référentiel $\mathcal{F}_b$ pour obtenir une équation sous la forme suivante :

$$\mathbf{H}^a_b = \mathbf{H}^a_1 \mathbf{H}^1_2 ... \mathbf{H}^{n-1}_n \mathbf{H}^n_b. \tag{6.5}$$

Grace au langage du programmation RAPID, nous n'avons pas souvent besoin d'utiliser ce type d'équation. Nous devons simplement utiliser judicieusement les référentiels de travail (wobjdata) et les calculs matriciels se feront automatiquement par le contrôleur du robot. Ce qui arrive plus souvent, c'est d'essayer de trouver la matrice homogène qui représente la pose d'un référentiel par rapport à un autre, en suivant une séquence de transformations imaginaires prédéterminée (comme dans le cas des angles d'Euler que nous étudierons plus tard dans ce chapitre) ou en s'imaginant une telle séquence. Dans ce cas, il n'y a pas vraiment d'autres référentiels intermédiaires, ou plutôt ces référentiels imaginaires ne sont pas attachés à des objets réels.

Par exemple, en se référant de nouveau à la figure 5.4, nous connaissons déjà la pose du référentiel $\mathcal{F}_{pièce}$ par rapport au référentiel $\mathcal{F}_0$, $\mathbf{H}^0_{pièce}$, et nous cherchons la pose d'un référentiel $\mathcal{F}_{pièce2}$ qui est obtenu en tournant le référentiel $\mathcal{F}_{pièce}$ de 180° autour de son axe $z_{pièce}$. Dans ce cas, nous pouvons utiliser la fonction RelTool dans le langage RAPID, mais imaginons que le référentiel $\mathcal{F}_{pièce2}$ dont nous cherchons la pose doit être obtenue en tournant le référentiel $\mathcal{F}_{pièce}$ de 30° autour de l'axe z_{table} du référentiel $\mathcal{F}_{table}$. Dans un tel cas, aucune fonction RAPID ne peut faire le calcul pour nous. Bien sûr, dans les deux cas, nous pouvons obtenir la matrice $\mathbf{H}^0_{pièce2}$ de la manière présentée dans le chapitre précédent, mais ce calcul peut s'avérer assez complexe et n'est pas recommandé. Heureusement, il existe deux règles assez simples qui nous permettrons de faire ce genre de transformations sans même utiliser la trigonométrie et le calcul vectoriel.

6.2.1 Transformations pures

Dans le contexte de l'utilisation des matrices homogènes en tant que matrice de transformation, nous aurons besoin de définir quatre matrices homogènes de transformations de base (ou pures). La première matrice est celle qui correspond à une translation pure :

$$\mathbf{H}_{trans}(p_x, p_y, p_z) = \begin{bmatrix} 1 & 0 & 0 & p_x \\ 0 & 1 & 0 & p_y \\ 0 & 0 & 1 & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{6.6}$$

Chacune des trois autres matrices correspond à une rotation autour d'un axe x, y ou z :

$$\mathbf{H}_{rot,x}(\theta) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{6.7}$$

$$\mathbf{H}_{rot,y}(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{6.8}$$

$$\mathbf{H}_{rot,z}(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{6.9}$$

6.2.2 Transformation par rapport au référentiel « mobile »

Soit les deux référentiels $\mathcal{F}_a$ et $\mathcal{F}_b$. Puisque nous allons considérer que nous connaissons la pose du référentiel $\mathcal{F}_b$ par rapport au référentiel $\mathcal{F}_a$, c'est-à-dire la matrice homogène $\mathbf{H}_b^a$, nous pouvons considérer le référentiel $\mathcal{F}_a$ comme étant « fixe » et le référentiel $\mathcal{F}_b$ comme étant « mobile ».

La pose, par rapport au référentiel $\mathcal{F}_a$, d'un référentiel $\mathcal{F}_c$ obtenu en appliquant une transformation $\mathbf{H}$ par rapport au référentiel $\mathcal{F}_b$ est définie par l'équation suivante :

$$\mathbf{H}_c^a = \mathbf{H}_b^a \mathbf{H}. \tag{6.10}$$

Comme exemple, considérons à nouveau la figure 5.4. Si nous désirons définir un référentiel $\mathcal{F}_{pièce2}$ qui est obtenu en tournant le référentiel $\mathcal{F}_{pièce1}$ de 180° autour de son axe $z_{pièce1}$, alors la pose de ce nouveau référentiel sera

$$\mathbf{H}^0_{pièce2} = \mathbf{H}^0_{pièce1}\mathbf{H}_{rot,z}(180°). \tag{6.11}$$

Comme déjà mentionné, la fonction RelTool de RAPID effectue exactement ce type de calcul. Par exemple, le calcul précédent peut se faire de cette manière simple :

Exemple 6.1 – Fonction RelTool

```
MODULE Exemple()
    PERS    robtarget rtPiece1 := [[300,0,500],[0,1,0,0],[0,-1,0,0],[9E9,9E9,9E9,9E9,9E9,9E9]];
    VAR     robtarget rtPiece2;

    rtPiece2 := RelTool(rtPiece1, 0, 0, 0 \Rz:=180);
ENDMODULE
```

La fonction RelTool peut être appelée avec trois arguments optionnels pour la rotation : « \Rx :=... », « \Ry :=... », « \Rz :=... ». Ces arguments doivent être présentés dans cet ordre, par exemple RelTool(rtPiece1, 5, 7, 2 \Rx :=90 \Rz :=180) et non RelTool(rtPiece1, 5, 7, 2 \Rz :=180 \Rx :=90). Lorsque ces arguments sont présents, l'ordre de transformations est : translation le long des axes x, y et z du référentiel du robtarget, suivie par une rotation autour du nouvel axe x, suivie par une rotation autour du nouvel axe y, suivie par une rotation autour du nouvel axe z. Par exemple, l'instruction RelTool(rtPiece1, 10, 20, 30 \Rx :=90 \Ry :=40 \Rz :=180) est équivalente à :

$$\mathbf{H}^0_{pièce1}\mathbf{H}_{trans}(5,7,2)\mathbf{H}_{rot,x}(90°)\mathbf{H}_{rot,y}(40°)\mathbf{H}_{rot,z}(180°). \tag{6.12}$$

6.2.3 Transformation par rapport au référentiel « fixe »

Admettons maintenant que nous désirons trouver la pose, par rapport au référentiel $\mathcal{F}_a$, d'un référentiel $\mathcal{F}_c$ obtenu en appliquant une transformation $\mathbf{H}$ par rapport au référentiel $\mathcal{F}_a$ (plutôt que $\mathcal{F}_b$). Cette pose est définie par l'équation suivante :

$$\mathbf{H}_c^a = \mathbf{H}\mathbf{H}_b^a. \tag{6.13}$$

Comme exemple concret, considérons à nouveau la figure 5.4. Si nous désirons définir un référentiel $\mathcal{F}_{pièce2}$ qui est obtenu en tournant le référentiel $\mathcal{F}_{pièce}$ de 30° autour de l'axe z_{table}, alors la pose de ce nouveau référentiel sera

$$\mathbf{H}_{pièce2}^0 = \mathbf{H}_{rot,z}(30°)\mathbf{H}_{pièce}^0. \tag{6.14}$$

Il n'existe pas de fonction en RAPID, équivalente à RelTool, pour faire ce type de transformation. Lorsque la transformation $\mathbf{H}$ est une translation pure, nous pouvons utiliser la fonction Offs en RAPID. Sinon, nous pouvons créer notre propre fonction :

Exemple 6.2 – Fonction RelWorld

```
FUNC robtarget RelWorld(robtarget rt1, num x, num y, num z \num Rx \num Ry \num Rz)
    VAR pose pResultat;
    VAR num anglex;
    VAR num angley;
    VAR num anglez;

    IF Present (Rx) THEN anglex := Rx; ELSE anglex := 0; ENDIF
    IF Present (Ry) THEN angley := Ry; ELSE angley := 0; ENDIF
    IF Present (Rz) THEN anglez := Rz; ELSE anglez := 0; ENDIF

    pResultat := PoseMult([[x,y,z],[1,0,0,0]],[rt1.trans, rt1.rot]);
    pResultat := PoseMult([[0,0,0],OrientZYX(anglez,angley,anglex)],pResultat);

    RETURN [pResultat.trans, pResultat.rot, rt1.robconf, rt1.extax];

ENDFUNC
```

Ainsi, par exemple, l'instruction RelWorld(rtPiece1, 10, 20, 30 \Rx :=90 \Ry :=40 \Rz :=180) est équivalente à :

$$\mathbf{H}_{rot,z}(180°)\mathbf{H}_{rot,y}(40°)\mathbf{H}_{rot,x}(90°)\mathbf{H}_{trans}(5,7,2)\mathbf{H}_{pièce1}^0. \tag{6.15}$$

6.2.4 Transformations composites

La situation où les notions théoriques des deux sections précédentes s'avèrent très utiles est lorsque nous désirons calculer la pose d'un référentiel $\mathcal{F}_b$ par rapport à un référentiel $\mathcal{F}_a$. Bien sûr, nous pouvons calculer les vecteurs unitaires le long des axes du référentiel $\mathcal{F}_b$ exprimés par rapport au référentiel $\mathcal{F}_a$, ainsi que le vecteur de position de l'origine du référentiel $\mathcal{F}_b$ par rapport au référentiel $\mathcal{F}_a$, mais ceci peut être assez complexe. Souvent, la manière la plus simple c'est de s'imaginer une série de transformations qu'un référentiel initialement coïncidant avec le référentiel $\mathcal{F}_a$ doit effectuer pour se rendre coïncidant avec le référentiel $\mathcal{F}_b$.

Considérons la figure 6.1. Notre objectif c'est de trouver la pose du référentiel $\mathcal{F}_b$ par rapport au référentiel $\mathcal{F}_a$. L'angle entre axe y_b et le plan $x_a y_b$ est 45°. Il y a plusieurs séquences de transformations qui peuvent amener un référentiel imaginaire $\mathcal{F}_m$, initialement coïncident avec $\mathcal{F}_a$, à $\mathcal{F}_b$. Admettons que la séquence que vous imaginer est la suivante :

- translation de 10 mm le long de l'axe z_a, donc $\mathbf{H}_m^a = \mathbf{H}_{trans}(0,0,10)$;
- rotation de $-90°$ autour de l'axe x_a, donc $\mathbf{H}_m^a = \mathbf{H}_{rot,x}(-90°)\mathbf{H}_{trans}(0,0,10)$;
- rotation de $-135°$ autour de l'axe z_m, donc $\mathbf{H}_m^a = \mathbf{H}_{rot,x}(-90°)\mathbf{H}_{trans}(0,0,10)\mathbf{H}_{rot,z}(-135°)$.

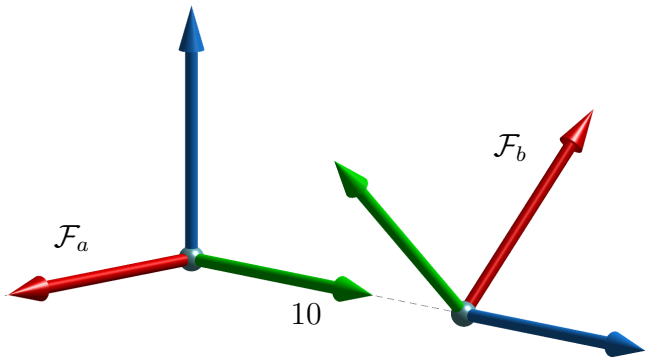


FIGURE 6.1 – Exemple de transformations composites.

Après ces trois transformations, le référentiel imaginaire $\mathcal{F}_m$ coïncidera avec le référentiel $\mathcal{F}_b$, ce qui veut dire que

$$\mathbf{H}_b^a = \mathbf{H}_{rot,x}(-90°)\mathbf{H}_{trans}(0,0,10)\mathbf{H}_{rot,z}(-135°). \tag{6.16}$$

Il existe plusieurs d'autres séquences de transformations qui donnent évidement le même résultat final, mais pas nécessairement avec la même séquence de trois matrices homogènes. Par exemple, il est possible de démontrer que $\mathbf{H}_b^a = \mathbf{H}_{trans}(0,10,0)\mathbf{H}_{rot,x}(-90°)\mathbf{H}_{rot,z}(-90°)$. Peu importe la séquence, nous pouvons faire le calcul pour obtenir la matrice $\mathbf{H}_b^a$ numériquement et ensuite en extraire le quaternion et la position comme nous allons voir dans la section suivante.

Il existe également une autre façon qui nous évitera de faire des calculs matriciels. Une fois que nous avons trouvé une expression comme celle de l'équation (6.16), nous pouvons l'interpréter comme une suite de transformations par rapport au référentiel « fixe » :

- rotation de $-90°$ autour de l'axe x_m, donc $\mathbf{H}_m^a = \mathbf{H}_{rot,x}(-90°)$;
- translation de 10 mm le long de l'axe z_m, donc $\mathbf{H}_m^a = \mathbf{H}_{rot,x}(-90°)\mathbf{H}_{trans}(0,0,10)$;
- rotation de $-135°$ autour de l'axe z_m, donc $\mathbf{H}_m^a = \mathbf{H}_{rot,x}(-90°)\mathbf{H}_{trans}(0,0,10)\mathbf{H}_{rot,z}(-135°)$.

Enfin, nous pouvons utiliser la fonction RelTool en cascade, comme montré dans l'exemple suivant :

Exemple 6.3 – Utilisation de la commande RelTool en cascade

```
MODULE Exemple()
    PERS    robtarget rtB := [[0,0,0],[1,0,0,0],[0,-1,0,0],[9E9,9E9,9E9,9E9,9E9,9E9]];

  rtB := RelTool(rtB, 0, 0, 0 \Rx:=-90);
  rtB := RelTool(rtB, 0, 0, 10);
  rtB := RelTool(rtB, 0, 0, 0 \Rz:=-135);
ENDMODULE
```

6.3 Représentation de l'orientation

L'autre raison pourquoi vous devez connaître les notions théoriques de la section précédente, c'est pour mieux comprendre les *angles d'Euler* et leurs multiples conventions et interprétations.

6.3.1 Angles d'Euler

Le théorème de rotation d'Euler stipule que tout déplacement d'un corps rigide, à l'exception d'une translation pure, peut être représenté par une rotation autour d'un certain axe, appelé *axe de rotation équivalant*. En conséquence, l'orientation d'un référentiel dans l'espace par rapport à un autre peut être décrite par trois paramètres au minimum. La représentation de l'orientation (dans l'espace) à trois paramètres la plus connue est celle des angles d'Euler.

Les angles d'Euler sont essentiellement trois angles, disons α, β, et γ, qui correspondent à une transformation composite de trois rotations pures (c'est-à-dire, autour d'un axe x, y ou z). Puisqu'il existe douze combinaisons différentes de produits de rotations pures non redondants, il y a douze conventions possibles pour les angles d'Euler :

$$\mathbf{H}_{xyz}(\alpha,\beta,\gamma) = \mathbf{H}_{rot,x}(\alpha)\,\mathbf{H}_{rot,y}(\beta)\,\mathbf{H}_{rot,z}(\gamma), \tag{6.17}$$
$$\mathbf{H}_{xzy}(\alpha,\beta,\gamma) = \mathbf{H}_{rot,x}(\alpha)\,\mathbf{H}_{rot,z}(\beta)\,\mathbf{H}_{rot,y}(\gamma), \tag{6.18}$$
$$\mathbf{H}_{yxz}(\alpha,\beta,\gamma) = \mathbf{H}_{rot,y}(\alpha)\,\mathbf{H}_{rot,x}(\beta)\,\mathbf{H}_{rot,z}(\gamma), \tag{6.19}$$
$$\mathbf{H}_{yzx}(\alpha,\beta,\gamma) = \mathbf{H}_{rot,y}(\alpha)\,\mathbf{H}_{rot,z}(\beta)\,\mathbf{H}_{rot,x}(\gamma), \tag{6.20}$$
$$\mathbf{H}_{zxy}(\alpha,\beta,\gamma) = \mathbf{H}_{rot,z}(\alpha)\,\mathbf{H}_{rot,x}(\beta)\,\mathbf{H}_{rot,y}(\gamma), \tag{6.21}$$
$$\mathbf{H}_{zyx}(\alpha,\beta,\gamma) = \mathbf{H}_{rot,z}(\alpha)\,\mathbf{H}_{rot,y}(\beta)\,\mathbf{H}_{rot,x}(\gamma), \tag{6.22}$$
$$\mathbf{H}_{xyx}(\alpha,\beta,\gamma) = \mathbf{H}_{rot,x}(\alpha)\,\mathbf{H}_{rot,y}(\beta)\,\mathbf{H}_{rot,x}(\gamma), \tag{6.23}$$
$$\mathbf{H}_{xzx}(\alpha,\beta,\gamma) = \mathbf{H}_{rot,x}(\alpha)\,\mathbf{H}_{rot,z}(\beta)\,\mathbf{H}_{rot,x}(\gamma), \tag{6.24}$$
$$\mathbf{H}_{yxy}(\alpha,\beta,\gamma) = \mathbf{H}_{rot,y}(\alpha)\,\mathbf{H}_{rot,x}(\beta)\,\mathbf{H}_{rot,y}(\gamma), \tag{6.25}$$
$$\mathbf{H}_{yzy}(\alpha,\beta,\gamma) = \mathbf{H}_{rot,y}(\alpha)\,\mathbf{H}_{rot,z}(\beta)\,\mathbf{H}_{rot,y}(\gamma), \tag{6.26}$$
$$\mathbf{H}_{zxz}(\alpha,\beta,\gamma) = \mathbf{H}_{rot,z}(\alpha)\,\mathbf{H}_{rot,x}(\beta)\,\mathbf{H}_{rot,z}(\gamma), \tag{6.27}$$
$$\mathbf{H}_{zyz}(\alpha,\beta,\gamma) = \mathbf{H}_{rot,z}(\alpha)\,\mathbf{H}_{rot,y}(\beta)\,\mathbf{H}_{rot,z}(\gamma). \tag{6.28}$$

Imaginons de nouveau deux référentiels qui coïncident initialement, $\mathcal{F}_0$ et $\mathcal{F}_m$. Pour chacune des douze conventions, on peut s'imaginer que le référentiel $\mathcal{F}_m$ effectue trois rotations. À la suite de la première rotation, le référentiel $\mathcal{F}_m$ coïncide avec un référentiel $\mathcal{F}_1$. À la suite de la deuxième rotation, le référentiel $\mathcal{F}_m$ coïncide avec un référentiel $\mathcal{F}_2$. Enfin, à la suite de la troisième rotation, le référentiel $\mathcal{F}_m$ coïncide avec un référentiel $\mathcal{F}_3$. C'est la pose de $\mathcal{F}_3$ par rapport à $\mathcal{F}_0$ qui est représentée par une des douze équations (6.17–6.28). Mais quel est l'ordre exacte de rotations dans chacune des douze conventions ?

Prenons, la première convention comme exemple. Elle peut correspondre à quatre séquences de rotations différentes, c'est-à-dire, à quatre interprétations possibles :

interprétation 1
- rotation de α autour de l'axe x_0 ($\mathbf{H}_{xyz}(\alpha,\beta,\gamma) = \mathbf{H}_{rot,x}(\alpha)$) ;
- rotation de β autour de l'axe y_1 ($\mathbf{H}_{xyz}(\alpha,\beta,\gamma) = \mathbf{H}_{rot,x}(\alpha)\mathbf{H}_{rot,y}(\beta)$) ;
- rotation de γ autour de l'axe z_2 ($\mathbf{H}_{xyz}(\alpha,\beta,\gamma) = \mathbf{H}_{rot,x}(\alpha)\,\mathbf{H}_{rot,y}(\beta)\,\mathbf{H}_{rot,z}(\gamma)$).

interprétation 2
- rotation de γ autour de l'axe z_0 ($\mathbf{H}_{xyz}(\alpha,\beta,\gamma) = \mathbf{H}_{rot,z}(\gamma)$) ;
- rotation de β autour de l'axe y_0 ($\mathbf{H}_{xyz}(\alpha,\beta,\gamma) = \mathbf{H}_{rot,y}(\beta)\mathbf{H}_{rot,z}(\gamma)$) ;
- rotation de α autour de l'axe x_0 ($\mathbf{H}_{xyz}(\alpha,\beta,\gamma) = \mathbf{H}_{rot,x}(\alpha)\,\mathbf{H}_{rot,y}(\beta)\,\mathbf{H}_{rot,z}(\gamma)$).

interprétation 3
- rotation de β autour de l'axe y_0 ($\mathbf{H}_{xyz}(\alpha,\beta,\gamma) = \mathbf{H}_{rot,y}(\beta)$) ;
- rotation de γ autour de l'axe z_1 ($\mathbf{H}_{xyz}(\alpha,\beta,\gamma) = \mathbf{H}_{rot,y}(\beta)\mathbf{H}_{rot,z}(\gamma)$) ;
- rotation de α autour de l'axe x_0 ($\mathbf{H}_{xyz}(\alpha,\beta,\gamma) = \mathbf{H}_{rot,x}(\alpha)\,\mathbf{H}_{rot,y}(\beta)\,\mathbf{H}_{rot,z}(\gamma)$).

interprétation 4
- rotation de γ autour de l'axe y_0 ($\mathbf{H}_{xyz}(\alpha,\beta,\gamma) = \mathbf{H}_{rot,y}(\beta)$) ;
- rotation de α autour de l'axe x_0 ($\mathbf{H}_{xyz}(\alpha,\beta,\gamma) = \mathbf{H}_{rot,x}(\alpha)\mathbf{H}_{rot,z}(\gamma)$) ;
- rotation de β autour de l'axe z_2 ($\mathbf{H}_{xyz}(\alpha,\beta,\gamma) = \mathbf{H}_{rot,x}(\alpha)\,\mathbf{H}_{rot,y}(\beta)\,\mathbf{H}_{rot,z}(\gamma)$).

De plus, il existe plusieurs façons de décrire chacune de quatre interprétations possibles. Des fois, des termes comme « nouveau référentiel », « référentiel précédent » ou « référentiel unique » sont utilisés.

D'autres fois, c'est un schéma qui est donné. Il est ainsi important de comprendre que les angles d'Euler ne sont jamais expliqués de la même façon. Pire encore, les différents fabricants de robots industriels et les différents développeurs de logiciels de CAO utilisent différentes conventions d'angles d'Euler. Voici une liste d'exemples concrets :

- les robots Kawasaki, Adept et Stäubli utilisent la convention décrite par l'équation (6.28) ;
- les robots Bosch utilisent la convention décrite par l'équation (6.21) ;
- les robots FANUC, KUKA et ABB utilisent la convention décrite par l'équation (6.22) ;
- les logiciels CATIA et SolidWorks utilisent la convention décrite par l'équation (6.27).

Pour mieux comprendre les angles d'Euler, nous vous conseillons fortement d'utiliser l'application EulerAngles pour iOS, ainsi que de vous fabriquer deux petits référentiels avec des cure-dents.

Comme vous avez probablement remarqué, nous ne donnons intentionnellement pas de noms à ces conventions, pour éviter toute confusion. Certains, par exemple, vont appeler la convention décrite par l'équation (6.22) « les angles d'Euler XYZ », alors que d'autres vont utiliser le terme « convention ZYX ».

Comme déjà mentionné, le contrôleur d'un robot ABB utilise plutôt les quaternions pour communiquer avec le programmeur du robot, mais l'orientation peut être aussi affichée sur le FlexPendant comme trois angles d'Euler selon la convention décrite par l'équation (6.22). De plus, RAPID offre les fonctions OrientZYX et EulerZYX qui servent à convertir les angles d'Euler (selon cette convention) en un quaternion et vice-versa.

Nous donnerons alors un peu plus de détails sur les angles d'Euler de la convention utilisée par ABB. L'équation pour la matrice homogène est :

$$\begin{aligned}\mathbf{H}_{zyx}(\alpha,\beta,\gamma) &= \mathbf{H}_{rot,z}(\alpha)\,\mathbf{H}_{rot,y}(\beta)\,\mathbf{H}_{rot,x}(\gamma)\\ &= \begin{bmatrix} \cos\alpha\cos\beta & -\sin\alpha\cos\gamma+\cos\alpha\sin\beta\sin\gamma & \sin\alpha\sin\gamma+\cos\alpha\sin\beta\cos\gamma & 0\\ \sin\alpha\cos\beta & \cos\alpha\cos\gamma+\sin\alpha\sin\beta\sin\gamma & -\cos\alpha\sin\gamma+\sin\alpha\sin\beta\cos\gamma & 0\\ -\sin\beta & \cos\beta\sin\gamma & \cos\beta\cos\gamma & 0\\ 0 & 0 & 0 & 1 \end{bmatrix}. \end{aligned} \tag{6.29}$$

C'est relativement rare que vous utiliserez cette équation en pratique. Ce qui vous arrivera beaucoup plus souvent, c'est de trouver la matrice homogène qui représente une pose et de vouloir en déduire les angles d'Euler (selon une convention donnée) afin de programmer votre robot. Bien sûr, dans certains situations simples, vous pourrez déduire les angles d'Euler sans effectuer aucun calcul, mais généralement, il faudra obtenir la matrice homogène d'abord. Voici alors les équations pour obtenir les trois angles d'Euler selon la convention utilisée par ABB, à partir d'une matrice homogène $\mathbf{H}$:

Cas 1 : $h_{3,1} \neq \pm 1$

$$\beta = \operatorname{atan2}\left(-h_{3,1}, \pm\sqrt{h_{1,1}^2+h_{2,1}^2}\right),\ \alpha = \operatorname{atan2}\left(\frac{h_{2,1}}{\cos\beta}, \frac{h_{1,1}}{\cos\beta}\right),\ \gamma = \operatorname{atan2}\left(\frac{h_{3,2}}{\cos\beta}, \frac{h_{3,3}}{\cos\beta}\right), \tag{6.30}$$

Cas 2 : $h_{3,1} = \pm 1$

$$\beta = -h_{3,1}90°,\ \gamma = \text{valeur arbitraire},\ \alpha = \operatorname{atan2}(-h_{3,1}h_{2,3}, h_{2,2}) - h_{3,1}\gamma, \tag{6.31}$$

où $h_{i,j}$ sont les éléments de la matrice homogène $\mathbf{H}$.

Nous verrons comment obtenir ces équations plus tard dans ce manuel. Pour l'instant, il faut seulement noter que lorsque l'axe x du référentiel dont nous voulons représenter l'orientation est parallèle à l'axe z du référentiel par rapport auquel nous nous référons, il existe une infinité de solutions pour les trois angles d'Euler selon la convention utilisée par ABB (cas 2). Dans tous les autres cas, il y a deux triplets possibles d'angles d'Euler, dans les plages $[-180°, 180°]$ (cas 1).

Le cas 2 s'appelle une *singularité de représentation*. Cette singularité existe dans toute représentation de l'orientation dans l'espace à trois paramètres. Elle est similaire à la discontinuité qui existe dans les coordonnées géographiques : au pôles Nord et Sud, la latitude est 90° ou −90°, respectivement, mais la longitude n'est pas définie (elle peut avoir une valeur arbitraire).

6.3.2 Quaternions (paramètres d'Euler-Rodrigues)

Pour éviter cette discontinuité dans la représentation de l'orientation dans l'espace à l'aide de trois paramètres, on utilise les quaternions, ou plus précisément les *paramètres d'Euler-Rodrigues*. Cette forme de représentation consiste en un vecteur normalisé de quatre scalaires. C'est le quaternion qui est utilisé dans les contrôleurs de robots, car il est non seulement plus compact que la matrice de rotation, mais aussi moins sensible aux erreurs approximatives. En plus, lors d'une interpolation entre deux orientations différentes, les éléments du quaternion changent sans interruption, évitant les sauts inhérents aux paramétrisations tridimensionnelles tels que les angles d'Euler.

Par contre, le quaternion est rarement utilisé comme moyen de communication entre l'usager et le contrôleur du robot, car il n'est pas très intuitive. Le langage de programmation RAPID est probablement le seul à utiliser les quaternions. Comme déjà discuté, dans ce langage, une orientation est enregistrée dans une variable de type orient qui est une liste de quatre valeurs de type num : $[q_1, q_2, q_3, q_4]$. Voici quelques formules qui rendent le quaternion un peu plus intuitif :

$$q_1 = \cos(\theta/2),\ q_2 = k_x \sin(\theta/2),\ q_3 = k_y \sin(\theta/2),\ q_4 = k_z \sin(\theta/2), \tag{6.32}$$

où $\mathbf{k} = [k_x, k_y, k_z]^T$ est le vecteur unitaire le long de l'axe de rotation équivalant et θ est l'angle de rotation autour de cet axe. Maintenant, il devient un peu plus facile de se rappeler que l'orientation nulle (c'est-à-dire, quand les deux référentiels coïncident) est représentée par le quaternion $[1, 0, 0, 0]$, car $\theta = 0$. Par contre, même une simple rotation de 90° autour de l'axe x n'est pas facile à déceler ou entrer dans une variable de type orient, car le quaternion associé est [0.707106781, 0.707106781, 0, 0].

Enfin, les formules dont vous aurez probablement besoin sont celles qui nous permettent de trouver le quaternion à partir d'une matrice homogène $\mathbf{H}$:

$$q_1 = \frac{\sqrt{1 + h_{1,1} + h_{2,2} + h_{3,3}}}{2}, \tag{6.33}$$

$$q_2 = \mathrm{sgn}(h_{3,2} - h_{2,3})\frac{\sqrt{1 + h_{1,1} - h_{2,2} - h_{3,3}}}{2}, \tag{6.34}$$

$$q_3 = \mathrm{sgn}(h_{1,3} - h_{3,1})\frac{\sqrt{1 - h_{1,1} + h_{2,2} - h_{3,3}}}{2}, \tag{6.35}$$

$$q_4 = \mathrm{sgn}(h_{2,1} - h_{1,2})\frac{\sqrt{1 - h_{1,1} - h_{2,2} + h_{3,3}}}{2}. \tag{6.36}$$

Comme déjà mentionné, en RAPID, un quaternion peut être obtenu directement à partir des trois angles d'Euler selon la convention décrite dans l'équation (6.29), tel que démontré dans l'exemple 6.4. À l'inverse, les angles d'Euler peuvent être obtenus à l'aide de la fonction EulerZYX, tel qu'illustré dans l'exemple 6.5.

Exemple 6.4 – Fonction OrientZYX

```
MODULE Exemple()
    VAR num anglex;
    VAR num angley;
    VAR num anglez;
    VAR orient quat;

  quat := OrientZYX(anglex, angley, anglex);
ENDMODULE
```

Exemple 6.5 – Fonction EulerZYX

```
MODULE Exemple()
    VAR num anglex;
    VAR num angley;
    VAR num anglez;
    VAR orient quat;

    anglex := EulerZYX(\X, quat);
    angley := EulerZYX(\Y, quat);
    anglez := EulerZYX(\Z, quat);
ENDMODULE
```

6.4 Exercices

6.1. Calculer les éléments des différentes matrices homogènes demandées.

(a) Dans le cas des deux référentiels montrés à la figure 6.2, trouver les matrices homogènes de transformation de base $\mathbf{H}_{trans}$ et $\mathbf{H}_{rot,z}$, et calculer la matrice résultante $\mathbf{H}_1^0$.

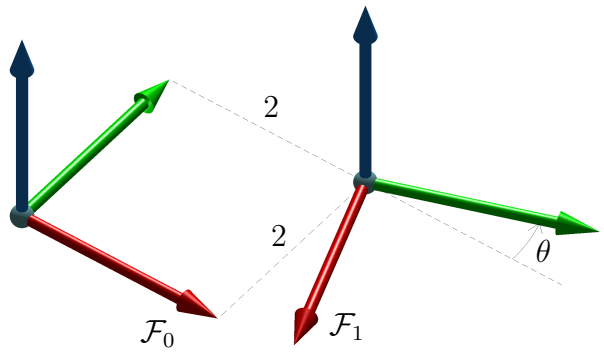


FIGURE 6.2 – Exercice 6.1(a).

(b) Dans le cas des deux référentiels montrés à la figure 6.3, trouver les matrices homogènes de transformation de base permettant de calculer $\mathbf{H}_1^0$.

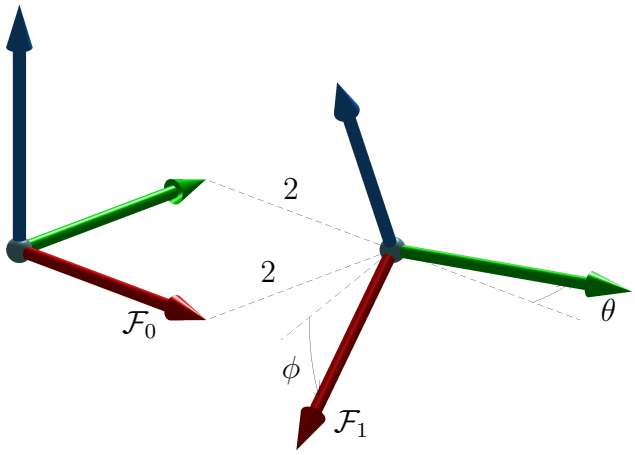


FIGURE 6.3 – Exercoce 6.1(b).

(c) Dans le cas des trois référentiels montrés à la figure 6.4, trouver les matrices suivantes :

i. $\mathbf{H}_1^0$;
ii. $\mathbf{H}_2^1$;
iii. $\mathbf{H}_2^0$, en fonction de $\mathbf{H}_1^0$ et $\mathbf{H}_2^1$.

(d) Dans le cas des trois référentiels montrés à la figure 6.5, trouver les matrices suivantes :

i. $\mathbf{H}_1^0$;
ii. $\mathbf{H}_2^0$;
iii. $\mathbf{H}_2^1$, en fonction de $\mathbf{H}_1^0$ et $\mathbf{H}_2^0$.

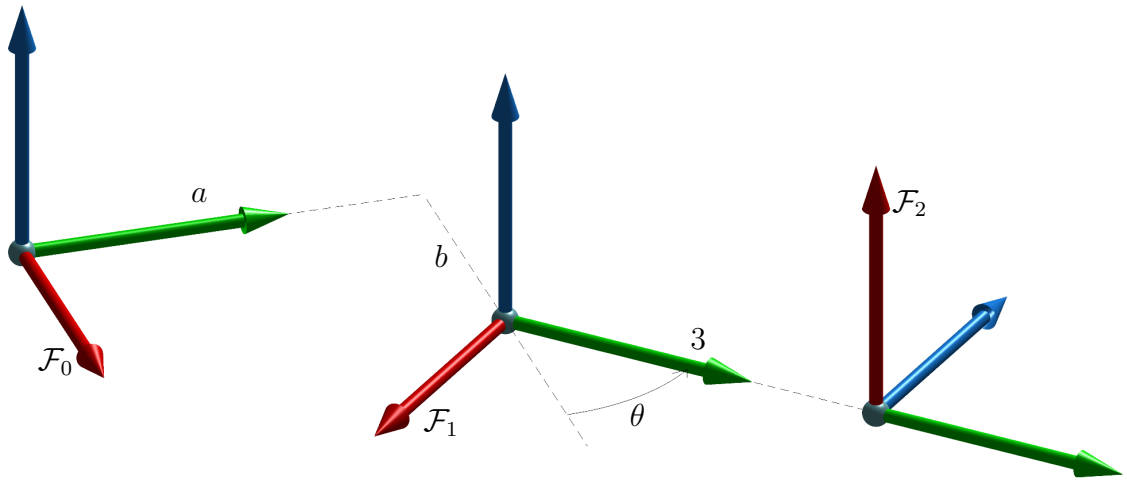


FIGURE 6.4 – Exercice 6.1(c).

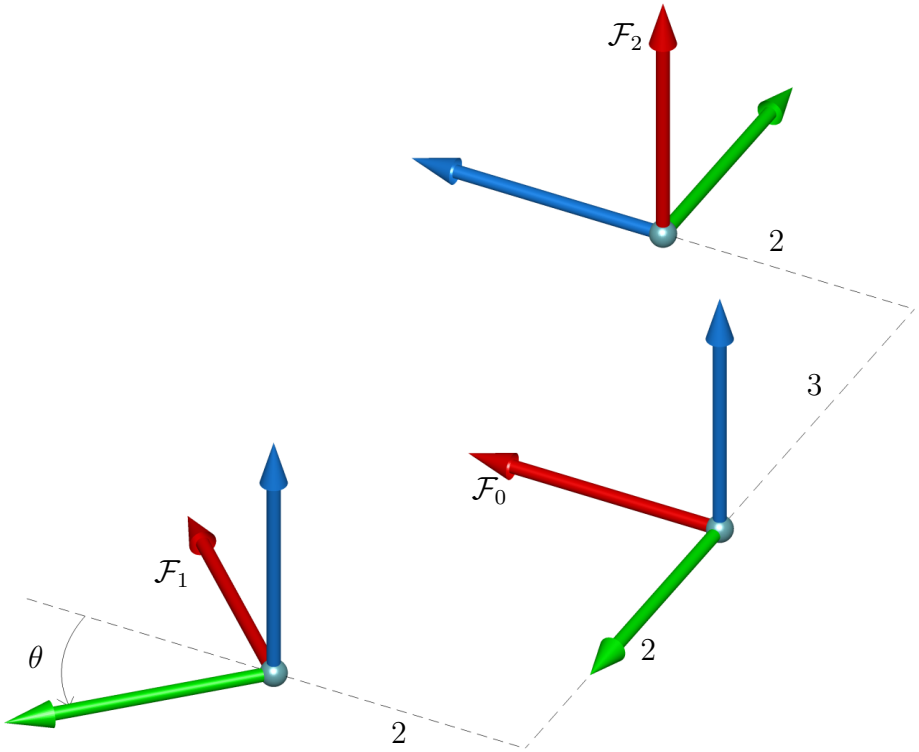


FIGURE 6.5 – Exercice 6.1(d).

6.2. Soit deux référentiels $\mathcal{F}_0$ et $\mathcal{F}_1$. À l'aide des matrices homogènes de transformation de base, faire les transformations demandées si la matrice $\mathbf{H}_1^0$ est :

$$\mathbf{H}_1^0 = \begin{bmatrix} 1 & 0 & 0 & 3 \\ 0 & 1 & 0 & 5 \\ 0 & 0 & 1 & 7 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{6.37}$$

Vérifier vos résultats à l'aide d'un graphique. Écrire la routine RAPID qui modifie la matrice $\mathbf{H}_1^0$.

(a) Faire une rotation de 30° autour de l'axe z_0 du référentiel $\mathcal{F}_0$.

Solution :

$$
\begin{aligned}
\mathbf{H}_2^0 &= \mathbf{H}_{rot,z}(30^\circ)\mathbf{H}_1^0 \\
&= \begin{bmatrix} \cos 30^\circ & -\sin 30^\circ & 0 & 0 \\ \sin 30^\circ & \cos 30^\circ & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 3 \\ 0 & 1 & 0 & 5 \\ 0 & 0 & 1 & 7 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\
&= \begin{bmatrix} \cos 30^\circ & -\sin 30^\circ & 0 & 3\cos 30^\circ - 5\sin 30^\circ \\ \sin 30^\circ & \cos 30^\circ & 0 & 3\sin 30^\circ + 5\cos 30^\circ \\ 0 & 0 & 1 & 7 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\
&= \begin{bmatrix} 0.866 & -0.500 & 0 & 0.098 \\ 0.500 & 0.866 & 0 & 5.830 \\ 0 & 0 & 1 & 7 \\ 0 & 0 & 0 & 1 \end{bmatrix}.
\end{aligned}
$$

La vérification graphique est montrée à la figure 6.6.

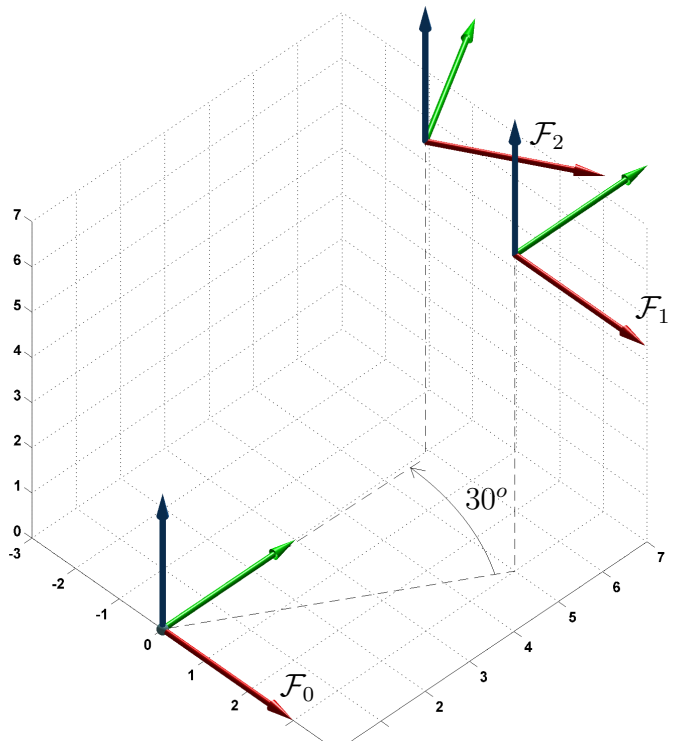


FIGURE 6.6 – Réponse à l'exercice 6.2(a).

Enfin voici le programme RAPID nécessaire pour faire le calcul de l'exercice 6.2(a), avec le résultat de l'exécution montré à la figure 6.7.

Exemple 6.6 – Code RAPID pour l'exercice 6.2(a)

```
MODULE Ex6_2a_solution

VAR pose V;
VAR pose Rot;
VAR pose reponse;
```

```
PROC main()
  ! le quaternion equivalent a aucune rotation est [1,0,0,0]
  H10:=[[3,5,7],[1,0,0,0]];
  Rot:=[[0,0,0],OrientZYX(30,0,0)];

  ! calcul
  reponse := PoseMult(Rot,H10);

  ! affiche le resultat
  TPErase;
  TPWrite "reponse ="\Pos:=reponse.trans;

ENDPROC
ENDMODULE
```

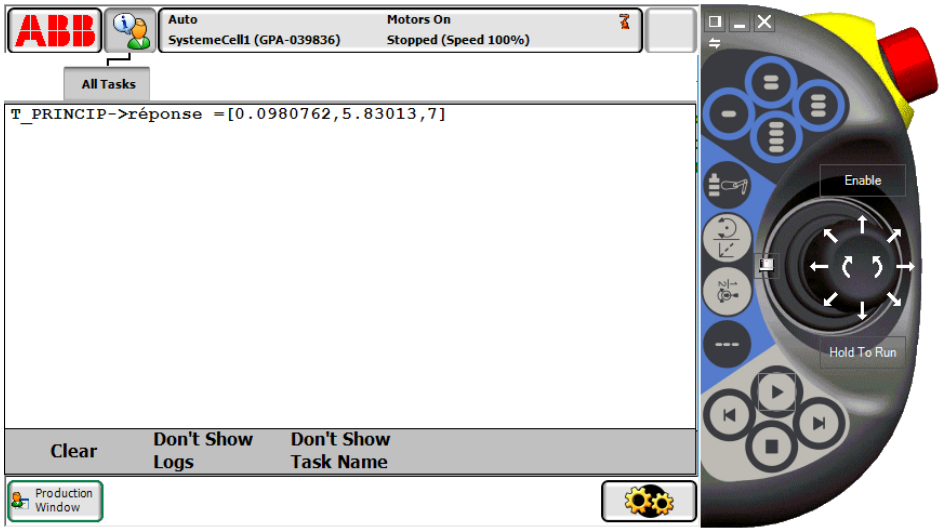


FIGURE 6.7 – Résultat de l'exécution du programme RAPID de l'exercice 6.2(a).

(b) Faire une rotation du référentiel $\mathcal{F}_1$ de $-25°$ autour de l'axe x_0 du référentiel $\mathcal{F}_0$.

(c) Faire une rotation du référentiel $\mathcal{F}_1$ de $40°$ autour de son axe y_1.

(d) Faire une translation du référentiel $\mathcal{F}_1$ de 8 mm le long de l'axe y_0 du référentiel $\mathcal{F}_0$.

(e) Faire une rotation du référentiel $\mathcal{F}_1$ de $30°$ autour de l'axe x_0 du référentiel $\mathcal{F}_0$, suivi d'une translation de 3 mm le long de l'axe y_0 du référentiel $\mathcal{F}_0$.

(f) Faire une rotation du référentiel $\mathcal{F}_1$ de $30°$ autour de l'axe y_0 du référentiel $\mathcal{F}_0$, suivi d'une translation de 5 mm le long de l'axe z du nouveau référentiel.

(g) Faire une translation du référentiel $\mathcal{F}_1$ de -2 mm le long de l'axe x_0 du référentiel $\mathcal{F}_0$, suivi d'une rotation de $15°$ autour de l'axe y_2 du nouveau référentiel.

6.3. Soit la cellule robotique montrée à la figure 6.8. Poser les équations qui permetraient de trouver les matrices homogènes demandées.

(a) Poser l'équation pour $\mathbf{H}^{cam}_{table}$, en fonction de $\mathbf{H}^{0}_{table}$ et $\mathbf{H}^{0}_{cam}$.

Solution : $\mathbf{H}^{cam}_{table} = (\mathbf{H}^{0}_{cam})^{-1}\mathbf{H}^{0}_{table}$.

(b) Poser l'équation pour $\mathbf{H}^{table}_{pièce}$, en fonction de $\mathbf{H}^{table}_{cam}$ et $\mathbf{H}^{cam}_{pièce}$.

(c) Poser l'équation pour $\mathbf{H}^{R}_{table}$, en fonction de $\mathbf{H}^{0}_{table}$ et $\mathbf{H}^{R}_{0}$.

(d) Poser l'équation pour $\mathbf{H}^{0}_{cam}$, en fonction de $\mathbf{H}^{R}_{0}$, $\mathbf{H}^{R}_{table}$ et $\mathbf{H}^{table}_{cam}$.

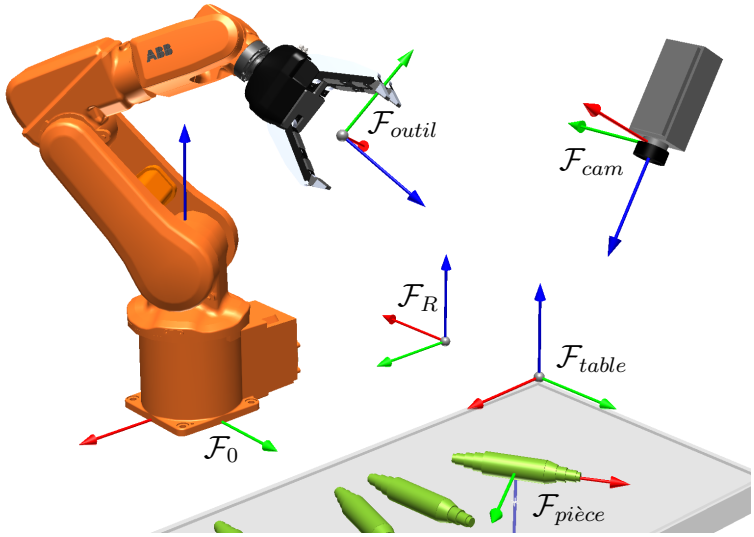


FIGURE 6.8 – Exercice 6.3.

6.4. Dessiner les référentiels (a) $\mathcal{F}_1$ et (b) $\mathcal{F}_2$ dont les poses par rapport au référentiel $\mathcal{F}_0$ sont définies par les deux matrices homogènes suivantes :

$$\mathbf{H}_1^0 = \begin{bmatrix} 0 & 0 & -1 & 20 \\ 1 & 0 & 0 & -30 \\ 0 & -1 & 0 & 55 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad \mathbf{H}_2^0 = \begin{bmatrix} 0.866 & 0 & -0.5 & 10 \\ 0.5 & 0 & 0.866 & -10 \\ 0 & -1 & 0 & 5 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

6.5. Un référentiel $\mathcal{F}_b$ coïncide au départ avec un référentiel $\mathcal{F}_a$. On fait tourner $\mathcal{F}_b$ de 30° autour de l'axe z_a du référentiel $\mathcal{F}_a$, puis de 45° autour de l'axe x du nouveau référentiel.

(a) Donner la matrice homogène $\mathbf{H}_b^a$.

(b) Écrire les instructions RAPID pour réaliser ce calcul, si $\mathcal{F}_a$ est le référentiel de l'atelier et $\mathcal{F}_b$ correspond à un robtarget.

6.6. Un référentiel est obtenu par une série de transformations à partir d'un référentiel de départ. Suite à la liste des transformations suivantes, définir la matrice homogène décrivant la pose du référentiel.

(a) Rotation de 180° autour de l'axe y du référentiel de départ.

(b) Translation de $(7, 12, -4)$ par rapport au référentiel de départ.

(c) Rotation de 30° autour de l'axe x du nouveau référentiel.

Écrire le programme RAPID réalisant ce calcul, à l'aide de la fonction RelTool.

6.7. Le dessin de la figure 6.9 illustre les référentiels wfeuille et wtable ainsi que le référentiel de l'outil du robot, tpince. La feuille sur la table est de format A4 (210 mm × 297 mm). Dessiner l'outil sur la figure, à la suite des séquences de commandes suivantes :

(a) 1er robtarget :

```
P1:=[[297,210,0],OrientZYX(180,0,0),[0,-1,2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
MoveL P1, v100, fine, tpince\wobj:=wtable;
```

(b) 2e robtarget :

```
P2:=[[297,210,0],OrientZYX(180,0,0),[0,-1,2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
MoveL P2, v100, fine, tpince\wobj:=wfeuille;
```

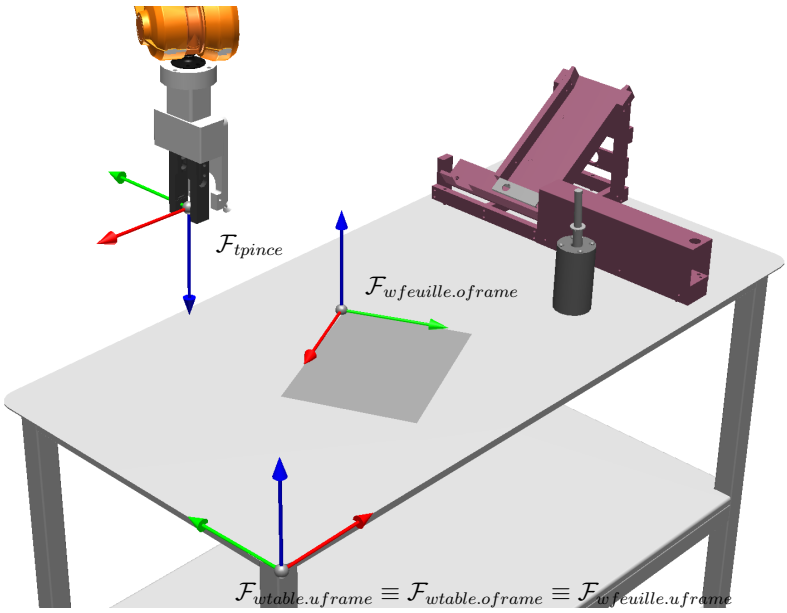


FIGURE 6.9 – Exercice 6.7.

(c) 3e robtarget :

```
P3:=[[297,210,0],OrientZYX(180,0,0),[0,-1,2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
MoveL RelTool(P3,0,0,-200 \Rz:=90), v100, fine, tpince\wobj:=wfeuille;
```

6.8. Soit la pose suivante de l'effecteur d'un robot ABB :

$$\begin{array}{ccc} \text{X} = 500 \text{ mm} & \text{Y} = -300 \text{ mm} & \text{Z} = 850 \text{ mm} \\ \text{AngleX} = 180° & \text{AngleY} = 0° & \text{AngleZ} = 90° \end{array}$$

On utilise le boîtier de commande et, selon le référentiel de l'outil, on effectue d'abord une rotation de 90° de l'outil autour de son axe x_{outil}, puis une translation $(150, 400, -100)$ selon le référentiel de l'outil. Trouver les valeurs X, Y, Z, AngleX, AngleY et AngleZ qui seront affichées à l'écran du FlexPendant après ces opérations.

6.9. La figure 6.10 montre le référentiel d'une buse de soudage et celui associé à la bride de montage, soit tool0. Calculer la matrice homogène $\mathbf{H}_{tbuse}^{tool0}$ sachant que le référentiel désiré a d'abord subi une rotation de $-30°$ autour de l'axe x_{tool0}, puis une rotation de 90° autour du nouvel axe z_1, et finalement une translation de 210 mm le long de l'axe z_{tool0} et de 120 mm le long de l'axe y_{tool0}.

6.10. Soit les quatre référentiels présentés à la figure 6.11. Donner les poses des référentiels $\mathcal{F}_a$, $\mathcal{F}_b$ et $\mathcal{F}_c$ par rapport à $\mathcal{F}_0$.

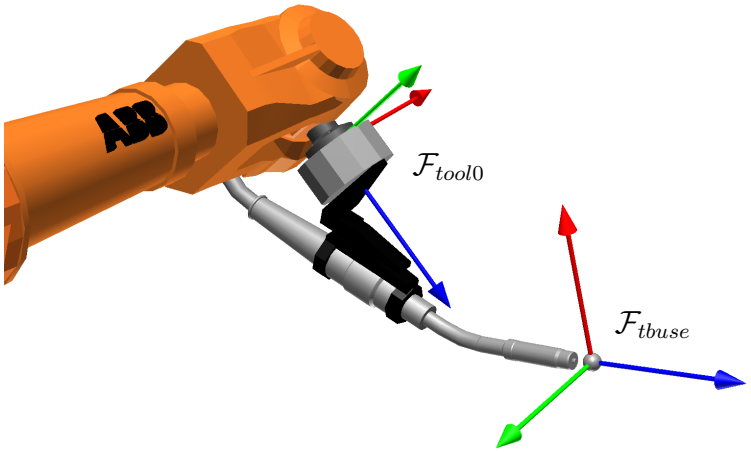


FIGURE 6.10 – Exercice 6.9.

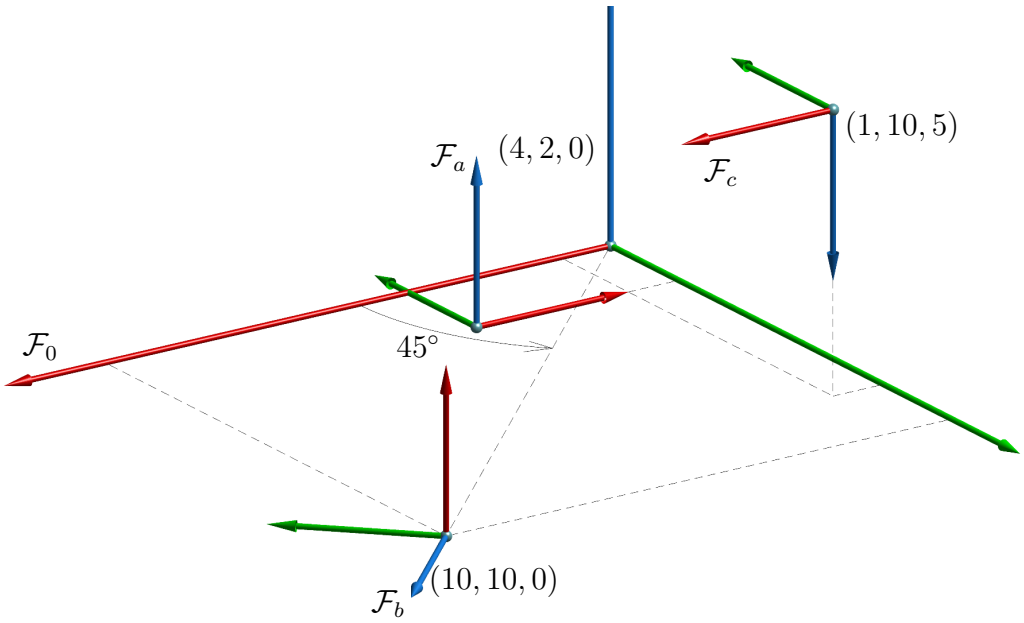


FIGURE 6.11 – Exercice 6.10.

6.5 Réponses aux exercices

6.1. (a)

$$\begin{aligned}\mathbf{H}_1^0 &= \mathbf{H}_{trans}(2,2,0)\mathbf{H}_{rot,z}(\theta-90^\circ)\\ &= \begin{bmatrix}1&0&0&2\\0&1&0&2\\0&0&1&0\\0&0&0&1\end{bmatrix}\begin{bmatrix}\cos(\theta-90^\circ)&-\sin(\theta-90^\circ)&0&0\\\sin(\theta-90^\circ)&\cos(\theta-90^\circ)&0&0\\0&0&1&0\\0&0&0&1\end{bmatrix} = \begin{bmatrix}\sin\theta&\cos\theta&0&2\\-\cos\theta&\sin\theta&0&2\\0&0&1&0\\0&0&0&1\end{bmatrix}\end{aligned}$$

(b)

$$\begin{aligned}\mathbf{H}_1^0 &= \mathbf{H}_{trans}(2,2,0)\mathbf{H}_{rot,z}(\theta-90^\circ)\mathbf{H}_{rot,y}(\phi)\\ &= \begin{bmatrix}1&0&0&2\\0&1&0&2\\0&0&1&0\\0&0&0&1\end{bmatrix}\begin{bmatrix}\cos(\theta-90^\circ)&-\sin(\theta-90^\circ)&0&0\\\sin(\theta-90^\circ)&\cos(\theta-90^\circ)&0&0\\0&0&1&0\\0&0&0&1\end{bmatrix}\begin{bmatrix}\cos\phi&0&\sin\phi&0\\0&1&0&0\\-\sin\phi&0&\cos\phi&0\\0&0&0&1\end{bmatrix}\\ &= \begin{bmatrix}\sin\theta\cos\phi&\cos\theta&\sin\theta\sin\phi&2\\-\cos\theta\cos\phi&\sin\theta&-\cos\theta\sin\phi&2\\-\sin\phi&0&\cos\phi&0\\0&0&0&1\end{bmatrix}\end{aligned}$$

(c) i.

$$\begin{aligned}\mathbf{H}_1^0 &= \mathbf{H}_{trans}(b,a,0)\mathbf{H}_{rot,z}(\theta-90^\circ)\\ &= \begin{bmatrix}1&0&0&b\\0&1&0&a\\0&0&1&0\\0&0&0&1\end{bmatrix}\begin{bmatrix}\cos(\theta-90^\circ)&-\sin(\theta-90^\circ)&0&0\\\sin(\theta-90^\circ)&\cos(\theta-90^\circ)&0&0\\0&0&1&0\\0&0&0&1\end{bmatrix} = \begin{bmatrix}\sin\theta&\cos\theta&0&b\\-\cos\theta&\sin\theta&0&a\\0&0&1&0\\0&0&0&1\end{bmatrix}\end{aligned}$$

ii.

$$\begin{aligned}\mathbf{H}_2^1 &= \mathbf{H}_{trans}(0,3,0)\mathbf{H}_{rot,y}(-90^\circ)\\ &= \begin{bmatrix}1&0&0&0\\0&1&0&3\\0&0&1&0\\0&0&0&1\end{bmatrix}\begin{bmatrix}\cos(-90^\circ)&0&\sin(-90^\circ)&0\\0&1&0&0\\-\sin(-90^\circ)&0&\cos(-90^\circ)&0\\0&0&0&1\end{bmatrix} = \begin{bmatrix}0&0&-1&0\\0&1&0&3\\1&0&0&0\\0&0&0&1\end{bmatrix}\end{aligned}$$

iii.

$$\begin{aligned}\mathbf{H}_2^0 &= \mathbf{H}_1^0\mathbf{H}_2^1\\ &= \begin{bmatrix}\cos(\theta-90^\circ)&-\sin(\theta-90^\circ)&0&b\\\sin(\theta-90^\circ)&\cos(\theta-90^\circ)&0&a\\0&0&1&0\\0&0&0&1\end{bmatrix}\begin{bmatrix}0&0&-1&0\\0&1&0&3\\1&0&0&0\\0&0&0&1\end{bmatrix}\\ &= \begin{bmatrix}0&\cos\theta&-\sin\theta&b+3\cos\theta\\0&\sin\theta&\cos\theta&a+3\sin\theta\\1&0&0&0\\0&0&0&1\end{bmatrix}\end{aligned}$$

(d) i.

$$\begin{aligned}\mathbf{H}_1^0 &= \mathbf{H}_{trans}(2,2,0)\mathbf{H}_{rot,z}(\theta-90^\circ)\\ &= \begin{bmatrix}1&0&0&2\\0&1&0&2\\0&0&1&0\\0&0&0&1\end{bmatrix}\begin{bmatrix}\cos(\theta-90^\circ)&-\sin(\theta-90^\circ)&0&0\\\sin(\theta-90^\circ)&\cos(\theta-90^\circ)&0&0\\0&0&1&0\\0&0&0&1\end{bmatrix}=\begin{bmatrix}\sin\theta&\cos\theta&0&2\\-\cos\theta&\sin\theta&0&2\\0&0&1&0\\0&0&0&1\end{bmatrix}\end{aligned}$$

ii.

$$\begin{aligned}\mathbf{H}_2^0 &= \mathbf{H}_{trans}(2,-3,0)\mathbf{H}_{rot,z}(180^\circ)\mathbf{H}_{rot,y}(-90^\circ)\\ &= \begin{bmatrix}1&0&0&2\\0&1&0&-3\\0&0&1&0\\0&0&0&1\end{bmatrix}\begin{bmatrix}\cos 180^\circ&-\sin 180^\circ&0&0\\\sin 180^\circ&\cos 180^\circ&0&0\\0&0&1&0\\0&0&0&1\end{bmatrix}\begin{bmatrix}\cos(-90^\circ)&0&\sin(-90^\circ)&0\\0&1&0&0\\-\sin(-90^\circ)&0&\cos(-90^\circ)&0\\0&0&0&1\end{bmatrix}\\ &= \begin{bmatrix}0&0&1&2\\0&-1&0&-3\\1&0&0&0\\0&0&0&1\end{bmatrix}\end{aligned}$$

iii.

$$\begin{aligned}\mathbf{H}_2^1 &= \mathbf{H}_0^1\mathbf{H}_2^0 = (\mathbf{H}_1^0)^{-1}\mathbf{H}_2^0\\ &= \begin{bmatrix}\sin\theta&-\cos\theta&0&-2\sin\theta+2\cos\theta\\\cos\theta&\sin\theta&0&-2\sin\theta-2\cos\theta\\0&0&1&0\\0&0&0&1\end{bmatrix}\begin{bmatrix}0&0&1&2\\0&-1&0&-3\\1&0&0&0\\0&0&0&1\end{bmatrix}\\ &= \begin{bmatrix}0&\cos\theta&\sin\theta&+5\cos\theta\\0&-\sin\theta&\cos\theta&-5\sin\theta\\1&0&0&0\\0&0&0&1\end{bmatrix}\end{aligned}$$

6.2. (a) Déjà solutionné

(b)

$$\begin{aligned}\mathbf{H}_2^0 &= \mathbf{H}_{rot,x}(-25^\circ)\mathbf{H}_1^0\\ &= \begin{bmatrix}1&0&0&0\\0&\cos(-25^\circ)&-\sin(-25^\circ)&0\\0&\sin(-25^\circ)&\cos(-25^\circ)&0\\0&0&0&1\end{bmatrix}\begin{bmatrix}1&0&0&3\\0&1&0&5\\0&0&1&7\\0&0&0&1\end{bmatrix}\\ &= \begin{bmatrix}1&0&0&3\\0&0.906&0.422&7.489\\0&-0.422&0.906&4.231\\0&0&0&1\end{bmatrix}\end{aligned}$$

Exemple 6.7 – Code RAPID pour l'exercice 6.2(b)

```
MODULE Ex6_2b_solution

VAR pose V;
VAR pose Rot;
VAR pose reponse;

PROC main()
  H10:=[[3,5,7],[1,0,0,0]];
  Rot:=[[0,0,0],OrientZYX(0,0,-25)];

  ! calcul
  reponse := PoseMult(Rot,H10);

  ! affiche le resultat
  TPErase;
  TPWrite "reponse ="\Pos:=reponse.trans;

ENDPROC
ENDMODULE
```

(c)

$$\begin{aligned}
\mathbf{H}_2^0 &= \mathbf{H}_1^0 \mathbf{H}_{rot,y}(40°) \\
&= \begin{bmatrix} 1 & 0 & 0 & 3 \\ 0 & 1 & 0 & 5 \\ 0 & 0 & 1 & 7 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos 40° & 0 & \sin 40° & 0 \\ 0 & 1 & 0 & 0 \\ -\sin 40° & 0 & \cos 40° & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\
&= \begin{bmatrix} \cos 40° & 0 & \sin 40° & 3 \\ 0 & 1 & 0 & 5 \\ -\sin 40° & 0 & \cos 40° & 7 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0.766 & 0 & 0.642 & 3 \\ 0 & 1 & 0 & 5 \\ -0.642 & 0 & 0.766 & 7 \\ 0 & 0 & 0 & 1 \end{bmatrix}
\end{aligned}$$

Exemple 6.8 – Code RAPID pour l'exercice 6.2(c)

```
MODULE Ex6_2c_solution

VAR pose V;
VAR pose Rot;
VAR pose reponse;

PROC main()
  H10:=[[3,5,7],[1,0,0,0]];
  Rot:=[[0,0,0],OrientZYX(0,40,0)];

  ! calcul
  reponse := PoseMult(H10,Rot);

  ! affiche le resultat
  TPErase;
  TPWrite "reponse ="\Pos:=reponse.trans;

ENDPROC
ENDMODULE
```

(d)

$$\mathbf{H}_2^0 = \mathbf{H}_{trans}(0,8,0)\mathbf{H}_1^0 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 8 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 3 \\ 0 & 1 & 0 & 5 \\ 0 & 0 & 1 & 7 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 3 \\ 0 & 1 & 0 & 13 \\ 0 & 0 & 1 & 7 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Exemple 6.9 – Code RAPID pour l'exercice 6.2(d)

```
MODULE Ex6_2d_solution

VAR pose V;
VAR pose T;
VAR pose reponse;

PROC main()
  H10:=[[3,5,7],[1,0,0,0]];
  T:=[[0,8,0],[1,0,0,0]];

  ! calcul
  reponse := PoseMult(T,H10);

  ! affiche le resultat
  TPErase;
  TPWrite "reponse ="\Pos:=reponse.trans;

ENDPROC
ENDMODULE
```

(e)

$$\mathbf{H}_3^0 = \mathbf{H}_{trans}(0,3,0)\mathbf{H}_{rot,x}(30^\circ)\mathbf{H}_1^0$$
$$= \begin{bmatrix}1&0&0&0\\0&1&0&3\\0&0&1&0\\0&0&0&1\end{bmatrix}\begin{bmatrix}1&0&0&0\\0&\cos 30^\circ&-\sin 30^\circ&0\\0&\sin 30^\circ&\cos 30^\circ&0\\0&0&0&1\end{bmatrix}\begin{bmatrix}1&0&0&3\\0&1&0&5\\0&0&1&7\\0&0&0&1\end{bmatrix} = \begin{bmatrix}1&0&0&3\\0&0.866&-0.5&3.83\\0&0.5&0.866&8.562\\0&0&0&1\end{bmatrix}$$

Exemple 6.10 – Code RAPID pour l'exercice 6.2(e)

```
MODULE Ex6_2e_solution

VAR pose V;
VAR pose T;
VAR pose Rot;
VAR pose reponse;

PROC main()
  H10:=[[3,5,7],[1,0,0,0]];
  T:=[[0,3,0],[1,0,0,0]];
  Rot:=[[0,0,0],OrientZYX(0,0,30)];

  ! calcul
  reponse := PoseMult(T,PoseMult(Rot,H10));

  ! affiche le resultat
  TPErase;
  TPWrite "reponse ="\Pos:=reponse.trans;

ENDPROC
ENDMODULE
```

(f)

$$\mathbf{H}_1^0 = \mathbf{H}_{rot,y}(30^\circ)\mathbf{H}_1^0\mathbf{H}_{trans}(0,0,5)$$
$$= \begin{bmatrix}\cos 30^\circ&0&\sin 30^\circ&0\\0&1&0&0\\-\sin 30^\circ&0&\cos 30^\circ&0\\0&0&0&1\end{bmatrix}\begin{bmatrix}1&0&0&3\\0&1&0&5\\0&0&1&7\\0&0&0&1\end{bmatrix}\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&1&5\\0&0&0&1\end{bmatrix} = \begin{bmatrix}0.866&0&0.5&8.598\\0&1&0&5\\-0.5&0&0.866&8.892\\0&0&0&1\end{bmatrix}$$

Exemple 6.11 – Code RAPID pour l'exercice 6.2(f)

```
MODULE Ex6_2f_solution

VAR pose V;
VAR pose T;
VAR pose Rot;
VAR pose reponse;

PROC main()
  H10:=[[3,5,7],[1,0,0,0]];
  T:=[[0,0,5],[1,0,0,0]];
  Rot:=[[0,0,0],OrientZYX(0,30,0)];

  ! calcul
  reponse := PoseMult(PoseMult(Rot,H10),T);

  ! affiche le resultat
  TPErase;
  TPWrite "reponse ="\Pos:=reponse.trans;

ENDPROC
ENDMODULE
```

(g)

$$\begin{aligned}\mathbf{H}_1^0 &= \mathbf{H}_{trans}(-2,0,0)\mathbf{H}_1^0\mathbf{H}_{rot,y}(15^\circ)\\ &= \begin{bmatrix}1&0&0&-2\\0&1&0&0\\0&0&1&0\\0&0&0&1\end{bmatrix}\begin{bmatrix}1&0&0&3\\0&1&0&5\\0&0&1&7\\0&0&0&1\end{bmatrix}\begin{bmatrix}\cos 15^\circ&0&\sin 15^\circ&0\\0&1&0&0\\-\sin 15^\circ&0&\cos 15^\circ&0\\0&0&0&1\end{bmatrix} = \begin{bmatrix}0.965&0&0.258&1\\0&1&0&5\\-0.258&0&0.965&7\\0&0&0&1\end{bmatrix}\end{aligned}$$

Exemple 6.12 – Code RAPID pour l'exercice 6.2(f)

```
MODULE Ex6_2f_solution

VAR pose V;
VAR pose T;
VAR pose Rot;
VAR pose reponse;

PROC main()
  H10:=[[3,5,7],[1,0,0,0]];
  T:=[[-2,0,0],[1,0,0,0]];
  Rot:=[[0,0,0],OrientZYX(0,15,0)];

  ! calcul
  reponse := PoseMult(PoseMult(T,H10),Rot);

  ! affiche le resultat
  TPErase;
  TPWrite "reponse ="\Pos:=reponse.trans;

ENDPROC
ENDMODULE
```

6.3. (a) Déjà solutionné

(b) $\mathbf{H}_{pièce}^{table} = \mathbf{H}_{cam}^{table}\mathbf{H}_{pièce}^{cam}$

(c) $\mathbf{H}_{table}^{R} = \mathbf{H}_{0}^{R}\mathbf{H}_{table}^{0}$

(d) $\mathbf{H}_{cam}^{0} = (\mathbf{H}_{0}^{R})^{-1}\mathbf{H}_{table}^{R}\mathbf{H}_{cam}^{table}$

6.4. Les réponses sont données aux figures 6.12 et 6.13.

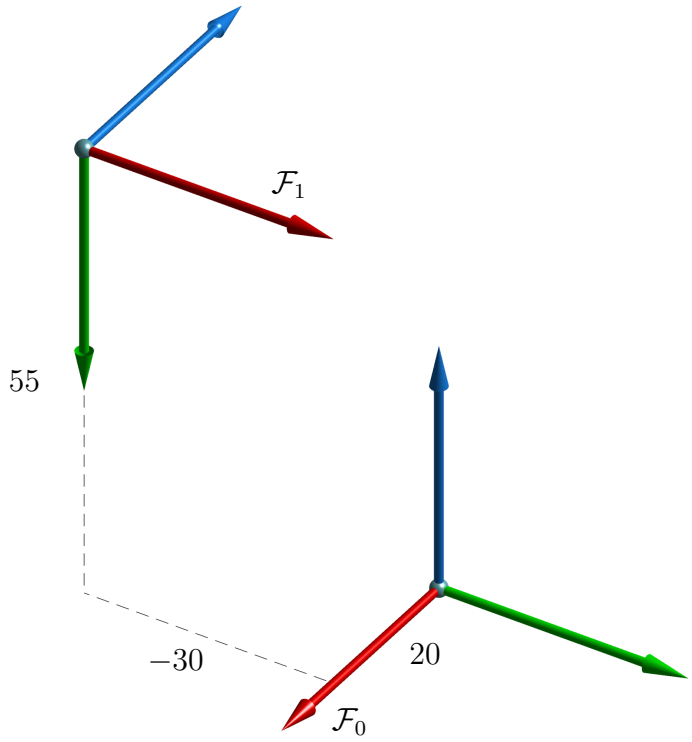


FIGURE 6.12 – Réponse à l'exercice 6.4(a).

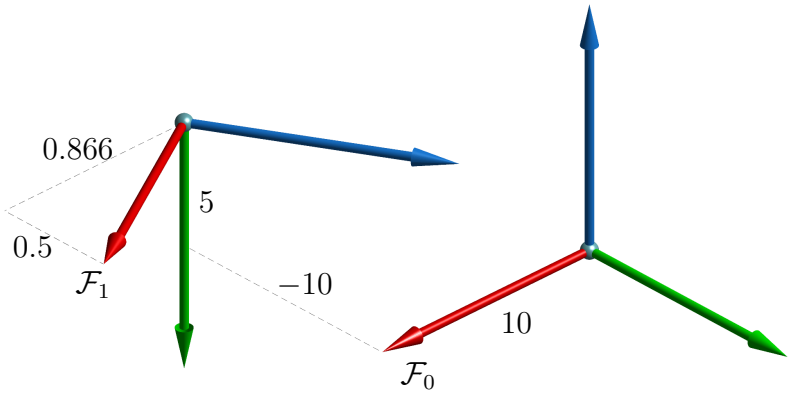


FIGURE 6.13 – Réponse à l'exercice 6.4(b).

6.5. (a)

$$\begin{aligned}\mathbf{H}_b^a &= \mathbf{H}_{rot,z}(30°)\mathbf{H}_{rot,x}(45°)\\ &= \begin{bmatrix}\cos 30° & -\sin 30° & 0 & 0\\ \sin 30° & \cos 30° & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{bmatrix}\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & \cos 45° & -\sin 45° & 0\\ 0 & \sin 45° & \cos 45° & 0\\ 0 & 0 & 0 & 1\end{bmatrix} = \begin{bmatrix}0.866 & -0.3535 & 0.3535 & 0\\ 0.5 & 0.612 & -0.612 & 0\\ 0 & 0.707 & 0.707 & 0\\ 0 & 0 & 0 & 1\end{bmatrix}\end{aligned}$$

(b) Le code RAPID est donnée dans l'exemple 6.13.

Exemple 6.13 – Code RAPID pour l'exercice 6.5

```
MODULE Ex6_5_solution

VAR robtarget Fb;
```

```
PROC main()
    Fb.trans:=[0,0,0];
    Fb.rot:=OrientZYX(30,0,45);
    ...
ENDPROC
ENDMODULE
```

6.6.

$$\begin{aligned}
\mathbf{H} &= \mathbf{H}_{trans}(7,12,-4)\mathbf{H}_{rot,y}(180°)\mathbf{H}_{rot,x}(30°) \\
&= \begin{bmatrix} 1 & 0 & 0 & 7 \\ 0 & 1 & 0 & 12 \\ 0 & 0 & 1 & -4 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos 180° & 0 & \sin 180° & 0 \\ 0 & 1 & 0 & 0 \\ -\sin 180° & 0 & \cos 180° & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos 30° & \sin 30° & 0 \\ 0 & \sin 30° & \cos 30° & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\
&= \begin{bmatrix} -1 & 0 & 0 & 7 \\ 0 & .866 & -.5 & 12 \\ 0 & -.5 & .866 & -4 \\ 0 & 0 & 0 & 1 \end{bmatrix}
\end{aligned}$$

Exemple 6.14 – Code RAPID pour l'exercice 6.6

```
MODULE Ex6_6_solution

VAR robtarget FDepart;
VAR robtarget FResultat;

PROC main()
    FResultat := RelTool(FDepart,7,12,-4 \Ry:=180);
    FResultat := RelTool(FResultat,0,0,0 \Rx:=30);
    ...
ENDPROC
ENDMODULE
```

6.7. Désolé, le dessin-réponse n'est pas disponible pour cet exercice.

6.8.

$$\begin{aligned}
\mathbf{H}^{?}_{final} &= \mathbf{H}^{?}_{début}\mathbf{H}_{rot,x}(90°)\mathbf{H}_{trans}(150,400,-100) \\
&= \begin{bmatrix} 0 & 1 & 0 & 500 \\ 1 & 0 & 0 & -300 \\ 0 & 0 & -1 & 850 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos 90° & -\sin 90° & 0 \\ 0 & \sin 90° & \cos 90° & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 150 \\ 0 & 1 & 0 & 400 \\ 0 & 0 & 1 & -100 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\
&= \begin{bmatrix} 0 & 0 & -1 & 600 \\ 1 & 0 & 0 & -150 \\ 0 & -1 & 0 & 450 \\ 0 & 0 & 0 & 1 \end{bmatrix}
\end{aligned}$$

$$\begin{array}{ccc} \text{X} = 600 \text{ mm} & \text{Y} = -150 \text{ mm} & \text{Z} = 450 \text{ mm} \\ \text{AngleX} = -90° & \text{AngleY} = 0° & \text{AngleZ} = 90° \end{array}$$

6.9.

$$\mathbf{H}_{tbuse}^{tool0} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 120 \\ 0 & 0 & 1 & 210 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(-30^\circ) & -\sin(-30^\circ) & 0 \\ 0 & \sin(-30^\circ) & \cos(-30^\circ) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos 90^\circ & -\sin 90^\circ & 0 & 0 \\ \sin 90^\circ & \cos 90^\circ & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 0 & -1 & 0 & 0 \\ \cos(-30^\circ) & 0 & -\sin(-30^\circ) & 120 \\ \sin(-30^\circ) & 0 & \cos(-30^\circ) & 210 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

6.10.

$$\mathbf{H}_a^0 = \begin{bmatrix} -1 & 0 & 0 & 4 \\ 0 & -1 & 0 & 2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{H}_b^0 = \begin{bmatrix} 0 & \sin 45^\circ & \cos 45^\circ & 10 \\ 0 & -\cos 45^\circ & \sin 45^\circ & 10 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{H}_c^0 = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & -1 & 0 & 10 \\ 0 & 0 & -1 & 5 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Chapitre 7

Cinématique directe

Nous avons déjà vu dans la section 5.1 comment calculer la cinématique directe dans le cas d'un robot plan à deux articulations motorisées. Cependant, l'approche utilisée était une approche ad hoc qui ne peut pas être généralisée. Dans ce chapitre, nous allons décrire une méthode systématique pour développer le modèle cinématique d'un robot sériel à n articulations. Cette méthode consiste en deux étapes principales : placer un référentiel sur chaque lien du robot et ensuite, pour chaque paire de référentiels consécutifs, trouver quatre paramètres géométriques qui décrivent la pose de l'un par rapport à l'autre. Il s'agit de la méthode Denavit-Hartenberg qui a été proposée en 1955 [15] et reste toujours la méthode de loin la plus utilisée en robotique sérielle. Il existe, cependant, quelques variations de cette méthode [16]. Dans ce cours, nous allons utiliser la méthode originale, pour être conforme avec les anciennes notes de cours. Cependant, il faut avouer que la variation proposée dans [17] est un peu plus intuitive.

7.1 La méthode Denavit-Hartenberg

Pour résoudre la cinématique directe d'un robot sériel spatial à n articulations (rotoïdes ou prismatiques), il est presque inévitable de placer un référentiel sur chaque lien du robot, chose que nous n'avons pas eu besoin de faire dans l'exemple trivial de la section 5.1. Ainsi, dans le cas d'un robot à n articulations, donc à $n+1$ liens, le modèle de la cinématique directe du robot sera exprimé par l'équation suivante :

$$\mathbf{H}_n^0 = \mathbf{H}_1^0 \mathbf{H}_2^1 \mathbf{H}_3^2 \dots \mathbf{H}_n^{n-1}, \tag{7.1}$$

où $\mathbf{H}_n^0$ représente la pose du référentiel $\mathcal{F}_n$ (celui de la bride du robot, qu'ABB appelle tool0) par rapport au référentiel $\mathcal{F}_0$ (celui de la base). Le problème principal avec un choix aléatoire de référentiels est que l'obtention des matrices $\mathbf{H}_i^{i-1}$ serait assez laborieux. Un autre problème est bien sûr le fait qu'une telle approche ne serait pas systématique.

Dans les années 1950s, les messieurs Jacques Denavit and Richard Hartenberg ont eu l'excellente idée de proposer une méthode simple et systématique pour placer des référentiels sur chaque lien d'un mécanisme sériel qui facilite énormément le calcul des matrices $\mathbf{H}_i^{i-1}$. Leur idée était de placer les référentiels de telle manière que l'axe z_i (où $i = 0, 1, 2, \dots n-1$) est le long de l'axe de l'articulation $i+1$, dans le cas d'une articulation rotoïde, ou parallèle à la direction de l'articulation $i+1$, dans le cas d'une articulation prismatique, et de contraindre l'axe x_i d'intersecter l'axe z_{i-1} à angle droit. Grace à cette double contrainte (non seulement intersecter mais aussi à angle droit), nous avons besoin de seulement quatre paramètres géométriques, plutôt que six, pour décrire la pose du référentiel $\mathcal{F}_i$ par rapport au référentiel $\mathcal{F}_{i-1}$. Ces quatre paramètres géométriques, deux distances et deux angles, sont en plus très faciles à trouver.

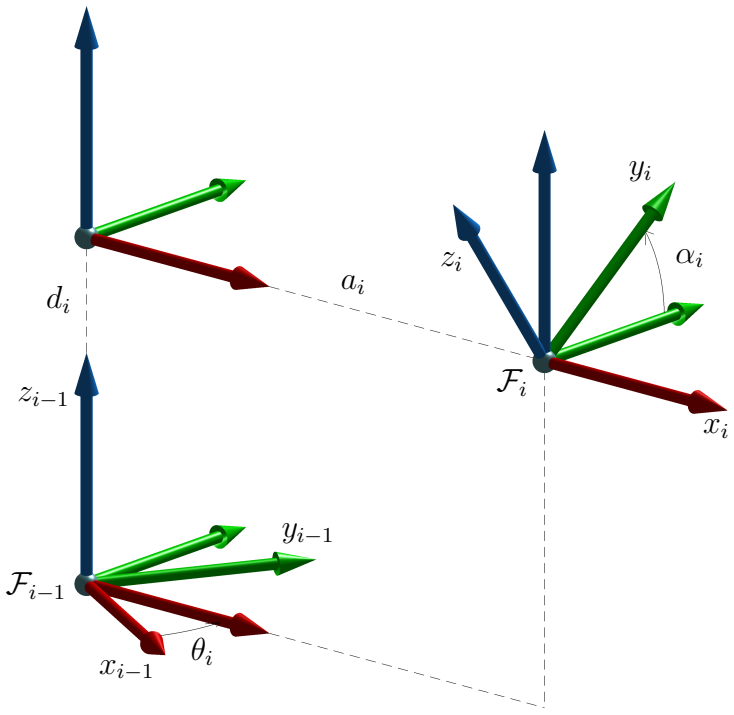


FIGURE 7.1 – Les quatre transformations consécutives de base pour obtenir $\mathcal{F}_i$ à partir de $\mathcal{F}_{i-1}$.

La figure 7.1 démontre deux référentiels arbitraires, $\mathcal{F}_i$ et $\mathcal{F}_{i-1}$, qui respectent cette contrainte (l'axe x_i intersecte l'axe z_{i-1} à angle droit). Pour l'instant, nous allons ignorer les articulations et les liens du robot. La figure 7.1 illustre clairement que nous pouvons obtenir le référentiel $\mathcal{F}_i$ à partir du référentiel $\mathcal{F}_{i-1}$, à l'aide de seulement quatre transformations consécutives simples (translation le long d'un axes x, y ou z, ou rotation autour d'un axe x, y ou z). Ces quatre transformations peuvent être faites dans l'ordre suivant :

1 rotation de θ_i autour de l'axe z_{i-1} ;

2 translation de d_i le long de l'axe z du nouveau référentiel ;

3 translation de a_i le long de l'axe x du nouveau référentiel ;

4 rotation de α_i autour de l'axe x du nouveau référentiel.

Bien sûr, il est possible de trouver d'autres séquences de quatre transformations consécutives, mais nous sommes intéressé uniquement par le résultat final.

Nous allons appeler θ_i, d_i, a_i et α_i, les quatre paramètres DH. Ainsi, la matrice $\mathbf{H}_i^{i-1}$ qui représente la pose du référentiel $\mathcal{F}_i$ par rapport au référentiel $\mathcal{F}_{i-1}$ peut être obtenue en fonction des quatre paramètres DH par l'équation suivante :

$$\begin{aligned}\mathbf{H}_i^{i-1} &= \mathbf{H}_{rot,z}(\theta_1)\mathbf{H}_{trans}(0,0,d_i)\mathbf{H}_{trans}(a_i,0,0)\mathbf{H}_{rot,x}(\alpha_1) \\ &= \begin{bmatrix} \cos\theta_i & -\sin\theta_i\cos\alpha_i & \sin\theta_i\sin\alpha_i & a_i\cos\theta_i \\ \sin\theta_i & \cos\theta_i\cos\alpha_i & -\cos\theta_i\sin\alpha_i & a_i\sin\theta_i \\ 0 & \sin\alpha_i & \cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}. \end{aligned} \tag{7.2}$$

Maintenant que nous avons vue l'idée derrière la méthode Denavit-Hartenberg, nous pouvons présenter l'algorithme pour placer ce que nous allons appeler les référentiels DH sur un robot sériel et ensuite la méthode pour le calcul des quatre paramètres DH.

7.1.1 Algorithme pour placer les référentiels DH

Pour rendre l'algorithme plus facile à expliquer, nous allons donner les étapes à suivre selon trois situations différentes.

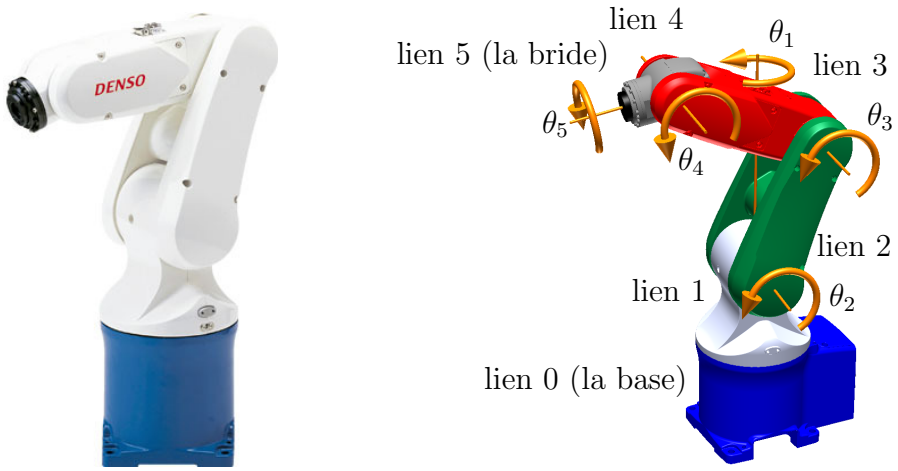
- Pour le référentiel $\mathcal{F}_0$, placez son axe z_0 le long de l'axe de l'articulation 1, dans le cas d'une articulation rotoïde, en respectant la règle de la main droite, ou parallèle à la direction de l'articulation $i+1$ et dans le même sens, dans le cas d'une articulation prismatique. Placez l'origine du référentiel $\mathcal{F}_0$ et les axes x_0 et y_0 selon vos préférences. ⚠ Il est important de comprendre que le référentiel $\mathcal{F}_0$ est fixé à la base du robot. Donc, lorsque les articulations du robot se déplacent, ce référentiel reste immobile par rapport à la base du robot.
- Pour le référentiel $\mathcal{F}_i$, où $i = 1, 2, \ldots, n-1$, placez son axe z_i le long de l'axe de l'articulation $i+1$, dans le cas d'une articulation rotoïde, en respectant la règle de la main droite, ou parallèle à la direction de l'articulation $i+1$ et dans le même sens, dans le cas d'une articulation prismatique. Choisissez l'origine du référentiel $\mathcal{F}_i$ de telle manière que l'axe x_i intersecte l'axe z_{i-1} à angle droit. Si les axes z_i et z_{i-1} sont parallèles ou confondues, il existe infiniment de possibilités. Sinon, il existe une seule possibilité pour l'origine $\mathcal{F}_i$ et deux pour l'axe x_i. Choisissez parmi ces possibilités selon vos préférences. ⚠ Il est très important de comprendre que le référentiel $\mathcal{F}_i$ est fixé au lien i. Donc, lorsque seul l'articulation $i+1$ se déplace, le référentiel $\mathcal{F}_i$ reste immobile par rapport à la base du robot.
- Pour le référentiel $\mathcal{F}_n$, placer l'axe x_n de telle manière qu'il intersecte l'axe z_{n-1} à angle droit. Placez l'origine du référentiel $\mathcal{F}_n$ et les axes z_n et y_n selon vos préférences. Normalement, on positionne l'origine au centre de l'extrémité de la bride du robot, comme dans le cas des robots ABB. ⚠ Il est important de comprendre que ce référentiel est fixé à la bride du robot (lien n).

7.1.2 Algorithme pour obtenir les paramètres DH

Une fois tous les $n+1$ référentiels DH placés, il faut trouver les paramètres DH pour chaque référentiel $\mathcal{F}_i$ (où $i = 1, 2, \ldots, n$). En se référant à la figure 7.1, il est facile de déduire les quatre règles suivantes pour les paramètres DH :

- θ_i est l'angle entre l'axe x_{i-1} et l'axe x_i, mesuré autour de l'axe z_{i-1} en utilisant la règle de la main droite. ⚠ Il est important de comprendre que si l'articulation i est rotoïde, θ_i est une *variable articulaire*, alors que si l'articulation i est prismatique, θ_i est une constante (positive, zéro ou négative).
- d_i est la distance *dirigée* de l'axe x_{i-1} vers l'axe x_i, mesurée le long de l'axe z_{i-1}. ⚠ Il est important de comprendre que si l'articulation i est prismatique, d_i est une *variable articulaire*, alors que si l'articulation i est rotoïde, d_i est une constante (positive, zéro ou négative).
- a_i est la distance dirigée de l'axe z_{i-1} vers l'axe z_i, mesurée le long de l'axe x_i. ⚠ Il est important de se rappeler que a_i est toujours une constante (positive, zéro ou négative). Par exemple, dans la figure 7.1, a_i est positive.
- α_i est l'angle entre l'axe z_{i-1} et l'axe z_i, mesuré autour de l'axe x_i en utilisant la règle de la main droite. ⚠ Il est important de se rappeler que α_i est toujours une constante. Par exemple, dans la figure 7.1, α_i est environ $+30°$.

Lorsque vous calculer les paramètres DH à partir d'une figure avec les référentiels DH, il faut se rappeler que le robot est articulé et non une structure. En autres mots, il faut toujours s'imaginer le robot bouger avec ses référentiels DH.



(a) (b)

FIGURE 7.2 – (a) Le robot VP-5243 de la compagnie DENSO Robotics et (b) un schéma illustrant les sens positives de rotations.

7.1.3 Équation de la cinématique directe

Une fois les paramètres DH obtenus, il ne reste que des les substituer dans l'équation (7.2) pour obtenir les matrices $\mathbf{H}_1^0$, $\mathbf{H}_2^1$, ..., $\mathbf{H}_n^{n-1}$, et les substituer dans l'équation (7.1). ⚠ Il est important de comprendre que même si les référentiels $\mathcal{F}_0$ et $\mathcal{F}_n$ sont déjà choisis (par exemple, par le fabricant du robot), il existe tout de même infiniment de possibilités pour les autres référentiels DH et, par conséquence, pour les matrices $\mathbf{H}_i^{i-1}$. Cependant, le résultat final (la matrice $\mathbf{H}_n^0$) devrait être le même.

Enfin, si vous avez un référentiel outil, et un référentiel atelier, la pose du premier par rapport au dernier est définie par l'équation suivante :

$$\mathbf{H}_{outil}^{atelier} = \mathbf{H}_0^{atelier}\mathbf{H}_n^0\mathbf{H}_{outil}^n. \tag{7.3}$$

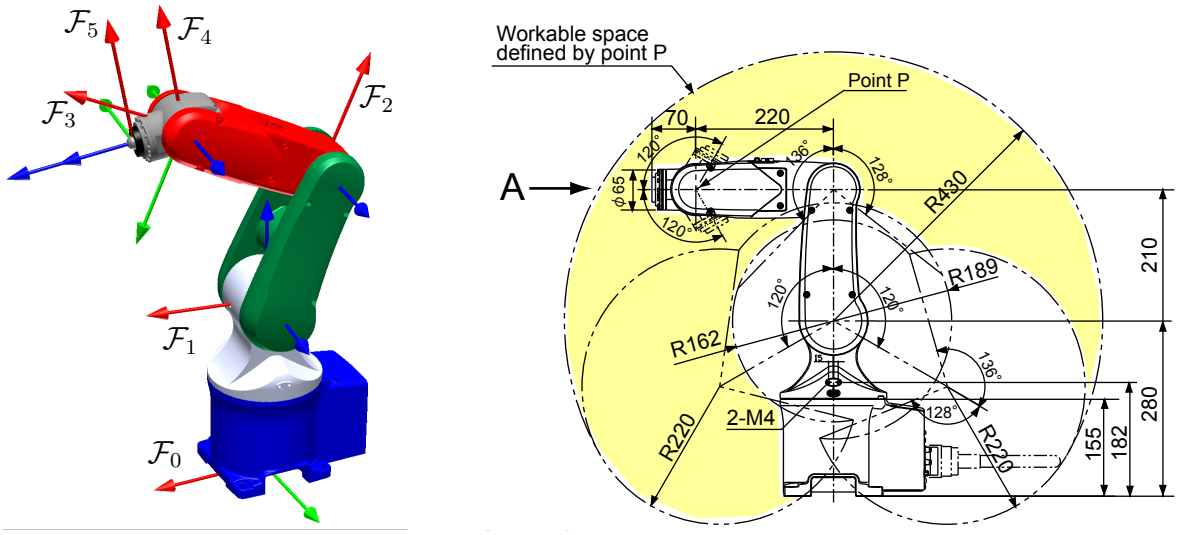
7.2 Exemple d'un robot sériel à cinq articulations rotoïdes

Pour mieux comprendre la méthode Denavit-Hartenberg, nous allons l'appliquer dans le cas d'un robot réel. Il s'agit d'un robot sériel à cinq articulations rotoïdes, et plus précisément de l'ancien modèle VP-5243 de la compagnie DENSO Robotics. Quelques autres fabricants de robots offrent également des modèles de robots à cinq articulations rotoïdes, destinés au transfert de matériel.

La figure 7.2 présente une photo du robot VP-5243, ainsi qu'un schéma qui illustre chacun des cinq axes des articulations rotoïdes avec le sens positive de rotation. Une vidéo démontrant les mouvements des articulations est également disponible. La figure 7.2(b) démontre également la façon selon laquelle on numérote les liens et les articulations. La base est toujours le lien 0 et la bride du robot, le lien n, alors que l'axe 1 est toujours celui de l'articulation à la base.

La procédure pour placer les référentiels DH sera détaillée en classe. Ici, nous allons simplement donner une des nombreuses solutions possibles à la figure 7.3(a). Il faut bien comprendre que le référentiel $\mathcal{F}_i$ est fixé au lien i, malgré le fait que son axe z_i, dans le cas $i < n$, est le long de l'axe de l'articulation $i+1$. Une vidéo est disponible pour bien illustrer ce fait.

Une fois les référentiels DH placés, la prochaine étape est de trouver les paramètres DH en se référant aux figures 7.3(a) et (b). Encore une fois, il est nécessaire de s'imaginer le robot avec ses référentiels DH bouger. Dans ce cas, nous vous avons fournie une vidéo, mais ce ne serait évidement pas le cas à l'examen final ou en pratique. Une erreur fréquente lors du calcul des paramètres d_i et a_i est de considérer les axes d'un référentiels comme des segments et non comme des lignes d'une longueur infinie.



(a) (b)

FIGURE 7.3 – (a) Une des solutions pour les référentiels DH et (b) les dimensions du robot .

TABLEAU 7.1 – Paramètres DH selon le choix de référentiels DH présenté à la figure 7.3(a).

i	θ_i	d_i	a_i	α_i
1	θ_1	280 mm	0 mm	$-90°$
2	θ_2	0 mm	210 mm	$0°$
3	θ_3	0 mm	220 mm	$0°$
4	θ_4	0 mm	0 mm	$-90°$
5	θ_5	70 mm	0 mm	$0°$

Les paramètres DH sont données au tableau 7.1. Selon notre expérience d'enseignement, la partie la plus difficile à comprendre est la colonne des θ_i. Si nous continuons avec les paramètres DH trouvés dans le tableau 7.1, la configuration du robot où toutes les articulations sont à zéro degrés sera un peu étrange. Essayez de la trouvez par vous-même. Dans cette configuration, tous les axes x_i doivent être parallèles et dans le même sens. De plus, à moins que vous ne décidiez pas de construire votre propre robot sériel, les seules fois que vous aurez à faire la cinématique directe (après avoir réussit ce cours), ce sera pour un robot déjà existant. Or, dans un robot existant, le choix de configuration zéro a été déjà fait par son fabricant et nous devons se conformer à ce choix.

Dans sa configuration zéro, le vrai robot VP-5243 est étiré vers le haut. En autres mots, dans la configuration illustrée à la figure 7.3(b), $\theta_1 = 0°$, $\theta_2 = 0°$, $\theta_3 = 90°$, $\theta_4 = 0°$, et $\theta_5 = 0°$. Or, selon notre choix de référentiels DH, dans la configuration 7.3(b), $\theta_1 = 0°$, $\theta_2 = -90°$, $\theta_3 = 90°$, $\theta_4 = -90°$, et $\theta_5 = 0°$. Ainsi, pour être conforme avec le vrai robot VP-5243, nous devons ajouter un décalage de $-90°$ sur θ_2 et un autre de $-90°$ aussi sur θ_4. Les nouveaux paramètres DH sont montrés au tableau 7.2.

TABLEAU 7.2 – Paramètres DH pour être conforme avec le vrai robot VP-5243.

i	θ_i	d_i	a_i	α_i
1	θ_1	280 mm	0 mm	$-90°$
2	$\theta_2 - 90°$	0 mm	210 mm	$0°$
3	θ_3	0 mm	220 mm	$0°$
4	$\theta_4 - 90°$	0 mm	0 mm	$-90°$
5	θ_5	70 mm	0 mm	$0°$

La prochaine étape est de substituer les paramètres DH du tableau 7.2 dans l'équation (7.2) :

$$\begin{aligned}
\mathbf{H}_1^0 &= \mathbf{H}_{rot,z}(\theta_1)\mathbf{H}_{trans}(0,0,280)\mathbf{H}_{trans}(0,0,0)\mathbf{H}_{rot,x}(-90^\circ) \\
&= \begin{bmatrix} \cos\theta_1 & 0 & -\sin\theta_1 & 0 \\ \sin\theta_1 & 0 & \cos\theta_1 & 0 \\ 0 & -1 & 0 & 280 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \qquad (7.4) \\
\mathbf{H}_2^1 &= \mathbf{H}_{rot,z}(\theta_2 - 90^\circ)\mathbf{H}_{trans}(0,0,0)\mathbf{H}_{trans}(210,0,0)\mathbf{H}_{rot,x}(0^\circ) \\
&= \begin{bmatrix} \sin\theta_2 & \cos\theta_2 & 0 & 210\sin\theta_2 \\ -\cos\theta_2 & \sin\theta_2 & 0 & -210\cos\theta_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \qquad (7.5) \\
\mathbf{H}_3^2 &= \mathbf{H}_{rot,z}(\theta_3)\mathbf{H}_{trans}(0,0,0)\mathbf{H}_{trans}(220,0,0)\mathbf{H}_{rot,x}(0^\circ) \\
&= \begin{bmatrix} \cos\theta_3 & -\sin\theta_3 & 0 & 220\cos\theta_3 \\ \sin\theta_3 & \cos\theta_3 & 0 & 220\sin\theta_3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \qquad (7.6) \\
\mathbf{H}_4^3 &= \mathbf{H}_{rot,z}(\theta_4 - 90^\circ)\mathbf{H}_{trans}(0,0,0)\mathbf{H}_{trans}(0,0,0)\mathbf{H}_{rot,x}(-90^\circ) \\
&= \begin{bmatrix} \sin\theta_4 & 0 & \cos\theta_4 & 0 \\ -\cos\theta_4 & 0 & \sin\theta_4 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \qquad (7.7) \\
\mathbf{H}_5^4 &= \mathbf{H}_{rot,z}(\theta_5)\mathbf{H}_{trans}(0,0,70)\mathbf{H}_{trans}(0,0,0)\mathbf{H}_{rot,x}(0^\circ) \\
&= \begin{bmatrix} \cos\theta_5 & -\sin\theta_5 & 0 & 0 \\ \sin\theta_5 & \cos\theta_5 & 0 & 0 \\ 0 & 0 & 1 & 70 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \qquad (7.8)
\end{aligned}$$

Enfin, il ne reste que de substituer les matrices ci-dessus dans l'équation (7.1) et faire quelques manipulations trigonométriques afin de rendre l'expression finale plus compacte :

$$\begin{aligned}
\mathbf{H}_5^0 &= \mathbf{H}_1^0\mathbf{H}_2^1\mathbf{H}_3^2\mathbf{H}_4^3\mathbf{H}_5^4 \\
&= \begin{bmatrix} -c_1c_{234}c_5 + s_1s_5 & c_1c_{234}s_5 + s_1c_5 & c_1s_{234} & c_1(70s_{234} + 220s_{23} + 210s_2) \\ -s_1c_{234}c_5 - c_1s_5 & s_1c_{234}s_5 - c_1c_5 & s_1s_{234} & s_1(70s_{234} + 220s_{23} + 210s_2) \\ s_{234}c_5 & -s_{234}s_5 & c_{234} & 280 + 70c_{234} + 220c_{23} + 210c_2 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \qquad (7.9)
\end{aligned}$$

où $s_1 = \sin\theta_1$, $c_1 = \cos\theta_1$, $s_2 = \sin\theta_2$, $c_2 = \cos\theta_2$, $s_5 = \sin\theta_5$, $c_5 = \cos\theta_5$, $s_{23} = \sin(\theta_2 + \theta_3)$, $c_{23} = \cos(\theta_2+\theta_3)$ $s_{234} = \sin(\theta_2+\theta_3+\theta_4)$, et $c_{234} = \cos(\theta_2+\theta_3+\theta_4)$. Pour faire ce type de simplification, ou plutôt substitution, il faut utiliser une fonction qui fait des regroupements trigonométriques uniquement sur les matrices qui correspondent à des axes qui restent toujours parallèles (dans notre cas, les axes 2, 3 et 4). Par exemple, le calcul à faire dans vos calculatrices TI est `H10*tCollect(H21*H32*H43)*H54`.

7.3 Exercices

7.1. Solutionner la cinématique directe du robot montré à la figure 7.4.

(a) Placer les référentiels DH sur le robot de la figure 7.4.

(b) Construire le tableau des paramètres DH en se basant sur la figure 7.5.

(c) Donner les valeurs approximatives des variables articulaires dans la configuration illustré à la figure 7.4, selon votre choix de référentiels DH.

(d) Définir les matrices $\mathbf{H}_0^{atelier}$ et $\mathbf{H}_{outil}^{n}$ en se basant sur la figure 7.6.

(e) Trouver l'expression pour la matrice $\mathbf{H}_{outil}^{atelier}$.

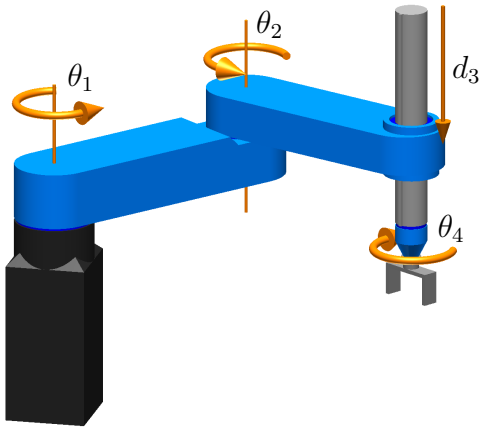


FIGURE 7.4 – Robot SCARA pour l'exercice 7.1.

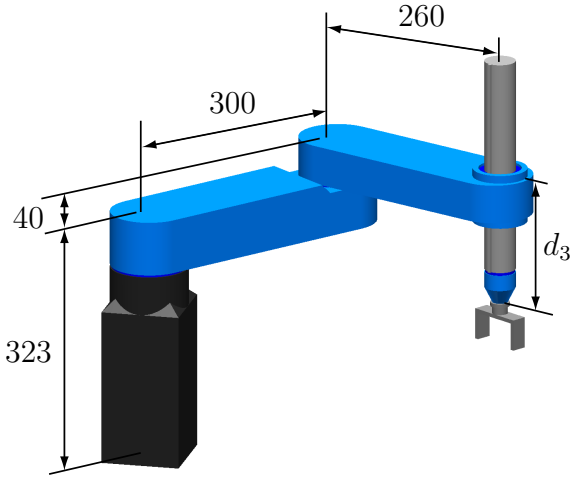


FIGURE 7.5 – Dimensions du robot de l'exercice 7.1.

FIGURE 7.6 – Référentiel de l'atelier et de l'outil pour le robot de l'exercice 7.1.

7.2. Solutionner la cinématique directe du robot à la figure 7.7.

(a) Placer les référentiels DH sur le robot de la figure 7.7.

(b) Construire le tableau des paramètres DH en se basant sur la figure 7.8.

(c) Définir les matrices $\mathbf{H}_0^{atelier}$ et $\mathbf{H}_{outil}^{n}$ en se basant sur la figure 7.9.

(d) Trouver l'expression pour la matrice $\mathbf{H}_{outil}^{atelier}$.

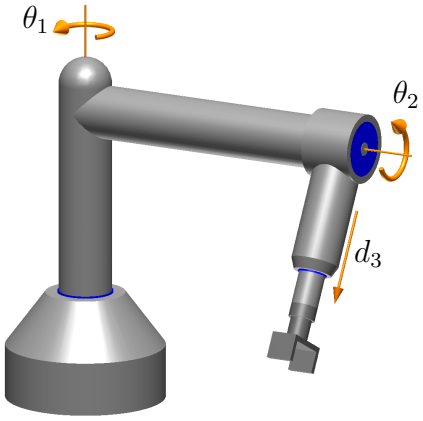


FIGURE 7.7 – Robot à trois degrés de liberté pour l'exercice 7.2.

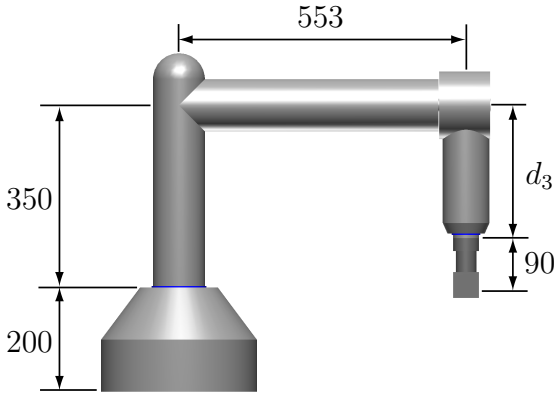


FIGURE 7.8 – Dimensions du robot de l'exercice 7.2.

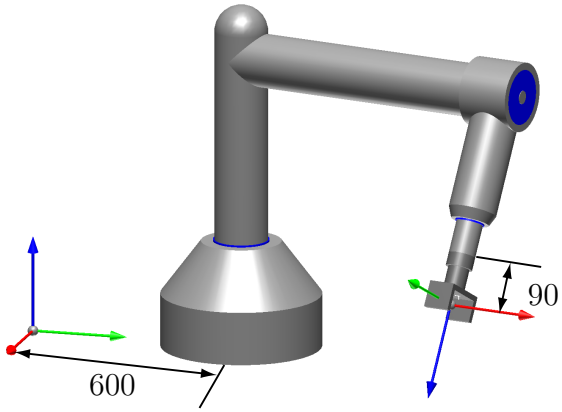


FIGURE 7.9 – Référentiels de l'atelier et de l'outil pour le robot de l'exercice 7.2.

7.3. Solutionner la cinématique directe du robot à la figure 7.10.

(a) Placer les référentiels DH sur le robot de la figure 7.10.

(b) Construire le tableau des paramètres DH en se basant sur la figure 7.11 pour connaître les dimensions. Afin de correctement définir les constantes à ajoutées dans votre tableau, il faut considerer que $\theta_1 = 0°$, $d_2 = 0$ mm, $d_3 = 0$ mm et $\theta_4 = 0°$, dans la configuration du robot montrée à la figure 7.11. En plus, dans cette configuration, la position du centre de l'outil par rapport à la base est de 412.5 mm vers l'avant, 148 mm sur le côté, et 405 mm en haut.

(c) Définir les matrices $\mathbf{H}_0^{atelier}$ et $\mathbf{H}_{outil}^{n}$ en se basant sur la figure 7.11.

(d) Trouver l'expression pour la matrice $\mathbf{H}_{outil}^{atelier}$.

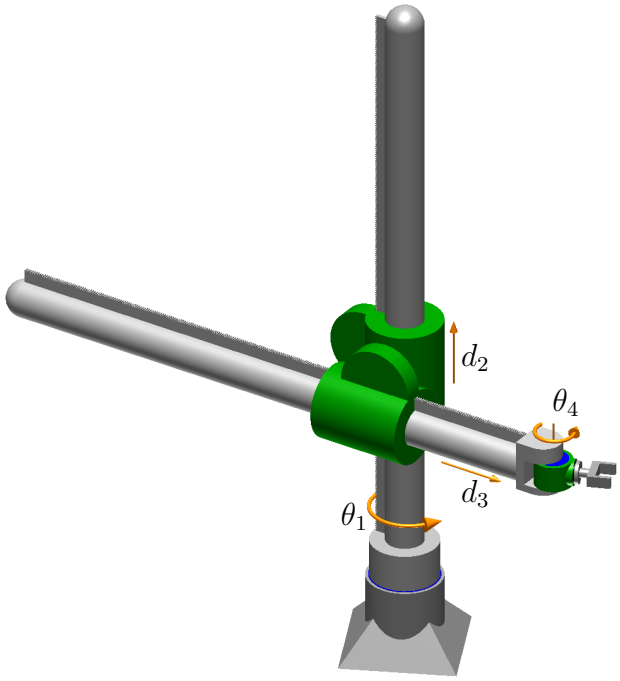


FIGURE 7.10 – Robot à quatre degrés de liberté pour l'exercice 7.3.

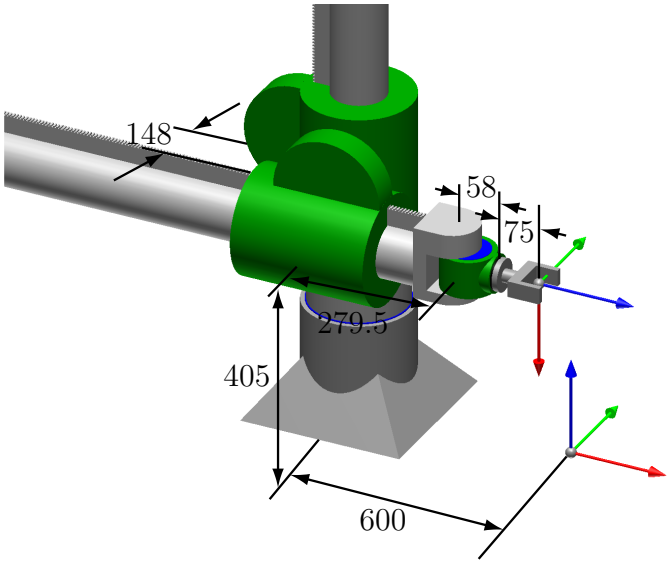


FIGURE 7.11 – Dimensions du robot de l'exercice 7.3.

7.4. Solutionner la cinématique directe du robot des figures 7.12 et 7.13. Trouver uniquement l'expression pour la position de l'effecteur par rapport à la base.

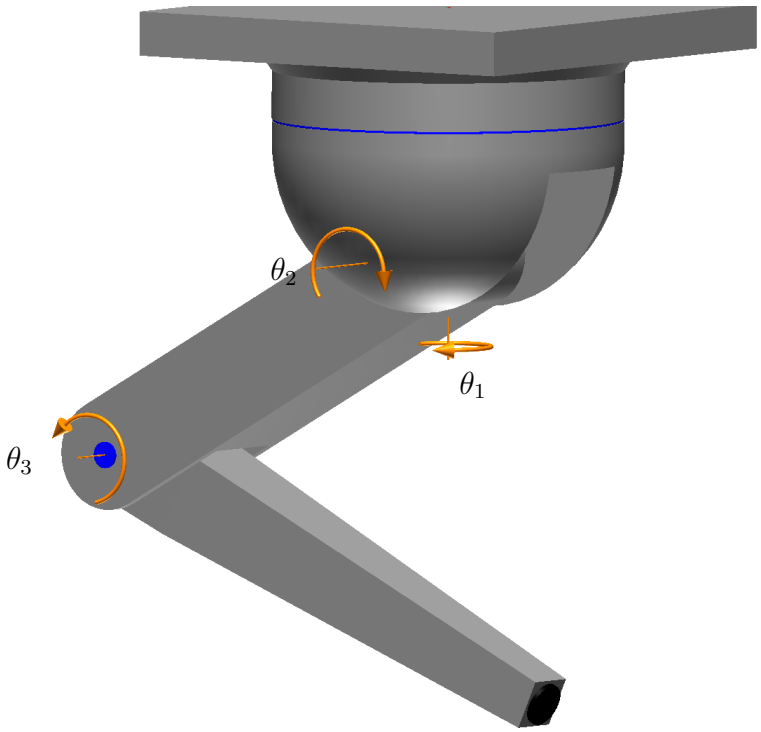


FIGURE 7.12 – Robot à trois degrés de liberté pour l'exercice 7.4.

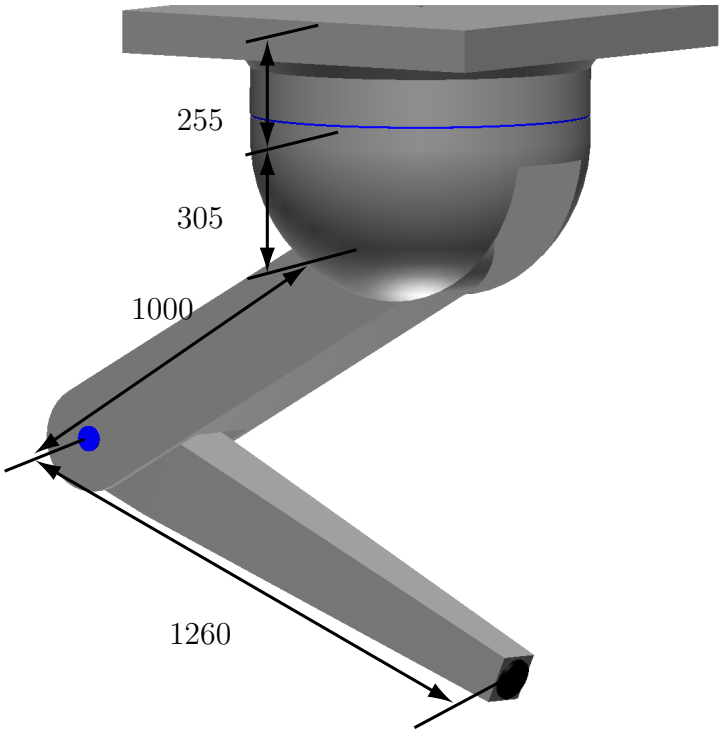


FIGURE 7.13 – Dimensions du robot de l'exercice 7.4.

7.5. Pour le robot des figures 7.14 et 7.15, placer les référentiels DH et trouver les paramètres DH, si dans la configuration montrée à la figure 7.15, toutes les variables articulaires sont égales à zéro.

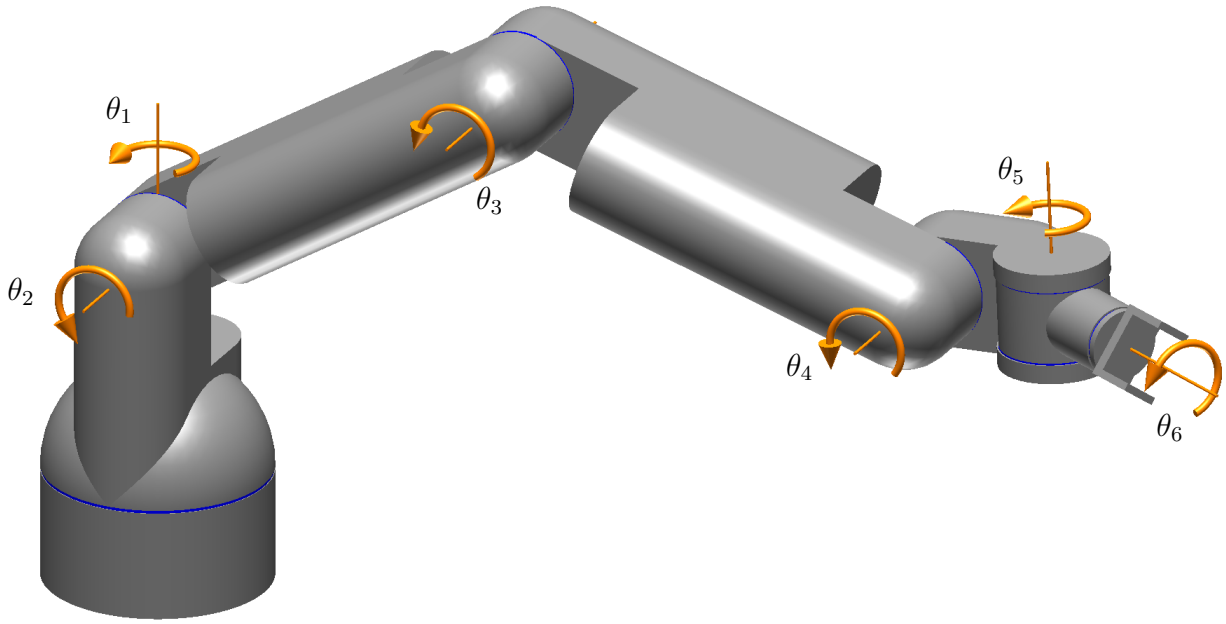


FIGURE 7.14 – Robot à six degrés de liberté pour l'exercice 7.5.

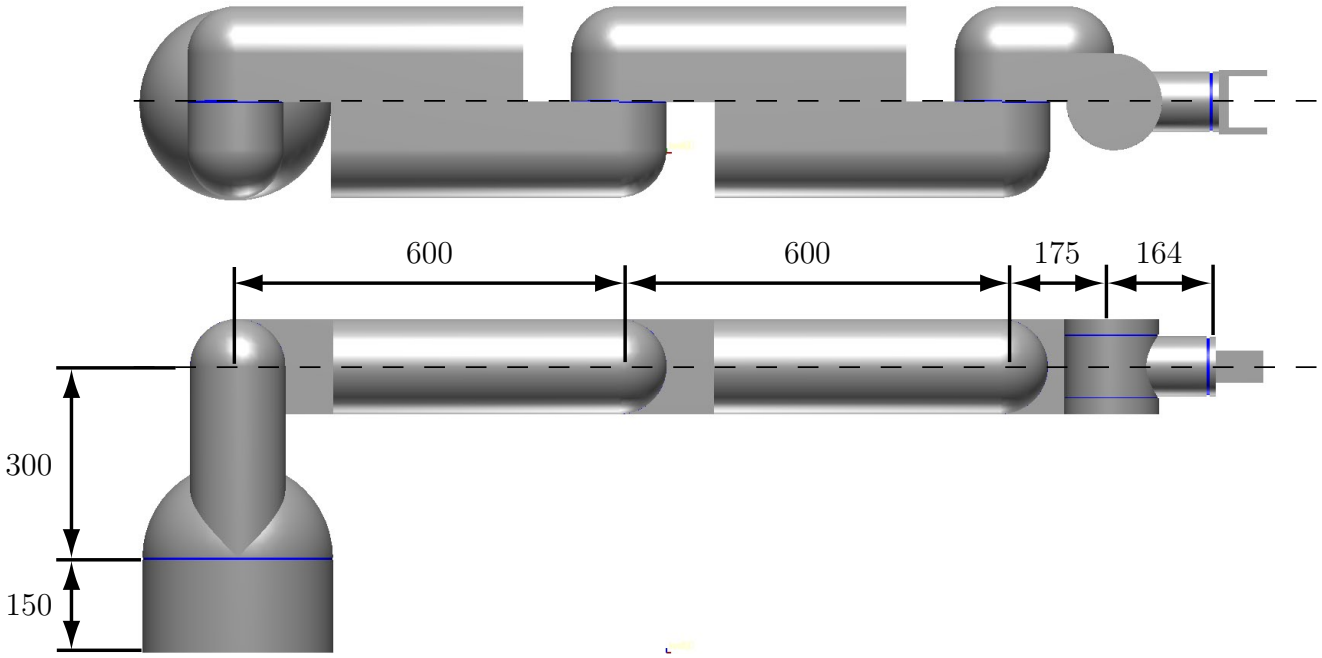


FIGURE 7.15 – Dimensions du robot de l'exercice 7.5.

7.6. Placer les référentiels DH sur le robot des figures 7.16 et 7.17 de telle manière que toutes les variables articulaires sont égales à zéro (c'est-à-dire, les axes x des sept référentiels DH ont la même direction) dans la configuration du robot montrée à la figure 7.17 (sans ajouter des décalages). Il vous êtes fortement suggéré d'utiliser le logiciel RoKiSim pour bien comprendre la notion des référentiels DH dans le cas des robots avec cette architecture.

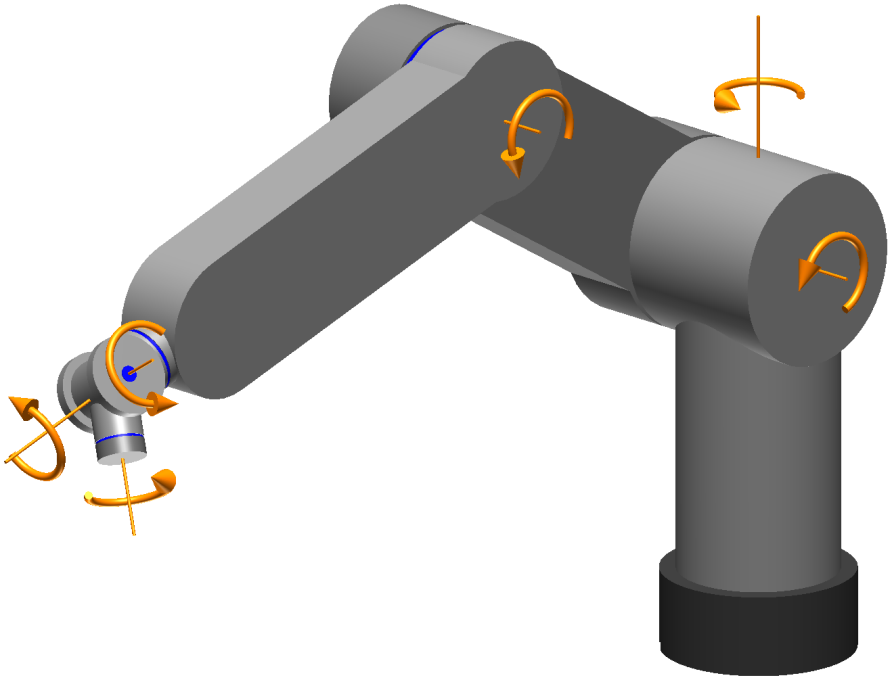


FIGURE 7.16 – Robot à six degrés de liberté pour l'exercice 7.6.

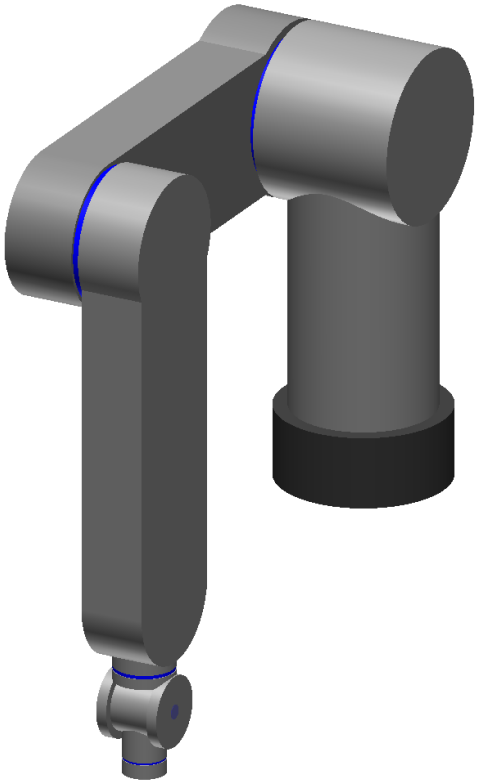


FIGURE 7.17 – Configuration zéro pour le robot de l'exercice 7.6.

7.4 Réponses aux exercices

7.1. La solution de la cinématique directe du robot à la figure 7.4 est donnée ci-dessous.

(a) Les référentiels DH sont illustrés à la figure 7.18.

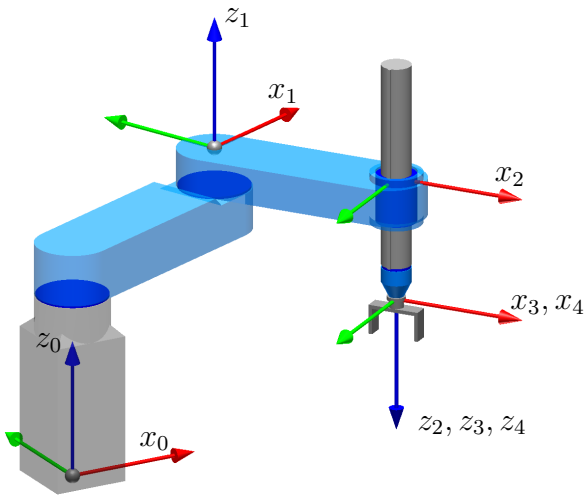


FIGURE 7.18 – Référentiels DH pour le robot de l'exercice 7.1.

(b) Les paramètres DH sont donnés ci-dessous :

TABLEAU 7.3 – Paramètres DH pour le robot de la figure 7.18.

i	θ_i	d_i	a_i	α_i
1	θ_1	363	300	0°
2	θ_2	0	260	180°
3	0°	d_3	0	0°
4	θ_4	0	0	0°

(c) Valeurs : $\theta_1 \approx 20°$, $\theta_2 \approx -60°$, $d_3 \approx 200$ et $\theta_4 \approx 0°$.

(d)

$$\mathbf{H}_0^{atelier} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{H}_{outil}^4 = \begin{bmatrix} 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 37.5 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

(e)

$$\mathbf{H}_{outil}^{atelier} = \begin{bmatrix} \cos(\theta_1+\theta_2-\theta_4) & \sin(\theta_1+\theta_2-\theta_4) & 0 & 260\sin(\theta_1+\theta_2)+300\sin\theta_1 \\ \sin(\theta_1+\theta_2-\theta_4) & -\cos(\theta_1+\theta_2-\theta_4) & 0 & -260\cos(\theta_1+\theta_2)-300\cos\theta_1 \\ 0 & 0 & -1 & 325.5-d_3 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

7.2. La solution de la cinématique directe du robot à la figure 7.7 est donnée ci-dessous.

(a) Les référentiels DH sont illustrés à la figure 7.19.

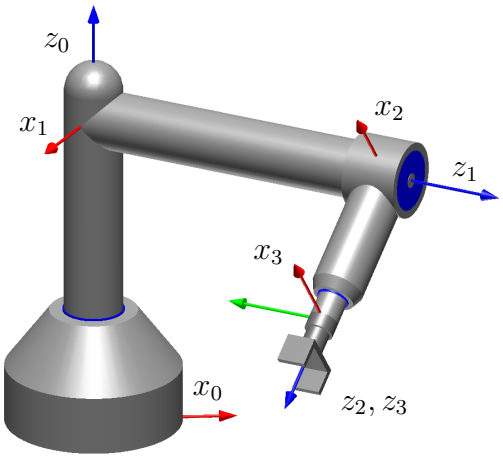


FIGURE 7.19 – Référentiels DH pour le robot de l'exercice 7.2.

(b) Les paramètres DH sont donnés ci-dessous :

TABLEAU 7.4 – Paramètres DH pour le robot de la figure 7.19.

i	θ_i	d_i	a_i	α_i
1	θ_1	550	0	$-90°$
2	θ_2	553	0	$-90°$
3	$0°$	d_3	0	$0°$

(c)

$$\mathbf{H}_0^{atelier} = \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 600 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{H}_{outil}^3 = \begin{bmatrix} 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 90 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

(d)

$$\mathbf{H}_{outil}^{atelier} = \begin{bmatrix} -c_1 & -s_1c_2 & s_1s_2 & (d_3+90)s_1s_2 - 553c_1 \\ -s_1 & c_1c_2 & -c_1s_2 & -(d_3+90)c_1s_2 - 553s_1 + 600 \\ 0 & -s_2 & -c_2 & 550 - (d_3+90)c_2 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

où $s_1 = \sin\theta_1$, $c_1 = \cos\theta_1$, $s_2 = \sin\theta_2$ et $c_2 = \cos\theta_2$.

7.3. La solution de la cinématique directe du robot à la figure 7.10 est donnée ci-dessous.

(a) Les référentiels DH sont illustrés à la figure 7.20.

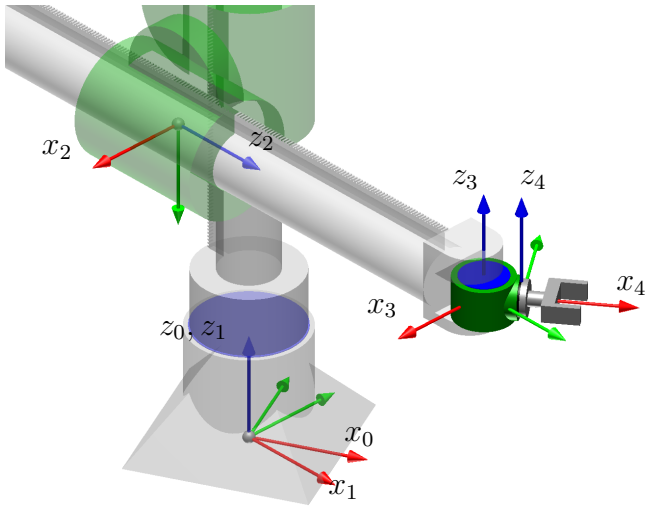


FIGURE 7.20 – Référentiels DH pour le robot de l'exercice 7.3.

(b) Les paramètres DH sont donnés ci-dessous :

TABLEAU 7.5 – Paramètres DH pour le robot de la figure 7.20.

i	θ_i	d_i	a_i	α_i
1	θ_1	0	0	$0°$
2	$-90°$	$d_2 + 405$	148	$-90°$
3	$0°$	$d_3 + 279.5$	0	$90°$
4	$\theta_4 + 90°$	0	58	$0°$

(c)

$$\mathbf{H}_0^{atelier} = \begin{bmatrix} 1 & 0 & 0 & -600 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{H}_{outil}^4 = \begin{bmatrix} 0 & 0 & 1 & 75 \\ 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

(d)

$$\mathbf{H}_{outil}^{atelier} = \begin{bmatrix} 0 & -s_{14} & c_{14} & (d_3 + 279.5)c_1 + 133c_{14} + 148s_1 - 600 \\ 0 & c_{14} & s_{14} & (d_3 + 279.5)s_1 + 133s_{14} - 148c_1 \\ -1 & 0 & 0 & d_2 + 405 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

où $s_1 = \sin\theta_1$, $c_1 = \cos\theta_1$, $s_{14} = \sin(\theta_1 + \theta_4)$ et $c_{14} = \cos(\theta_1 + \theta_4)$.

7.4. Le vecteur de position de la bride du robot est donné selon les référentiels DH illustrés à la figure 7.21.

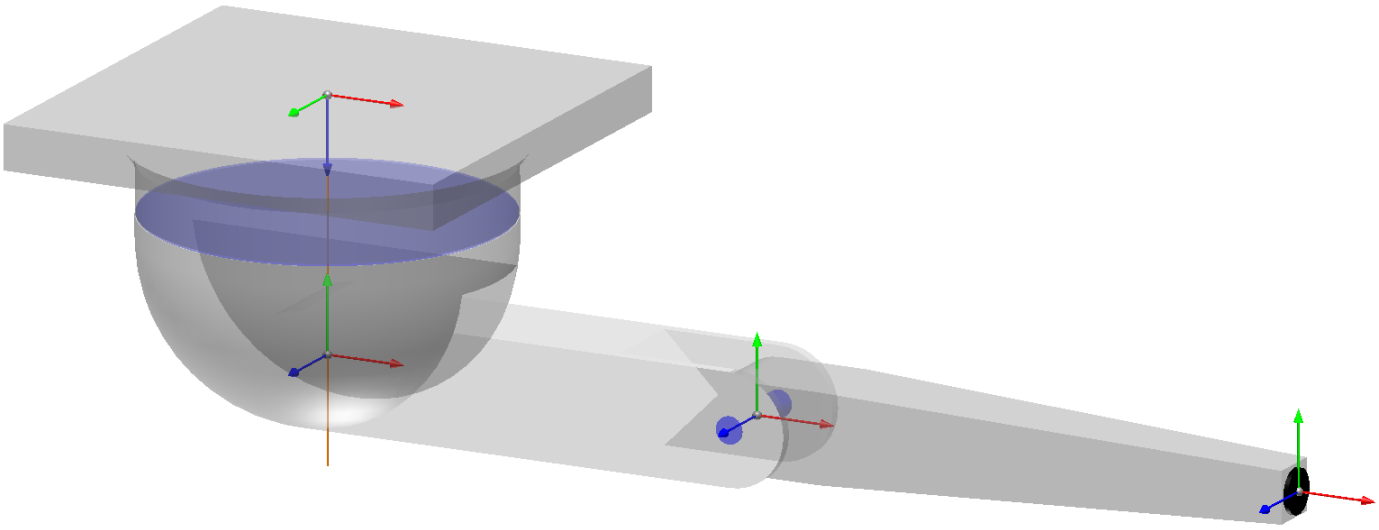


FIGURE 7.21 – Référentiels DH pour le robot de l'exercice 7.4.

$$\mathbf{H}_3^0 = \begin{bmatrix} c_1c_{23} & -c_1s_{23} & -s_1 & c_1(1260c_{23}+1000c_2) \\ s_1c_{23} & -s_1s_{23} & c_1 & s_1(1260c_{23}+1000c_2) \\ -s_{23} & -c_{23} & 0 & 560-(1260s_{23}+1000s_2) \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{v}_3^0 = \begin{bmatrix} c_1(1260c_{23}+1000c_2) \\ s_1(1260c_{23}+1000c_2) \\ 560-(1260s_{23}+1000s_2) \end{bmatrix}$$

où $s_1 = \sin\theta_1$, $c_1 = \cos\theta_1$, $s_2 = \sin\theta_2$, $c_2 = \cos\theta_2$, $s_{23} = \sin(\theta_2+\theta_3)$ et $c_{23} = \cos(\theta_2+\theta_3)$.

7.5. Les référentiels DH sont illustrés à la figure 7.22, alors que les paramètres DH sont donnés dans le tableau 7.6.

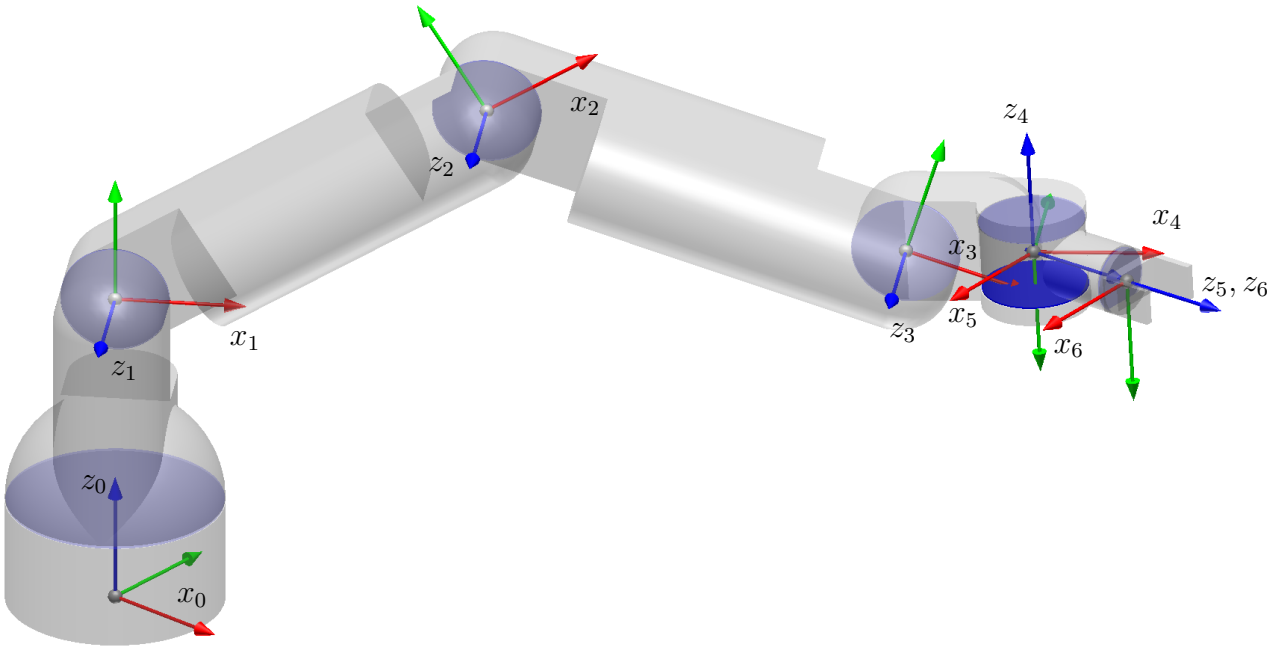


FIGURE 7.22 – Référentiels DH pour le robot de l'exercice 7.5.

TABLEAU 7.6 – Paramètres DH pour le robot de la figure 7.22.

i	θ_i	d_i	a_i	α_i
1	θ_1	450	0	90°
2	θ_2	0	600	0°
3	θ_3	0	600	0°
4	θ_4	0	175	−90°
5	$\theta_5 - 90°$	0	0	−90°
6	θ_6	164	0	0°

7.6. Les référentiels DH sont illustrés à la figure 7.23.

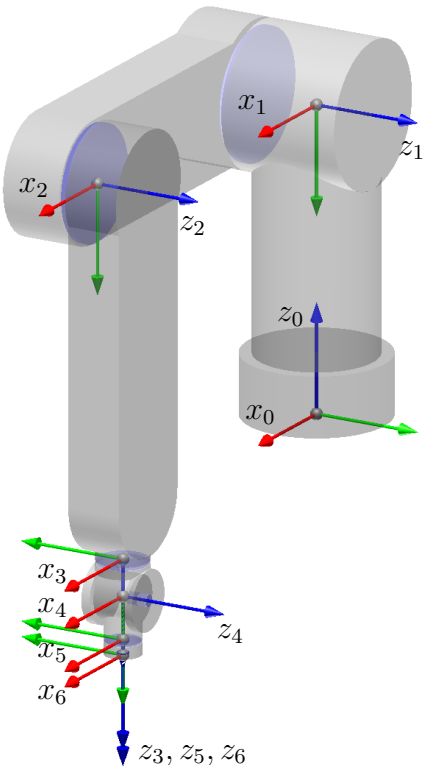


FIGURE 7.23 – Référentiels DH pour le robot de l'exercice 7.6.

Chapitre 8

Cinématique inverse

La cinématique inverse est, sans aucune doute, la partie la plus difficile d'un cours en robotique. Contrairement à la cinématique directe, il n'existe pas de méthodes systématiques pour résoudre la cinématique inverse d'un robot sériel sous forme analytique. D'ailleurs, il n'est même pas possible de résoudre la cinématique inverse, sous forme analytique, de certains robots sériels (par exemple, le vieux robot FANUC en face du local A-3336). Par contre, comme nous l'avons déjà vu dans la section 5.1, la solution de la cinématique inverse d'un robot sériel implique uniquement la solution des équations trigonométriques relativement simples. Souvent, ce qui est difficile n'est pas de résoudre la cinématique inverse sous forme analytique, mais de le faire de manière élégante, avec une compréhension géométrique des résultats, et sans oublier aucun cas particulier.

Dans ce chapitre, nous allons présenter la solution de la cinématique inverse du robot Denso présenté dans le chapitre précédent. La solution semble complexe, mais sachez que ceci est un des exemples les plus complexes.

8.1 Cinématique inverse du robot VP-5243

Pour résoudre la cinématique inverse d'un robot à n articulation, il faut généralement résoudre sa cinématique directe et poser l'équation suivante :

$$\mathbf{H}_{outil}^{atelier} = \mathbf{H}_{0}^{atelier}\mathbf{H}_{n}^{0}\mathbf{H}_{outil}^{n}. \tag{8.1}$$

où les $\mathbf{H}_{0}^{atelier}$ et $\mathbf{H}_{outil}^{n}$ sont connues, ainsi que la matrice $\mathbf{H}_{outil}^{atelier}$ (c'est la pose désirée), alors que la matrice $\mathbf{H}_{n}^{0}$ est connue sous forme d'une expression en fonction des n variables articulaires qui sont maintenant considérées comme inconnues.

Cette équation peut être réécrit sous la forme suivante :

$$\mathbf{H}_{n}^{0} = \left(\mathbf{H}_{0}^{atelier}\right)^{-1}\mathbf{H}_{outil}^{atelier}\left(\mathbf{H}_{outil}^{n}\right)^{-1} \equiv \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{8.2}$$

où n_x, n_y, n_z, o_x, o_y, o_z, a_x, a_y, a_z, p_x, p_y et p_z seront considérés comme connues.

Ainsi, pour résoudre la cinématique inverse du robot VP-5243, il faut « simplement » résoudre l'équation matricielle ci-dessous :

$$\begin{bmatrix} -c_1c_{234}c_5 + s_1s_5 & c_1c_{234}s_5 + s_1c_5 & c_1s_{234} & c_1(70s_{234} + 220s_{23} + 210s_2) \\ -s_1c_{234}c_5 - c_1s_5 & s_1c_{234}s_5 - c_1c_5 & s_1s_{234} & s_1(70s_{234} + 220s_{23} + 210s_2) \\ s_{234}c_5 & -s_{234}s_5 & c_{234} & 280 + 70c_{234} + 220c_{23} + 210c_2 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{8.3}$$

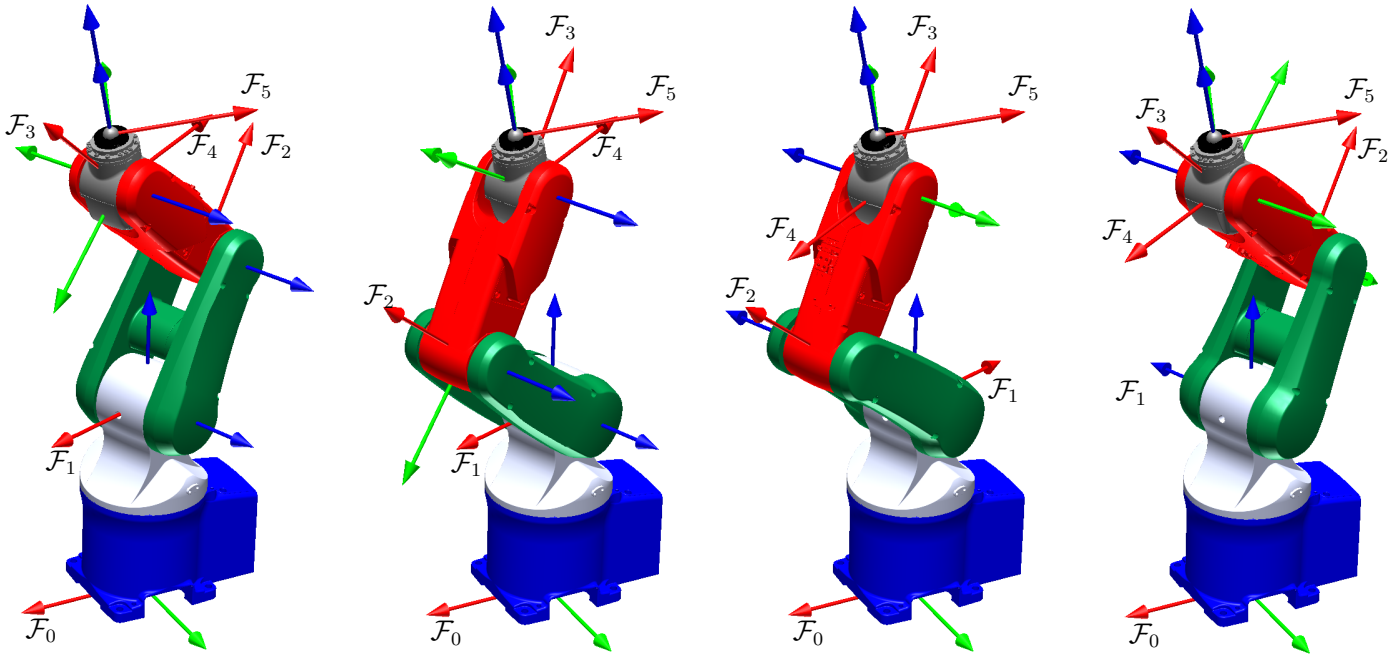


FIGURE 8.1 – Les quatre configurations non-singulières possible pour le robot VP-5243.

Cette équation matricielle représente en fait douze équations trigonométriques, en cinq inconnues : θ_1, θ_2, θ_3, θ_4 et θ_5. Malheureusement, ces douze équations sont trop complexes. Heureusement, il existe une façon simple de simplifier certaines de ces équations, en les découplant. Il faut simplement prémultiplier les deux côtés de l'équation matricielle (8.3) par $(\mathbf{H}_1^0)^{-1}$. Cette opération va nous débarrasser des expressions en θ_1 dans la matrice à gauche :

$$\begin{bmatrix} -c_{234}c_5 & c_{234}s_5 & s_{234} & 70s_{234}+220s_{23}+210s_2 \\ -s_{234}c_5 & s_{234}s_5 & -c_{234} & -70c_{234}-220c_{23}-210c_2 \\ -s_5 & -c_5 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} =$$

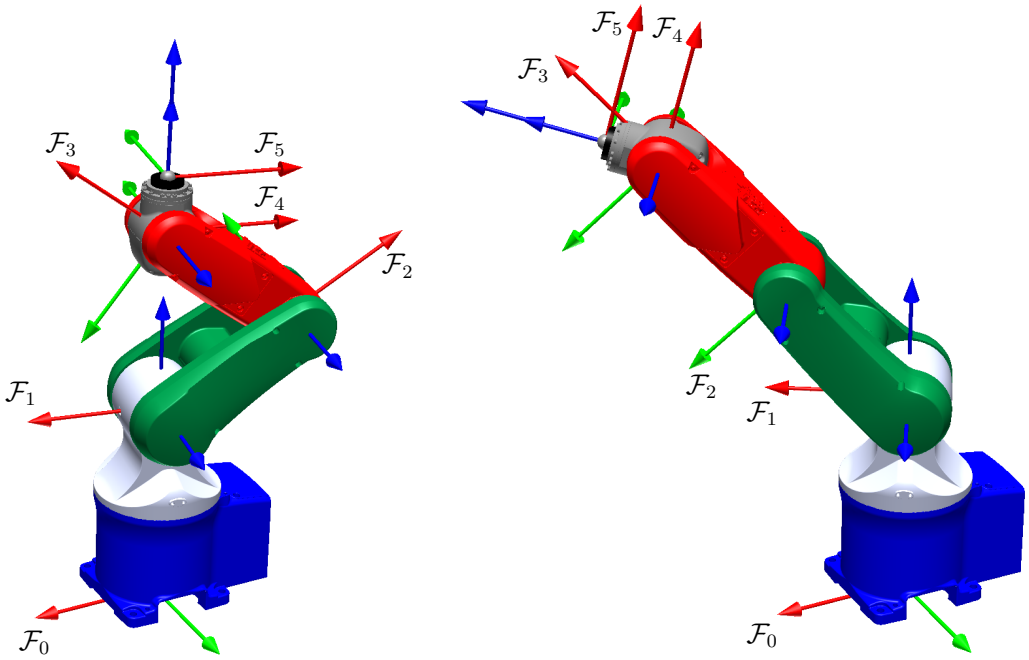
$$\begin{bmatrix} n_xc_1+n_ys_1 & o_xc_1+o_ys_1 & a_xc_1+a_ys_1 & p_xc_1+p_ys_1 \\ -n_z & -o_z & -a_z & 280-p_z \\ -n_xs_1+n_yc_1 & -o_xs_1+o_yc_1 & -a_xs_1+a_yc_1 & -p_xs_1+p_yc_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \quad (8.4)$$

Pour continuer, il faut absolument se faire une idée du nombre de solutions possibles à la cinématique inverse de notre robot. C'est probablement la partie la plus difficile ; elle nécessite nécessite de l'expérience. La figure 8.1 illustre les quatre solutions possibles à la cinématique inverse pour le robot VP-5243, dans le cas général. Ainsi, un roboticien expérimenté saura toute de suite que l'angle ta_1 acceptera deux solutions, avec un décalage de 180°.

Cette observation est très importante, car elle nous indique que nous n'avons que trouver une seule équation trigonométrique linéaire en $\cos\theta_1$ et $\sin\theta_1$. Pourtant, nous avons les deux équations suivantes, tirée de l'équation matricielle (8.4) :

$$-a_x\sin\theta_1 + a_y\cos\theta_1 = 0, \quad (8.5)$$
$$-p_x\sin\theta_1 + p_y\cos\theta_1 = 0. \quad (8.6)$$

Pourquoi nous ne pouvons pas résoudre ces deux équations en θ_1 avec nos calculatrices TI ? Tout d'abord, il faut réaliser que le vecteur $[a_x, a_y]^T$ est la projection du vecteur unitaire le long de l'axe z_5 du



(a) (b)

FIGURE 8.2 – Les deux types de configurations singulières du robot VP-5243 : (a) singularité d'épaule ; (b) singularité de coude.

référentiel $\mathcal{F}_5$ sur le plan x_0y_0 du référentiel $\mathcal{F}_0$. De façon similaire, le vecteur $[p_x, p_y]^T$ est la projection du vecteur qui relie l'origine du référentiel $\mathcal{F}_0$ avec l'origine du référentiel $\mathcal{F}_5$ sur le plan x_0y_0 du référentiel $\mathcal{F}_0$. Il est facile de voir que, en général, ces deux projections sont colinéaires (c'est-à-dire que le ratio $a_x : a_y$ est égale au ratio $p_x : p_y$). Ceci veut dire que les deux équations ci-dessus sont dépendantes et que nous ne pouvons pas les considérer simultanément.

Cependant, à cette étape, une erreur fréquence sera de considérer seulement une des deux équations, par exemple l'équation 8.5. Mais qu'est-ce qu'il va arriver si l'axe z_5 pointe vers le haut (c'est-à-dire que $a_x = a_y = 0$) ? Notre équation aura la forme $0 = 0$. De façon similaire, si l'origine du référentiel $\mathcal{F}_5$ se trouve sur l'axe 1 du robot, $p_x = p_y = 0$, et l'équation 8.6 dégénéra (aura la forme $0 = 0$). Voici pourquoi, la solution pour la variable articulaire θ_1 devra prendre en compte les deux équations :

$$\theta_1 = \begin{cases} \text{atan2}(a_y, a_x) \text{ et } \text{atan2}(-a_y, -a_x), \text{ si } a_x^2 + a_y^2 \neq 0 \\ \text{atan2}(p_y, p_x) \text{ et } \text{atan2}(-p_y, -p_x), \text{ si } a_x^2 + a_y^2 = 0 \text{ et } p_x^2 + p_y^2 \neq 0 \end{cases} \tag{8.7}$$

Oui, mais qu'est-ce qu'il va arriver si $a_x = a_y = p_x = p_y = 0$, c'est-à-dire si l'axe 5 du robot coïncide avec l'axe 1, tel qu'illustré à la figure 8.2(a) ? Ce type de configuration s'appelle une *singularité d'épaule*. Dans une telle configurations, nous pouvons varier θ_1 et θ_5 avec la même vitesse mais en sens inverse, sans que la pose de la bride (du référentiel $\mathcal{F}_5$) change par rapport à la base du robot (référentiel $\mathcal{F}_0$). Ainsi, la solution complète pour θ_1 sera :

$$\theta_1 = \begin{cases} \text{atan2}(a_y, a_x) \text{ et } \text{atan2}(-a_y, -a_x), \text{ si } a_x^2 + a_y^2 \neq 0 \\ \text{atan2}(p_y, p_x) \text{ et } \text{atan2}(-p_y, -p_x), \text{ si } a_x^2 + a_y^2 = 0 \text{ et } p_x^2 + p_y^2 \neq 0 \\ \text{arbitraire} \qquad\qquad\qquad\qquad\quad \text{, si } a_x^2 + a_y^2 = 0 \text{ et } p_x^2 + p_y^2 = 0 \end{cases} \tag{8.8}$$

Une fois les valeurs de θ_1 obtenues, nous pouvons maintenant trouver θ_5 à partir des deux équations

suivantes tirées de l'équation matricielle (8.4) :

$$-\sin\theta_5 = -n_x\sin\theta_1 + n_y\cos\theta_1, \tag{8.9}$$
$$-\cos\theta_5 = -o_x\sin\theta_1 + o_y\cos\theta_1. \tag{8.10}$$

Ainsi pour chaque solution de θ_1, nous pouvons trouver une seule solution pour θ_5 :

$$\theta_5 = \text{atan2}\left(n_x\sin\theta_1 - n_y\cos\theta_1, o_x\sin\theta_1 - o_y\cos\theta_1\right). \tag{8.11}$$

Il est important de comprendre que l'équation 8.11 est toujours définie (c'est-à-dire que les arguments de la fonction atan2 ne sont jamais zéro en même temps). En effet, il faut noter que les matrices dans l'équation (8.4) correspondent à la pose du référentiel $\mathcal{F}_5$ par rapport au référentiel $\mathcal{F}_1$. Ainsi, les expressions $-n_x\sin\theta_1 + n_y\cos\theta_1$ et $-o_x\sin\theta_1 + o_y\cos\theta_1$ correspondent aux projections des vecteurs unitaires le long des axes x_5 et y_5, respectivement, sur l'axe z_0. Or, il est évident que x_5 et y_5 ne peuvent pas être normaux à l'axe z_1 (celui qui est le long de l'axe 2 du robot) en même temps.

De façon similaire, nous pouvons trouver θ_{234} ($\theta_{234} = \theta_2 + \theta_3 + \theta_4$) à partir des deux équations suivantes tirées de l'équation matricielle (8.4) :

$$\sin\theta_{234} = a_x\cos\theta_1 + a_y\sin\theta_1, \tag{8.12}$$
$$-\cos\theta_{234} = -a_z. \tag{8.13}$$

Ainsi pour chaque solution de θ_1, nous pouvons trouver une seule solution pour θ_{234} :

$$\theta_{234} = \text{atan2}\left(a_x\cos\theta_1 + a_y\sin\theta_1, a_z\right). \tag{8.14}$$

Suivant la même logique, les deux expressions $a_x\cos\theta_1 + a_y\sin\theta_1$ et $-a_z$ correspondent aux projection du vecteur unitaire le long de l'axe z_5 sur les axes x_1 et y_1, respectivement. Pour que ces deux expression soient simultanément zéro, il faut que l'axe z_5 soit parallèle à l'axe z_1, ce qu'il est impossible, car ces axes sont toujours normaux.

Nous pouvons continuer avec l'équation matricielle (8.4) pour trouver θ_{23} et θ_3, mais il nous sera plus difficile de faire des observations géométriques. Nous allons plutôt post-multiplié les deux cotés de cette équation matricielle par la matrice $\left(\mathbf{H}_5^4\right)^{-1}$, pour nous débarrasser de θ_5 du côté gauche :

$$\begin{bmatrix} -c_{234} & 0 & s_{234} & 220s_{23} + 210s_2 \\ -s_{234} & 0 & -c_{234} & -220c_{23} - 210c_2 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} =$$
$$\begin{bmatrix} -q_2s_5 + q_1c_5 & q_1s_5 + q_2c_5 & a_ys_1 + a_xc_1 & -70a_xc_1 - 70a_ys_1 + p_xc_1 + p_ys_1 \\ o_zs_5 - n_zc_5 & -n_zs_5 - o_zc_5 & -a_z & 280 + 70a_z - p_z \\ q_3s_5 + q_4c_5 & q_4s_5 - q_3c_5 & -a_xs_1 + a_yc_1 & 70a_xs_1 - 70a_yc_1 - p_xs_1 + p_yc_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{8.15}$$

où $q_1 = n_ys_1 + n_xc_1$, $q_2 = o_ys_1 + o_xc_1$, $q_3 = o_xs_1 - o_yc_1$ et $q_4 = -n_xs_1 + n_yc_1$.

Avant de continuer, il faut noter que les matrice homogène de cette dernière équation matricielle correspondent à la pose du référentiel $\mathcal{F}_4$ par rapport au référentiel $\mathcal{F}_1$. Or, il est évident que la distance entre les origines de ces deux référentiels dépendent uniquement de l'angle θ_3. Ainsi, pour obtenir une équation en θ_3 seulement, nous allons mettre au carré les composantes en position de l'équation matricielle (8.15) et prendre leur somme :

$$(220s_{23} + 210s_2)^2 + (220c_{23} - 210c_2)^2 + 0^2 =$$
$$(-70a_xc_1 - 70a_ys_1 + p_xc_1 + p_ys_1)^2 + (280 + 70a_z - p_z)^2 + (70a_xs_1 - 70a_yc_1 - p_xs_1 + p_yc_1)^2. \tag{8.16}$$

Après simplifications (par exemple, en utilisant la fonction `tSimplify` de vos calculatrices TI) et quelques réarrangement manuels, cette équation devient :

$$220^2 + 210^2 + 2(210)(220)\cos\theta_3 = (p_x - 70a_x)^2 + (p_y - 70a_y)^2 + (pz - 280 - 70a_z)^2. \tag{8.17}$$

L'équation (8.19) n'est rien d'autre que la lois des cosinus appliquée au triangle composé par les origines des référentiels $\mathcal{F}_1$, $\mathcal{F}_2$, et $\mathcal{F}_3$. Cette équation a une solution si est seulement si

$$(220 - 210)^2 \leq (p_x - 70a_x)^2 + (p_y - 70a_y)^2 + (pz - 280 - 70a_z)^2 \leq (220 + 210)^2. \tag{8.18}$$

Autrement dit, les deux inégalités (8.18) définissent l'enveloppe du travail du robot. Si ces deux inégalités sont satisfaites, la solution pour l'angle θ_3 est

$$\theta_3 = \pm \arccos\left(\frac{(p_x - 70a_x)^2 + (p_y - 70a_y)^2 + (pz - 280 - 70a_z)^2 - 92500}{92400}\right). \tag{8.19}$$

En pratique, lorsque le bras proximal et le bras distal sont alignés tel qu'illustré à la figure 8.2(b), c'est-à-dire lorsque les origines des référentiels $\mathcal{F}_1$, $\mathcal{F}_2$, et $\mathcal{F}_3$ sont colinéaires, avec l'origine du référentiel $\mathcal{F}_2$ entre les deux autres, il existe une seule solution pour θ_5, $\theta_5 = 0$. Cette configuration s'appelle une *singularité de coude*.

Tout ce qui reste maintenant c'est de trouver θ_2. Nous pouvons faire ça en considérant les deux équations suivantes tirées de l'équation matricielle (8.15) :

$$\begin{aligned} 220\sin\theta_{23} + 210\sin\theta_2 &= -70a_x\cos\theta_1 - 70a_y\sin\theta_1 + p_x\cos\theta_1 + p_y\sin\theta_1, & (8.20)\\ -220\cos\theta_{23} - 210\cos\theta_2 &= 280 + 70a_z - p_z, & (8.21)\end{aligned}$$

que nous pouvons réarranger pour obtenir :

$$\begin{aligned} (220\cos\theta_3 + 210)\sin\theta_2 + 220\sin\theta_3\cos\theta_2 &= -70a_x\cos\theta_1 - 70a_y\sin\theta_1 + p_x\cos\theta_1 + p_y\sin\theta_1, & (8.22)\\ 220\sin\theta_3\sin\theta_2 - (220\cos\theta_3 + 210)\cos\theta_2 &= 280 + 70a_z - p_z. & (8.23)\end{aligned}$$

La solution de ce système de deux équations linéaires en $\sin\theta_2$ et $\cos\theta_2$ est

$$\begin{aligned} \sin\theta_2 &= \frac{220r_2\sin\theta_3 + 220r_1\cos\theta_3 + 210r_1}{92500 + 92400\cos\theta_3}, & (8.24)\\ \cos\theta_2 &= \frac{-220r_2\cos\theta_3 + 220r_1\sin\theta_3 - 210r_2}{92500 + 92400\cos\theta_3}, & (8.25)\end{aligned}$$

où $r_1 = -70a_x\cos\theta_1 - 70a_y\sin\theta_1 + p_x\cos\theta_1 + p_y\sin\theta_1$ et $r_2 = 280 + 70a_z - p_z$. Puisque le dénominateur dans les deux équations ci-dessus est toujours positive, la solution pour θ_2 est

$$\theta_2 = \operatorname{atan2}\left(220r_2\sin\theta_3 + 220r_1\cos\theta_3 + 210r_1, -220r_2\cos\theta_3 + 220r_1\sin\theta_3 - 210r_2\right). \tag{8.26}$$

Il faut bien comprendre que, en général, il y aura quatre solutions différentes pour θ_2, puisqu'il y a deux solutions pour θ_3 et deux pour θ_1.

Finalement, il ne reste qu'obtenir θ_4 :

$$\theta_4 = \theta_{234} - \theta_2 - \theta_3. \tag{8.27}$$

Pour conclure la cinématique inverse, nous allons présenter les variables articulaires pour les quatre solutions qui correspondent à la figure 8.1 au tableau 8.1. Quant à la pose du référentiel $\mathcal{F}_5$ par rapport au référentiel $\mathcal{F}_0$ dans ces quatre configurations, elle est :

$$\mathbf{H}_5^0 \approx \begin{bmatrix} -0.400 & 0.884 & 0.243 & 76.723 \\ -0.898 & -0.431 & 0.089 & 27.925 \\ 0.183 & -0.183 & 0.966 & 670.893 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{8.28}$$

Tableau 8.1 – Les quatre solutions de la cinématique inverse présentée à la figure 8.1.

solution	θ_1	θ_2	θ_3	θ_4	θ_5
1	20.000°	−30.000°	80.000°	−35.000°	45.000°
2	20.000°	52.236°	−80.000°	42.764°	45.000°
3	−160.000°	−52.236°	80.000°	−42.764°	−135.000°
4	−160.000°	30.000°	−80.000°	35.000°	−135.000°

Comme vous avez vu, la solution de la cinématique inverse d'un robot sériel à six articulations rotoïde est assez compliquée. La solution de la cinématique inverse d'un robot sériel à six articulations rotoïde est très similaire, quoiqu'un peu plus difficile. Cette solution, dans le cas du robot IRB 120 vous sera présentée en classe.

8.2 La cinématique inverse et le langage RAPID

En pratique, il peut vous arriver parfois de vouloir résoudre la cinématique inverse d'un robot industriel. Le premier exemple sera évidement lorsque vous voulez savoir les différentes configurations possibles pour une pose donnée de l'effecteur du robot. Bien sûr, vous pouvez même visualiser ces configurations en utilisant RobotStudio ou RoKiSim, mais ceci ne peut pas être automatisée. Un deuxième exemple sera lorsque vous voulez convertir du code G en langage RAPID. Bien sûr, vous pouvez utiliser un logiciel comme Robotmaster, mais ce logiciel est assez dispendieux.

La meilleure solution sera évidement de résoudre la cinématique inverse par vous même, mais il existe une fonction en RAPID qui peut le faire à votre place :

Exemple 8.1 – Fonction CalcJointT

```
MODULE Exemple()
   VAR jointtarget jointpos1;
   CONST robtarget P1:=[[300,0,500],[0,1,0,0],[0,-1,0,0],[9E9,9E9,9E9,9E9,9E9,9E9]];;

   jointpos1 := CalcJointT(P1, tPince_bloc \WObj:=wobj1);
ENDMODULE
```

Le paradoxe de cette fonction est que vous devez lui fournir non seulement la pose, mais aussi le confdata (cf1, cf4, cf6, et cfx). Comment est-ce qu'on peut savoir dans quel cadran se trouve l'angle θ_1, par exemple, alors qu'on cherche cet angle ? La solution c'est d'exécuter la fonction CalcJointT avec toutes les combinaisons de confdata possibles. L'exemple 8.2 vous présente une telle solution possible. Ce code est une adaptation du code présenté dans le rapport final d'Éric Cotnoir, dans le cadre du cours GPA792 (décembre 2012). Avec une telle approche exhaustive, nous pouvons obtenir toutes les solutions de la cinématique inverse d'un robot ABB, sans même connaître la fonction arccos. Cependant, cette approche ne vous sera d'aucune utilité à l'examen final...

Exemple 8.2 – Astuce pour calculer les huit solutions de la cinématique inverse en RAPID

```
MODULE MainModule

  VAR iodev jtfile;
  VAR robtarget r;
  VAR jointtarget tryconf;
  VAR bool notfound := TRUE;
  VAR bool goodconf := TRUE;
  VAR num cf1 := -2;
  VAR num cf4 := -2;
  VAR num cf6 := -2;
  VAR num idx;
  VAR jointtarget j{8};

  PROC Main()
```

```
    Open "U:"\File:="jointtargets.txt",jtfile\Write;
    idx := 1;

    !Calculer un rabtarget qui est a l'interieur de l'espace de travail du robot IRB 120
    r := CalcRobT([[100,40,-89,150,-40,-70], [9E9,9E9,9E9,9E9,9E9,9E9] ], tool0);
    Write jtfile, "Position (x,y,z) : " + NumToStr(r.trans.x,3) + " mm, " \NoNewLine;
    Write jtfile, NumToStr(r.trans.y,3) + " mm, " + NumToStr(r.trans.z,3) + " mm";
    Write jtfile, "Orientation (az, ay, ax) : " + NumToStr(EulerZYX(\Z, r.rot),3) +" deg, " \NoNewLine;
    Write jtfile, NumToStr(EulerZYX(\Y, r.rot),3) + " deg, " + NumToStr(EulerZYX(\X, r.rot),3) + " deg";
    Write jtfile,"";
    notfound := TRUE;
    goodconf := TRUE;

    FOR cfx FROM 0 TO 7 DO !on cherche les huit solutions de la cinematique inverse
      notfound := TRUE;
      WHILE notfound AND cf1 <= 1 DO
        WHILE notfound AND cf4 <= 1 DO
          WHILE notfound AND cf6 <= 1 DO
            tryconf := CalcJointT([r.trans,r.rot,[cf1,cf4,cf6,cfx],[9E9,9E9,9E9,9E9,9E9,9E9]],tool0);
            !il faut absolument ajouter ces cinq conditions en plus de goodconf
            IF goodconf AND tryconf.robax.rax_1 <= 180
                        AND tryconf.robax.rax_4 >= -180 AND tryconf.robax.rax_4 < 180
                        AND tryconf.robax.rax_6 >= -180 AND tryconf.robax.rax_6 < 180 THEN
              j{idx} := tryconf;
              Write jtfile,"j{" + NumToStr(idx,0) + "} := [[" \NoNewLine;
              Write jtfile,NumToStr(tryconf.robax.rax_1,3) + "," \NoNewLine;
              Write jtfile,NumToStr(tryconf.robax.rax_2,3) + "," \NoNewLine;
              Write jtfile,NumToStr(tryconf.robax.rax_3,3) + "," \NoNewLine;
              Write jtfile,NumToStr(tryconf.robax.rax_4,3) + "," \NoNewLine;
              Write jtfile,NumToStr(tryconf.robax.rax_5,3) + "," \NoNewLine;
              Write jtfile,NumToStr(tryconf.robax.rax_6,3) + "]," \NoNewLine;
              Write jtfile,"[9E9,9E9,9E9,9E9,9E9,9E9]];";
              idx := idx + 1;
              notfound := FALSE; !des qu'on trouve une solution, on passe au prochain cfx
            ENDIF
            goodconf := TRUE;
            cf6 := cf6 + 1;
          ENDWHILE
          cf6 := -2;
          cf4 := cf4 + 1;
        ENDWHILE
        cf4 := -2;
        cf1 := cf1 + 1;
      ENDWHILE
      cf1 := -2;
    ENDFOR

    Close jtfile;

    ERROR
    !Lorsque le confdata dans CalcJointT n'est pas bon, RAPID genere l'erreur numero 1074 (ERR_ROBLIMIT)
    IF ERRNO = ERR_ROBLIMIT THEN
      goodconf := FALSE;
      TryNext;
    ENDIF

  ENDPROC
ENDMODULE
```

Le résultat du programme est présenté ci-dessous.

Exemple 8.3 – Résultat de l'exécution du code précédent

```
Position (x,y,z) : -44.116 mm, 383.452 mm, 786.010 mm
Orientation (az, ay, ax) : 170.877 deg, -0.374 deg, 77.705 deg

j{1} := [[100.000,27.116,-64.900,-38.989,30.718,120.975],[9E9,9E9,9E9,9E9,9E9,9E9]];
j{2} := [[100.000,27.116,-64.900,141.011,-30.718,-59.025],[9E9,9E9,9E9,9E9,9E9,9E9]];
j{3} := [[100.000,40.000,-89.000,-30.000,40.000,110.000],[9E9,9E9,9E9,9E9,9E9,9E9]];
j{4} := [[100.000,40.000,-89.000,150.000,-40.000,-70.000],[9E9,9E9,9E9,9E9,9E9,9E9]];
j{5} := [[-80.000,-40.000,-64.900,158.992,63.703,95.797],[9E9,9E9,9E9,9E9,9E9,9E9]];
j{6} := [[-80.000,-40.000,-64.900,-21.008,-63.703,-84.203],[9E9,9E9,9E9,9E9,9E9,9E9]];
j{7} := [[-80.000,-27.116,-89.000,156.375,53.320,100.785],[9E9,9E9,9E9,9E9,9E9,9E9]];
j{8} := [[-80.000,-27.116,-89.000,-23.625,-53.320,-79.215],[9E9,9E9,9E9,9E9,9E9,9E9]];
```

8.3 Exercices

8.1. Solutionner la cinématique inverse du robot de l'exercice 7.1, si la pose du référentiel de l'outil du robot par rapport au référentiel de l'atelier est donnée par

$$\mathbf{H}_{outil}^{atelier} = \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

8.2. Solutionner la cinématique inverse du robot de l'exercice 7.2, si la pose du référentiel de l'outil du robot par rapport au référentiel de l'atelier est donnée par

$$\mathbf{H}_{outil}^{atelier} = \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

8.3. Solutionner la cinématique inverse du robot de l'exercice 7.3, si la pose du référentiel de l'outil du robot par rapport au référentiel de l'atelier est donnée par

$$\mathbf{H}_{outil}^{atelier} = \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

8.4. Solutionner la cinématique inverse du robot de l'exercice 7.4, si la position de la bride du robot par rapport au référentiel de base du robot est donnée par

$$\mathbf{v}_3^0 = \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix}.$$

8.4 Réponses aux exercices

8.1. Les expressions pour les quatre variables articulaires sont données ci-dessous :

$$\begin{aligned}
\theta_2 &= \pm \arccos\left(\frac{p_x^2 + p_y^2 - 157600}{15600}\right), \\
\theta_1 &= \text{atan2}\left(13p_y \sin\theta_2 + 13p_x \cos\theta_2 + 15p_x, 13p_x \sin\theta_2 - 13p_y \cos\theta_2 - 15p_y\right), \\
d_3 &= 325.5 - p_z, \\
\theta_4 &= \text{atan2}(n_y, n_x) - \theta_1 - \theta_2.
\end{aligned}$$

Pour obtenir θ_2, vous devez mettre deux équations au carré et prendre leur somme.

8.2. Les expressions pour les trois variables articulaires sont données ci-dessous :

$$\begin{aligned}
\theta_1 &= \text{atan2}(-n_y, -n_x), \\
\theta_2 &= \text{atan2}(-o_z, -a_z), \\
d_3 &= p_x n_y o_z + (600 - p_y) n_x o_z + (p_z - 550) a_z.
\end{aligned}$$

Pour obtenir d_3, vous devez pré-multiplier votre équation matricielle initiale par

$$\left(\mathbf{H}_2^1\right)^{-1} \left(\mathbf{H}_1^0\right)^{-1} \left(\mathbf{H}_0^{atelier}\right)^{-1}.$$

8.3. Les expressions pour les quatre variables articulaires sont données ci-dessous :

$$\begin{aligned}
d_2 &= p_z - 405, \\
d_3 &= -279.5 \pm \sqrt{-21904 + q_1^2 + q_2^2}, \\
\theta_1 &= \text{atan2}(2d_3 q_2 + 559 q_2 + 296 q_1, 2d_3 q_1 - 296 q_2 + 559 q_1), \\
\theta_4 &= \text{atan2}(a_y, a_x) - \theta_2,
\end{aligned}$$

où $q_1 = -133a_x + p_x + 600$ et $q_2 = -133a_y + p_y$. Il est très facile d'obtenir d_2 et $(\theta_1 + \theta_2)$ à partir de l'équation matricielle initiale. Pour obtenir les autres variables articulaires, il faut pré-multiplier l'équation matricielle initiale par $\left(\mathbf{H}_0^{atelier}\right)^{-1}$ et la post-multiplier par $\left(\mathbf{H}_{outil}^4\right)^{-1} \left(\mathbf{H}_4^3\right)^{-1}$. La nouvelle équation matricielle représentera la matrice homogène $\mathbf{H}_3^0$. Or, il est clair que la distance entre l'origine du référentiel $\mathcal{F}_3$ et l'axe z_0 ne dépend que de la distance d_3. Ainsi, il faut prendre la somme des carrés des composantes en x et en y de la nouvelle équation matricielle pour obtenir une équation en d_3, comme seule inconnue. Une fois d_3 trouvée, il est facile de trouver θ_1 et, ensuite, θ_4.

8.4. Les expressions pour les trois variables articulaires sont données ci-dessous :

$$\begin{aligned}
\theta_1 &= \begin{cases} \text{atan2}(p_y, p_x) \text{ et } \text{atan2}(-p_y, -p_x), & \text{si } p_x^2 + p_y^2 \neq 0 \\ \text{arbitraire} & \text{, si } p_x^2 + p_y^2 = 0 \end{cases}, \\
\theta_3 &= \pm \arccos\left(\frac{p_x^2 + p_y^2 + q^2 - 2587600}{2520000}\right), \\
\theta_2 &= \text{atan2}\left(q(63c_3 + 50) - 63(c_1 p_x + s_1 p_y)s_3, (c_1 p_x + s_1 p_y)(63c_3 + 50) + 63 q s_3\right),
\end{aligned}$$

où $q = 560 - p_z$, $s_1 = \sin\theta_1$, $c_1 = \cos\theta_1$, $s_3 = \sin\theta_3$ et $c_3 = \cos\theta_3$. Pour obtenir ces trois variables articulaires, vous devez pré-multiplier votre équation matricielle initiale par $\left(\mathbf{H}_1^0\right)^{-1}$.

Chapitre 9

Programmation avancée

Ce chapitre aborde les fonctions plus avancées de la réalisation d'une tâche (« task ») dans un contrôleur de robot ABB.

9.1 Structure de programmation

9.1.1 L'utilisation de plusieurs modules

Il est possible de créer ou de charger plusieurs modules dans une tâche (« task ») du robot. Par défaut, le contenu des modules est en déclaration globale, ce qui signifie qu'une procédure ou une fonction dans un module peut être appelée dans un autre module. Les registres mémoires (VAR, CONST et PERS) déclarés de manière globale dans l'entête d'un module sont également accessibles dans tous les autres modules.

La segmentation d'un programme en modules est surtout utile pour le support et le déverminage. Par exemple, il devient facile de recharger ou modifier un module sans altérer le contenu des registres d'un autre module. La segmentation permet également d'épurer le module principal en déplaçant toutes les routines non critiques dans un ou plusieurs modules connexes.

Il est également possible de changer le comportement d'un module afin de faciliter l'exécution d'un programme en mode manuel, ou de protéger le contenu d'un module que l'on ne désire pas dévoiler. Il existe cinq attributs possibles pour un module, comme montré au tableau 9.1.

Tableau 9.1 – Descriptions des attributs d'un module.

Attribut	Description
NoStepIn	le code du module n'est pas affiché lors d'une exécution pas à pas.
ViewOnly	le code du module ne peut pas être modifié dans le FlexPendant.
ReadOnly	le code du module ne peut pas être modifié dans le FlexPendant ; toutefois, cet attribut peut être supprimé à partir du FlexPendant.
NoView	le code du module ne peut être visualisé ; uniquement exécuté. Les routines globales peuvent être appelées à partir d'autres modules et sont toujours exécutées avec l'attribut NoStepIn. Il est possible d'atteindre les valeurs courantes concernant les données globales à partir d'autres modules ou depuis la fenêtre de données du boîtier de commandes. NoView peut être défini uniquement hors ligne (avec un éditeur de texte).
SysModule	le module est chargé de façon permanente dans la partie système de la tâche du robot.

L'exemple 9.1 montre l'utilisation d'un attribut NoStepIn dans un module contenant les affichages nécessaires d'un programme classique.

Exemple 9.1 – Exemple d'attribut d'un module en langage RAPID.

```
MODULE MainModule

 PERS robtarget rRetrait:=[[337.69,-859.05,774.51], [0.000762984,0.852406,-0.522879,0.000750765],
     [-1,-1,-2,0], [9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];

!***************************************************************************************
PROC main()

 ! Deplacement au robtarget repos
 MoveJ rRetrait, V1000, fine, tPince_bloc\wobj:=wobj0;
 TPMessage RobotARepos;

 IF DI09_ZS0101 = 0 THEN
  TPMessage RobotAttentePieceMagasin;
  WaitDI DI09_ZS0101, 1; ! Attente d'un bloc
 ENDIF

ENDPROC
!***************************************************************************************
ENDMODULE


MODULE Affichage(NOSTEPIN)

CONST NUM RobotAttentePieceMagasin:=1;
CONST NUM RobotARepos:=2;

! ************************************************************************
! *** appel de la bonne routine pour afficher du texte a l'operateur     **
PROC TPMessage(num NumeroMessage)
 Test NumeroMessage

 Case RobotAttentePieceMagasin:
  Message1;
 Case RobotARepos:
  Message2;
 DEFAULT:
  TPWrite "Erreur de message par le programmeur";
  WaitTime 2;
  Stop;
 EndTest
ENDPROC

PROC Message1()
 TPErase;
 ! s'assure de rester dans les 40 carac. d'une ligne
 !       "1234567890123456789012345678901234567890"
 TPWRITE "**************************************";
 TPWRITE "**    En attente d'une piece dans    **";
 TPWRITE "**            le magasin.            **";
 TPWRITE "**                                   **";
 TPWRITE "**************************************";
ENDPROC

PROC Message2()
 TPErase;
 !       "1234567890123456789012345678901234567890"
 TPWRITE "**************************************";
 TPWRITE "**    Le robot est au robtarget       **";
 TPWRITE "**    -->       REPOS.      <--       **";
 TPWRITE "**                                   **";
 TPWRITE "**************************************";
ENDPROC

ENDMODULE
```

9.1.2 La gestion d'erreurs

Dans un programme RAPID, il est normal de gérer les erreurs qui se produisent et qui sont prévisibles comme le montre l'exemple 9.2. Il est possible de mettre en place un gestionnaire d'erreurs local dans chaque routine. Cette section de la routine s'exécute lorsqu'une erreur dans la couche programme se produit dans la routine, telle qu'une division par zéro. Il faut comprendre que ce n'est pas le rôle du gestionnaire d'erreurs de gérer des erreurs de plus haut niveau, tel qu'une collision ou un passage par une configuration singulière.

Exemple 9.2 – Exemple de gestionnaire d'erreurs en RAPID.

```
MODULE MainModule
    VAR robtarget Rob1;

PROC main()
    ! definit la pose du robtarget
    Rob1.trans:=[500,500,50]; Rob1.rot:=OrientZYX(0,0,180);
    ! desactive l'axe externe
    Rob1.extax:=[9E9,9E9,9E9,9E9,9E9,9E9];
    ! definit la config mecanique souhaite
    Rob1.robconf:=[-1,0,2,0];

 if Atteignable(Rob1) then
  MoveJ Rob1,v1000,fine,tPince_bout;
 else
  TPWrite "Le robtarget n'est pas atteignable.";
 endif

 !...

ENDPROC

FUNC bool Atteignable(robtarget rTmp)
  VAR jointtarget q;
  q := CalcJointT(rTmp,Tpince_Bout,\Wobj:=wobj0);
  Return (true);
ERROR
  ! Erreur de programme du au calcul de la cinematique inverse qui ne se resout pas
  Return (false);
ENDFUNC

ENDMODULE
```

Quand le pointeur de programme est catapulté dans le gestionnaire d'erreurs d'une routine, il est possible de consulter la variable système ERRNO qui contient le numéro de la faute commise (par exemple, 1074). Il en existe plusieurs centaines de numéros d'erreurs, dont la plupart ont une constante associée (par exemple, la constante ERR_ROBLIM est déclarée avec la valeur 1074). Il est possible de faire différentes actions, commandes ou mouvements dans un gestionnaire d'erreurs, et ce afin de répondre aux différents besoins face à l'erreur commise. Une fois l'erreur sous contrôle, on doit renvoyer le pointeur de programme au bon endroit afin de reprendre l'exécution normale du programme.

Pour définir comment on désire reprendre l'exécution du programme, les instructions possibles sont Retry, TryNext, Return et Raise. Une erreur survenue dans une routine A peut être renvoyée au gestionnaire d'erreurs de la routine B qui a appelé la routine A, à l'aide de l'instruction Raise. Si, lors du renvoi, aucun gestionnaire d'erreurs n'est présent dans la routine B, alors le gestionnaire du robot reprend le contrôle et arrête le robot. La figure 9.1 nous montre où il est possible de renvoyer le pointeur de programme afin de reprendre l'exécution normale du programme.

Il est possible de faire sa propre bibliothèque d'erreurs à l'aide de l'instruction BookErrNo afin de mieux gérer un programme. Cette instruction est régulièrement utilisée quand on développe des modules complets qui seront utilisés dans plusieurs projets différents.

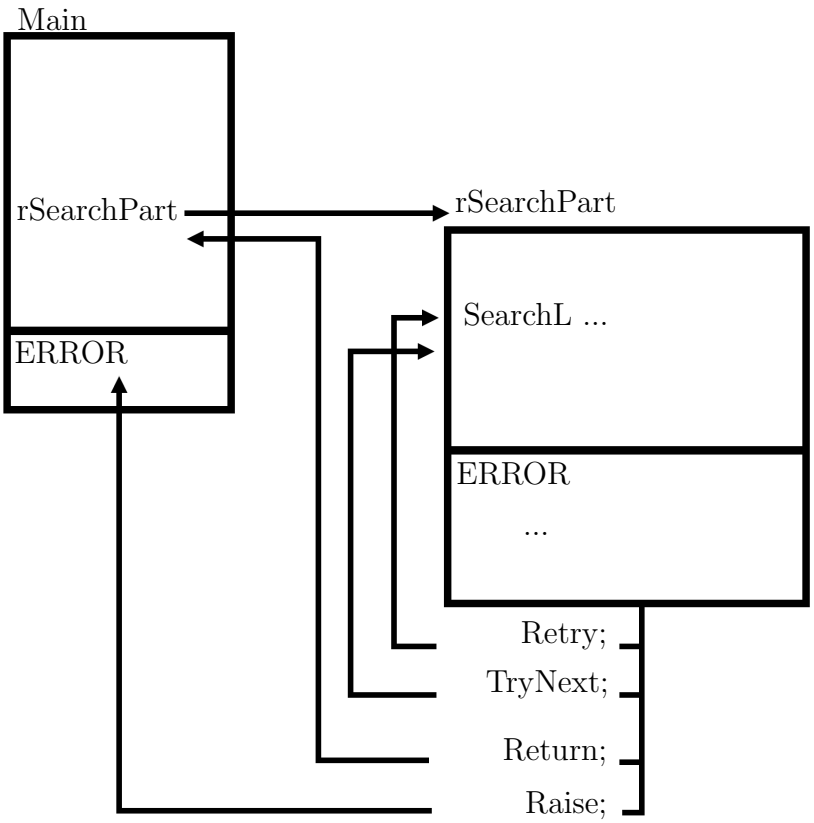


FIGURE 9.1 – Choix de reprise du programme.

9.2 Instructions de haut niveau

9.2.1 Les interruptions

Les interruptions ont pour mandat de mettre temporairement la séquence normale du programme en pause (pas nécessairement le pointeur de mouvement) afin d'exécuter immédiatement une routine bien précise que l'on appelle une trappe (« TRAP »), ou une routine d'interruption. Il est important de savoir que dans un contrôleur ABB, il n'y a qu'un seul niveau de priorité pour les interruptions, qu'elles soient globales ou locales. Donc, si plusieurs interruptions se déclenchent dans un court laps de temps, leurs trappes respectives seront exécutées une après l'autre, dans l'ordre du déclenchement.

Il est important de comprendre que les interruptions n'ont pas pour mandat de remplacer des systèmes plus complexes comme les automates programmables, souvent présents dans une cellule robotisée. Le contrôleur percevra l'activation d'une interruption, mais exécutera la routine d'interruption qu'une fois les opérations minimales accomplit, ce qui signifie qu'il peut exister un délai entre 2 ms et 30 ms entre le déclenchement de l'interruption et l'exécution de la routine d'interruption. Ce délai n'est pas constant ; il dépend de ce que le contrôleur est en train d'exécuter. Par exemple, si le contrôleur est en train d'exécuter une instruction MoveL, ce délai sera beaucoup plus grand que si le contrôleur est en train d'exécuter l'instruction WaitDI.

Dans un robot ABB, ce qu'on appelle interruptions globales sont des variables de type intnum. C'est par cette variable que l'on peut appliquer un certain contrôle (expliqués dans le tableau 9.4), comme l'activation, la mise en sommeil et autres. Il y a deux autres informations importantes à associer à l'interruption. La première information est le déclencheur (expliqués dans le tableau 9.2), qui peut être une entrée, une sortie, une persistante, du temps ou encore un état du contrôleur. Il ne peut avoir qu'un seul déclencheur par interruption. La deuxième information à associer à l'interruption est le nom de la routine d'interruption qui contient le code à exécuter. Il est donc possible d'associer une routine d'interruption à plus d'une interruption. L'exemple 9.3 montre bien l'exercice d'association du déclencheur et de la routine d'interruption à une interruption.

Exemple 9.3 – Interruption en RAPID

```
MODULE MainModule
VAR bool Bouton:=false;
VAR intnum intBoutonPresse; !le nom de l'interruption

PROC main()
 IDelete intBoutonPresse; ! pour s'assurer que l'interruption n'existe pas
 CONNECT intBoutonPresse WITH TrBouton; ! associe a l'interruption la routine d'interruption
 ISignalDI di_virtuel1_bouton1, 1, intBoutonPresse; ! associe l'entree a surveiller a l'interruption
 ...

ENDPROC

! Routine d'interruption execute des qu'on presse le bouton "-" du FlexPendant
TRAP TrBouton
 ISleep intBoutonPresse;
 Bouton:=true;
ENDTRAP

ENDMODULE
```

9.2.2 Interruptions locales

Comme expliqué précédemment, le contrôleur d'un robot ABB a pour habitude de laisser son pointeur de programme prendre de l'avance sur le pointeur de mouvement aussitôt qu'un mouvement n'a pas une zone demandant l'arrêt du robot (par exemple, lorsqu'on utilise l'instruction MoveL avec le paramètre z0 plutôt que le paramètre fine). Ceci signifie que si dans les lignes de code qui suivent une instruction

Tableau 9.2 – Descriptions des types de déclencheur d'interruption.

Déclencheur	Description
ISignalAI	Interruption à partir d'un signal d'entrée analogique.
ISignalAO	Interruption à partir d'un signal de sortie analogique.
ISignalDI	Interruption à partir d'un signal d'entrée numérique, déclenché sur un front.
ISignalDO	Interruption à partir d'un signal de sortie numérique, déclenché sur un front.
ISignalGI	Interruption à partir d'un groupe de signaux d'entrée numériques.
ISignalGO	Interruption à partir d'un groupe de signaux de sortie numériques.
IPers	Interruption lors d'un changement de valeur d'une variable persistante.
ITimer	Interruption à partir d'un chronomètre (précision d'environ 10 ms pour une exécution unique ou 100 ms pour une exécution cyclique).
IError	Interruption à partir d'une erreur du contrôleur (par exemple, une collision ou un passage par une configuration singulière). Exige une programmation multitâche que nous n'abordons pas dans ce cours.

Tableau 9.3 – Instructions pour le contrôle des interruptions.

Instruction de contrôle	Description
IDelete	Suppression d'une interruption, si existante.
Connect	Création et association d'une interruption à une routine d'interruption. Il est important de noter que l'on ne peut changer l'association sans supprimer l'interruption au préalable. Par exemple, dans le cas d'une exécution continue, il faut commencer par un IDelete ou utiliser l'instruction Connect dans une zone qui est exécutée une seule fois.
ISleep	Utilisé pour mettre en sommeil une interruption. Donc, une activation du déclencheur n'a aucun effet durant le sommeil.
IWatch	Utilisé pour mettre en mode surveillance une interruption (après avoir utilisé l'instruction ISleep), les déclenchements durant le sommeil ne sont pas repris.
IDisable	Utilisé pour mettre en attente l'exécution des routines d'interruption. Les interruptions sont vues, mais les routines d'interruption « TRAP » sont exécutées seulement après la réactivation par l'instruction IEnable.
IEnable	Utilisé pour mettre en activité toutes les interruptions.

Tableau 9.4 – Instructions pour le contrôle des interruptions.

de mouvement (par exemple, MoveL) font une action physique, comme une activation d'une sortie, il est certain que celle-ci sera exécutée avant la fin du mouvement du robot et rarement au même endroit du parcours.

Pour arriver à synchroniser une sortie ou l'exécution d'une routine simple, comme montrée à la figure 9.2, il faut utiliser des instructions de mouvement spécifiques. Le premier groupe d'instructions nous permet de synchroniser un changement d'état sur une sortie et un mouvement : MoveLDO, MoveJDO et MoveCDO. Le deuxième groupe d'instructions nous permet de synchroniser l'exécution d'une routine et un mouvement : MoveLSync, MoveJSync et MoveCSync. Il faut noter que si l'exécution de la routine n'est pas finie avant la fin du mouvement, cela force le robot à s'arrêter avant de continuer sur les prochaines instructions du programme.

Le troisième groupe concerne les amorces de type « Trigger », soit des interruptions qui ne s'activent pas de manière systématique, mais bien seulement lors d'événements bien spécifiés par le programmeur. Nous n'allons aborder que la plus commune des instructions de ce groupe, soit l'instruction TriggIO. Elle

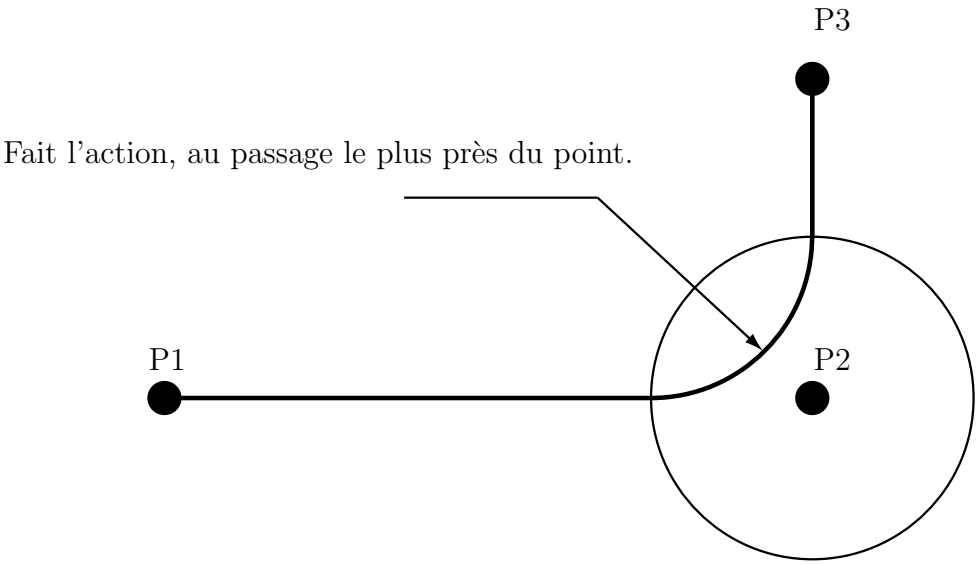


FIGURE 9.2 – Lieu d'une action sur les mouvements avec interruptions locales

sert à une action qu'y s'amorce à un(e) certain(e) temps/distance avant la fin ou après le début d'un déplacement. Une fois l'interruption définie, il faut l'associer à un mouvement. On fait cette association à l'aide de l'instruction TriggL, qui est l'équivalent de l'instruction MoveL additionné d'une interruption, comme le montre l'exemple 9.4.

La notion d'interruption amorcée avec le temps se fait uniquement avant la fin du mouvement. La valeur ne peut pas excéder le temps de freinage du robot, soit 0,5 seconde. S'il a une immobilisation du robot à la fin de déplacement (par exemple, si le paramètre fine est utilisé), la précision sera d'environ 5 ms. Imprécision sera nettement plus élevé s' il y a une zone d'interpolation dans le mouvement n'immobilisant pas le robot (par exemple, si le paramètre z50 est utilisé).

Exemple 9.4 – L'instruction TriggIO en RAPID.

```
VAR triggdata gunon;
VAR triggdata gunoff
 ...
!partir la colle 50~mm apres le point de depart
TriggIO gunon,  50 \Start \Dop := do20, 1;
!arreter la colle 0,1 s avant la fin du mouvement
TriggIO gunoff, 0.1\Time  \Dop := do20, 0;
!
TriggL pStart, v500, gunon, fine, tGun;
TriggL pEnd, v100, gunoff, fine, tGun;
```

⚠ Il est important de comprendre que ces actions synchronisées sont en fait des interruptions locales créées et déclenchées au bon moment dans le mouvement. Elles sont donc régies par la règle du premier arrivé, premier servi. Il est donc fortement déconseillé d'utiliser ces instructions à l'intérieur d'une routine d'interruption.

```
VAR triggdata Variable;
 ...
TriggIO Variable, Distance \Dop := Sortie, Valeur;
```

1. TriggIO : Instruction qui créer l'interruption locale (amorce).
2. Variable : variable qui contient la description et comportement de l'interruption.
3. Distance : valeur en mm du point d'arrivée ou de départ si la clef est spécifiée. Cette valeur peut également être du temps en seconde, si la clef est spécifiée.

4. \Start : Clef optionnelle qui spécifie si travail à partir du point de départ.
5. \Time : Clef optionnelle qui spécifie que la valeur de distance est en temps.
6. \Dop : Permet de dire sur quelle sortie l'interruption agira. Le paramètre est optionnel, car on pourrait agir sur un autre type de sortie, exemple analogique.
7. Valeur : valeur que l'on désire mettre dans la sortie au moment de l'interruption.

```
TriggL robtarget, speeddata, amorce, zonedata, tooldata, \wobj:=wobjdata;
```

1. TriggL : Commande pour un déplacement linéaire de l'endroit où il se trouve vers le robtarget spécifié, en ayant une ou plusieurs interruptions locales associées.
2. robtarget : Valeur venant d'une variable, fonction ou autre qui indique la destination de l'outil.
3. speeddata : Vitesse de croisière maximale que l'outil peut d'atteindre durant le déplacement.
4. amorce : amorce (au préalablement créée) qui peut s'activer durant le mouvement.
5. zonedata : Précision d'arrêt ou d'approche de l'origine du référentiel de l'outil sur un robtarget à atteindre.
6. tooldata : Le référentiel de l'outil et les caractéristiques physiques de l'outil.
7. wobjdata : Paramètre optionnel qui défini par rapport à quel référentiel de travail on veut positionner le référentiel de l'outil à la pose définie dans le robtarget. Si ce paramètre est omis, le référentiel par défaut est celui de l'atelier (appelée wobj0).

9.2.3 La recherche (instructions SearchL et SearchC)

Dans nombreuses applications industrielles, un robot doit faire un exercice de localisation (par exemple, à l'aide de capteurs montés sur l'effecteur, tels qu'un palpeur), avant de démarrer une séquence. Pour arriver à cela, le robot procède à un déplacement contrôlé (linéaire ou circulaire) avec une surveillance de l'état d'un capteur. Dès que le capteur atteint une valeur souhaitée, le contrôleur mémorise instantanément la pose de l'effecteur du robot (plus précisément, le robtarget de l'outil). Plusieurs options existent, comme immobiliser le robot lors du changement d'état du capteur, faire la recherche du front montant ou descendant sur le capteur, etc. Cependant, il est important de comprendre que le capteur a un temps de réaction. En plus, celui-ci est souvent branché sur une carte qui communique par un bus de terrain. Ainsi, des petits délais de réponse s'accumulent et si en plus le robot est en train de se déplacer rapidement, l'erreur de l'exercice de déplacement peut être assez importante (par exemple, 1 mm). C'est pour cette raison qu'un exercice de localisation se fait souvent en rebond : on trouve grossièrement l'objet, on recule un peu et on reprendre la recherche, mais avec une vitesse moins élevée.

Dans le langage RAPID, ce sont les instructions SearchL et SearchC qui sont utilisées pour la localisation (la recherche). Ces deux instructions effectuent un MoveL ou MoveC, respectivement, tout en observant une entrée du système. Si jamais le robot atteint le robtarget final sans avoir perçu l'état recherché sur le capteur, une erreur est levée et le registre ERRNO prend la valeur de la constante ERR_WHLSEARCH.

Les options classiques d'une instruction SearchL sont les suivantes :

```
SearchL  \Stop Entree, robVu, robObjectif, vitesse, Outil, \WObj;
```

1. SearchL : Instruction de recherche durant un déplacement vers robObjectif.
2. \Stop : Clef optionnelle qui spécifie si le robot arrête son déplacement ou non.
3. Entree : signal d'entrée sous surveillance.
4. robVu : variable dans laquelle le robtarget du robot sera sauvegardé.
5. robObjectif : objectif du déplacement linéaire.
6. vitesse : vitesse linéaire de l'outil.
7. Outil : outil actif pour le déplacement.
8. \Wobj : référentiel nécessaire si différent du wobj0.

L'exemple 9.5 illustre l'utilisation d'un de nos robots IRB 1600 qui recherche le bloc le plus haut d'une pile de blocs. La logique de détection est que quand la ventouse No. 1 est activée et touche à une surface (celle d'un bloc ou de la table), la pression de l'air monte et la sortie DI01_EE_VAC01 devient activée. Une fois un bloc trouvé, le robot le saisit à l'aide des deux ventouses et le place dans la glissoire. La prise du bloc suivant se fait à partir du robtarget précédent, décalé en z négatif par l'épaisseur d'un bloc. Une fois la pile finie, le robot retourne à sa configuration de repos.

Exemple 9.5 – Exemple d'un exercice de recherche en RAPID.

```
MODULE MainModule
 ! Donnees de type robtarget enseignes
 PERS robtarget rPriseB:=[[87.70,655.55,382.72], [0.00877888,0.0413324,0.998982,0.0157715], [0,0,-3,0],
     [9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
 PERS robtarget rCourant:=[[87.6979,655.556,393.592], [0.00876966,0.0413314,0.998982,0.0157768], [0,0,-3,0],
     [9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
 PERS robtarget rDepot:=[[-456.27,779.08,567.71], [0.00880226,0.0413122,0.998982,0.0157957], [1,0,-3,0],
     [9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
 PERS robtarget rRetrait:=[[115.95,660.35,868.08], [0.399696,0.0309939,0.915456,-0.0349546], [0,-1,-3,0],
     [9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
 ! Donnees de type constante
 CONST num Decalage:=-200; ! Distance d'approche ou de retrait (mm)
 ! Donnees de type variable
 CONST num EpaisMM:=25.4;

!*********************************************************************************
PROC main()
 Accset 50,50;
 ! lance la recherche au dessus de la pile
 rCourant:=Offs(rPriseB,0,0,3.5*EpaisMM);
 WHILE (true) DO
  Prise;
  Depot;
 ENDWHILE
ENDPROC

!*********************************************************************************
PROC Prise()
 MoveJ RelTool(rCourant,0,0,Decalage),v1000,z50,tSuce1\wobj:=wobj0;
 ! se replace a la derniere position
 MoveL rCourant, v1000, fine, tSuce1\WObj:=wobj0;
 Set DO02_EE_VENT01;
 Set DO03_EE_VENT02;
 ! recherche un bloc
 SearchL \stop, DI01_EE_VAC01, rCourant, rPriseB, v10, tSuce1\wobj:=wobj0;
 MoveL RelTool(rCourant,0,0,Decalage), v1000, z50, tSuce1\wobj:=wobj0;
ERROR
 IF ERRNO = ERR_WHLSEARCH THEN
  Reset DO02_EE_VENT01;
  Reset DO03_EE_VENT02;
        MoveL RelTool(rPriseB,0,0,Decalage), v1000, z50, tSuce1\wobj:=wobj0;
```

```
        MoveJ rRetrait, v1000, fine, tSuce1\wobj:=wobj0;
  TPWrite "Pile vide...";
  WaitTime 3;
  Exit;
 ELSE
  Raise;
 ENDIF
ENDPROC

!***********************************************************************************
PROC Depot()
 MoveJ RelTool(rDepot,0,0,Decalage), v1000, z50, tSuce1\wobj:=wobj0;
 MoveL rDepot, V100, fine, tSuce1;
 Reset DO02_EE_VENT01;
 Reset DO03_EE_VENT02;
 MoveL RelTool(rDepot,0,0,Decalage), v1000, z50, tSuce1\wobj:=wobj0;
ENDPROC
!***********************************************************************************

ENDMODULE
```

9.3 Intervention sur le mouvement du robot

Il est possible de modifier un mouvement du robot en cours à l'aide d'une interruption. Le tableau 9.5 énumère les instructions permettant d'arrêter, reprendre ou abandonner un mouvement qui est en cours d'exécution. Ces instructions peuvent être utilisées à l'intérieur d'une routine d'interruption ou d'une trap d'erreur.

Tableau 9.5 – Instructions pour le changement d'un mouvement du robot.

Instruction de contrôle	Description
StopMove	Arrêt du mouvement courant, en gardant le reste de la trajectoire actif dans la carte de contrôle.
StartMove	Reprise de la trajectoire active dans la carte de contrôle.
ClearPath	Efface la trajectoire active et/ou auxiliaire dans la carte de contrôle.
StorePath	Enregistre la trajectoire active dans la carte de contrôle, dans une mémoire auxiliaire (ne peut avoir qu'une à la fois).
RestoPath	Transfert la trajectoire stockée dans la mémoire auxiliaire vers la carte de contrôle.

L'instruction StopMove est valable que pour le mouvement courant. Cela n'empêche pas le robot d'effectuer d'autres mouvements. Cependant, il est important de comprendre que si l'on désire reprendre la trajectoire interrompue, il faut la stocker à l'aide de l'instruction StorePath (immédiatement après l'instruction StopMove), afin de pouvoir la reprendre plus tard à l'aide de l'instruction RestoPath.

L'exemple 9.6 illustre l'utilisation des instructions StopMove et ClearPath, dans une situation où le mouvement interrompu ne doit pas être repris (car les ventouses ont échappé la pièce). Cet exemple illustre également l'utilisation d'une interruption et d'un gestionnaire d'erreurs.

L'exemple 9.7 illustre l'exemple typique d'un mouvement interrompu, mais qui doit être repris. Puisque le robot sera déplacé avant d'être retourné à la configuration initiale (lorsque l'interruption est survenue), on doit utiliser les instructions CRobt, StorePath et RestorePath. Il faut noter que dans ce type de situation, il faut absolument utiliser l'instruction CRobT pour pouvoir enregistrer la configuration initiale (lorsque l'interruption est survenue). Si l'on utilise simplement l'instruction RestoPath, le robot n'ira pas nécessairement à la configuration initiale ou ne sera pas en mesure de reprendre le parcours.

Exemple 9.6 – Annulation d'un mouvement.

```
MODULE MainModule
VAR intnum drop_payload;
CONST errnum ERR_DROP_LOAD := -1;

PROC minicycle()
 BookErrNo ERR_DROP_LOAD;
 Transport;
ERROR (ERR_DROP_LOAD)
 Retry; !Transport sera re-executee
ENDPROC

PROC Transport()
 Set DO02_EE_VENT01;
 Set DO03_EE_VENT02;
 CONNECT drop_payload WITH gohome;
 ISignalDI \Single, DI01_EE_VAC01, 0, drop_payload;
 MoveL p1, v500, fine, tSuce1;
 ...
 IDelete drop_payload;
 Reset DO02_EE_VENT01;
 Reset DO03_EE_VENT02;
ENDPROC

TRAP gohome
 StopMove \Quick;
 ClearPath;
 IDelete drop_payload;
 MoveL pRetrait, v500, fine, tSuce1;
 Raise ERR_DROP_LOAD;
ERROR
 Raise; !propage l'erreur directement au gestionnaire d'erreurs de minicycle()
ENDTRAP

ENDMODULE
```

Exemple 9.7 – Interruption temporaire d'un mouvement.

```
TRAP BuseObstruee
 VAR robtarget pTemp;
 StopMove; ! arrete le mouvement courant
 StorePath; ! enregistre le reste de la trajectoire
 pTemp := CRobT(\tool := tPince); ! capture la pose courante (absolument necessaire)
 MoveL Reltool(pTemp,0,0,-200), v100, z50, tPince; ! degagement
 NettoyageOutil; ! fait l'exercice de nettoyer l'outil, suite au probleme
 MoveJ Reltool(pTemp,0,0,-200), v100, z50, tPince; ! va a l'approche de la position d'arret
 MoveL pTemp, v100, fine, tPince; ! va exactement a l'endroit ou le robot a ete arrete
 RestoPath; ! retablit la trajectoire stocke
 StartMove; ! reprend la trajectoire stocke
ENDTRAP
```

9.4 Zones atelier

Dans la plupart des robots industriels, il existe des moyens (souvent en option payante) pour vérifier si un PDO (point de l'outil, c.-à-d., l'origine du référentiel de l'outil) se trouve dans une zone ou non. Dans le cas des robots ABB, cette notion s'appelle WorldZone. Une « zone atelier » a deux mandats possibles : (1) la limitation d'accès ou (2) la surveillance de présence. En pratique, une zone atelier est utilisée principalement pour empêcher que le PDO de rentrer dans une zone.

En RAPID, il existe trois zones cartésiennes (montré à la figure 9.3) : la boîte, la sphère ou le cylindre vertical. Il existe également deux zones articulaires. Un robot peut avoir plusieurs zones atelier actives en même temps. Ils peuvent être permanentes, activée en tout temps dès la mise sous-tension d'un robot, ou temporaires, que l'on peut créer, supprimer, activer ou désactiver selon les besoins du projet.

Il est important de comprendre que ces zones ne peuvent en aucun cas être considérées 100% sécuritaires, car seule la position d'un point de l'effecteur du robot est surveillée (dans le cas des zones

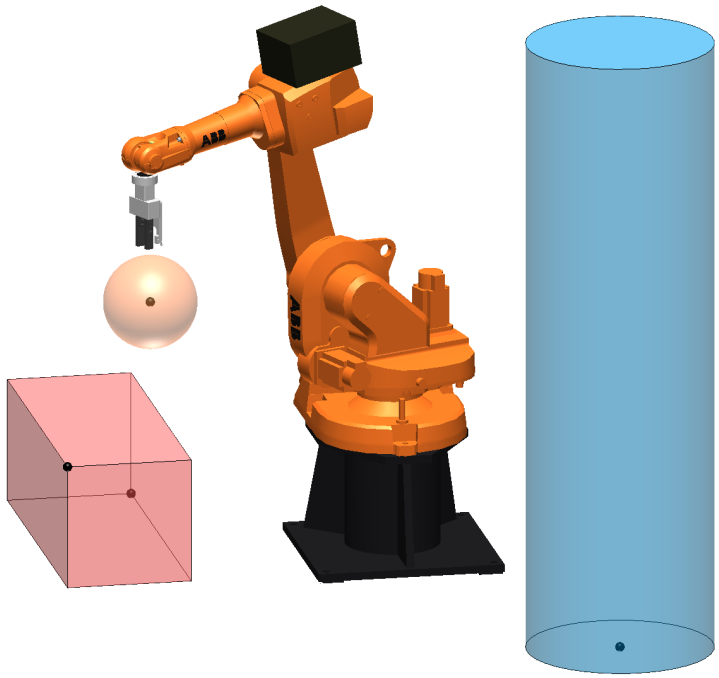


FIGURE 9.3 – Les trois zones atelier cartésiennes possibles.

cartésiennes) et non l'ensemble du robot. Ainsi, ces zones permettent uniquement de diminuer les risques de collision.

On doit définir un volume de type shapedata et savoir si c'est l'intérieur ou l'extérieur de ce volume que l'on désire surveiller, comme le montre l'exemple 9.8.

Exemple 9.8 – Création d'un shapedata en RAPID.

```
VAR shapedata Table;
CONST robtarget Coin1Table:=[[-102.00,-166.00,230.44], [0.533634,-0.506523,-0.473548,0.48417], [-1,-1,0,0],
    [9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
VAR  robtarget Coin2Table:=[[302.560,-365.08,230.44], [0.533634,-0.506523,-0.473548,0.48417], [-1,-1,0,0],
    [9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];

PROC Main()
    Coin2Table.trans.z:=-230;
    WZBoxDef \Inside, Table, Coin1Table.trans, Coin2Table.trans;
    ...
ENDPROC
```

Les paramètres d'un shapedata de type cartésien sont limités aux options suivantes :

```
WzBoxDef \Inside | \Outside, shapedata, coin1, coin2;
WzSphDef \Inside | \Outside, shapedata, centre, rayon;
WzCylDef \Inside | \Outside, shapedata, centre, rayon, hauteur;
```

1. WzBoxDef, WzSphDef ou WzCylDef : paramètre pour la forme voulue lors de la création d'un volume.
2. \Inside | \Outside : clef mutuellement exclusive qui définit si l'on surveille l'intérieur ou l'extérieur du volume.
3. shapedata : variable contenant les caractéristiques du volume créé.
4. coin1, coin2 et centre : vecteurs de type POS qui définissent la position du volume par rapport à au référentiel de l'atelier (Wobj0).
5. rayon, hauteur : dimensions en millimètres définissant les caractéristiques du volume.

Les paramètres d'un shapedata de type articulaire sont limités aux options suivantes :

```
WZLimJointDef  \Inside | \Outside, shapedata, JointInf, JointSup;
WZHomeJointDef \Inside | \Outside, shapedata, JointPosition, Delta;
```

1. WZLimJointDef ou WZHomeJointDef : paramètre pour la limitation de la zone (min/max ou autour d'un JointTarget).
2. \Inside | \Outside : Clef mutuellement exclusive qui définit si l'on surveille l'intérieur ou l'extérieur du volume.
3. shapedata : variable contenant les caractéristiques du volume créé.
4. JointInf et JointSup : vecteur articulaire (de type JointTarget) définissant les valeurs minimum et maximum atteignables pour chacune des articulations.
5. JointPosition : Position articulaire de type JointTarget.
6. Delta : la valeur de divergence absolue de la position articulaire.

À partir d'un volume créé, on peut définir une zone atelier temporaire (wzTemporary) ou stationnaire (wzStationary), avec un comportement informatif (reflétant la présence ou l'absence de la PDO dans la zone) ou restrictif (le PDO ne peut y aller ou ne peut pas en sortir). L'exemple 9.9 montre la création d'une restriction d'accès pour que le robot ne percute pas une porte de machine, s'il y a erreur de mouvement. Dans le cadre du laboratoire A-0610, nous devons qu'utiliser UNIQUEMENT les zones atelier temporaires puisque nous n'avons pas accès aux options permettant de mettre en place les zones atelier stationnaires.

Exemple 9.9 – Création d'une restriction en RAPID.

```
VAR wzTemporary Limite;
VAR shapedata porte;

PROC main()
WZBoxDef \Inside, porte, corner1, corner2;
WZLimSup \Temp, Limite, porte;
...
ENDPROC
```

L'instruction WZLimSup nécessite les paramètres suivants :

```
WZLimSup \Temp | \Stat, wzTemporary ou wzStationary, shapedata;
```

1. WZLimSup : Instruction pour créer une zone atelier en limitation d'accès (actif dès la création).
2. \Temp | \Stat : Définit si cette zone est toujours active ou est manipulable par le programmeur.
3. wzTemporary ou wzStationary : Variable contenant la zone créée.
4. shapedata : Le volume qui va servir de frontière limitante.

L'instruction WZDOSet nécessite les paramètres suivants :

```
WZDoSet \Temp | \Stat, wzTemporary ou wzStationary, shapedata, signal, valeur;
```

1. WZDoSet : Instruction pour créer une zone agissant sur le signal de sortie afin de refléter si le TCP est ou non dans la zone (actif dès la création).
2. \Temp | \Stat : Définit si cette zone est toujours active ou est manipulable par le programmeur.
3. wzTemporary ou wzStationary : Variable contenant la zone créée.

4. shapedata : Le volume qui va servir de frontière de permutation sur l'état du signal, peut importe la direction quand on la traverse.
5. signal : La sortie du robot qui reflétera la présence ou pas du TCP dans la zone.
6. valeur : La valeur booléenne correspondant au choix intérieur ou extérieur choisi lors de la création du shapedata .

Enfin, les instructions au tableau 9.6 permettent d'interagir avec une zone atelier temporaire, qu'elle soit informative ou restrictive.

Tableau 9.6 – Instructions pour le contrôle des zones atelier.

Instruction de contrôle	Description
WZFree	Annulation d'une zone temporaire.
WZEnable	Activation du comportement d'une zone temporaire.
WZDisable	Désactivation du comportement d'une zone temporaire.

9.5 Exercices

9.1. Une cellule robotisée est composée d'un robot IRB 6600 et de deux centres d'usinage, tel qu'illustré à la figure 9.4. Le mandat du robot est de procéder à l'évacuation des pièces usinées par chaque centre d'usinage en les déposant sur le convoyeur de sortie correspondant. Les deux centres d'usinage produisent sur demande un produit A ou un produit B, dans un ordre quelconque. Les centres d'usinage n'étant pas complètement compatible avec le robot, ceux-ci ne fournissent qu'un pulse d'une durée d'une seconde pour indiquer qu'une pièce est prête pour la prise. Pendant la durée de ce pulse, chaque centre d'usinage indique également le type de pièce (A ou B) via deux signaux digitaux distincts. Une fois la prise complétée par le robot, celui-ci doit aviser le centre d'usinage qu'il peut se remettre en production. Pour ce faire, le contrôleur du robot doit envoyer un pulse d'une durée de 10 secondes via une sortie digitale connectée au centre d'usinage. Le tableau 9.7 donne la liste des entrées/sorties définies dans le contrôleur du robot. Écrire le programme pour décharger le plus rapidement possibles les centres d'usinage. Ajouter au programme une protection immobilisant le robot, si celui-ci s'approche d'un centre d'usinage en fonction. Les routines de prise et dépôt sont récupérées d'un autre projet et disponibles dans un autre module (pas besoin de les coder) : PriseC1_ProdA, PriseC1_ProdB, PriseC2_ProdA, PriseC2_ProdB et DepotProd.

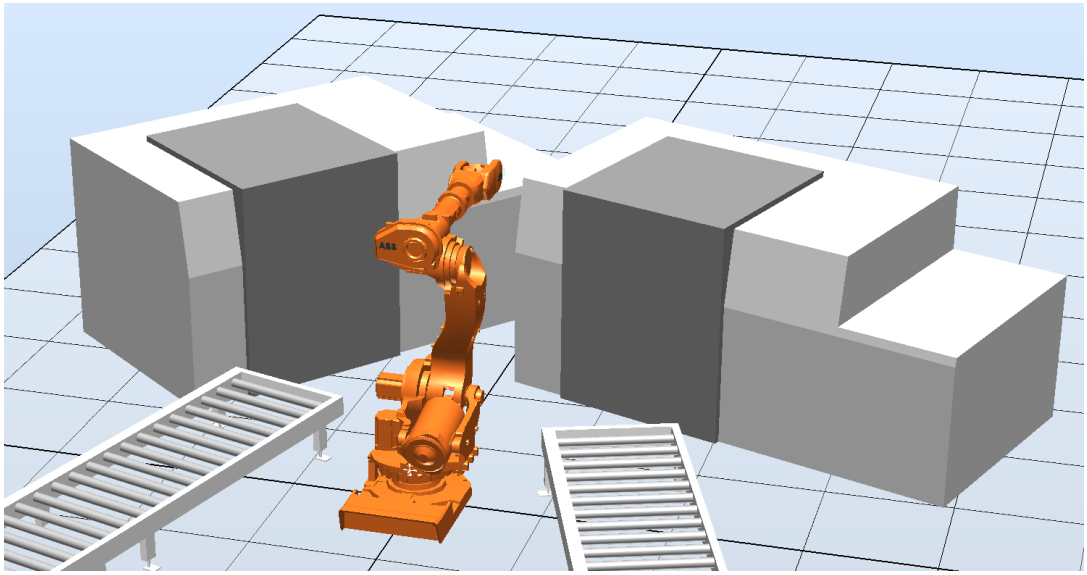


FIGURE 9.4 – Cellule robotisée pour l'exercice 9.1.

Signal	Type	Description
PcOkCentre1	entrée	Une pièce est présente au centre d'usinage 1.
PcACentre1	entrée	Il s'agit du produit A.
PcBCentre1	entrée	Il s'agit du produit B.
Centre1Libre	sortie	Le centre d'usinage 1 peut redemarer.
PcOkCentre2	entrée	Une pièce est présente au centre d'usinage 2.
PcACentre2	entrée	Il s'agit du produit A.
PcBCentre2	entrée	Il s'agit du produit B.
Centre2Libre	sortie	Le centre d'usinage 2 peut redémarrer.
ConvLibre	entrée	Les convoyeurs de sortie sont libres.
diFinCycle	entrée	Le cycle de production est terminé.

TABLEAU 9.7 – Liste des entrées/sorties pour l'exercice 1

9.2. Un robot de palletisation (à quatre degrés de liberté) a pour mandat de prendre une par une des plaques rectangulaires horizontales et de les charger de manière précise dans une machine d'usinage (figure 9.5). Malheureusement, ces plaques ne sont pas bien justifiées sur la palette et la quantité de plaques sur la palette n'est pas toujours la même. Pour cette raison, un capteur laser (capteur de présence) a été ajouté sur l'outil. Ce capteur doit être utilisé pour mesurer de façon précise l'emplacement exacte de chaque plaque.

Les plaques ont le mêmes dimensions ($48" \times 40" \times 1"$), mais leur épaisseur peut varier de quelques millièmes. Le robot possède déjà un outil à vacuum, défini dans le programme comme toolSuces. La position du capteur laser est $[0, -175, 80]$ (en mm) par rapport à tool0. Le capteur détecte la présence d'un objet de 0 mm à 100 mm en dessous du capteur (figure 9.6).

Écrire le programme qui vide la palette vers le système d'usinage. Avant de saisir une pièce, le robot devrait localiser le dessus de la pile, puis le centre et l'orientation de la première pièce en détectant les arrêtes de la pièce à l'aide du capteur laser. Le signal de détection laser est di_LaserTrouvee et l'activation des ventouses se fait avec la sortie do_VacOn. Le système possède déjà un robtaget robCentrePalette (montré à la figure 9.7). Lorsque l'outil toolSuce se trouve à ce robtarget, l'outil est au bon endroit pour la prise d'une plaque parfaitement alignée, mais à hauteur de la palette. Une palette pleine ne fera jamais plus de 1,3 m de haut.

Le signal di_SysOk est vrai si la zone de dépôt est libre. Le programme s'arrête après avoir terminé la pile sur la palette ou quand un opérateur demande la fin de cycle via le bouton di_FinCycle. Le système termine toujours sa séquence en retournant à au robtarget repos. Si une plaque est échappée durant le déplacement, le robot s'arrête immédiatement et met fin au programme.

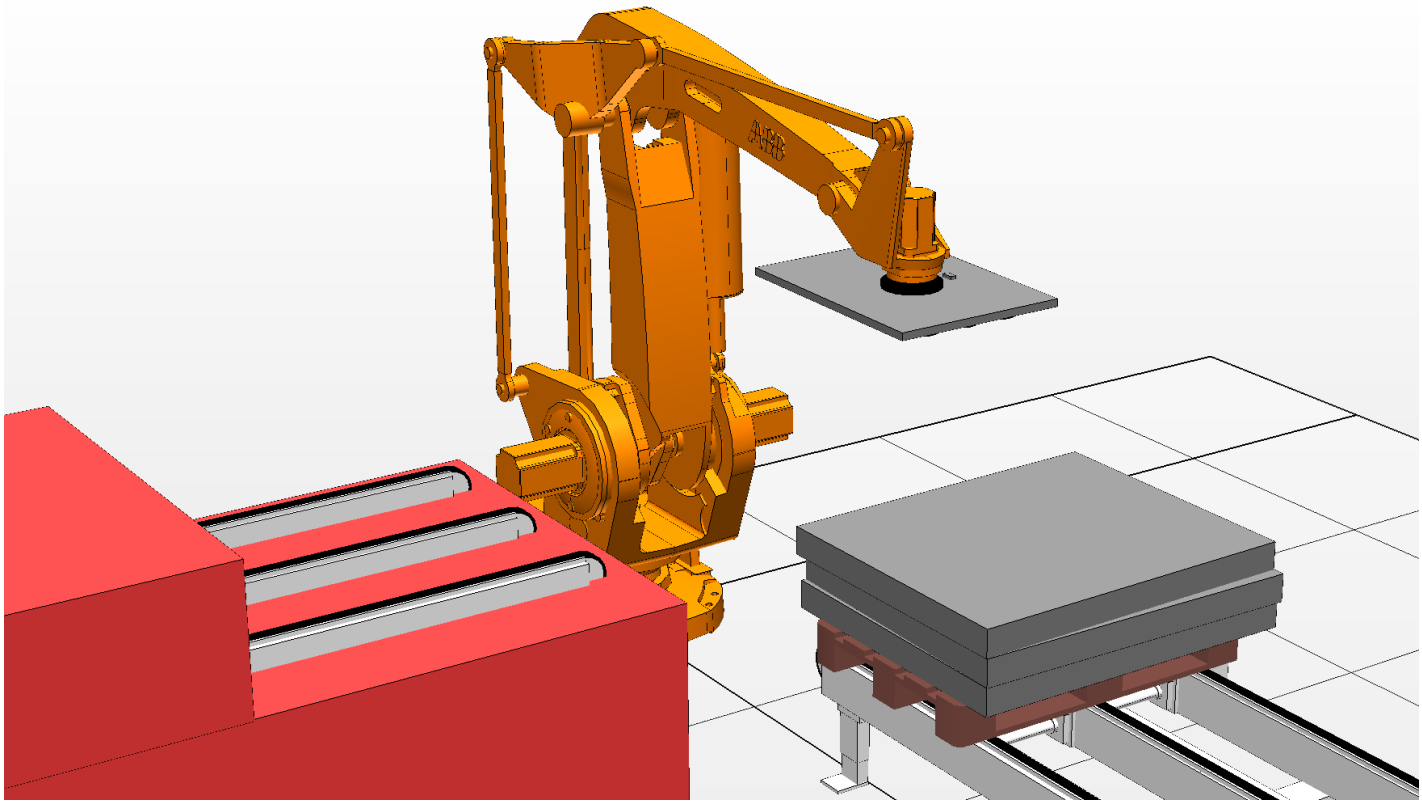


Figure 9.5 – Cellule robotisée pour l'exercice 9.2.

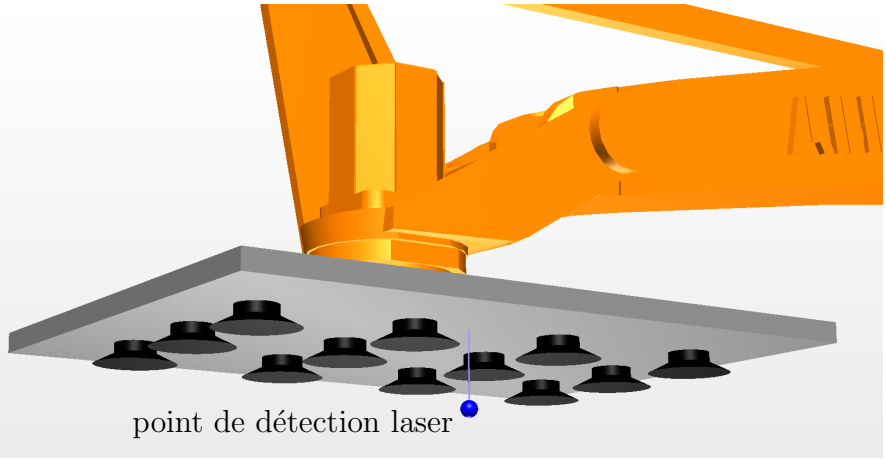


FIGURE 9.6 – Point de détection le plus éloigné du capteur laser.

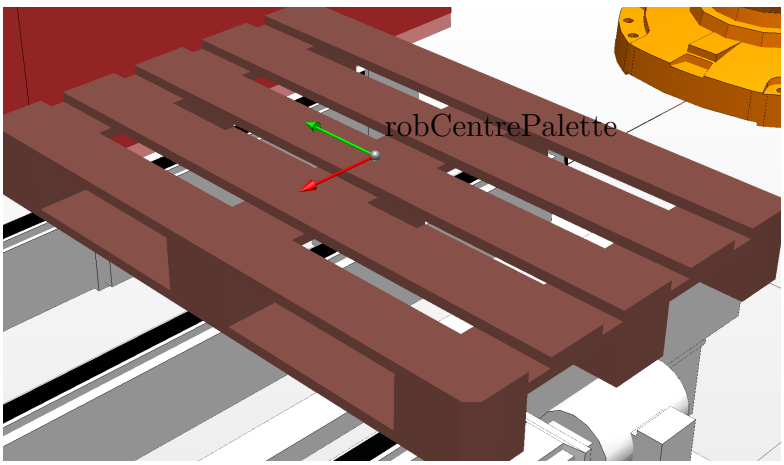


FIGURE 9.7 – Robtarget au centre de la palette.

9.6 Réponses aux exercices

9.1. Le code est présenté à l'exemple 9.10.

Exemple 9.10 – Programme pour l'exercice 9.1.

```
MODULE MainModule
  VAR num Centre1:=0;
  VAR num Centre2:=0;
  CONST num PieceA:=1;
  CONST num PieceB:=2;
  CONST num AucunePiece:=0;
  VAR intnum intCentre1;
  VAR intnum intCentre2;
  VAR wzTemporary wzCentre1;
  VAR wzTemporary wzCentre2;

!***********************************************************************************
PROC main()
  Init;
  WHILE (NOT TestDI(diFinCycle)) DO
    ! Verifie si Centre 1 a termine
    If NOT (Centre1 = AucunePiece) then
      WZDisable wzCentre1;                ! donne acces Centre1
      If (Centre1 = PieceA) PriseC1_ProdA; ! vide produit de type A
      If (Centre1 = PieceB) PriseC1_ProdB; ! vide produit de type B
      WZEnable wzCentre1;                 ! securise centre 1
      Centre1 := AucunePiece;
      IWatch intCentre1;
      PulseDO \PLength:=10, Centre1Libre;  ! avise le centre qu'il est vide
      DepotProd \Gauche;                  ! depose a gauche
    ! Verifie si Centre 2 a termine
    ElseIf NOT (Centre2 = AucunePiece) then
      WZDisable wzCentre2;                ! donne acces Centre2
      If (Centre2 = PieceA) PriseC2_ProdA; ! vide produit de type A
      If (Centre2 = PieceB) PriseC2_ProdB; ! vide produit de type B
      WZEnable wzCentre2;                 ! securise centre 2
      Centre2 := AucunePiece;
      IWatch intCentre2;
      PulseDO \PLength:=10, Centre2Libre;  ! avise le centre qu'il est vide
      DepotProd \Droit;                   ! depose a droite
    Else
      WaitTime 1;                         ! relache le processeur
    ENDIF
  ENDWHILE
  Exit;                                   ! Force le PP pour les inter. et les zones
ENDPROC

!***********************************************************************************
PROC Init()
  VAR shapedata boite1;
  VAR shapedata boite2;
  ! active les interruptions
  CONNECT intCentre1 WITH trCentre1;
  ISignalDi PcOkCentre1,1,intCentre1;
  CONNECT intCentre2 WITH trCentre2;
  ISignalDi PcOkCentre2,1,intCentre2;
  ! active la supervision des zones atelier
  WZBoxDef \Inside, boite1, [1500,100,0], [4000,4000,4000];
  WZBoxDef \Inside, boite2, [1500,-100,0], [4000,-4000,4000];
  WZLimSup \temp, wzCentre1, boite1;
  WZLimSup \temp, wzCentre2, boite2;
ENDPROC

!***********************************************************************************
TRAP trCentre1
  TPErase;
  TPWrite "Centre 1 termine";
  If (TestDi(PcACentre1) AND TestDi(PcBCentre1)) Stop; ! Probleme de transmission
  If (TestDi(PcACentre1)) Centre1 := PieceA; ! assigne produit de type A
  If (TestDi(PcBCentre1)) Centre1 := PieceB; ! assigne produit de type B
  Isleep intCentre1;
ENDTRAP
```

```
!***********************************************************************************
TRAP trCentre2
  TPErase;
  TPWrite "Centre 2 termine";
  If (TestDi(PcACentre2) AND TestDi(PcBCentre2)) Stop; ! Probleme de transmission
  If (TestDi(PcACentre2)) Centre2 := PieceA; ! assigne produit de type A
  If (TestDi(PcBCentre2)) Centre2 := PieceB; ! assigne produit de type B
  ISleep intCentre1;
ENDTRAP

ENDMODULE
```

9.2. Le code est présenté à l'exemple 9.11.

Exemple 9.11 – Programme pour l'exercice 9.2.

```
MODULE Ex2Solution
  VAR bool drapeau;
  VAR intnum intEchape;
  VAR intnum intFC;
  VAR bool FinDeCycle:=false;

  PERS robtarget robCentrePalette := [[2011.66,430.40,666.56], [0,0,-1,0], [0,0,0,0],
                                     [9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
  PERS robtarget robCourant := [[2011.66,430.40,786.23], [0,0,-1,0], [0,0,0,0],
                               [9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
  VAR robtarget robX1;
  VAR robtarget robX2;
  VAR robtarget robY;
  PERS robtarget robMachine := [[1163.42,-1449.28,1109.53], [0,0,-1,0], [-1,0,-1,0],
                               [9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
  PERS tooldata tLaser := [TRUE,[[-38.1,42.545,222.174], [0.653281,-0.270598,-0.270598,-0.653281]],
                                [1.7,[12.2,0,158],[1,0,0,0],0,0,0]];

!***********************************************************************************
PROC main()
  Init;
  WHILE (NOT FinDeCycle) DO
    Prise;
    Depot;
  ENDWHILE
  MoveL Offs(robCentrePalette,0,0,1500), v1000, fine, toolSuces\wobj:=wobj0;
  Exit; ! Force le PP pour les interruption et les zones atelier
ENDPROC

!***********************************************************************************
PROC Init()
  ! active les interruptions
  CONNECT intFC WITH trFC;
  ISignalDi di_FinCycle,1,intFC;
  CONNECT intEchape WITH trEchape;
  ISignalDi di_LaserTrouvee,0,intEchape;
  ISleep intEchape;

  ! definit le debut de la recherche 1,4 mm en haut de la palette
  robCourant := Offs(robCentrePalette,0,0,1400);
  ! definit l'outil laser a partir de toolSuces
  tLaser:=toolSuces;
  tLaser.tframe :=[[0,-175,80],[1,0,0,0]];
ENDPROC

!***********************************************************************************
PROC Prise()
  VAR num nLigneCode:=0;
  VAR num Hauteur:=0;
  VAR pose Coin;

  MoveJ robCourant, vmax, fine, tLaser;
  ! Trouve la hauteur
  SearchL\Stop, di_LaserTrouvee, robCourant, robCentrePalette, v100, tLaser;
  Hauteur := robCourant.trans.z;
  nLigneCode:=1;
  MoveL Offs(robCourant,-200,-25*25.4,-10), v1000, fine, tLaser;
```

```
  ! recherche le premier point sur l'arrete X
  SearchL\Stop, di_LaserTrouvee, robX1, Offs(robCourant,-200,-15*25.4,-5), v100, tLaser;
  nLigneCode:=2;
  MoveL Offs(robCourant,200,-25*25.4,-10),v1000,fine,tLaser;
  ! recherche le deuxieme point sur la meme arrete
  SearchL\Stop, di_LaserTrouvee, robX2, Offs(robCourant,200,-15*25.4,-5), v100, tLaser;
  nLigneCode:=3;
  MoveL Offs(robCourant,-29*25.4,0,-10),v1000,fine,tLaser;
  ! recherche le troisieme point, sur une autre arrete
  SearchL\Stop, di_LaserTrouvee, robY, Offs(robCourant,-19*25.4,0,-5), v100, tLaser;

  Coin := DefFrame(robX1, robX2, robY \Origin:=3);
  ! sans danger puisque plaque horizontale
  Coin.trans.z := Hauteur;
  robCourant.trans := Coin.trans;
  RobCourant.rot := Coin.rot;
  robCourant := Reltool(robCourant, 24*25.4, 20*25.4,0);

  MoveL Offs(robCourant,0,0,50), v1000, z10, toolSuces;
  MoveL robCourant, v100, fine, toolSuces;

  IWatch intEchape;
  Set do_VacOn;
  MoveL Offs(robCourant,0,0,300), v1000, z100, toolSuces;

ERROR
  If (ERRNO = Err_WhlSearch) and (nLigneCode = 0) then
    MoveL Offs(robCentrePalette,0,0,1500), v1000, fine, toolSuces\wobj:=wobj0;
      Exit;
  ELSE
    RAISE;
  ENDIF
ENDPROC

!***********************************************************************************
PROC Depot()
  MoveJ Offs(robMachine,0,0,300), v5000, z100, toolSuces;
  drapeau := false;
  WaitDI di_SysOk, 1, \MaxTime:=5, \TimeFlag:=drapeau;
  If drapeau then
    TPWrite "Attendez que le systeme est de la place";
    WaitDI di_SysOk, 1;
  ENDIF
  MoveL robMachine, v100, fine, toolSuces;
  ISleep intEchape;
  Reset do_VacOn;
  MoveL Offs(robMachine,0,0,100), v1000, z50, toolSuces;
ENDPROC

!***********************************************************************************
TRAP trFC
  TPErase;
  TPWrite "Demande vue. Le cycle en cours va d'abord terminer.";
  FinDeCycle := true;
  Isleep intFC;
ENDTRAP

!***********************************************************************************
TRAP trEchape
  TPErase;
  TPWrite "Piece echapee...";
  StopMove;
  Exit;
ENDTRAP

ENDMODULE
```

Chapitre 10

Matrice jacobienne et singularités

Lorsque l'opérateur d'un robot spécifie la vitesse cartésienne de l'outil, le contrôleur du robot doit calculer les vitesses articulaires nécessaires. Il est possible d'obtenir la relation entre le vecteur des vitesses de l'outil et le vecteur des vitesses articulaires en différentiant l'équation de la cinématique directe par rapport au temps. Cependant, une telle opération s'avère trop complexe dans le cas des robots à plus de deux articulations rotoïdes. Heureusement, il existe une matrice qu'on appelle jacobienne qui peut convertir le vecteur de vitesses articulaires en vecteur de vitesses de l'outil. Elle peut être calculée de manière symbolique à l'aide de formules simples, que nous allons présenter dans ce chapitre.

Lorsque le rang de la matrice jacobienne devient plus petit que le nombre d'articulations motorisées du robot sériel, on parle d'une singularité. Nous avons déjà présenté la notion d'une configuration singulière dans le chapitre précédant, comme étant une discontinuité dans le problème de la cinématique inverse. Dans ce chapitre, nous allons nous intéresser plutôt par les configurations qui sont proches d'une configuration singulière. Nous allons voir pourquoi la performance du robot se détériore dans la proximité des configurations singulières.

10.1 Matrice jacobienne

L'équation du vecteur des vitesses de l'outil du robot, $\mathbf{t}$, en fonction du vecteur des vitesses articulaires, $\dot{\mathbf{q}}$, a la forme générale suivante :

$$\mathbf{t} = \mathbf{J}\dot{\mathbf{q}}, \tag{10.1}$$

où $\mathbf{J}$ et la matrice Jacobienne. Dans le cas d'un robot sériel à n articulations motorisées, cette équation peut être exprimée comme

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} = \begin{bmatrix} \mathbf{j}_{L,1} & \mathbf{j}_{L,2} & \cdots & \mathbf{j}_{L,n} \\ \mathbf{j}_{A,1} & \mathbf{j}_{A,2} & \cdots & \mathbf{j}_{A,n} \end{bmatrix} \begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \vdots \\ \dot{q}_n \end{bmatrix}, \tag{10.2}$$

où $\dot{x}$, $\dot{y}$ et $\dot{z}$ sont les dérivés par rapport au temps des coordonnées x, y et z de l'origine du référentiel de l'outil par rapport au référentiel de l'atelier, et ω_x, ω_y et ω_z sont les composantes du vecteur de la vitesse angulaire de l'outil. Il est important de noter que les composantes de la vitesse angulaires ne sont pas les dérivés par rapport au temps des angles d'Euler (peut importe la convention).

Les vecteurs $\mathbf{j}_{L,i}$ et $\mathbf{j}_{A,i}$ $(i = 1, 2, \cdots, n)$ sont 3×1. Le vecteur $\mathbf{j}_{L,i}$, multiplié à la vitesse articulaire $\dot{q}_i$, représente la contribution de l'articulation i à la vitesse linéaire de l'outil. De même, le vecteur $\mathbf{j}_{A,i}$, multiplié à la vitesse articulaire $\dot{q}_i$, représente la contribution de l'articulation i à la vitesse angulaire de l'outil. C'est grâce à ces deux observations que nous pourrons trouver des formules pour ces vecteurs. Cependant, nous devons considérer les articulations rotoïdes et prismatiques séparément.

10.1.1 Articulation prismatique

Nous cherchons d'abord la contribution de l'articulation prismatique i à la vitesse angulaire de l'outil, soit $\mathbf{j}_{A,i}\dot{d}_i$, afin de trouver l'expression générique pour $\mathbf{j}_{A,i}$. Si toutes les articulations restent immobiles, à l'exception de l'articulation prismatique $i-1$, l'orientation de l'outil du robot restera la même. En autres mots, la contribution de l'articulation prismatique i à la vitesse angulaire de l'outil est nulle et

$$\mathbf{j}_{A,i} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}. \tag{10.3}$$

De même, si toutes les articulations restent immobiles, sauf l'articulation prismatique i, l'outil du robot se déplacera dans la direction de l'articulation prismatique i, soit le long de l'axe z_{i-1} du référentiel $\mathcal{F}_{i-1}$, et avec la même vitesse, soit $\dot{d}_i$. En autre mois, nous avons

$$\mathbf{j}_{L,i} = \mathbf{e}_{i-1}^{atelier}, \tag{10.4}$$

où $\mathbf{e}_{i-1}^{atelier}$ est le vecteur unitaire le long de l'axe z_{i-1}, exprimé dans le référentiel $\mathcal{F}_{atelier}$.

Ce qui reste maintenant, c'est de trouver une formule pour $\mathbf{e}_{i-1}^{atelier}$. Les composantes de ce vecteur de dimension 3×1 sont les premières trois composantes de la troisième colonne de la matrice homogène $\mathbf{H}_{i-1}^{atelier}$. Il est alors possible d'extraire $\mathbf{e}_{i-1}^{atelier}$ via l'équation suivante :

$$\mathbf{e}_{i-1}^{atelier} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \mathbf{H}_{i-1}^{atelier} \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}. \tag{10.5}$$

10.1.2 Articulation rotoïde

Nous cherchons maintenant la contribution de l'articulation rotoïde i à la vitesse angulaire de l'outil, soit $\mathbf{j}_{A,i}\dot{\theta}_i$, afin de trouver l'expression pour $\mathbf{j}_{A,i}$. Si toutes les articulations restent immobiles, sauf l'articulation rotoïde i, l'outil du robot tournera autour de l'axe de cette articulation, soit l'axe z_{i-1}. Ainsi, le vecteur de la vitesse angulaire de l'outil sera $\mathbf{e}_{i-1}\dot{\theta}_i$ et nous aurons

$$\mathbf{j}_{A,i} = \mathbf{e}_{i-1}^{atelier}, \tag{10.6}$$

où le vecteur $\mathbf{e}_{i-1}^{atelier}$ est donné par l'équation (10.5).

De même, si toutes les articulations du robot restent immobiles, à l'exception de l'articulation rotoïde i, l'origine du référentiel de l'outil suivra le même mouvement circulaire qu'un pendule, dans un plan normal à l'axe z_{i-1}. Ainsi, le vecteur de la vitesse linéaire sera dans le même plan, tangent à la trajectoire circulaire et défini par l'expression $\mathbf{e}_{i-1}^{atelier} \times \mathbf{p}_{i-1,outil}^{atelier}\dot{\theta}_i$, où $\mathbf{p}_{i-1,outil}^{atelier}$ est le vecteur qui relie l'origine du référentiel $\mathcal{F}_{i-1}$ avec l'origine du référentiel $\mathcal{F}_{outil}$. Ainsi, nous avons

$$\mathbf{j}_{L,i} = \mathbf{e}_{i-1}^{atelier} \times \mathbf{p}_{i-1,outil}^{atelier}. \tag{10.7}$$

Il ne nous reste que de trouver une formule pour le vecteur $\mathbf{p}_{i-1,outil}^{atelier}$. Ce vecteur peut être obtenu en soustrayant le vecteur de position de l'origine du référentiel $\mathcal{F}_{i-1}$ du vecteur de position de l'origine du référentiel $\mathcal{F}_{outil}$. Le premier se trouve dans la quatrième colonne de la matrice $\mathbf{H}_{i-1}^{atelier}$, alors que le deuxième est dans la quatrième colonne de la matrice $\mathbf{H}_{outil}^{atelier}$. Ainsi, nous avons l'équation suivante :

$$\mathbf{p}_{i-1,outil}^{atelier} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \left(\mathbf{H}_{outil}^{atelier} - \mathbf{H}_{i-1}^{atelier}\right) \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}. \tag{10.8}$$

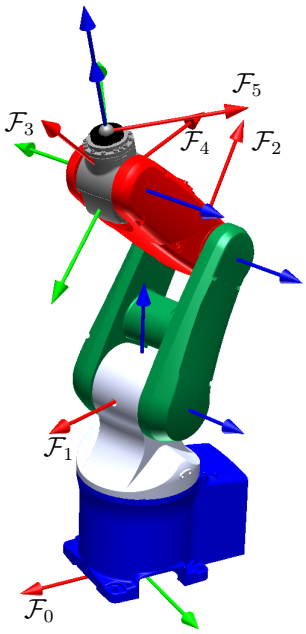


FIGURE 10.1 – Le robot VP-5243 avec ses référentiels DH.

10.2 Matrice jacobienne du robot VP-5243

Nous allons considérer de nouveau le robot VP-5243 de la compagnie Denso, montré à la figure 10.1. Rappelons que pour ce robot nous avons considéré aux chapitres 7 et 8 que $\mathbf{H}_0^{atelier} = \mathbf{H}_{outil}^5 = \mathbf{I}$, où $\mathbf{I}$ est la matrice d'identité 4×4. Nous allons d'abord calculer les vecteurs $\mathbf{e}_{i-1}^{atelier}$ $(i = 1, 2, 3, 4, 5)$, en utilisant les expressions pour $\mathbf{H}_{i-1}^{i-2}$ $(i = 2, 3, 4, 5)$ que nous avons trouvé dans la section 7.2 :

$$\mathbf{e}_0^{atelier} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \mathbf{H}_0^{atelier} \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, \tag{10.9}$$

$$\mathbf{e}_1^{atelier} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \mathbf{H}_0^{atelier} \mathbf{H}_1^0 \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} -s_1 \\ c_1 \\ 0 \end{bmatrix}, \tag{10.10}$$

$$\mathbf{e}_2^{atelier} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \mathbf{H}_0^{atelier} \mathbf{H}_1^0 \mathbf{H}_2^1 \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} -s_1 \\ c_1 \\ 0 \end{bmatrix}, \tag{10.11}$$

$$\mathbf{e}_3^{atelier} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \mathbf{H}_0^{atelier} \mathbf{H}_1^0 \mathbf{H}_2^1 \mathbf{H}_3^2 \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} -s_1 \\ c_1 \\ 0 \end{bmatrix}, \tag{10.12}$$

$$\mathbf{e}_4^{atelier} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \mathbf{H}_0^{atelier} \mathbf{H}_1^0 \mathbf{H}_2^1 \mathbf{H}_3^2 \mathbf{H}_4^3 \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} c_1 s_{234} \\ s_1 s_{234} \\ c_{234} \end{bmatrix}. \tag{10.13}$$

Notez bien que $\mathbf{e}_1^{atelier} = \mathbf{e}_2^{atelier} = \mathbf{e}_3^{atelier}$, puisque les axes z_1, z_2 et z_3 ont toujours la même direction.

Ainsi, si vous remarquez qu'un axe z_i est toujours dans la même direction qu'un autre axe z_j, vous n'avez pas besoin de calculer $\mathbf{e}_i^{atelier}$, mais seulement $\mathbf{e}_j^{atelier}$, puisque $\mathbf{e}_i^{atelier} = \mathbf{e}_j^{atelier}$.

Puisque toutes les cinq articulations du robot VP-5243 sont rotoïdes, nous allons calculer maintenant les vecteurs $\mathbf{p}_{i-1,outil}^{atelier}$ $(i = 1, 2, 3, 4, 5)$:

$$\mathbf{p}_{0,outil}^{atelier} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \left(\mathbf{H}_{outil}^{atelier} - \mathbf{H}_0^{atelier}\right) \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} c_1(70s_{234} + 220s_{23} + 210s_2) \\ s_1(70s_{234} + 220s_{23} + 210s_2) \\ 280 + 70c_{234} + 220c_{23} + 210c_2 \end{bmatrix}, \quad (10.14)$$

$$\mathbf{p}_{1,outil}^{atelier} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \left(\mathbf{H}_{outil}^{atelier} - \mathbf{H}_0^{atelier}\mathbf{H}_1^0\right) \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} c_1(70s_{234} + 220s_{23} + 210s_2) \\ s_1(70s_{234} + 220s_{23} + 210s_2) \\ 70c_{234} + 220c_{23} + 210c_2 \end{bmatrix}, \quad (10.15)$$

$$\mathbf{p}_{2,outil}^{atelier} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \left(\mathbf{H}_{outil}^{atelier} - \mathbf{H}_0^{atelier}\mathbf{H}_1^0\mathbf{H}_2^1\right) \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} c_1(70s_{234} + 220s_{23}) \\ s_1(70s_{234} + 220s_{23}) \\ 70c_{234} + 220c_{23} \end{bmatrix}, \quad (10.16)$$

$$\mathbf{p}_{3,outil}^{atelier} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \left(\mathbf{H}_{outil}^{atelier} - \mathbf{H}_0^{atelier}\mathbf{H}_1^0\mathbf{H}_2^1\mathbf{H}_3^2\right) \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} c_1 70s_{234} \\ s_1 70s_{234} \\ 70c_{234} \end{bmatrix}, \quad (10.17)$$

$$\mathbf{p}_{4,outil}^{atelier} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \left(\mathbf{H}_{outil}^{atelier} - \mathbf{H}_0^{atelier}\mathbf{H}_1^0\mathbf{H}_2^1\mathbf{H}_3^2\mathbf{H}_4^3\right) \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} c_1 70s_{234} \\ s_1 70s_{234} \\ 70c_{234} \end{bmatrix}. \quad (10.18)$$

Notez bien que $\mathbf{p}_{4,outil}^{atelier} = \mathbf{p}_{3,outil}^{atelier}$ puisque les origines des référentiels $\mathcal{F}_4$ et $\mathcal{F}_3$ coïncident toujours.

Enfin, pour obtenir les vecteurs $\mathbf{j}_{L,i} = \mathbf{e}_{i-1}^{atelier} \times \mathbf{p}_{i-1,outil}^{atelier}$ $(i = 1, 2, ..., 5)$, il faut que prendre les produits vectoriels des expressions déjà trouvées pour $\mathbf{e}_{i-1}^{atelier}$ et $\mathbf{p}_{i-1,outil}^{atelier}$. La réponse finale pour la matrice jacobienne du robot VP-5243 est ainsi :

$$\mathbf{J} = \begin{bmatrix} -s_1(70s_{234} + 220s_{23} + 210s_2) & c_1(70c_{234} + 220c_{23} + 210c_2) & c_1(70c_{234} + 220c_{23}) & 70c_1c_{234} & 0 \\ c_1(70s_{234} + 220s_{23} + 210s_2) & s_1(70c_{234} + 220c_{23} + 210c_2) & s_1(70c_{234} + 220c_{23}) & 70s_1c_{234} & 0 \\ 0 & -(70s_{234} + 220s_{23} + 210s_2) & -(70s_{234} + 220s_{23}) & -70s_{234} & 0 \\ 0 & -s_1 & -s_1 & -s_1 & c_1s_{234} \\ 0 & c_1 & c_1 & c_1 & s_1s_{234} \\ 1 & 0 & 0 & 0 & c_{234} \end{bmatrix}. \quad (10.19)$$

Notez bien que $\mathbf{j}_{L,5} = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}^T$ puisque dans notre cas, l'origine du référentiel de l'outil est sur l'axe 5 du robot. Donc, la contribution de l'articulation 5 à la vitesse linéaire de cette origine est nulle.

10.3 Configurations singulières

Le problème des singularités peut survenir uniquement lorsque nous voulons imposer un vecteur de vitesse donnée à l'outil du robot. En autre mots, ce problème peut arriver lorsque nous utilisons les instructions MoveL, MoveJ ou MoveC ou lorsque nous pilotons le robot en mode cartésien. Dans le cas

d'un robot sériel à six articulations motorisée, nous pouvons obtenir les vitesses articulaires à l'aide de l'équation suivante :

$$\dot{\mathbf{q}} = \mathbf{J}^{-1}\mathbf{t}. \tag{10.20}$$

Lorsque la matrice jacobienne est singulière (c'est-à-dire, lorsque $\det(\mathbf{J}) = 0$), nous ne pouvons pas utiliser cette équation pour trouver le vecteur des vitesses articulaires. En fait, dans une singularité, le robot perd sa capacité de déplacer son outil dans certaines directions. On dit souvent que l'outil du robot perd un ou plusieurs dégrés de libertés lorsque le robot est dans une configuration singulière. On peut ainsi penser aux configurations singulières comme la frontière (externe ou interne) de l'espace de travail du robot. Enfin, seuls les robots sériels dont la cinématique inverse peut avoir plus d'une solution ont des singularités.

Dans le cas de robots sériels à moins de six articulations, nous ne pouvons pas inverser la matrice jacobienne. C'est plutôt l'équation suivante qui est utilisée pour trouver le vecteur des vitesses articulaires :

$$\dot{\mathbf{q}} = \left(\mathbf{J}^T\mathbf{J}\right)^{-1}\mathbf{J}^T\mathbf{t}. \tag{10.21}$$

On appelle la matrice $\mathbf{J}^\dagger = \left(\mathbf{J}^T\mathbf{J}\right)^{-1}\mathbf{J}^T$ la pseudo-inverse de la matrice $\mathbf{J}$.

Dans le cas des singularités de robots sériels à $n < 6$ articulations, on ne parle pas d'une matrice jacobienne singulière, mais d'une matrice dont le rang devient plus petit que n. Le phénomène est exactement le même que dans le cas de robots sériels à six articulations. Un robot sériel à $n < 6$ articulations est en configuration singulière lorsque $\det\left(\mathbf{J}^T\mathbf{J}\right) = 0$.

Pour mieux comprendre le problème de singularités, nous allons considérer de nouveau le robot VP-5243. Pour ce robot, il est possible de démontrer que

$$\det\left(\mathbf{J}^T\mathbf{J}\right) = 46200^2 s_3^2\left(s_{234}^2 + (70s_{234} + 220s_{23} + 210s_2)^2\right). \tag{10.22}$$

L'expression à droite est égale à zéro lorsque $s_{234} = 0$ (les axes 1 et 6 du robot sont parallèles) et $70s_{234} + 220s_{23} + 210s_2 = 0$ (l'origine du référentiel $\mathcal{F}_5$ est sur l'axe 1 du robot), ce qui correspond à une singularité d'épaule (figure 8.2a) ou lorsque $\theta_3 = 0°$, ce qui correspond à une singularité de coude (figure 8.2b).

Enfin, nous allons considérer le robot VP-5243 dans une configuration proche d'une singularité de coude. Plus précisément, nous allons considérer la configuration suivante :

$$\mathbf{H}_5^0 = \begin{bmatrix} -1 & 0 & 0 & 5\text{ mm} \\ 0 & -1 & 0 & 0\text{ mm} \\ 0 & 0 & 1 & 220\text{ mm} \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{10.23}$$

$$\mathbf{q} \approx \begin{bmatrix} 0° \\ 101.157° \\ 144.876° \\ -246.033° \\ 0° \end{bmatrix}. \tag{10.24}$$

En autres mots, le centre de la bride du robot est à 5 mm de l'axe 1 du robot.

Dans cette configuration, il est possible de calculer que même si l'on désire que l'outil du robot se déplace à seulement 30 mm/s le long de l'axe y_0, c'est-à-dire $\mathbf{t} = \begin{bmatrix} 0 & 30\text{ mm/s} & 0 & 0 & 0 & 0 \end{bmatrix}^T$, le vecteur des vitesses articulaires sera $\dot{\mathbf{q}} \approx \begin{bmatrix} 343.775°/\text{s} & 0 & 0 & 0 & -343.775°/\text{s} \end{bmatrix}^T$. De telles vitesses articulaires sont bien au delà des capacités du robot (à titre d'exemple, la vitesse maximale de l'axe 1 du robot IRB 120 est $250°/s$.

Considérons maintenant une configuration proche d'une singularité de coude :

$$\mathbf{H}_5^0 \approx \begin{bmatrix} 0 & 0 & 1 & 499.699\text{ mm} \\ 0 & -1 & 0 & 0\text{ mm} \\ 1 & 0 & 0 & 268.486\text{ mm} \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{10.25}$$

$$\mathbf{q} = \begin{bmatrix} 0° \\ 90° \\ 3° \\ -3° \\ 0° \end{bmatrix}. \tag{10.26}$$

Dans cette configuration, il est possible de calculer que même si l'on désire que l'outil du robot se déplace à seulement 30 mm/s le long de l'axe x_0, c'est-à-dire $\mathbf{t} = \begin{bmatrix} 30\text{ mm/s} & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^T$, le vecteur des vitesses articulaires sera $\dot{\mathbf{q}} \approx \begin{bmatrix} 0 & 156.181°\text{/s} & -305.468°\text{/s} & 149.287°\text{/s} & 0 \end{bmatrix}^T$. Encore une fois, de telles vitesses articulaires dépassent les capacités du robot.

En conclusion, nous pouvons généraliser que lorsqu'un robot sériel est proche d'une configuration singulière, sa capacité de déplacer son outil rapidement peut diminuer considérablement. C'est pour cette raison que, lorsque vous utilisez les instructions MoveL, MoveJ ou MoveC, vous devez vous assurer que le robot ne passe pas proche d'une configuration singulière.

De façon plus générale, il faut noter que les performances d'un robot varient énormément d'une configuration à l'autre. Un emplacement optimal d'une série de déplacements à effectuer par l'outil du robot peut améliorer grandement le temps de cycle (temps d'exécution). Malheureusement, les logiciels de simulation comme RobotStudio ne disposent pas de telles fonctionnalités d'emplacement optimal. Afin de résoudre ce problème, vous pouvez utiliser RobotStudio pour tester de façon automatique un grand nombre d'emplacements possibles d'un workobject et, pour chaque emplacement, calculer le temps de cycle à l'aide des instructions ClkReset, ClkStart, ClkStop et ClkRead.

10.4 Exercices

10.1. Trouver l'expression symbolique de la matrice jacobienne, $\mathbf{J}$, du robot de l'exercice 7.1, en considérant le référentiel de l'outil et le référentiel de l'atelier. Donner l'expression symbolique pour $\det\left(\mathbf{J}^T\mathbf{J}\right)$ et commenter les configurations singulières du robot, s'il y a lieu.

10.2. Trouver l'expression symbolique de la matrice jacobienne, $\mathbf{J}$, du robot de l'exercice 7.2, en considérant le référentiel de l'outil et le référentiel de l'atelier. Donner l'expression symbolique pour $\det\left(\mathbf{J}^T\mathbf{J}\right)$ et commenter les configurations singulières du robot, s'il y a lieu.

10.3. Trouver l'expression symbolique de la matrice jacobienne, $\mathbf{J}$, du robot de l'exercice 7.3, en considérant le référentiel de l'outil et le référentiel de l'atelier. Donner l'expression symbolique pour $\det\left(\mathbf{J}^T\mathbf{J}\right)$ et commenter les configurations singulières du robot, s'il y a lieu.

10.4. Trouver l'expression symbolique de la matrice jacobienne de dimensions 3×3, $\mathbf{J}$, qui transforme le vecteur de vitesses articulaires au vecteur de la vitesse linéaire de l'effecteur, pour le robot de l'exercice 7.4. Donner l'expression symbolique pour $\det\left(\mathbf{J}\right)$ et commenter les configurations singulières du robot, s'il y a lieu.

10.5 Réponses aux exercices

10.1. Les expressions nécessaires pour la construction de la matrice jacobienne, ainsi que l'expression pour cette dernière, sont données ci-dessous :

$$\mathbf{e}_0^{atelier} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, \tag{10.27}$$

$$\mathbf{e}_1^{atelier} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, \tag{10.28}$$

$$\mathbf{e}_2^{atelier} = \begin{bmatrix} 0 \\ 0 \\ -1 \end{bmatrix}, \tag{10.29}$$

$$\mathbf{e}_3^{atelier} = \begin{bmatrix} 0 \\ 0 \\ -1 \end{bmatrix}, \tag{10.30}$$

$$\mathbf{p}_{0,outil}^{atelier} = \begin{bmatrix} 260\sin(\theta_1+\theta_2)+300\sin\theta_1 \\ -260\cos(\theta_1+\theta_2)-300\cos\theta_1 \\ 325.5-d_3 \end{bmatrix}, \tag{10.31}$$

$$\mathbf{p}_{1,outil}^{atelier} = \begin{bmatrix} 260\sin(\theta_1+\theta_2) \\ -260\cos(\theta_1+\theta_2) \\ -37.5-d_3 \end{bmatrix}, \tag{10.32}$$

$$\mathbf{p}_{3,outil}^{atelier} = \begin{bmatrix} 0 \\ 0 \\ -37.5-d_3 \end{bmatrix}, \tag{10.33}$$

$$\mathbf{J} = \begin{bmatrix} 260\cos(\theta_1+\theta_2)+300\cos\theta_1 & 260\cos(\theta_1+\theta_2) & 0 & 0 \\ 260\sin(\theta_1+\theta_2)+300\sin\theta_1 & 260\sin(\theta_1+\theta_2) & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & -1 \end{bmatrix}. \tag{10.34}$$

L'expression pour la déterminante de $\mathbf{J}^T\mathbf{J}$ est

$$\det\left(\mathbf{J}^T\mathbf{J}\right) = 6084000000\sin^2\theta_2, \tag{10.35}$$

ce qui veut dire que le robot est en configuration singulière si, est seulement si, $\sin\theta_2 = 0$. En pratique, cette condition est remplie uniquement lorsque le bras est complètement étendu.

10.2. Les expressions nécessaires pour la construction de la matrice jacobienne, ainsi que l'expression pour cette dernière, sont données ci-dessous :

$$\mathbf{e}_0^{atelier} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, \tag{10.36}$$

$$\mathbf{e}_1^{atelier} = \begin{bmatrix} -\cos\theta_1 \\ -\sin\theta_1 \\ 0 \end{bmatrix}, \tag{10.37}$$

$$\mathbf{e}_2^{atelier} = \begin{bmatrix} \sin\theta_1 \sin\theta_2 \\ -\cos\theta_1 \sin\theta_2 \\ -\cos\theta_2 \end{bmatrix}, \tag{10.38}$$

$$\mathbf{p}_{0,outil}^{atelier} = \begin{bmatrix} \sin\theta_1 \sin\theta_2(90+d_3) - 553\cos\theta_1 \\ -\cos\theta_1 \sin\theta_2(90+d_3) - 553\sin\theta_1 \\ 550 - \cos\theta_2(90+d_3) \end{bmatrix}, \tag{10.39}$$

$$\mathbf{p}_{1,outil}^{atelier} = \begin{bmatrix} \sin\theta_1 \sin\theta_2(90+d_3) - 553\cos\theta_1 \\ -\cos\theta_1 \sin\theta_2(90+d_3) - 553\sin\theta_1 \\ -\cos\theta_2(90+d_3) \end{bmatrix}, \tag{10.40}$$

$$\mathbf{J} = \begin{bmatrix} \cos\theta_1 \sin\theta_2(90+d_3) + 553\sin\theta_1 & \sin\theta_1\cos\theta_2(90+d_3) & \sin\theta_1\sin\theta_2 \\ \sin\theta_1 \sin\theta_2(90+d_3) - 553\cos\theta_1 & -\cos\theta_1\cos\theta_2(90+d_3) & -\cos\theta_1\sin\theta_2 \\ 0 & \sin\theta_2(90+d_3) & -\cos\theta_2 \\ 0 & -\cos\theta_1 & 0 \\ 0 & -\sin\theta_1 & 0 \\ 1 & 0 & 0 \end{bmatrix}. \tag{10.41}$$

L'expression pour la déterminante de $\mathbf{J}^T\mathbf{J}$ est

$$\det\left(\mathbf{J}^T\mathbf{J}\right) = \sin^2\theta_2(d_3+90)^4 + (1+\sin^2\theta_2)(d_3+90)^2 + 1 + 305809\cos^2\theta_2. \tag{10.42}$$

Évidement, cette expression est toujours positive, son minimum étant 1, ce qui veut dire que ce robot n'a pas de singularités.

10.3. Les expressions nécessaires pour la construction de la matrice jacobienne, ainsi que l'expression pour cette dernière, sont données ci-dessous :

$$\mathbf{e}_0^{atelier} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, \tag{10.43}$$

$$\mathbf{e}_2^{atelier} = \begin{bmatrix} \cos\theta_1 \\ \sin\theta_1 \\ 0 \end{bmatrix}, \tag{10.44}$$

$$\mathbf{e}_3^{atelier} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, \tag{10.45}$$

$$\mathbf{p}_{0,outil}^{atelier} = \begin{bmatrix} 148\sin\theta_1 + (d_3 + 279.5)\cos\theta_1 + 133\cos(\theta_1 + \theta_4) \\ -148\cos\theta_1 + (d_3 + 279.5)\sin\theta_1 + 133\sin(\theta_1 + \theta_4) \\ d_2 + 405 \end{bmatrix}, \tag{10.46}$$

$$\mathbf{p}_{3,outil}^{atelier} = \begin{bmatrix} 133\cos(\theta_1 + \theta_4) \\ 133\sin(\theta_1 + \theta_4) \\ 0 \end{bmatrix}, \tag{10.47}$$

$$\mathbf{J} = \begin{bmatrix} 148\cos\theta_1 - (d_3 + 279.5)\sin\theta_1 - 133\sin(\theta_1 + \theta_4) & 0 & \cos\theta_1 & -133\sin(\theta_1 + \theta_4) \\ 148\sin\theta_1 + (d_3 + 279.5)\cos\theta_1 + 133\cos(\theta_1 + \theta_4) & 0 & \sin\theta_1 & 133\cos(\theta_1 + \theta_4) \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 \end{bmatrix}. \tag{10.48}$$

L'expression pour la déterminante de $\mathbf{J}^T\mathbf{J}$ est

$$\det\left(\mathbf{J}^T\mathbf{J}\right) = (d_3 + 279.5)^2, \tag{10.49}$$

ce qui veut dire que le robot est en singularité lorsque $d_3 = -279.5$ (c'est-à-dire, lorsque la distance entre les axes z_3 and z_0 est le minimum, soit 148).

10.4. Les expressions nécessaires pour la construction de la matrice jacobienne, ainsi que l'expression pour cette dernière, sont données ci-dessous :

$$\mathbf{e}_0^{atelier} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, \tag{10.50}$$

$$\mathbf{e}_2^{atelier} = \begin{bmatrix} -s_1 \\ c_1 \\ 0 \end{bmatrix}, \tag{10.51}$$

$$\mathbf{e}_3^{atelier} = \begin{bmatrix} -s_1 \\ c_1 \\ 0 \end{bmatrix}, \tag{10.52}$$

$$\mathbf{p}_{0,outil}^{atelier} = \begin{bmatrix} c_1(1260c_{23} + 1000c_2) \\ s_1(1260c_{23} + 1000c_2) \\ 560 - 1260s_{23} - 1000s_2 \end{bmatrix}, \tag{10.53}$$

$$\mathbf{p}_{1,outil}^{atelier} = \begin{bmatrix} c_1(1260c_{23} + 1000c_2) \\ s_1(1260c_{23} + 1000c_2) \\ -1260s_{23} - 1000s_2 \end{bmatrix}, \tag{10.54}$$

$$\mathbf{p}_{2,outil}^{atelier} = \begin{bmatrix} 1260c_1c_{23} \\ 1260s_1c_{23} \\ -1260s_{23} \end{bmatrix}, \tag{10.55}$$

$$\mathbf{J} = \begin{bmatrix} -s_1(1260c_{23} + 1000c_2) & -c_1(1260s_{23} + 1000s_2) & -1260c_1s_{23} \\ c_1(1260c_{23} + 1000c_2) & -s_1(1260s_{23} + 1000s_2) & -1260s_1s_{23} \\ 0 & -1260c_{23} - 1000c_2 & -1260c_{23} \end{bmatrix}. \tag{10.56}$$

où $s_1 = \sin\theta_1$, $c_1 = \cos\theta_1$, $s_2 = \sin\theta_2$, $c_2 = \cos\theta_2$, $s_{23} = \sin(\theta_2 + \theta_3)$ et $c_{23} = \cos(\theta_2 + \theta_3)$.
L'expression pour la déterminante de la matrice $\mathbf{J}$ est

$$\det(\mathbf{J}) = 1260000s_3(1260c_{23} + 1000c_2), \tag{10.57}$$

ce qui veut dire que le robot est en singularité lorsque $s_3 = 0$ (c'est-à-dire, en pratique, lorsque le bras du robot est complètement étendu) ou lorsque $1260c_{23} + 1000c_2 = 0$ (c'est-à-dire, lorsque le point de l'effecteur se trouve sur l'axe 1 du robot).

Bibliographie

[1] ISO, Robots et composants robotiques – Vocabulaire, ISO 8373, 2012.

[2] Pollard Jr., W.L.G., « Spray painting machine », brevet américain No. 2 213 108, déposé le 29 octobre 1934, accepté le 27 août 1940.

[3] Roselund, H.A., « Means for moving spray guns or other devices through predetermined paths », brevet américain No. 2 344 108, déposé le 17 août 1939, accepté le 14 mars 1944.

[4] Pollard, W.L.V., « Position controlling apparatus », brevet américain No. 2 286 571, déposé le 22 avril 1935, accepté le 16 juin 1942.

[5] Devol Jr., G.C., « Programmed article transfer », brevet américain No. 2 988 237, déposé le 10 décembre 1954, accepté le 13 juin 1961.

[6] Rosheim, M.E., Robot Evolution – The Development of Anthrobotics, John Wiley & Sons, 1994.

[7] Westerlund, L., The Extended Arm of Man – A History of the Industrial Robot, Informationsförlaget, 2000.

[8] Makino, H., Kato, A., et Yamazaki, Y., « Research and commercialization of SCARA robot », International Journal of the Robotics Society of Japan, Vol. 23, No. 5, pages 61-62, 2007.

[9] IFR, World of robotics 2013.

[10] ABB, Caractéristiques du produit IRB 1600, 2010.

[11] Slamani, M., Nubiola, A., et Bonev, I.A., « Assessment of the positioning performance of an industrial robot », Industrial Robot, Vol. 39, No. 1, pages 57–68, 2012.

[12] Nubiola, A., et Bonev, I.A., « Absolute calibration of an ABB IRB 1600 robot using a laser tracker », Robotics and Computer-Integrated Manufacturing, Vol. 29, No. 1, pages 236–245, 2013.

[13] ABB, IRC5 avec FlexPendant, 2010.

[14] CSA, Robots industriels et systèmes robotiques : exigences générales de sécurité, Z434-03, 2004.

[15] Denavit, J., et Hartenberg, R.S., « A kinematic notation for lower pair mechanisms based on matrices », ASME Journal of Applied Mechanics, Vol. 6, pp. 215–221, 1955.

[16] Lipkin, H., « A note on Denavit-Hartenberg notation in robotics », Proceedings of IDETC/CIE 2005 ASME 2005 International Design Engineering Technical Conferences, Long Beach, Californie, États-Unis, 24–28 septembre 2005.

[17] Craig, J.J., Introduction to Robotics : Mechanics and Control, $3^{ème}$ édition, Pearson Education, 2005.