الجمهورية الجزائرية الديمقراطية الشعبية

République Algérienne Démocratique Et Populaire

وزارة التعـــليـم العالـــــــــي
والبحـــــث العلمـــــــي

Ministère De L'enseignement Supérieur

Et De La Recherche Scientifique

Université d'Oran 2

Institut de Maintenance et de Sécurité Industrielle

Département de maintenance en instrumentation

MEMOIRE

Pour l'obtention du diplôme de Master

Spécialité : Génie Industriel

Thème

Simulation de coopération de robots manipulateurs dans une chaine robotisée

Présenté et soutenu publiquement par :

Nom : EL RALI **Prénom : Mohamed Amine Abdelouahab**

Devant le jury composé de :

Nom et Prénom	Grade	Etablissement	Qualité
…….	…….	…….	Président
KACIMI Abderrahmane	M C B	…….	Encadreur
……	……..	……..	Examinateur

2021/2022

Sommaire

CHAPITRE III : Programmation graphique des taches avec Robotstudio

CHAPITRE IV : Simulation (Robot Studio)

Conclusion générale

Annexe

Remerciements

Je remercie tout d'abord Dieu tout puissant de m'avoir donné le courage, la force et la patience d'achever ce modeste travail.

Je souhaite ainsi adresser mes remerciements à mon directeur de mémoire ***M. KACIMI Abderrahmane***, à l'université d'Oran, pour sa patience, sa disponibilité et surtout ses judicieux conseils, qui ont contribué à alimenter ma réflexion.

Je tiens à remercier ***Ma Chère Mère AMRANI Noria***, pour son soutien constant et son encouragement durant toute ma période de travail

Je tiens à remercier toutes les personnes qui ont contribué au succès de mon stage et qui m'ont aidée lors de la rédaction de ce mémoire.

Je remercie également toute l'équipe pédagogique de l'université d'Oran et l'intervenant professionnel ***Mlle Dr. HEDIA Nacera***, pour avoir assuré la partie théorique de celle-ci.

EL RALI Mohamed Amine Abdelouaheb

Dédicace

Avec un énorme plaisir, un cœur ouvert et une immense joie, que je dédie mon travail à :

Mon très cher Grand père **AMRANI Ramadan** que dieu le bénisse dans son vaste paradis.

Ma très chère mère **AMRANI Noria**

En témoignage et en gratitude à leur aide morale et matérielle, leur soutien pendant les bons

Et les mauvais moments de ma vie et durant toute ma carrière d'étude.

Notre encadreur **Mr. KACIMI Abderrahmane**,

Ma chère amie **HEDIA Nacera**

Qui m'a tenu compagne et qui m'a entouré de sa sympathie

Et de sa sollicitude.

A tous ceux qui me connaissent de près ou de loin.

Merci.

EL RALI Mohamed Amine Abdelouaheb

Nomenclature

Symbole	Désignation
P	Un point de l'extrémité.
R_0	Un repère lié au bâti.
C_n	Corps.
j	L'articulation.
R_j	Le repère.
Z_j	L'axe porté par l'axe de l'articulation j.
X_j	L'axe porté par la perpendiculaire.
$\propto_j$	L'angle entre les axes Zj-1 et Zj correspondant à une rotation autour de Xj-1.
d_j	La distance entre Z_{j-1} et Z_j le long de X_{j-1}.
θ_j	L'angle entre l'axe x_{j-1} et x et x_j correspondant à une rotation autour de z_j.
r_j	La distance entre x_{j-1} et x et x_j le long de z_j.
q_j	La variable articulaire.
$^{j-1}T_j$	La matrice de transformation.
^0T_n	La matrice de passage.
X	Les coordonnées opérationnelles.
q	Le vecteur des variables articulaires.
$\dot{q}$	Les vitesses articulaires.

J(q)	La matrice jacobéenne.
dq	La différentielle articulaire.
$^{n}P_{E}$	L'origine du repère RE.
R_{E}	Le Repère.
N	Le vecteur unitaire.

Glossaire des acronymes

DDL	Degrés de liberté
AMF	American Machine and Foundry
SCARA	Sélective Compliance Assembly Robot Arm
MGI	Le modèle géométrique inverse
MGD	Le modèle géométrique direct
ARTE	A Robotic Toolbox for Education
GUI	Graphical User Interface
PEL	La programmation en ligne
PHL	La programmation hors-ligne
CFAO	Conception et Fabrication Assistée par Ordinateur
VBA	Visual Basic for Application

Liste des Figures

Liste des Figures

Liste des Figures

Liste des Tableaux

Résumé

Le travail présenté dans ce manuscrit porte sur la planification des tâches pour un robot industriel type ABB avec simulation par le logiciel de programmation graphique d'ABB "Robot Studio".
Notre travail consiste à simuler une coopération de robots manipulateurs dans une cellule industrielle robotisée, en utilisant le logiciel Robot Studio d'ABB.
L'objectif de ce projet de fin d'étude est de planifier, programmer et simuler des tâches industrielles parmi les plus utilisées, d'un robot manipulateur d'ABB.

Mots-clés : Robot Studio, cellule, ABB, synchronisation, programmation, industrie, articulation, apprentissage, commande.

Introduction Générale

Introduction Générale

Le concept de robot date de plusieurs siècles, mais le terme robot fut inventé par le tchèque Karel Capek dans une pièce de théâtre écrite en 1920 : "RUR ou les robots universels de Rossum".

Ce terme est dérivé du verbe tchèque "robota" signifiant travail forcé ou corvée. Il est certain que depuis fort longtemps, les réalisateurs d'automates ont cherché à pouvoir insuffler à leurs machines des comportements adaptés aux circonstances.

Malheureusement, jusqu'au vingtième siècle, les techniques étaient trop primitives pour permettre de telles réalisations. Il fallut attendre 1959 pour que Georges Devol invente une machine originale, polyvalente et reprogrammable, ce qui a permis au robot d'acquérir une réalité industrielle. Mais en fait ce ne fut que vers la fin des années 1970 que les robots industriels de première génération ont vu le jour.

Généralement, un robot manipulateur est considéré comme un système articulérigide c'est pour qu'il est développé dans le cadre d'application industrielle ; Cedernier a prouvé leur importance puisqu'il substitue efficacement l'homme dansla réalisation des tâches tel que la soudure dans les usines automobiles ou lapalettisation, assemblage…

La modélisation et la simulation des systèmes robotiques à l'aide de divers logiciels de programmation facilite le processus de conception, de construction etinspectant les robots dans le monde réel. Une simulation est importante pour lesprogrammeurs de robot dans leur permettant d'évaluer et de prédire lecomportement d'un robot, et en outre de vérifier et d'optimiser la planification detrajectoire de leur processus. En outre, cela permettra d'économiser le temps etl'argent, et jouer un rôle important dans l'évaluation de la fabrication D'automatisation. Être capable de simuler ouvre une large gamme d'options, enaidant à résoudre de nombreux problèmes créatifs. On peut étudier, concevoir,visualiser et tester un objet avant de faire une réalité.

Le travail réalisé et présenté dans ce mémoire s'articule de la façon suivante :

Le premier chapitre l'état de l'art de la programmation des robots industriels.
Ensuite dans le deuxième chapitre on a essayé de citer les Outils mathématiques pour la planification des taches industrielles.

Dans le troisième chapitre on s'est focalisé sur la programmation graphique des taches avec le logiciel robot Studio.

Enfin dans le quatrième et le dernier chapitre On a réussi à créer un programme simulant la tâche de palettisation pour déplacer des boites sur la palette dans un convoyeur avec une rentabilité, précision et rapidité remarquable assuré par le logiciel ABB Robot-Studio.

Chapitre I

Introduction

Un robot est un système mécanique poly-articulé mû par des actionneurs et commandé par un calculateur qui est destiné à effectuer une grande variété de tâches.

Selon la norme internationale ISO 8373, un robot industriel (figure I.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].

Figure (I.1) : Robots à six et cinq ddl : (a) robot sériel à six articulations et (b) robot sériel à cinq articulations.

I.1. Définitions générales

La robotique est une science pluridisciplinaire qui comprend la mécanique, l'automatique l'électrotechnique, le traitement de signal, l'informatique, la communication ... etc. Un robot se compose de :

1.1.1. Le mécanisme

Structure plus au moins proche de celle du bras humain, on dit aussi manipulateur quand il ne s'agit pas d'un robot mobile.

Sa motorisation est réalisée par des actionneurs électriques, pneumatiques ou hydrauliques qui transmettent leur mouvement aux articulations par des systèmes appropriés.

1.1.2. La perception

Permet de gérer les relations entre le robot et son environnement. Les organes de perception sont des capteurs dits « proprioceptifs » lorsqu'ils mesurent l'état interne du robot (position et vitesses des articulations) ou « extéroceptifs » lorsqu'ils recueillent des informations sur l'environnement (détection de présence, mesure de distance, vision artificielle).

1.1.3. La commande

Qui synthétise les consignes des asservissements pilotant les actionneurs. A partir de la fonction de perception et des ordres de l'utilisateur, elle permet d'engendrer les actions du robot.

1.1.4. L'interface homme-machine

A travers laquelle l'utilisateur programme les taches que le robot doit exécuter.

1.1.5. Le poste de travail et les dispositifs per robotique

Qui constituent l'environnement dans lequel évolue le robot. [2]

I.2. Le vocabulaire de la robotique [2]

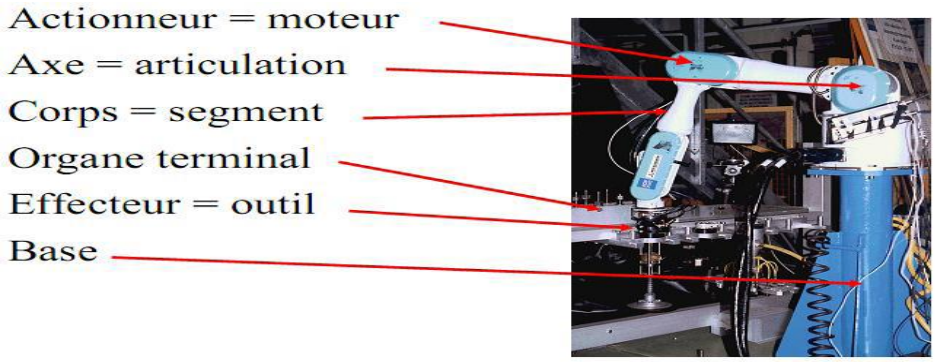


Figure (I.2) : Vocabulaire de la robotique.

I.3. La caractérisation des robots

I.3.1. Description de sa géométrie

Robot = système mécanique poly-articulé :

1.3.1.1. L'articulation prismatique

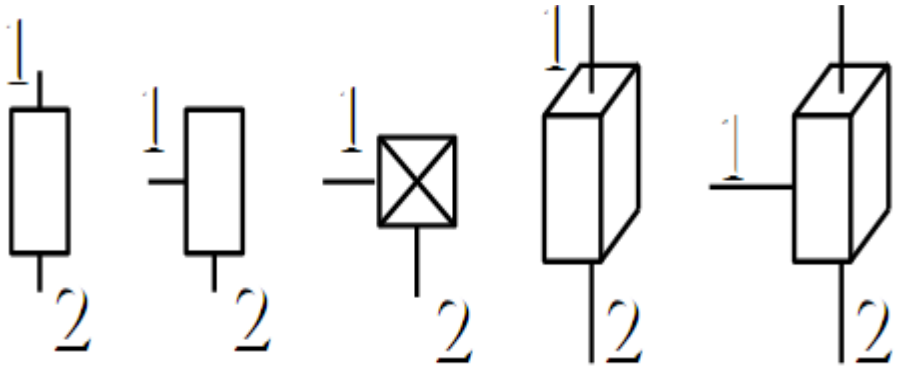


Figure (I.3) : Articulation prismatique.

1.3.1.2. L'articulation rotoïde

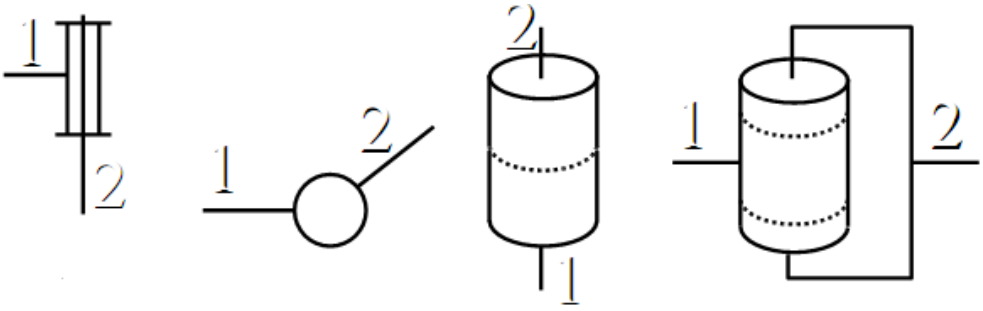


Figure (I.4) : Articulation rotoïde.

I.3.2. Les caractéristiques géométriques

• Nombre d'axes (propulsai par un actionneur).
• Architecture (série ou parallèle).
• Chaînage des articulations.
• Nombre de degrés de liberté.[2]

Exemples

– 3 axes, série, RRR, 3DL.

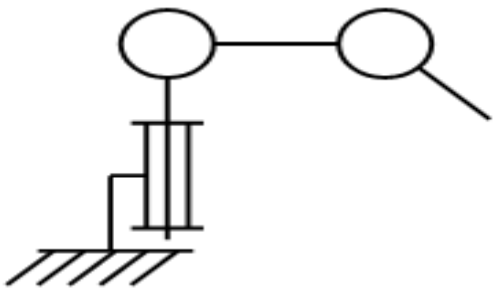


Figure (I.5) : Robot avec 3 axes, série, RRR et 3DL.

-3 axes, série, PPP, 3DL.

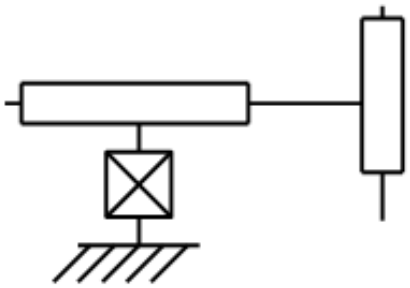


Figure (I.6) : Robot avec 3 axes, série, PPP, 3DL.

– 4 axes, parallèle, RP+RP, 3DL.

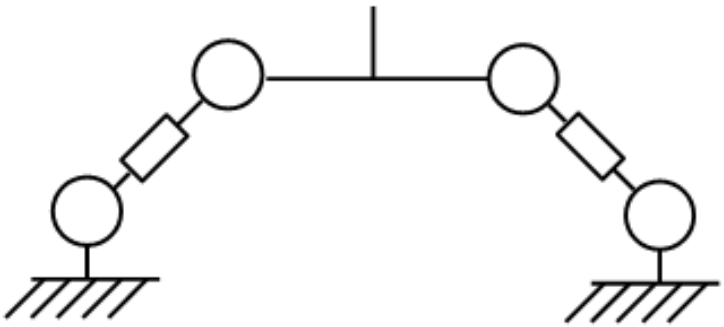


Figure (I.7) : Robot avec 4 axes, parallèle, RP+RP, 3DL.

I.3.3. L'architecture des robots

Un robot comporte 2 parties essentielles :

1.3.3.1. Le porteur

Structure mécanique articulée constituée des 3 premiers degrés de liberté à partir du bâti. Si P est un point de l'extrémité et R_0 un repère lié au bâti, le rôle du porteur est de fixer la position de P dans R_0.

1.3.3.2. Le poignet

Il est destiné à l'orientation de la pince ou de l'outil porté par le robot. La façon dont les liaisons motorisées sont reparties du bâti au poignet défini trois grandes classes d'architecture :

- **Architecture série** (ou chaine cinématique ouverte)
- **Architecture parallèle** (ou chaine cinématique multi-boucle)
- **Architecture mixte** (série-parallèle ou parallèle-série). [2]

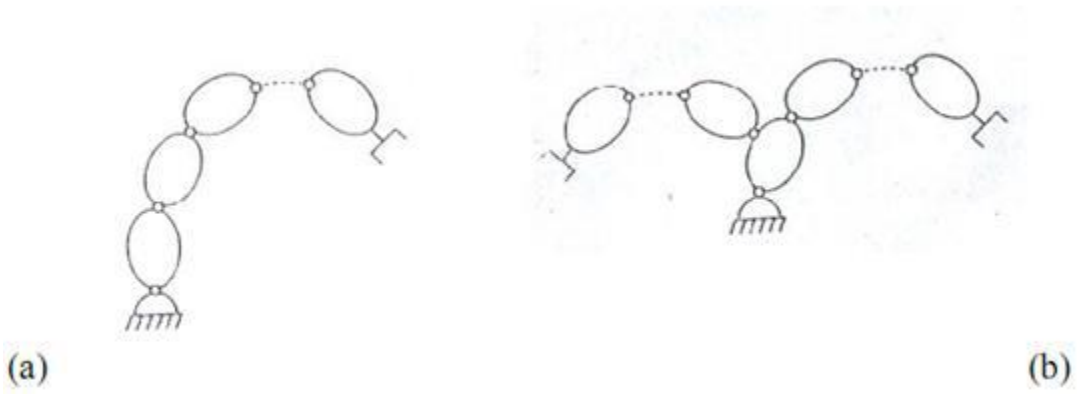


Figure (I.8) : Structure ouverte simple (a), structure arborescente (b).

I.4. La description 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.
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 (I.1) (a).
La grande majorité des robots industriels sont de type sériel. [3]

I.4.1. Le robot sériel

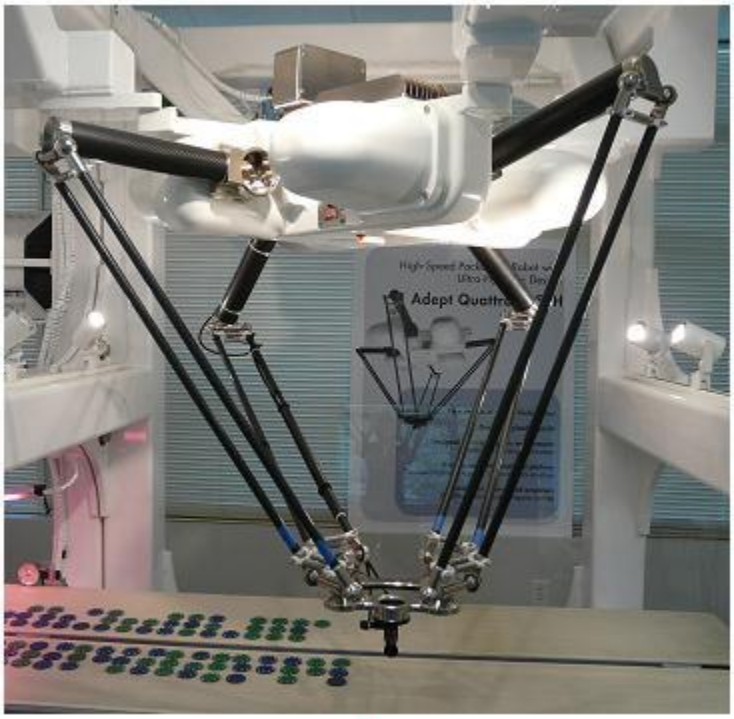
Un robot sériel est composé d'une série de segments reliés par des articulations motorisées rotoïde (en rotation) ou prismatiques (en translation). Dans certaines applications, on n'a pas besoin de déplacer les objets selon six ddl mais seulement selon cinq ou même quatre ou trois ddl. La figure (I.1) (b) illustre un robot sériel à cinq ddl utilisés pour la palettisation.
Le robot de la figure (1.9) est lui aussi utilisé pour la palettisation, mais il a seulement quatre ddl. [3]

(a)

Figure (I.9) : Robot sériel pour palettisation à quatre ddl.

I.4.2. Le robot parallèle

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.[3]



(b)

Figure (I.10) : Robot parallèle.

I.4.3. Le robot cartésien et le robot SCARA

Le robot illustré à la figure (I.11) (c) est un robot cartésien alors que le robot de la figure (I.11) (d) est un robot SCARA. Ces robots sont tous à quatre ddl (trois translations et une rotation autour d'un axe vertical) et servent à déplacer rapidement des petits objets.[3]

(c) (d)

Figure (I.11) : (c) Robot cartésien, (d) Robot SCARA.

I.4.4. Les robots redondants à sept ddl

Enfin, il existe aussi des robots à sept ddl (c.à.d. avec sept articulations motorisées) comme le robot illustré à la figure (I.12) (a), mais ils sont rarement utilisés. L'avantage d'un tel robot redondant est l'existence d'une infinité de possibilités pour atteindre une pose désirée, ce qui permet au robot de contourner des obstacles. Beaucoup plus souvent, on monte un robot a six articulations sur un guide linéaire la figure (I.12) (b) 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).[3]

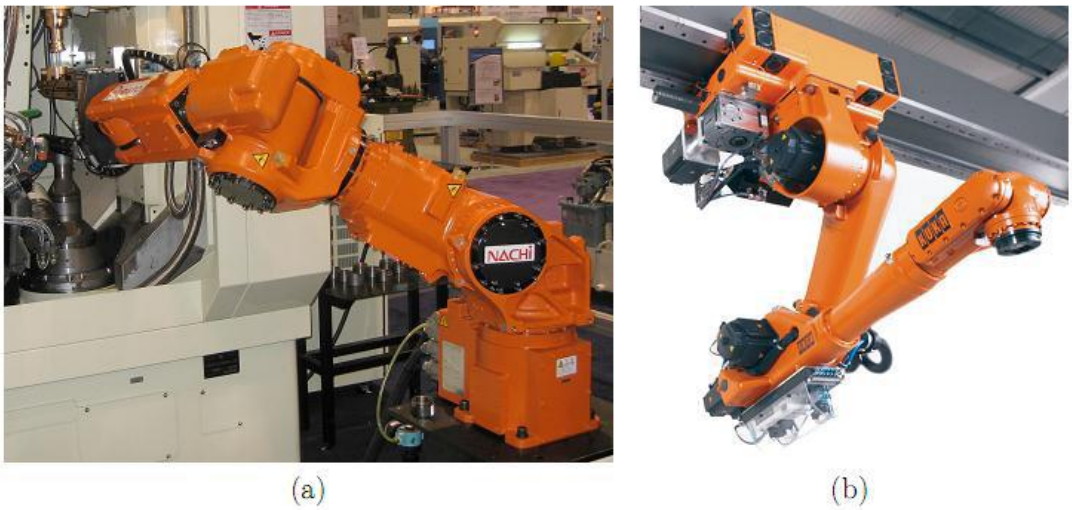


Figure (I.12) : Robots redondants.

I.5. Historique des robots industriels

Un des premiers robots industriels à être conçu par l'américain Willard L. G. Pollard Jr. qui a fait une demande de brevet pour son invention en 1934 [4]. Il s'agit d'un robot parallèle à deux ddl destinés à l'application de peinture sur la carrosserie d'une automobile. Une licence de ce brevet à 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 Mr Harold Roselund. Ce robot n'était pas le robot de Pollard, mais un robot sériel inventé par Mr. Roselund lui-même [5].

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.

En 1956 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. C'est en 1961 qu' Unimation développe leur premier prototype, l'Unimate, et le vend (à forte perte) à General Motors figure (I.13) (a). Ce robot a été utilisé pour assister une machine à couler par injection.
En 1969, Unimation installe 26 robots soudeurs sur une ligne d'assemblage de Chevrolet Vega, chez General Motors figure (I.13) (b).

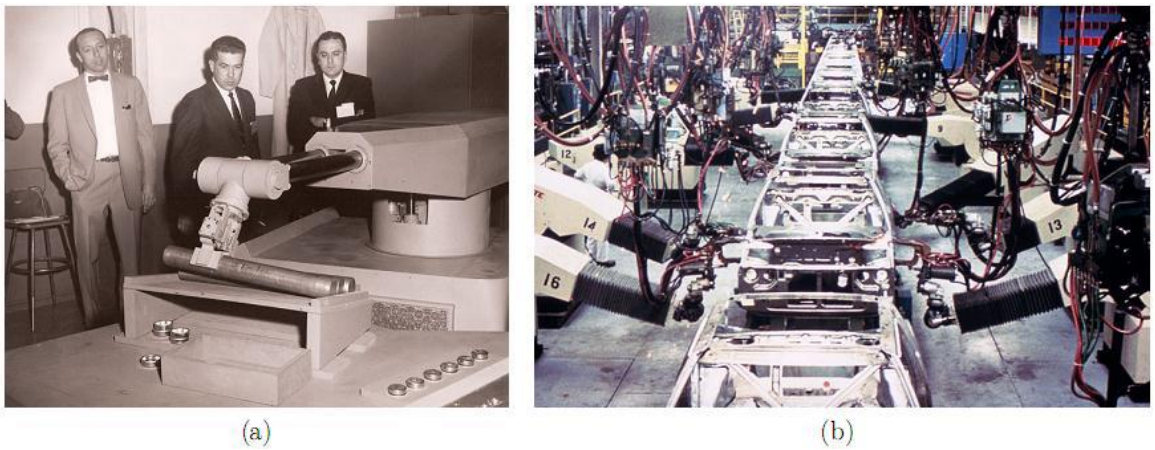


Figure (I.13) : Les premiers robots industriels

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. Unimation achète le Vicarm en 1977, l'améliore, et le renomme PUMA pour Programmable Universal Machine for Assembly figure (I.14). Le premier robot PUMA est installé en 1979, dans une usine de General Motors. Unimation finit par être vendu à l'entreprise suisse Staubli en 1989.

Figure (I.14) : Les trois premiers modèles du robot PUMA commercialisés par Unimation.

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 son propre robot à six ddl en 1973, l'IRB 6, le premier robot à six ddl avec un parallélogramme figure (I.15) (a).
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.

En 1980, 19 000 robots industriels ont 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, 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 (I.15c).

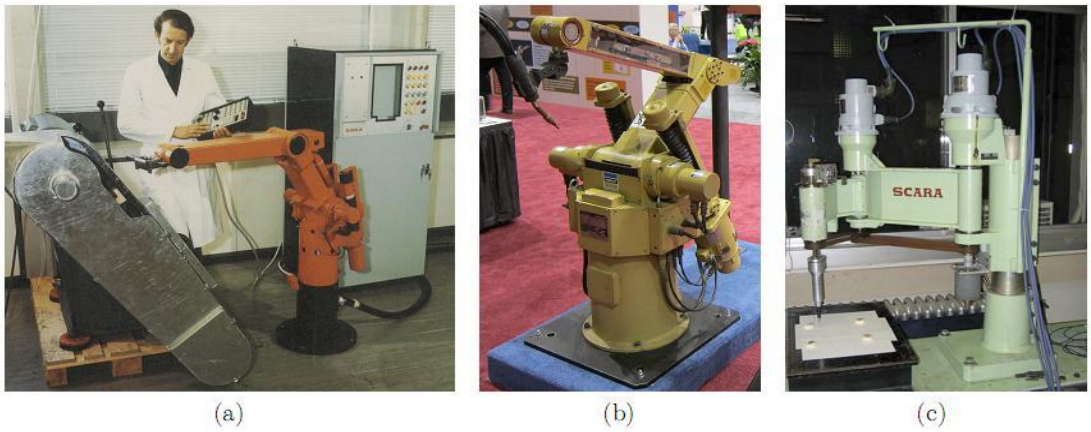


Figure (I.15) : Les premiers robots de la nouvelle ère de la robotique industrielle.

En 2012 [9] quelque 159 346 robots industriels ont été installés à travers le monde. 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 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, bien évidemment, les robots d'aujourd'hui sont nettement plus performants. Les avances en vision robotique sont aussi spectaculaires. Enfin, de plus en plus la programmation des robots se fait à l'aide de logiciels 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.[3]

I.6. Domaine d'application

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

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 .[3]

Figure (I.16) : Robotique industrielle dans les hôpitaux.

I.7. Le robot industriel IRB 1600 de la compagnie ABB

Le robot industriel que nous allons utiliser est le modèle IRB 1600 du fabricant ABB. Il est conçu spécifiquement pour les industries manufacturières qui utilisent des installations robotisées flexibles. Le robot a une structure ouverte spécialement adaptée à une utilisation souple. Le système est composé d'un manipulateur sériel à six ddl et d'un contrôleur IRC5.[3]

I.7.1. Le manipulateur

Le modèle 3D du manipulateur est montré à la figure (I.17), il s'agit d'un manipulateur à six articulations rotoïdes actionnées par des moteurs AC. 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.[3]

Figure (I.17) : Modèle 3D du robot IRB 1600.

I.7.2. L'espace de travail et de 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 rotoides 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 (I.1):[3]

Articulation	**Plage (par défaut)**	**Vitesse**
1	+ ou - 180 °	150°/S
2	-90° à 150°	160°/S
3	-245° à 65 °	170°/S
4	+ou - 200°	320°/S
5	+ou - 115°	400°/S
6	+ ou - 400 °	460°/S

Tableau (I.1) : Capacité des articulations du robot IRB1600.

La figure (I.18) (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 a lui une entité géométrique à six dimensions fort complexes 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 (I.18) (b). [3]

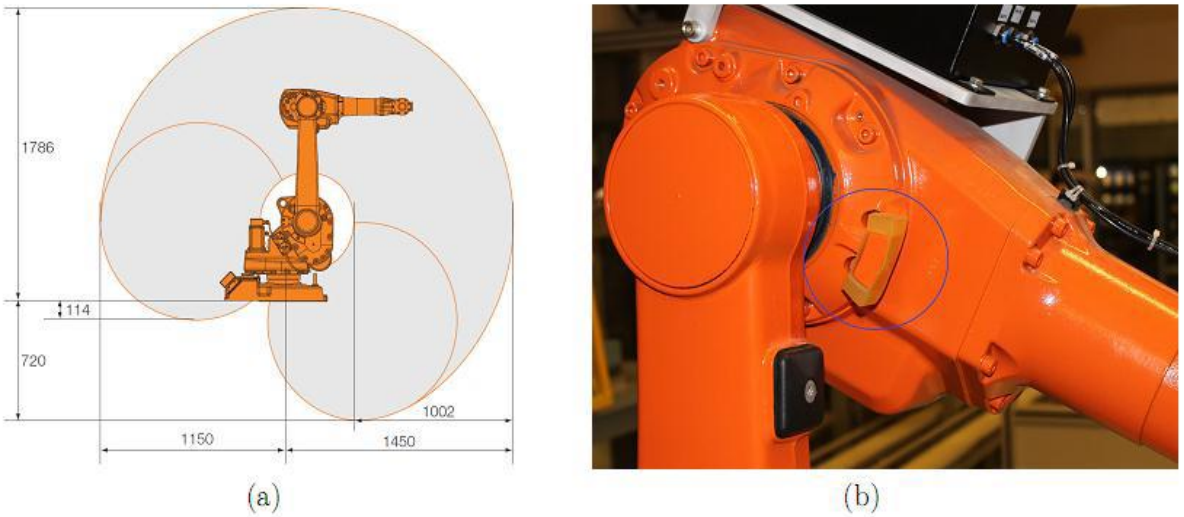


Figure (I.18) : (a) Section verticale de l'enveloppe de travail du robot IRB 1600 et (b) une butée mécanique sur ce même robot.

I.7.3. Description des composants principaux

I.7.3.1. La bride de montage

Par définition, un robot industriel est une machine multifonction 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.[3]
La figure (I.19) a montré un autre exemple de changeur d'outil monté sur le robot IRB 1600.

I.7.3.2. 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 (I.19) (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 une conduite flexible répondant à tous les besoins de mouvement sans toutefois risquer de coincer et/ou se faire arracher durant un mouvement.[3]

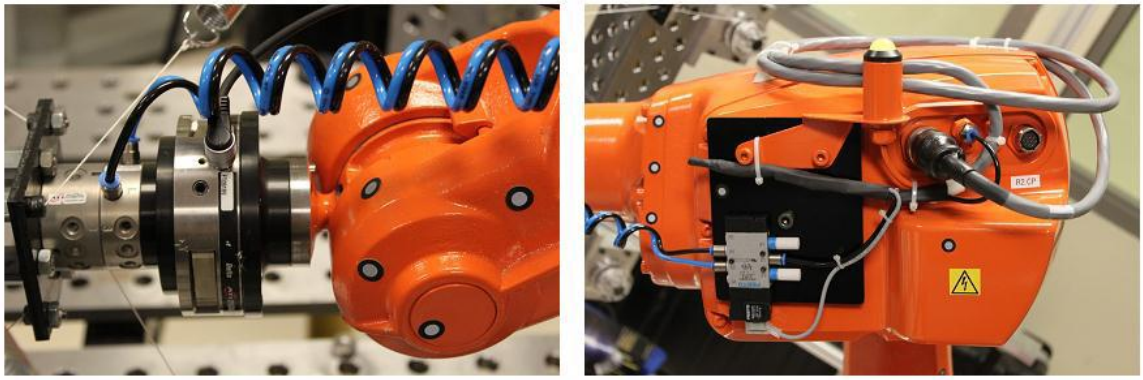


Figure (I.19) : L'effecteur du robot IRB 1600.

I.7.3.3. 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. [3]

I.7.3.4. Les freins électriques

Pour des raisons de sécurité, il est strictement interdit d'actionner les boutons de relâchement mécanique des freins figure (I.20) des robots IRB 1600. 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.).[3]

Figure (I.20) : Bouton pour relâcher les freins à ne jamais actionner.

I.7.4. La répétabilité

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 multidirectionnelle. ABB spécifie que la répétabilité du robot IRB 1600 est de ±0.050 *mm*. Au fait, il s'agit de la répétabilité unidirectionnelle. [3]

I.7.5. Le contrôleur IRC5 et le FlexPendant

I.7.5.1. 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 boitier 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. Le contrôleur est principalement composé des items suivants figure (I.21) (a) :

- Un ordinateur industriel de type Pentium 4 cadencé à 1.4 GHz, protégé par une carte Compact Flash 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 , 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. [3]

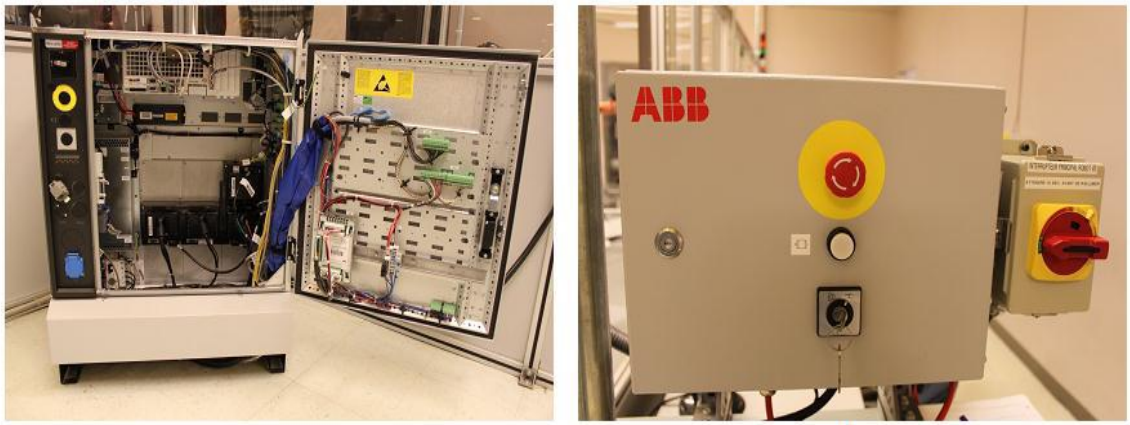


Figure (I.21) :(a) Contrôleur IRC5 standard de la compagnie ABB et (b) panneau opérateur externe d'un des quatre robots au laboratoire A-0610.

I.7.5.2. Le boitier de commande

Le boitier de commande appelé FlexPendant par ABB permet à l'opérateur d'interagir avec le contrôleur du robot figure (I.22). 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 boitier relié en tout temps avec le robot.[3]

Figure (I.22) : Boitier de commande FlexPendant de ABB.

Voici quelques règles et informations concernant le FlexPendant :

- Le boitier de commande est relié au contrôleur par un câble multibrin 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.

- Dans le coin du boitier 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.
- Sur le côté du boitier 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 boitier : départ, arrêt, avancement pas-à-pas ou recul pas-à-pas. Il est important de comprendre que l'arrêt est 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.[3]

Conclusion

Dans ce chapitre on a présenté les robots industriels les plus utilisés à travers le monde ainsi que leurs fabricants, leur historique en industrie et les différents champs d'applications. Quelques propriétés de ces robots sont évoquées dans ce chapitre qui vont servir dans les prochains chapitres comme le Flex Pendant.

Chapitre II

Introduction

Pour commander ou simuler le comportement d'un système mécanique articulé (robot), on doit disposer d'un modèle. Plusieurs niveaux de modélisation sont possibles selon les objectifs, les contraintes de la tâche et les performances recherchées.

Les modèles géométriques direct et inverse qui expriment la situation de l'organe terminal en fonction des variables articulaire et inversement.

Les modèles cinématiques directes et inverse qui expriment les vitesses de l'organe terminal en fonction des variables articulaires et inversement

Les modèles dynamiques direct et inverse définissant les équations du mouvement du robot qui permettent d'établir les relations entre les couples ou forces exercées par les actionneurs et les positions, vitesses, accélérations des articulations. [2]

II.1. Modèle Géométrique des robots

La Conception et la commande des robots nécessitent le calcul de certains modèles mathématiques tel que le modèle géométrique direct qui expriment la situation de l'organe terminal en fonction des vitesses articulaires du mécanisme et inversement.

La modélisation du robot de façon systématique et automatique exige une méthode adéquate pour la description de leur morphologie. Plusieurs méthodes et notion ont été proposées, la plus répandue est celle de Denavit-Hartenberg mais cette méthode, développée pour des structures ouvertes simples, présente des ambigüités lorsqu'elle est appliquées sur des robots ayant des structures fermées ou arborescente. C'est pourquoi, on utilise la notion de Khalil et Kleinfinger qui permet la description homogène, et avec un nombre minimum de paramètres, des architectures ouvertes simples et complexes des systèmes mécaniques articulés.[2]

II.1.1. Convention de Denavit-Hartenberg

Méthodologie à suivre pour décrire les robots à structure ouverte simple.

Une structure ouverte simple est composée de n+1 corps noté C_0,.....,C_n et de n articulations, le corps C_0 désigne la base du robot et le corps C_n le corps qui porte l'organe terminal.

L'articulation j connecte le corps C_j au corps C_{j-1} figure (II.1). [2]

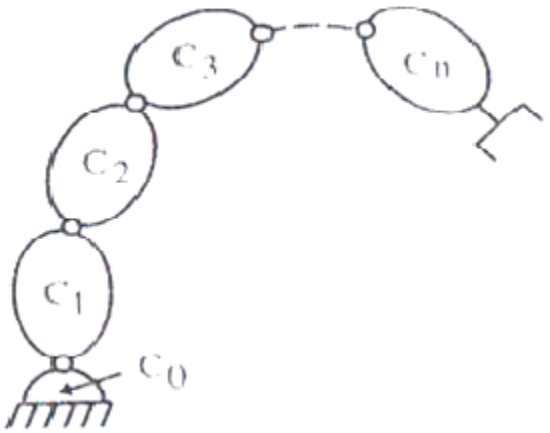


Figure (II.1) : Robot a structure ouverte simple.

La méthode de description est basée sur le principe suivant :

II.1.1.1. Principe

-Fixer des repères à chaque corps du robot.
-Calculer les matrices homogènes entre chaque corps.
-Calculer la matrice homogène entre base et organe terminal

II.1.1.2. Hypothèses

On suppose que le robot est constitué d'un chainage de n+1 corps liés entre eux par $n$articulations rotoïde ou prismatiques. A chaque corps, on associe un repère R_i. Les repères sont numérotés de 0 à n. La $i^{ème}$ articulation, dont la position est notée q_i, est le point qui relie les corps i-1 et i.
Le repère R_j fixé au corps C_j, est défini de sorte que :
-L'axe Z_j est porté par l'axe de l'articulation j.
-L'axe X_j est porté par la perpendiculaire commune aux axes Z_j et Z_{j+1} .Si les axes Z_j et Z_{j+1} sont parallèles ou colinéaires, le choix de X_i n'est pas unique. [2]

II.1.2. Les paramètres de Denavit-Hartenberg

Le passage du repère R_{j-1} eu repère R_j s'exprime en fonction des quatre paramètres géométrique suivants: figure (II.2)

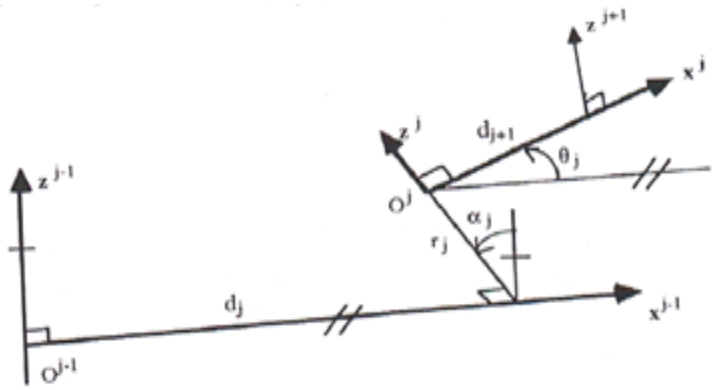


Figure (II.2) : Passage du repère.

1/ $\propto_j$: Angle entre les axes Z_{j-1} et Z_j correspondant à une rotation autour de X_{j-1}
2/ d_j : Distance entre Z_{j-1} et Z_j le long de X_{j-1}
3/ θ_j : Angle entre l'axe x_{j-1} et x et x_j correspondant à une rotation autour de z_j.
4/ r_j : Distance entre x_{j-1} et x et x_j le long de z_j.
-Si l'articulation i est de type prismatique, alors d_i est variable
-Si l'articulation i est de type rotoide, alors θ_i est variable.
La variable articulaire q_j associée à la $j^{ième}$articulation est définie par :

$$q_j = \sigma_j \theta_j + \sigma_j r_j \quad \text{(II.1)}$$

Avec :

σj= 0 si l'articulation *j* est rotoide
σj = 1 si l'articulation est prismatique.

A partir de cette description on peut définir la matrice de transformation homogène définissant le repère R_j dans le repère R_{j-1}
On a:

$$^{j}T_{j-1} = Rot\ (x, \propto_j) Trans(x, d_j) Rot(\ z, \theta_j) tras(z; r_j))\ \text{(II.2)}$$

$$^{i-1}T_i = \begin{bmatrix} c\theta_i & -s\theta_i & 0 & d_i \\ c\alpha_i s\theta_i & c\alpha_i c\theta_i & -s\alpha_i & -r_i s\alpha_i \\ s\alpha_i s\theta_i & s\alpha_i c\theta_i & c\alpha_i & r_i c\alpha_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \text{(II.3)}$$

On remarque que la matrice de rotation (3x3) $^{J-1}A_J$peut être obtenue par :

$$^{J-1}A_J = rot(\ x, \alpha_J)\text{Rot (Z,}\boldsymbol{\theta}_j\text{)}.\text{(II.4)}$$

La matrice de transformation définissant R_{j-1} dans R_j est donnée par :

$$J^T_{J-1} = \begin{pmatrix} j-1_{A^T_J} & -j-1_{A^T_J - j-1_{Pj}} \\ 0_{1.3} & 1 \end{pmatrix} \text{(II.5)}$$

$$J^T_{J-1} = \begin{pmatrix} A^T_J & -D_J C\theta_J \\ & -D_J S\theta_J \\ & -r_J \\ 0 \quad 0 & 0 \quad 1 \end{pmatrix} \text{(II.6)}$$

II.2. Le modèle géométrique direct

Le modèle géométrique direct (MDG) est l'ensemble des relations qui permettent d'exprimer la situation de l'organe terminal, c.à.d les coordonnées opérationnelles du robot, en fonction des coordonnées articulaires. Dans le cas d'une chaine ouverte simple, il peut être représenté par la matrice de passage ^0T_n : [2]
$^0T_n = {}^0T_1\ (q_1)\ {}^1T_2\ (q_2)$ ..... $^{n-1}T_n\ (q_n)$ (II.7)

Le modèle géométrique direct du robot peut être représenté par la relation :

$$X = (q)\ \text{(II.8)}$$

q étant le vecteur des variables articulaires tel que :

$$q = [q_1 q_2 \ldots \ldots qn]\ \text{(II.9)}$$

Les coordonnées opérationnelles sont définies par:

$$X = [X_1 X_2 \ldots \ldots . X_n] = \quad \text{(II.10)}$$

On a s= n x a :

$$X = [P_x P_y P_z \; s_x \; s_y \; s_z \; n_x n_y n_z a_x a_y a_z] \quad \text{(II.11)}$$

II.3. Le modèle géométrique inverse

Le modèle géométrique direct d'un robot permet de calculer les coordonnées opérationnelles donnant la situation de l'organe terminal en fonction des coordonnées articulaire, le modèle inverse consiste à calculer les coordonnées articulaires correspondant à une situation données de l'organe terminal. Lorsqu'elle existe, la forme explicite qui donne toutes les solutions possibles constitue le modèle géométrique inverse.

$q = f^{-1}(x)$ (II.12)

Pas de méthode analytique systématique pour calculer le MGI. Le mieux est de reprendre les équations du MGD et de mener le calcul à l'envers. Dans le cas où n = 6,

L'existence d'un poignet sphérique permet de débuter la résolution par :

$$Px = x_1 an_{xx} m + 1_{zx} \quad \text{(II.13)}$$

$$Py = x_2 an_{xy} m + 1_{zy} \quad \text{(II.14)}$$

$$Pz = x_3 an_{xz} m + 1_{zz} \quad \text{(II.15)}$$

Ensuite résolution au cas par cas pour exprimer les q_i, pour i = 1,2,…, n en fonction de P_x, P_y, P_z et des cosinus directeurs. [2]

II.4. Le modèle cinématique du robot

II.4.1. Le modèle cinématique directe

Le modèle cinématique directe d'un robot -manipulateur décrit les vitesses des Coordonnées opérationnelles en fonction des vitesses articulaires. Il est noté :

$$X = (q)\dot{q} \quad \text{(II.16)}$$

Ou J(q) désigne la matrice jacobéenne de dimension (m×n) du mécanisme, égale à $\frac{\vartheta X}{\vartheta q}$ et fonction de la configuration articulaire q. La même matrice jacobéenne intervient dans le calcul du modèle différentiel direct qui donne la variation élémentaire dX des coordonnées opérationnelles en fonction des variations élémentaires des coordonnées articulaires dq, soit : [2]

$$\mathrm{dX} = \mathrm{J(q)\ dq} \quad \text{(II.17)}$$

$$J(q) = \frac{\partial \mathrm{fi\ (q)}}{\partial\ \mathrm{qj}} \quad \text{(II.18)}$$

Avec i=1,....., m et j=1,.....,n

Le modèle cinématique directe peut être mit sous la forme :

$$\begin{pmatrix} \Omega p & 0_3 \\ 0_3 & \Omega r \end{pmatrix} \begin{pmatrix} {}^{\circ}Aj & 0_3 \\ 0_3 & {}^{\circ}Aj \end{pmatrix} \begin{pmatrix} I_3 & -ILjn \\ 0_3 & I_3 \end{pmatrix} iJ_{\mathrm{n}} \quad \mathrm{Jq} \quad \text{(II.19)}$$

Ou sous forme condensée :

$$X = {}^{0}J_n\dot{q} \qquad \text{(II.20)}$$

II.4.1.1. Intérêts de la matrice jacobienne

a/ Elle est la base du modèle différentiel inverse, permettant de calculer une solution local des variables articulaires q connaissant les coordonnées opérationnelles X.

b/ En statique, on utilise le jacobien pour établir la relation liant les efforts exercés par l'organe terminal sur l'environnement aux forces et couples des actionneurs.

c/ Elle facilite le calcul des singularités et de la dimension de l'espace opérationnel accessible du robot.

II.4.2. Modèle cinématique inverse

L'objectif du modèle cinématique inverse est de calculer, à partir d'une configuration q donnée, les vitesses articulaires q qui assurent au repère terminal une vitesse optimale X imposée. Cette définition est analogue à celle du modèle différentiel inverse : ce dernier permet de déterminer la différentielle articulaire dq correspondant à une différentielle des coordonnées opérationnelles dX spécifiée.

Pour obtenir le modèle cinématique inverse, on inverse le modèle cinématique directe en résolvant un système d'équation linéaires ; la mise en œuvre peut être faite de façon analytique ou numérique.

- la solution analytique a pour avantage de diminuer considérablement le nombre d'opération, mais on doit traiter tous les cas singuliers
- Les méthodes numériques sont plus générales, la plus répandue étant fondée sur la notion de pseudo- inverse : les algorithmes traitent de façon unifiée les cas réguliers, singuliers et redondants ; elles nécessitent un temps de calcul relativement important .

Dans ce cas, la matrice jacobienne J est carrée d'ordre n et son déterminant est non nul.

On calcule J^{-1} , la matrice inverse de J, qui permet de déterminer les vitesses articulaire *q* grâce à la relation:

$$\dot{q} = J^{-1} X \qquad \text{(II.21)}$$

Lorsque la matrice J a la forme suivante :

$$J = \begin{pmatrix} A & 0 \\ B & C \end{pmatrix} \qquad \text{(II.22)}$$

Les matrices A et C étant carrées inversibles, il est facile de montrer que l'inverse de cette matrice s'écrit :

$$J^{-1} = \begin{pmatrix} A^{-1} & 0 \\ -C^{-1}BA^{-1} & C^{-1} \end{pmatrix} \qquad \text{(II.23)}$$

A et *C* étant de dimension (3x3). [10]

II.5. L'orientation

On peut représenter une orientation (rotation) par plusieurs représentations mathématiques, les plus utilisées sont :

- Les trois angles d'EULER
- Les trois angles de CARDAN
- Les matrices des cosinus directeurs

• Les quaternions (voir ANNEXE)

II.6. Le principe de la description des tâches

Dans la plupart des contrôleurs de robot actuels, la géométrie des trajectoires est décrite par une succession de repères. Suivre une trajectoire consiste alors à amener en coïncidence sur ces repères un référentiel lié à l'organe terminal. Lorsque le nombre de composantes utilisées pour spécifier la tâche est inférieur au nombre de degrés de liberté du robot, il y a redondance de celui-ci vis-à-vis de la tâche et donc, une infinité de solution pour la réaliser. Cette description, minimale en ce sens qu'elle ne contraint que les degrés de liberté de la tâche ayant un rôle fonctionnel, est intéressante car elle permet de satisfaire des critères d'optimisation supplémentaires lors de son exécution.

Le formalisme retenu est celui du contact des surfaces usuelles de la mécanique (plan, cylindre, sphère) qui décrit les liaisons mécaniques simples tableau (II.2) illustrées par la figure (II.3). A ces six liaisons, on adjoint la liaison totale correspondant à un encastrement et les deux liaisons composées à un degré de mobilité :la liaison rotoïde et la liaison prismatique figure (II .4). [10]

La description des situations successives du repère terminal lié au robot est réalisée par une séquence de liaisons simples et composées, le choix d'une liaison étant dicté par les contraintes locales associe à la tâche. [10]

	Plan	Plan	Plan
Plan	Appui plan	Appui linéaire	Appui simple
Cylindre	Verrou	Anneau	Sphère
Rotule			

Tableau (II .1) : Liaison mécaniques simples

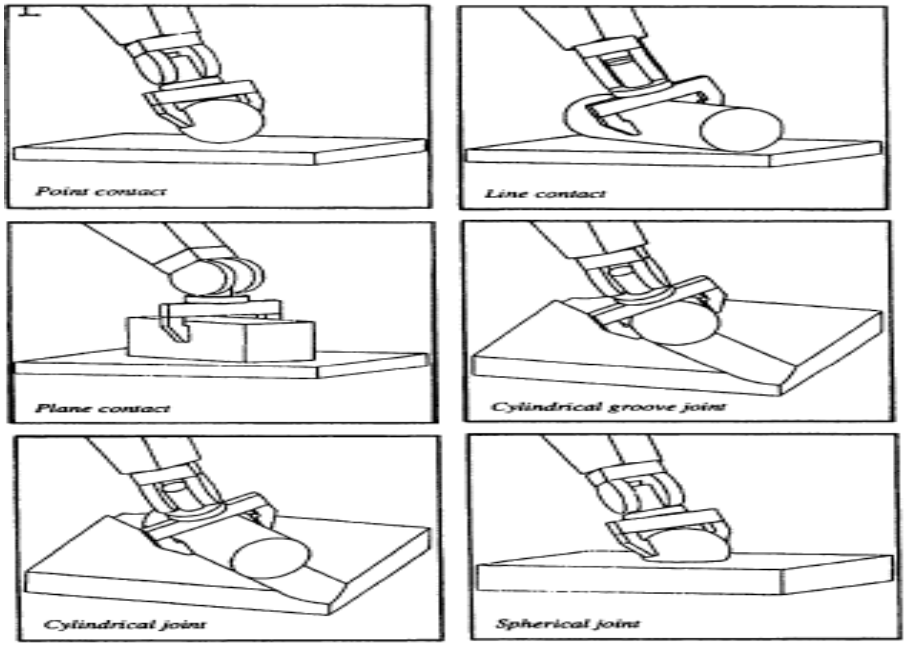


Figure (II.3) : Liaison mécaniques simples

Le formalisme des liaisons mécaniques peut paraître difficile à utiliser. Une solution plus ergonomique consiste à décrire un déplacement en termes de liaison entre deux entités géométriques simples (point, droite, plan) appartenant l'une au robot, l'autre à l'environnement :
Une liaison rotule, par exemple, correspond à la mise en coïncidence de deux points. De même, les liaisons composées seront spécifiées par deux combinaisons simultanées d'éléments géométriques.
Le choix n'est pas unique : ainsi, une liaison rotoïde par exemple peut être issue de la réalisation d'une liaison droite sur droite et d'une liaison point sur plan ou d'une liaison droite sur droite et d'une liaison point sur plan ou d'une liaison droite sur droite et d'une liaison point sur point.[10]

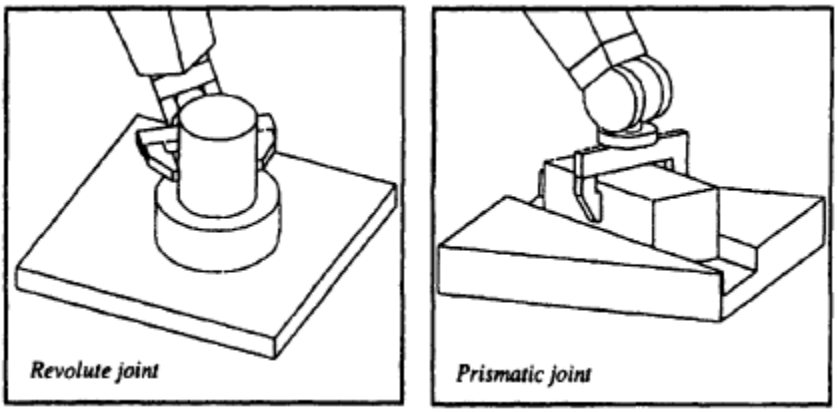


Figure (II .4) : Liaison rotoïde et prismatique

Cette description géométrique est particulièrement bien adaptée à une programmation graphique des tâches. La figure (II .5) montre l'exemple d'un assemblage cylindrique, réalisé avec l'application robotique du logiciel de CFAO CATIA dans lequel est implanté ce formalisme. Les différentes étapes sont les suivants :
a) Etat initial : sélection d'un point du robot et d'un point de l'environnement (rotule)
b) Le cylindre est positionné, avec une orientation quelconque, au-dessus du site d'assemblage ; sélection d'une droite du robot et d'une droite de l'environnement (verrou)
c) Les axes du tampon cylindrique et de l'alésage sont alignés ; sélection d'un point et d'une droite du robot d'une part, de l'environnement d'autre part (rotoïde)
d) Etat final : la tâche d'assemblage est terminée. On peut remarquer la rotation laissée libre autour de l'axe principal du cylindre entre les états c) et d). [10]

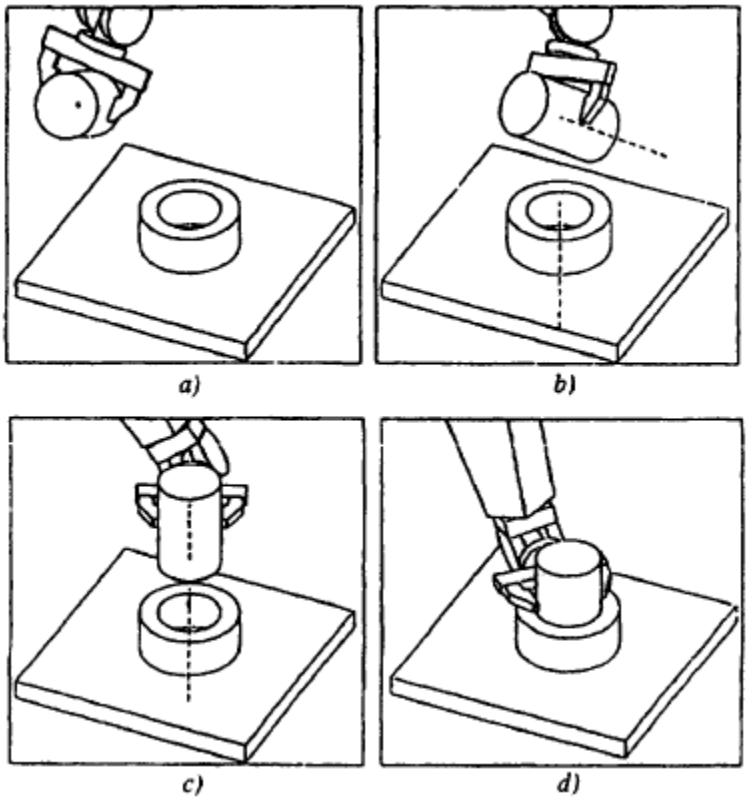


Figure (II.5) : Programmation graphique d'un assemblage à partir d'une description minimale des tâches.

II.7. Modèles différentiels associe à une description minimale

Pour prendre en compte ce type de description de tâche, on écrit le modèle différentiel définissant les relations différentielles du repère outil R_E de la façon suivante :

$$\begin{pmatrix} {}^{0}_{d}p^{e} \\ {}^{0}_{\delta E} \end{pmatrix} = \begin{pmatrix} {}^{\circ}An & 0_3 \\ 0_3 & {}^{\circ}A_n \end{pmatrix} \begin{pmatrix} {}^{n}_{d}E \\ n_{\delta E} \end{pmatrix} = \begin{pmatrix} 0_{An} & 0^3 \\ 0_3 & 0_{An} \end{pmatrix} \begin{pmatrix} I_3 & -n_{PE} \\ 0_3 & I_3 \end{pmatrix} \begin{pmatrix} {}^{n}_{d}p^{n} \\ n_{\delta^n} \end{pmatrix} = \begin{pmatrix} {}^{\circ}An & -0_{An\,P_E} \\ 0_3 & 0_{An} \end{pmatrix} nJ_n dq \quad \text{(II. 24)}$$

Le terme ${}^{n}P_E$définissant l'origine du repère R_E exprimée dans R_n.

Avec une description minimale de la tâche le modèle différentiel s'écrit :

$dX = H^{n}J_n dq$(II.25)

Où${}^{n}J_n$est de dimension (6 x *n*) et où H est une matrice de dimension (c x 6).

La caractérisation de la liaison c dépend du nombre d'équations de certaine associées à la tâche.

On va commencer par la description de l'exemple le plus simple qui est le suivant. [10]

II.8. Appui simple (point. Plan)

Cette liaison consiste à amener un point O_E de l'outil dans le plan Q. sa position dansle plan Q étant quelconque figure (II.6). Soit N le vecteur unitaire selon la normale auplan Q et soit O_D un point arbitraire du plan Q. Le déplacement global à effecteur pourréaliser l'appui simple s'exprime dans le repère R_0 par :

$R={}^{0}N^{t}({}^{0}P_{d}-{}^{0}P_{E})$ (II.26)

${}^{0}P_d$ et ${}^{0}P_E$ définissant les points OD et OE dans le repère R0. [10]

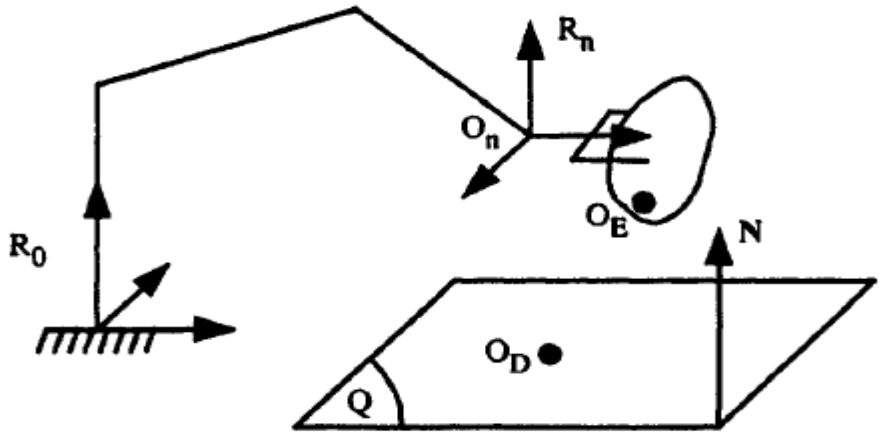


Figure (II.6) : Réalisation d'un appui simple.

Le déplacement global *r* est réalisé par une succession de déplacements élémentaires àune seule composante tels que :

$$dX = dr = {}^{\circ}N_T \, {}^{\circ}dP_E = ({}^{\circ}N_T \, {}^{\circ}A_N - {}^{\circ}N_T \, {}^{\circ}A_N \, {}^{n}p_E) \, {}^{n}J_n \, dq = {}^{\circ}N_T \, {}^{\circ}A_N (I_3 - {}^{n}p_E) \, {}^{n}J_n \, dq \quad \text{(II.27)}$$

La matrice H est alors identifiée et n'a qu'une seule ligne.L'expression (II.27) constitue le modèle différentiel de l'appui simple.

Remarque : On peut résumer les équivalences liaison mécanique/spécification géométrique par le tableau ci-dessous :

Type de liaison	Elément de l'effecteur	Elément de l'environnement	Nombre d'équations indépendantes	Nombre total d'équations
Type de liaison	Point	Plan	1	1
Elément de	Droite	Plan	2	2
l'effecteur	Plan	Plan	3	3
Elément de	Point	Droite	2	2
l'environnement	Point	Point	3	3
Nombre	Droite	Droite	4	4
d'équations	Droite-Point	Droite-Point	5	7
indépendantes	Plan-Plan	Plan-Plan	5	6
Nombre total d'équations				

Tableau (II.2) : Equivalence liaison mécanique/spécification géométrique.

Remarque : L'inconvénient majeur du modèle cinématique pour le transformateur de coordonnées est qu'il ne traduit qu'un comportement local du robot autour d'une configuration articulaire donnée. La trajectoire engendrée dépend donc de la configuration initiale du mécanisme.

Conclusion

Ce chapitre a été consacré aux outils mathématiques utilisés dans la modélisation, la simulation et la planification des robots manipulateurs.

Les programmes des différentes taches planifiées dans la robotique industrielle, que ça soit graphique ou par langage procédurale sont développés via ces outils mathématiques.

Nous avons montré comment, à partir d'une description utilisant le formalisme des liaisons mécaniques, il était possible de contraindre uniquement les degrés de liberté fonctionnels de la tâche.

Chapitre III

Introduction

Dans le processus continu, le robot est manipulé par une série de commandes et de fonctions stockées dans un fichier dit << programme robot>>. La trajectoire du robot doit être programmée avant son exécution. Deux modes de programmation des trajectoires robot sont couramment utilisés, la programmation en ligne (PEL) et la programmation hors-ligne (PHL).

III.1 La programmation en ligne

La programmation en ligne est aujourd'hui le mode de programmation le plus utilisé industriellement, il existe deux méthodes de programmation en ligne: l'apprentissage direct et l'apprentissage indirect point à point.

Figure (III.1) : Programmation en ligne.

III.1.1. L'apprentissage direct

Le principe de cette méthode de programmation consiste à mémoriser, lors de l'apprentissage, l'ensemble des configurations articulaires du robot pendant l'exécution de la tache ou pendant une simulation de celle-ci. L'opérateur manipule directement le robot à l'aide d'un pupitre d'apprentissage pour lui "apprendre" la tâche à accomplir. L'acquisition de la trajectoire est continue. Ce type de programmation est simple et demande peu de formation pour le programmeur. Cette méthode est plutôt utilisée pour des applications qui exigent des mouvements de précision limitée, dans la projection sur les pièces simples ou le domaine de la peinture.

III.1.2. L'apprentissage indirect point par point

Le principe utilisé par cette méthode est également basé sur une démonstration matérielle de la tâche à réaliser. L'opérateur déplace le robot en mode manuel, utilisant pour cela le pupitre de commande à touche. Cependant, la configuration du robot n'est enregistrée qu'en certains points caractéristiques de la trajectoire. La liaison entre ces différents points est spécifiée par les caractéristiques de la trajectoire entre ces points (linéaire, articulaire voire circulaire) et par la définition d'une vitesse de déplacement spécifiée par l'opérateur.

III.2. La programmation hors-ligne

La programmation hors-ligne permet la génération des programmes avec une précision beaucoup plus importante. Le temps consacré sur site est fortement réduit. Le positionnement des trajectoires ainsi que la conception des outils peuvent être optimisés.
Un des principaux intérêts de ce type de méthode est qu'il ne nécessite aucune immobilisation de l'outil de production durant la phase de programmation du robot. On distingue 2 approches principales pour la programmation hors-ligne des robots : la programmation par langage et la programmation graphique à partir d'un système de CFAO (Conception et Fabrication Assistée par Ordinateur). [3]

III.2.1. La programmation par langage robotique

Les langages de programmation des robots sont proches des langages informatiques classique. Ils proposent des instructions particulières pour commander les mouvements des robots. Ils sont performants pour définir des positions par calculs ou pour implanter des instructions conditionnelles ou des boucles.
Le principal inconvénient de ces langages est qu'ils restent généralement spécifiques à chaque marque de robot ; il n'existe pas de standard reconnu. Cela oblige l'opérateur à connaitre un langage pour communiquer avec le robot. Les langages suivants peuvent être cités : RAPID (ABB); VAL II (UNIMATION), RAIL (AUTOMATIX), PALAW (KOMATSU). Dans notre étude nous allons utiliser le langage RAPID.
Ce type de programmation nécessite de connaitre à l'avance les points de passage, l'orientation de l'outil ainsi que la vitesse locale de déplacement pour chaque point.

III.2.2. La programmation graphique

Cette méthode de programmation est une méthode plus pratique pour programmer des trajectoires robot sur les pièces de forme complexe. Les logiciels de CFAO actuels permettent de représenter et de modéliser les robots, les pièces à revêtir et l'environnement. La figure (II.2) présente la programmation hors-ligne à l'aide d'un logiciel CFAO robotique.

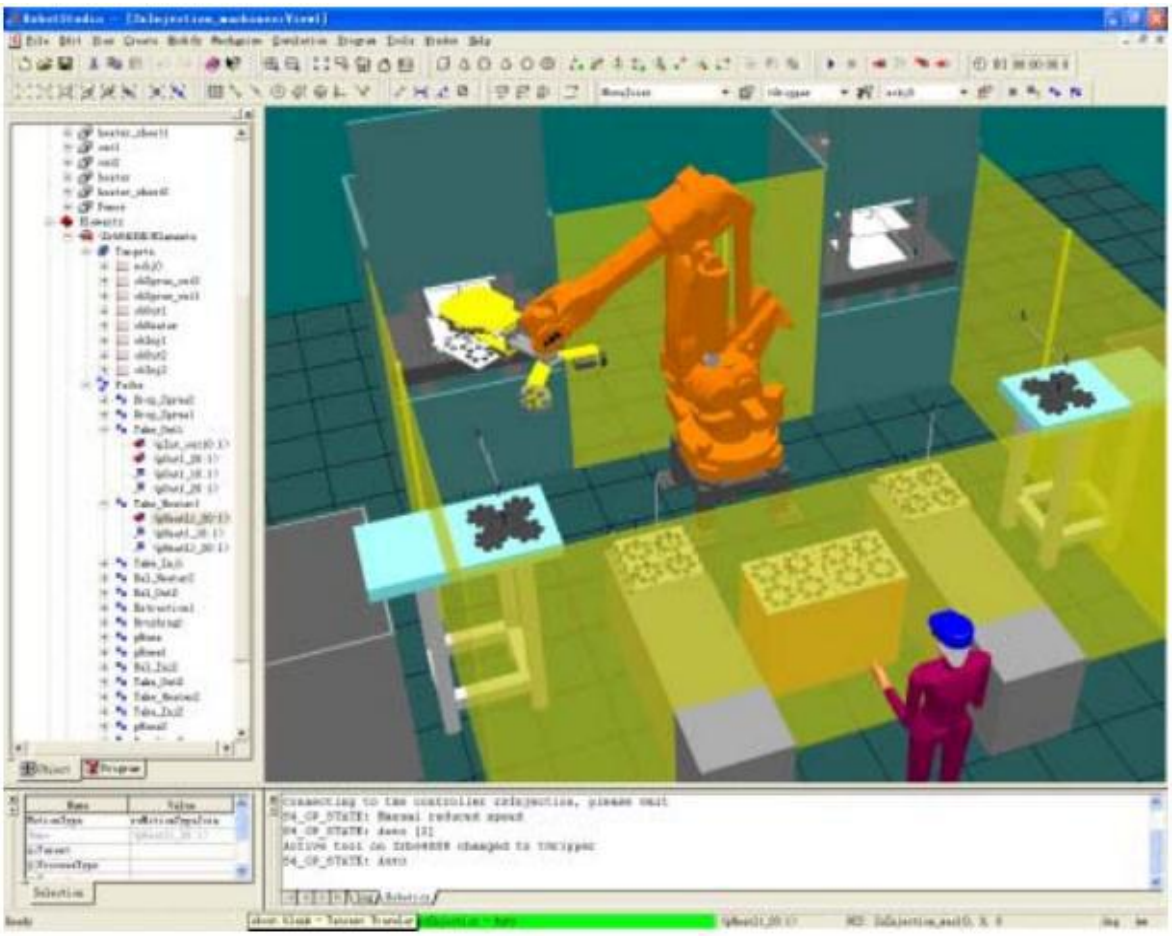


Figure (III.2) : Programmation graphique.

Dans le cadre de cette étude, le logiciel de CFAO robotique dit "RobotStudioTM " est utilisé en raison de sa bonne compatibilité entre les robots virtuels et les robots réels.
RobotStudio est un logiciel d'ABB qui permet de simuler et de programmer hors-ligne des systèmes robots à l'aide d'un PC standard fonctionnant sous Windows.

III.3. Le RobotStudio

RobotStudio est fondé sur le " VIRTUAL CONTROLLER" d'ABB qui est une copie exacte du logiciel réel qui commande le robot en production. Il est ainsi possible d'effectuer des simulations très réalistes à l'aide de programmes de robots réels et de fichiers de configuration identiques à ceux utilisés dans l'atelier. Le Virtual Controller contient ainsi un " pupitre mobile d'apprentissage" virtuel qui permet d'utiliser le robot simulé exactement comme un vrai robot.
RobotStudio augmente la rentabilité du système robot en permettant d'effectuer des taches comme la formation, la programmation et l'optimisation sans gêner la production.
RobotStudio offre ainsi plusieurs avantages comme :

- Réduction des risques- vérification de nouvelles installations en créant rapidement sa propre station, en menant des études de faisabilité précises, et en vérifiant l'accessibilité, les collisions et les problèmes de singularité.

- Mise en service plus rapide- le temps consacré à la programmation peut être réduit et, plus important encore, cette tâche peut être effectuée avant l'installation du système.
- Changement plus rapide- il est possible de tester les modifications apportées à des installations existantes ou de mettre à jour les programmes de production de nouvelles pièces.
- Optimisation des programmes- disposition de plusieurs outils qui facilitent l'optimisation des programmes de robot.
- Environnement de simulation 3D- représentation 3D complète des systèmes robotiques.
- Système de commande virtuel- système de commande de robot intégré à un PC standard.
- Bibliothèque de robots- modèles exacts de la gamme de produit complète ABB.
- Pupitre mobile virtuel d'apprentissage- parfait pour la formation hors -ligne des opérateurs.
- Editeur de programmes RAPID- vérification automatique des erreurs de programme.
- Simulateur des E/S- interaction avec des entrées/sorties simulées.
- VBA (Visual Basic for Application)- permet de développer des fonctionnalités particulières [3].

Son inconvénient principal est l'impossibilité de choisir les vitesses et les accélérations optimales de l'exécution des différentes taches planifiées, RobotStudio permet le choix de vitesses prédéfinies, néanmoins une solution consiste à lui ajouter des extensions sous forme de programmes.

III.3.1. La station de RobotStudio

D'une certaine façon, une station de RobotStudio correspond à une cellule de robot dans une usine. La station contient le robot, l'outil, les pièces de travail, les programmes et tout ce qui peut être utile lors de la simulation. Une seule station peut être ouverte lors de l'utilisation du logiciel. La station peut être enregistrée dans un fichier. Les fichiers de station portent l'extension << .stn >>. Un fichier de station contient :

- Un système de coordonnées dans lequel sont positionnés les objets de la station.
- Des références aux objets de la station.
- Des informations sur le système de commande virtuel utilisé dans la station. [3]

III.4. La programmation

III.4.1. Le langage RAPID

Le langage de programmation des robots ABB s'appelle RAPID. Un programme en RAPID est un ou plusieurs fichiers de type texte (ASCII), non compilées. Le contrôleur du robot interprète 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.
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.
Le langage RAPID possède tous les types de données atomiques classiques : booléen, numérique, chaine de caractères, etc. [3]

III.4.1.1. Syntaxe d'une déclaration

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 pas commencer par un caractère numérique ou par un caractère spécial. Cependant, noter que le tiret bas (<_ >) est toléré.
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.
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.
Il est impossible de déclarer une variable à mi- programme. L'ordre de classement des registres dans la déclaration n'est pas imposé. Il est possible de les regrouper par type, besoin, équipement, etc. [3]

III.4.1.2. Le registre mémoire

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

III.4.1.2.1. La 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.

III.4.1.2.2. La 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 >).

III.4.1.2.3. La 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 Les persistantes ne peuvent être déclarées qu'au niveau d'un module, et non à l'intérieur d'une procédure. [3]

III.4.1.3. Les enregistrements

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

III.4.1.3.1. La 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]]. Le langage RAPID utilise la représentation par quaternions (voir ANNEXE).

III.4.1.3.2. La wobjdata

L'enregistrement wobjdata (< work object data>) permet 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.
- ✓ ufprog (bool) : Sert à définir si le référentiel se déplace dans l'espace ou s'il est fixe.
- ✓ ufmec (string) : Sert à définir le mécanisme à suivre si ufprog est faux.
- ✓ uframe (pose) : Sert à définir la pose du référentiel du lieu de travail (< user frame >) par rapport au référentiel de l'atelier wobj0 (figure III.4). 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 à l'uframe (figure III.3). 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. [3]

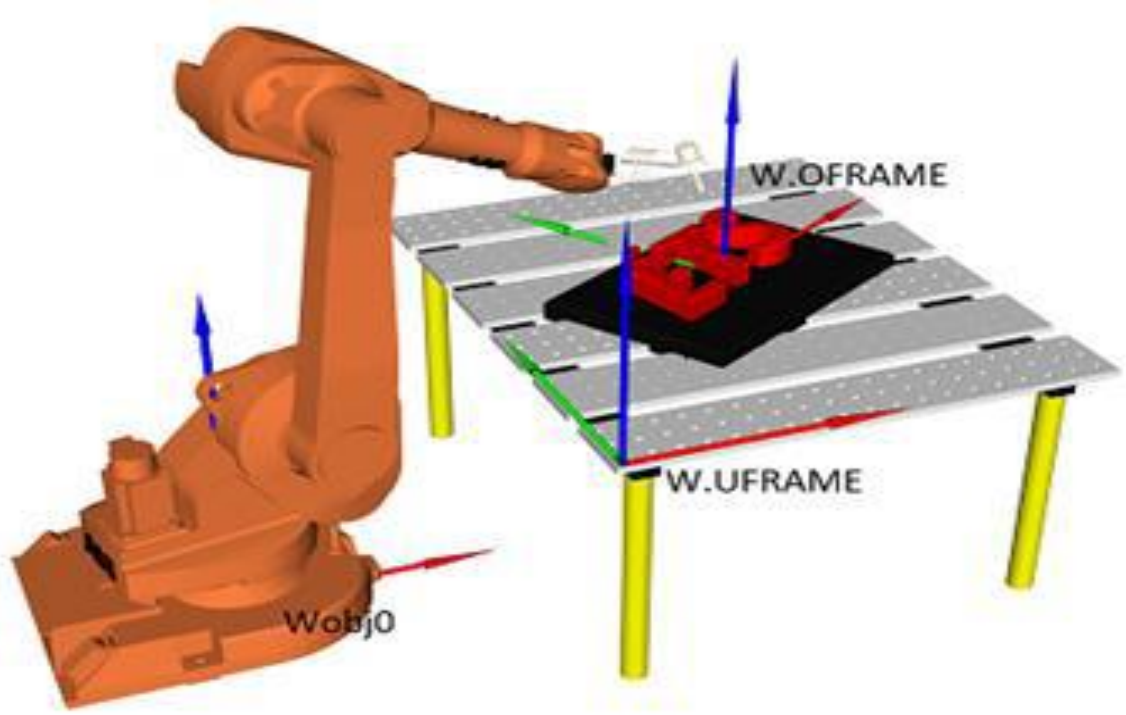


Figure (III.3) : Référentiels faisant partie d'un enregistrement de type workoject,dont le nom est W.

III.4.1.3.3. Le 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).
- ✓ extax (extjoint) : Permets d'interagir avec les articulations externes du robot. [3]

III.4.1.3.4. La tooldata

L'enregistrement de type tooldata permet de définir un référentiel sur un outil ainsi que les caractéristiques physiques de l'outil.

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. 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 ou le robot tient l'outil cette valeur sera toujours égale à false.

- ✓ 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 connaitre les forces physiques générées par l'outil lors de son déplacement et ainsi optimiser sa vitesse et sa trajectoire. [3]

III.4.1.4. Les 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. [3]

III.4.1.4.1. Les types de déplacements

III.4.1.4.1.1. Le déplacement linéaire

Dans un déplacement linéaire, l'origine du référentiel de l'outil spécifié suit une trajectoire linéaire (figure III.4). 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 une 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. 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.

Figure (III.4) : Déplacement linéaire de l'outil (commande moveL).

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 atteindre durant le déplacement.
4. zonedata : Précision d'arrêt ou d'approche de l'origine du référentiel de l'outil sur un robtarget à atteindre.
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).

III.4.1.4.1.2. Le déplacement articulaire

Dans un déplacement articulaire, la trajectoire suivie par l'effecteur du robot est difficilement prévisible (figure III.5). 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 ou 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).

Figure (III.5) : Déplacement articulaire de l'outil (commande MoveJ).

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 ou 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 atteindre durant le déplacement.
4. zonedata : Précision d'arrêt ou d'approche de l'origine du référentiel de l'outil sur un robtarget à atteindre.
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).

III.4.1.4.1.3. Le 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 (1er) : Valeur venant d'une variable, fonction ou autre qui indique le robtarget intermédiaire par lequel l'outil doit passer.
3. robtarget (2e) : 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.
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 définit 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).

III.4.1.5. La configuration mécanique

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. [3]

III.4.1.6. La vitesse et précision du déplacement

III.4.1.6.1. La 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 rotoides des externes en mm/s. [3]

III.4.1.7. 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 l'optimisation des 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. (Figure III.6).

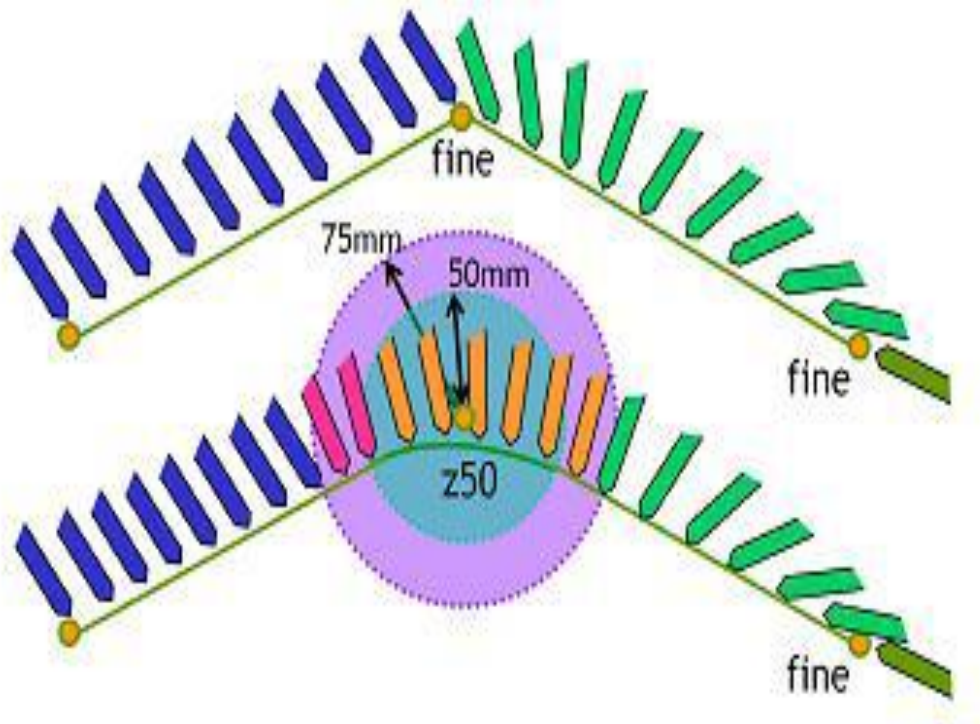


Figure (III.6) : Interpolation entre deux déplacements linéaires.

1. finep : Variable booléenne définissant si le robot doit s'arrêter ;
2. pzone-tcp : rayon en mm de la sphère ou l'origine du référentiel de l'outil peut commencer l'interpolation vers le prochain trajet ;

3. pzone-ori : rayon en mm de la sphère ou 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. [3]

III.4.1.8. Les 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 : [3]

III.4.1.8.1. La RelTool

Cette fonction retourne un robtarget en faisant une transformation (par exemple, une rotation) par rapport au référentiel d'un robtarget donné.
Exemple : 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.

III.4.1.8.2. L'Offs

Cette fonction retourne un robtarget décalé dans le repère du robtarget donné.
Exemple: r1:=Offs(robtarget, Dx, Dy, Dz); [3]
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 ;

III.4.1.8.3. 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 orientation contrôlée à partir de la translation des robtargets. [3]
Exemple: 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 (Ypositif) ;
(c) \Origin :=3 : Méthode utilisée lors de l'enseignement avec la méthode de 3 points dans le boitier 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.

Conclusion

La programmation graphique sous tous ses aspects, ses avantages, ses inconvénients et ses contraintes ont été évoqués dans ce chapitre. Les commandes de déplacement ont une importance capitale dans la programmation et la planification d'une tache, dans ce chapitre une attention particulière a été donnée à ce sujet.

Chapitre IV

Introduction

Les systèmes à convoyeurs sont indispensables dans l'industrie (manutention, stockage, déstockage...etc.), ils nécessitent une bonne et fiable programmation pour assurer leur fonctionnement. La programmation en ligne est la plus utilisée dans l'industrie, il existe deux méthodes de programmation en ligne : l'apprentissage direct et l'apprentissage indirect point à point, tandis que la programmation hors-ligne permet la génération des programmes avec une précision beaucoup plus importante, le temps consacré sur site est fortement réduit et le positionnement des trajectoires ainsi que la conception des outils peuvent être optimisés.

Robot sur table IRB 660 il transporte les boites de la ligne de production pour les mettre sur la palette dans une autre ligne de production. L'objectif principal du projet est de créer une ligne de production de tri simple.

Principaux défis de la mise en œuvre du projet : Manipulation d'objets à l'aide d'une pince intelligente (ABB), simple pince à vide communication de données entre les multiples bras robotiques contrôle de la bande transporteuse programme rapide et propre utilisant des fonctions, des structures, etc.

Le projet a été créé pour améliorer le VRM.

IV.1. Robot type IRB 660 Description

IV.1.1. Famille de robots

L'IRB 660 est le dernier robot de palettisation à quatre axes d'ABB. Il se distingue par une capacité de production élevée, un temps de cycle réduit avec une charge utile élevée, une longue portée et un temps productif très élevé. Il existe en deux versions (capacités de manutention de 180 et 250 kg) disposant chacune d'une portée de 3,15 m. Les connexions client (alimentation, signaux, signaux bus) sont intégrées au robot, de la base de celui-ci aux connexions effectuées au niveau de la bride d'outil.

IV.1.2. Système d'exploitation

Le robot est équipé du système de commande IRC5 et du logiciel de commande du robot, RobotWare. RobotWare prend en charge tous les aspects du système de robot, notamment le contrôle des mouvements, le développement et l'exécution des programmes applicatifs, la communication, etc. Pour en savoir plus, voir Caractéristiques du produit – Système de commande IRC5 avec Flex Pendant.

IV.1.3. Sécurité

Les normes de sécurité concernent le robot, le manipulateur et le système de commande complets.

IV.1.4. Fonctionnalités complémentaires

Pour offrir des fonctionnalités supplémentaires, le robot peut être équipé d'un logiciel optionnel de prise en charge d'applications. Par exemple, l'encollage et le soudage, des fonctions de communication comme la communication réseau, ainsi que des fonctions avancées telles le multitâches, le contrôle par capteurs et autres. Pour obtenir la description complète des logiciels optionnels, reportez-vous à la section Caractéristiques du produit - Logiciel du système de commande IRC5.

Figure (IV.1) : Robot manipulateur d'ABB L'IRB 660.

IV.1.5. Axes du manipulateur

Le manipulateur IRB 660 est équipé de 4 axes, comme indiqué dans la figure ci-dessus.

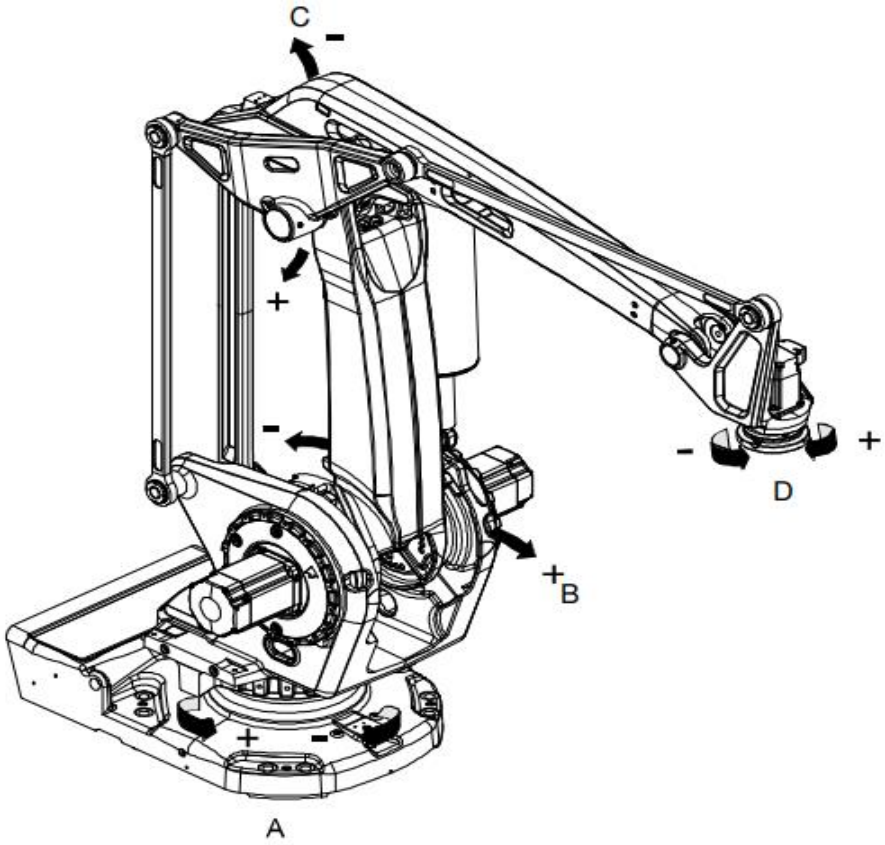


Figure (IV.2) : Robot manipulateur d'ABB L'IRB 660 schéma descriptif.

Position	Description	Zone de travail	Vitesse maximum
A	Axe 1	+180° à -180°	130°/s
B	Axe 2	+85° à -42°	130°/s
C	Axe 3	+120° à -20°	130°/s
D	Axe 4	+300° à -300°	300°/s

Tableau VI.1 : caractéristiques géométriques et cinématique du manipulateur IRB 660.

IV.1.6. Dimensions de l'IRB 660

Les figures suivantes montrent des vues de l'avant, des côtés et du haut du manipulateur IRB660 (dimensions en mm). Derrière le pied du manipulateur, laissez 200 mm pour les câbles.

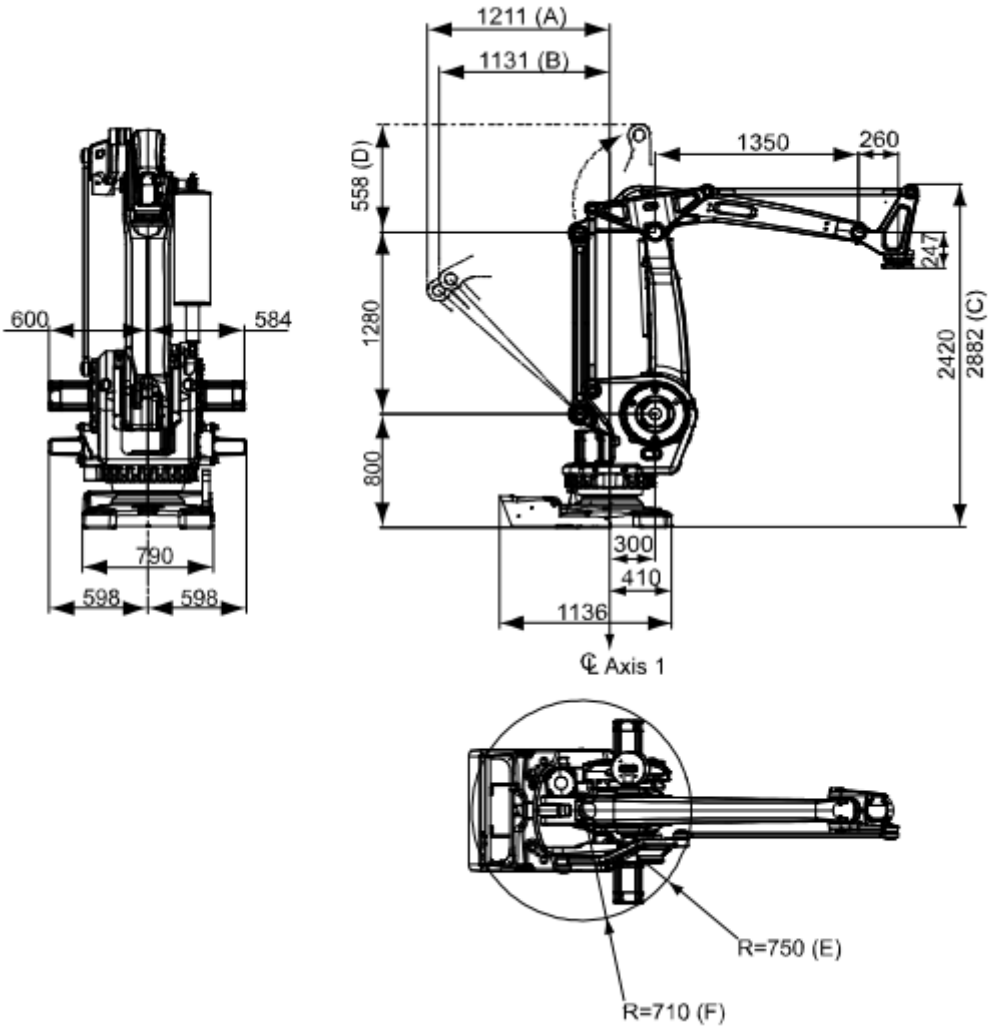


Figure (IV.3) : Robot manipulateur d'ABB L'IRB 660 schéma du constructeur.

IV.1.7. Diagrammes des charges

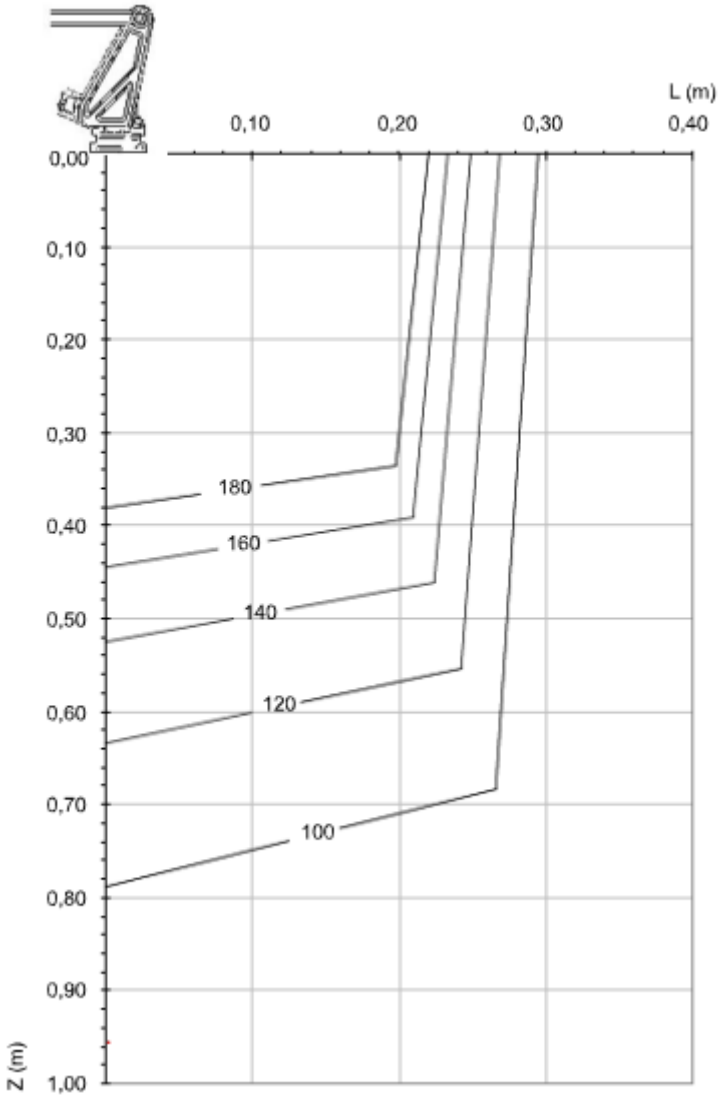


Figure (IV.4) : Robot manipulateur d'ABB L'IRB 660 diagramme des charges.

IV.2. Application

Dans cette partie nous allons effectuer la simulation avec le logiciel ABB robot studio un exemple simple d'une ligne de production de tri (IRB 660)

• En premier temps on va créer une station vide.

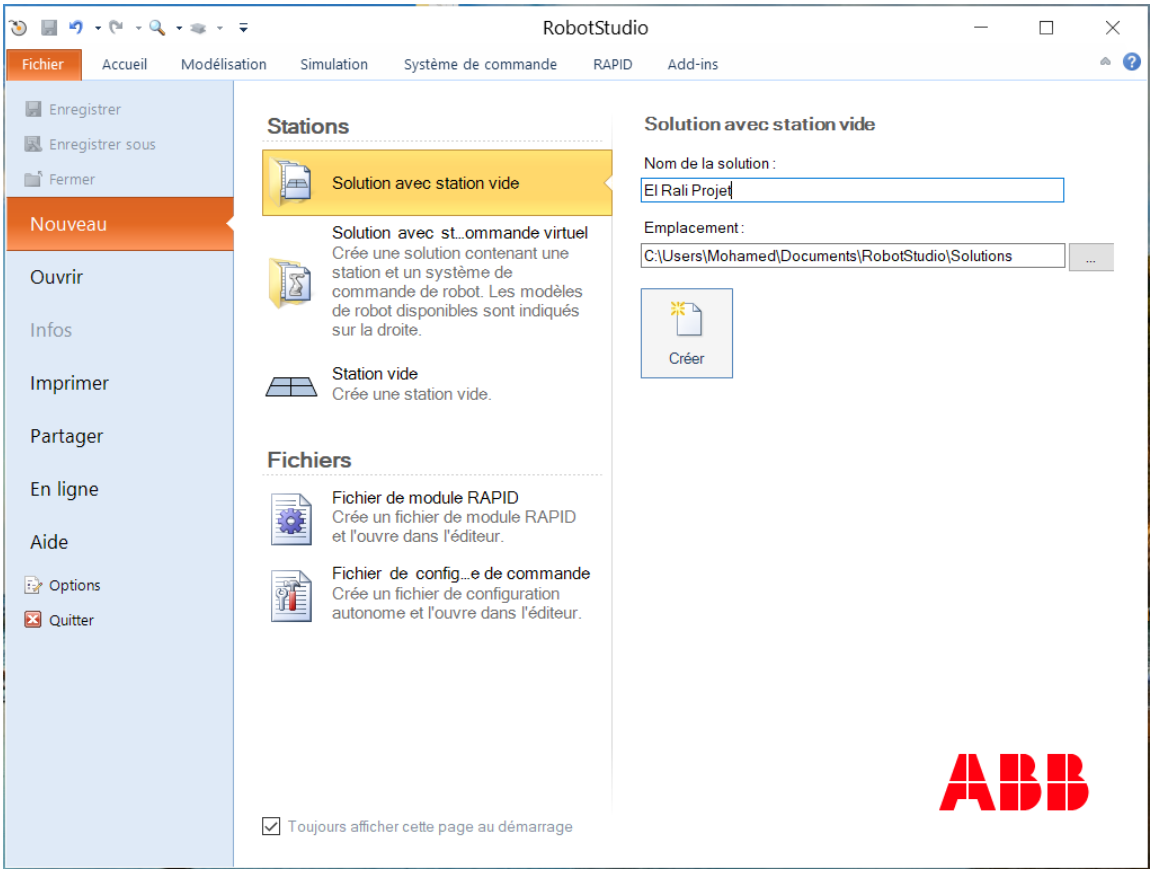


Figure (IV.5) : Création d'une station vide

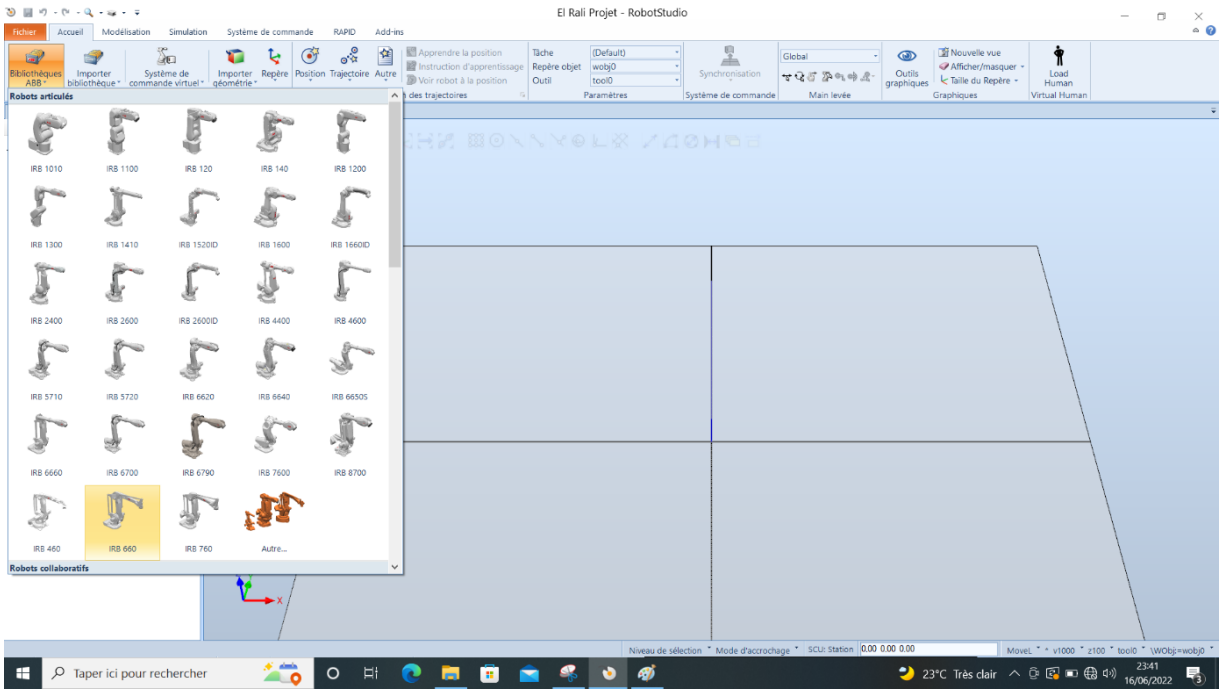


Figure (IV.6) : Importation un robot IRB660

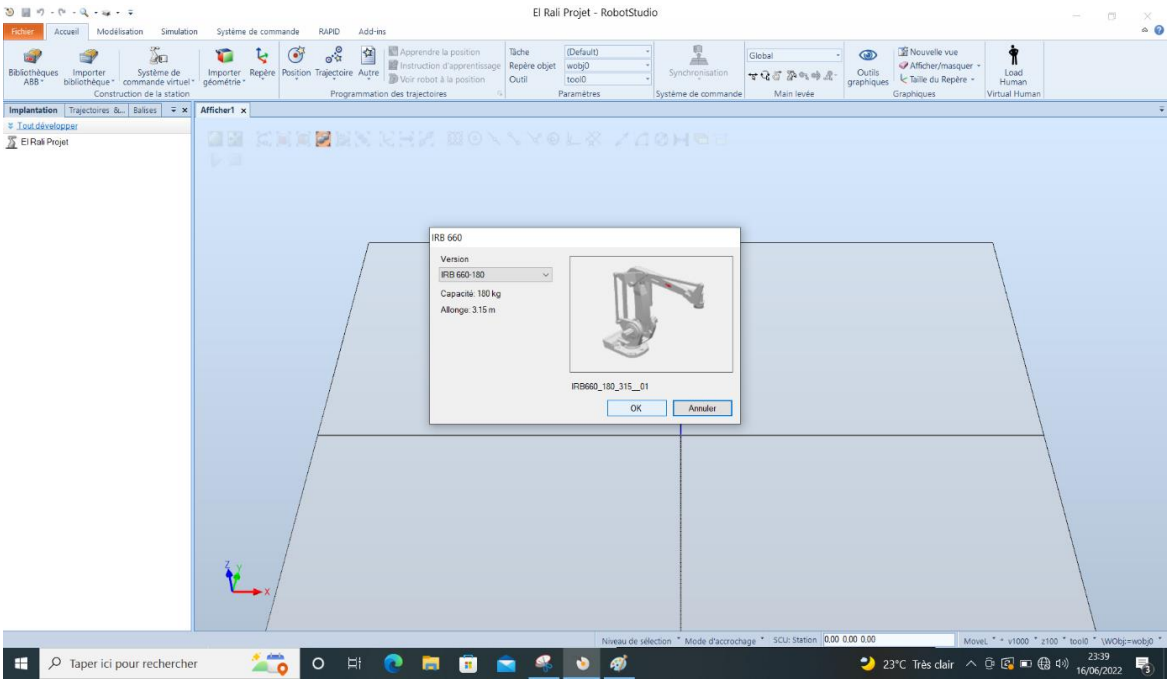


Figure (IV.7) : Définir la version de robot IRB 660

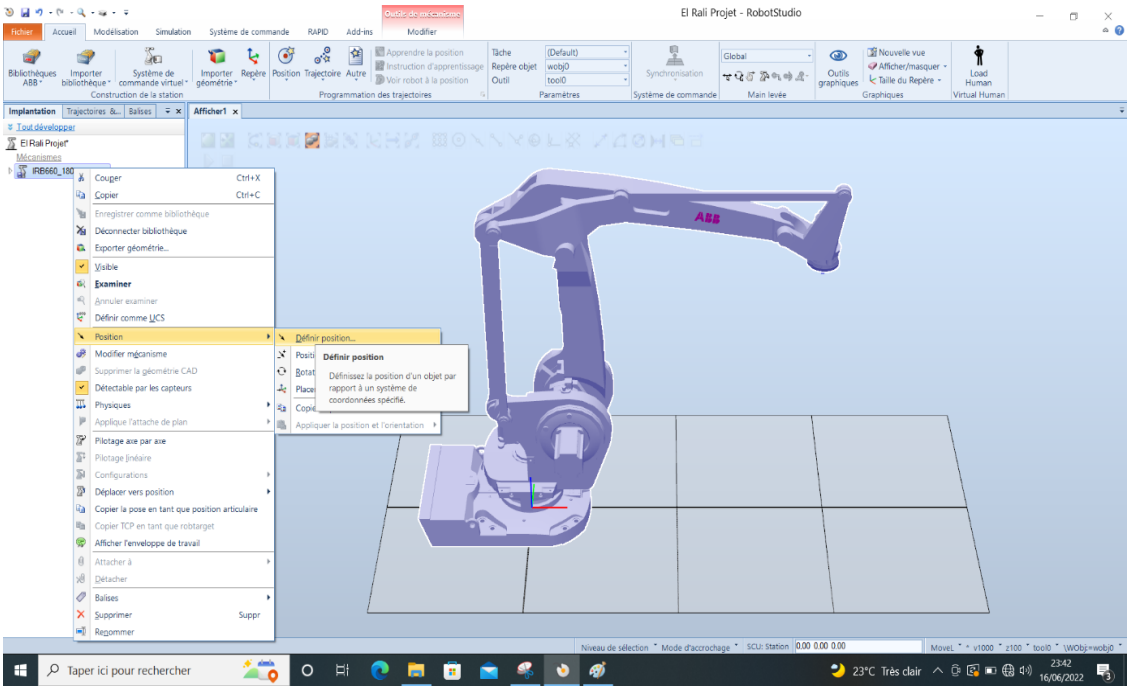


Figure (IV.8) : Définir la position de robot IRB 660

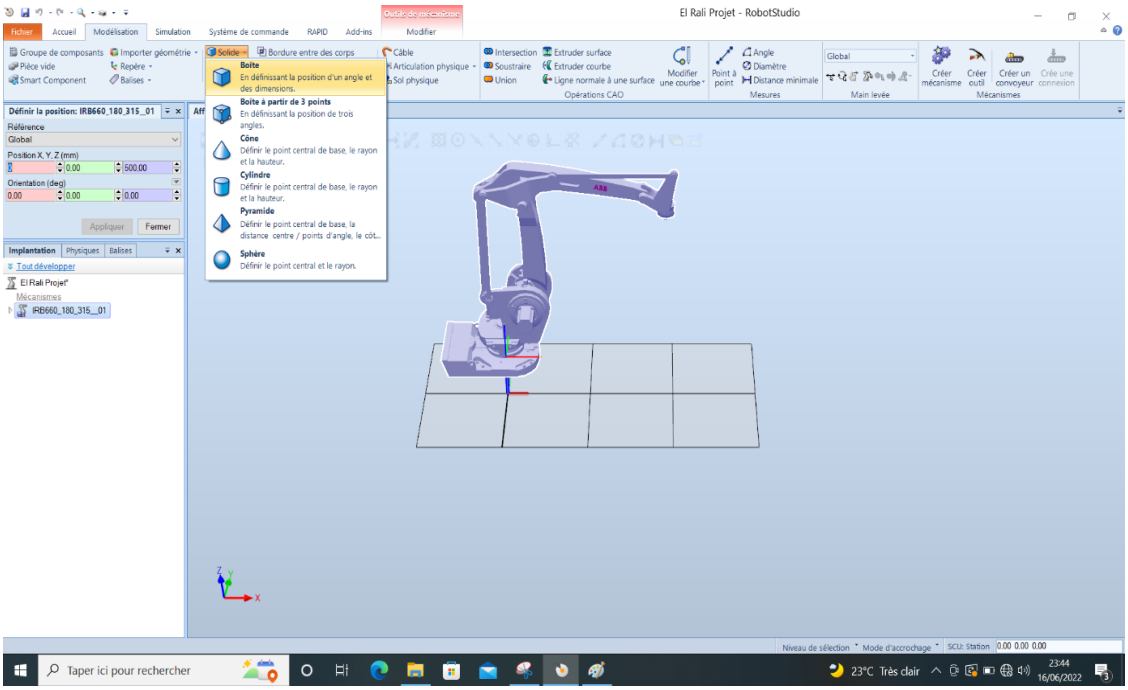


Figure (IV.9) : Ajouté les Coordonnées de robot et une boite

• Créez un système de commande à partir de l'agencement, je choisisse un système de commande existant.

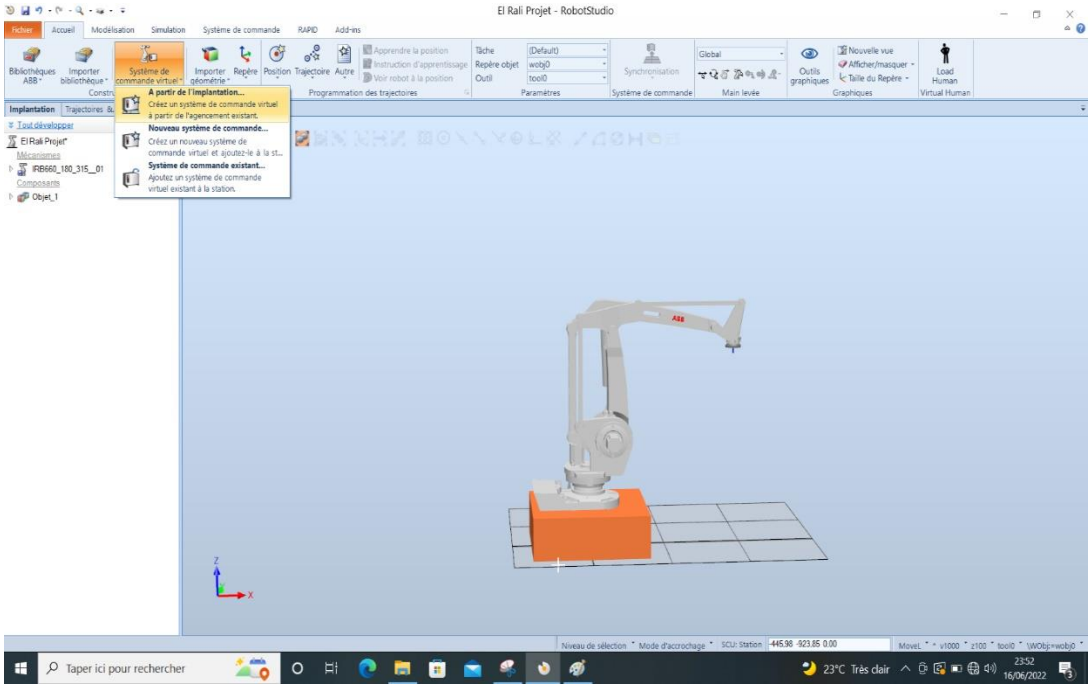


Figure (IV.10) : Créez un système de commande virtuel

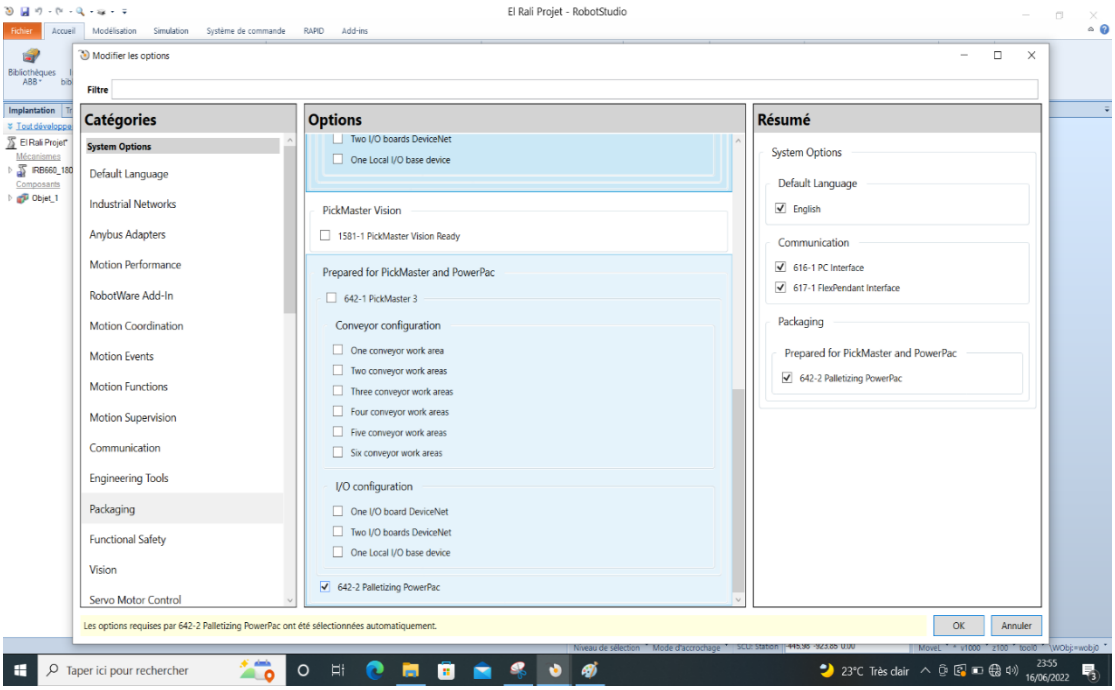


Figure (IV.11) : Cochez sur l'option 642-2 Palletizing PowerPac

• Maintenant on importe l'outil « Flex Gripper 2.0 »

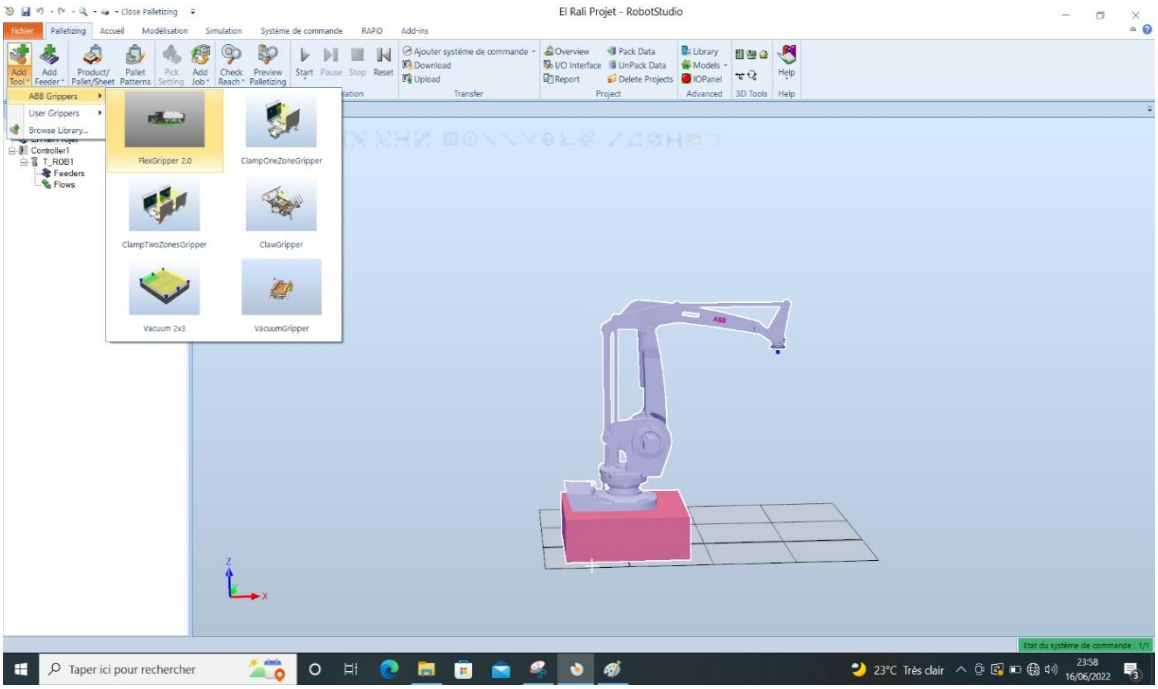


Figure (IV.12) : Importation et attachement du robot avec l'outil

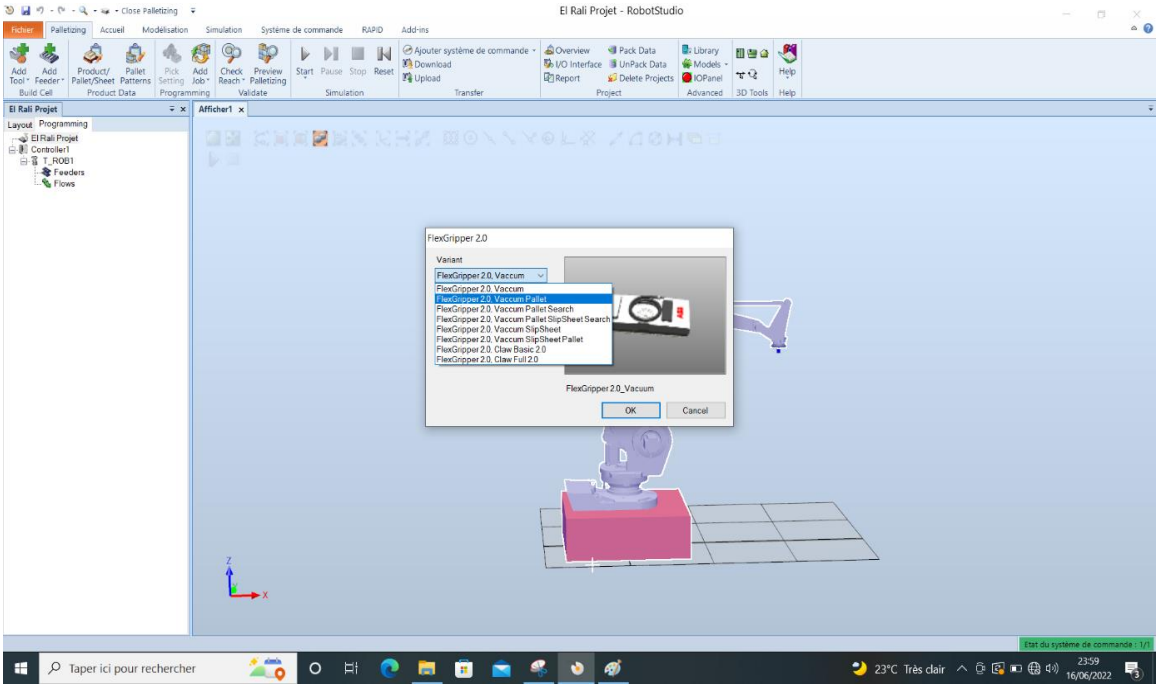


Figure (IV.13) : Ajoute le type de « Flex gripper 2.0 »

• Maintenant on va ajouter les deux convoyeur (conveyor_600), (Outfeeder_1320) et Pallet (Pallet Dispenser 1300_1300_300) et définir la position leur :

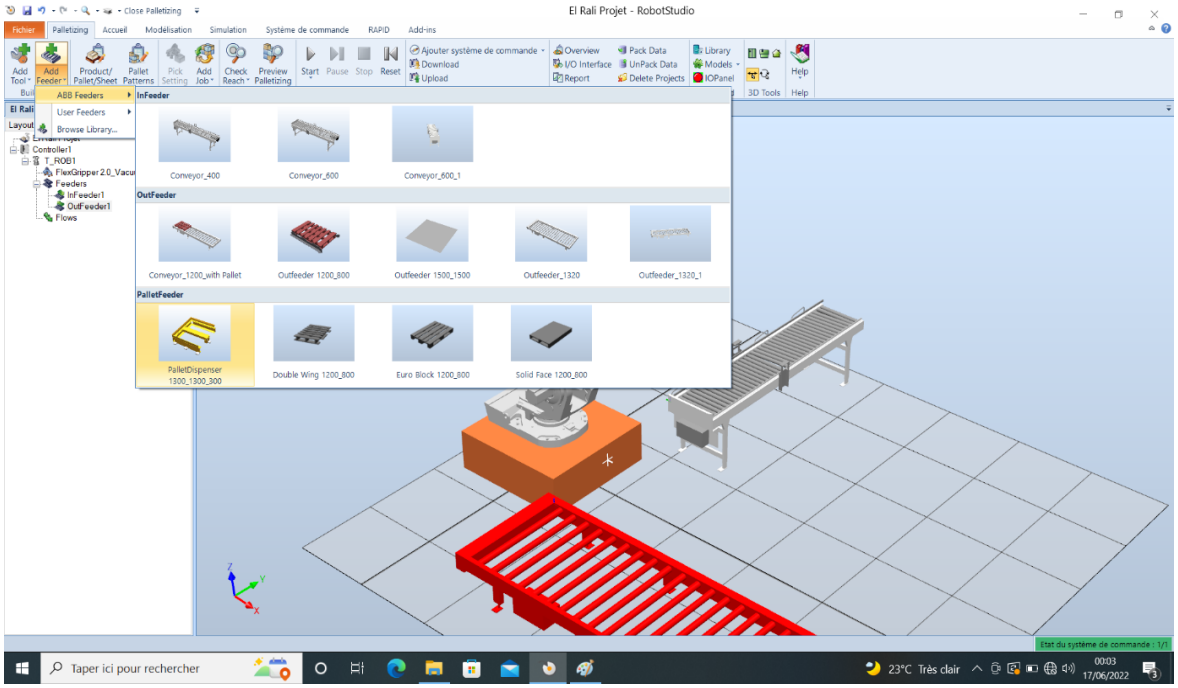


Figure (IV.14) : Ajouter les convoyeurs et le pallet

Abb Feeders	Position (X ,Y,Z)			Orientation (deg)		
Conveyor_600	800	2100	0	0	0	90
Outfeeder_1320	1600	-700	0	0	0	0
Pallet Dispenser 1300_1300_300	-300	-1600	0	0	0	-90

Tableau (IV.2) : Caractéristiques géométriques du convoyeur, palette et évacuer.

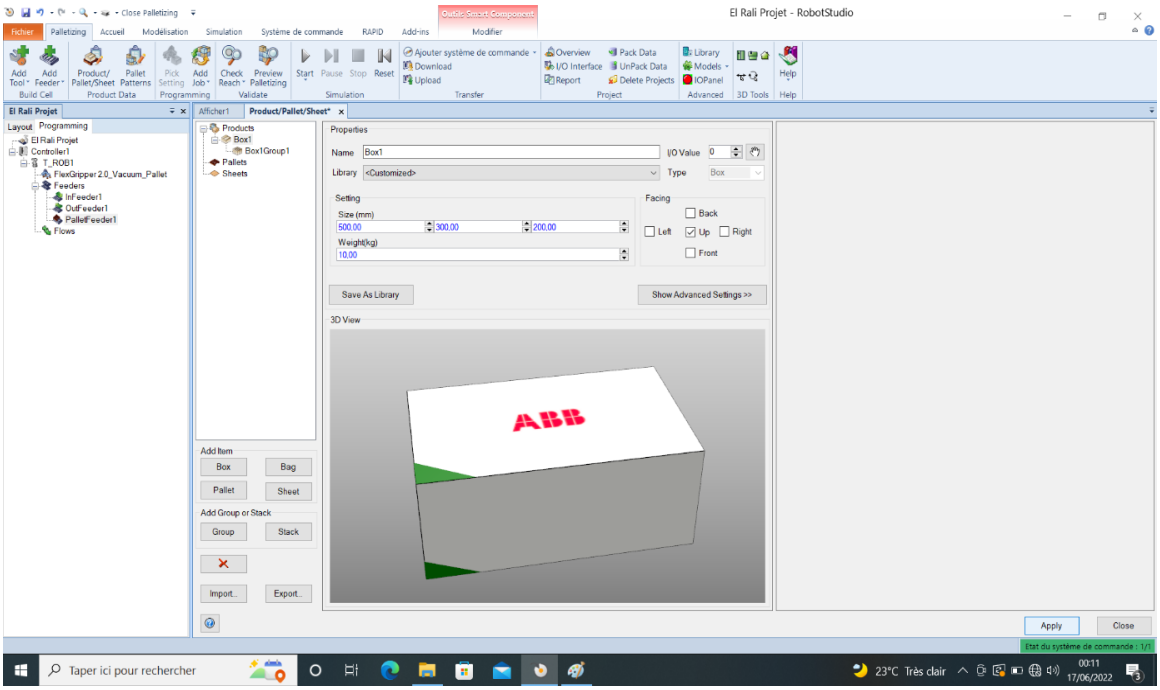


Figure (IV.15) : Définir les propriétés de la palette de produits utilisés

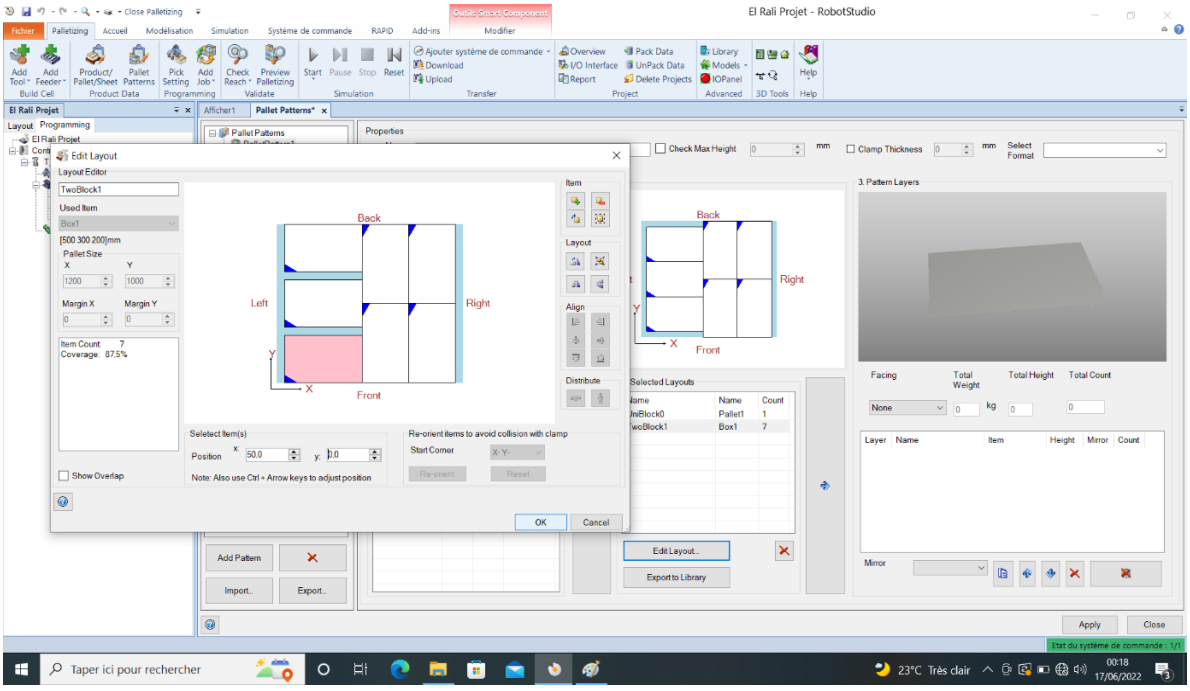


Figure (IV.16) : Modèles de palettes

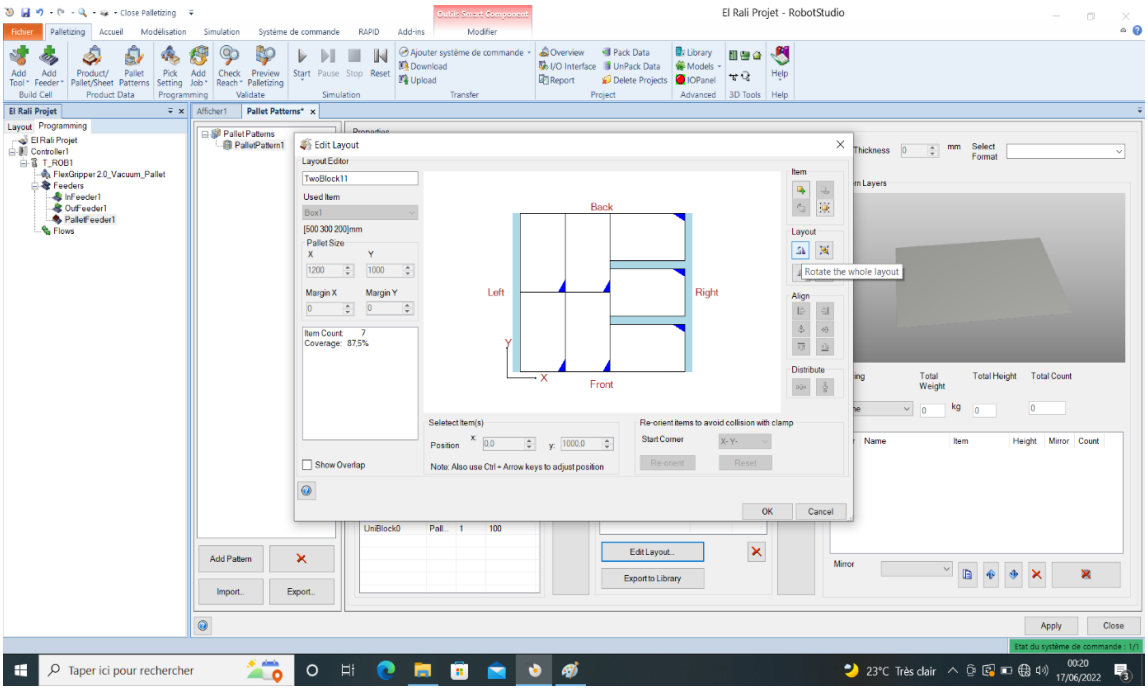


Figure (IV.17) : Modifier la mise en page (faire pivoter toute la mise en page)

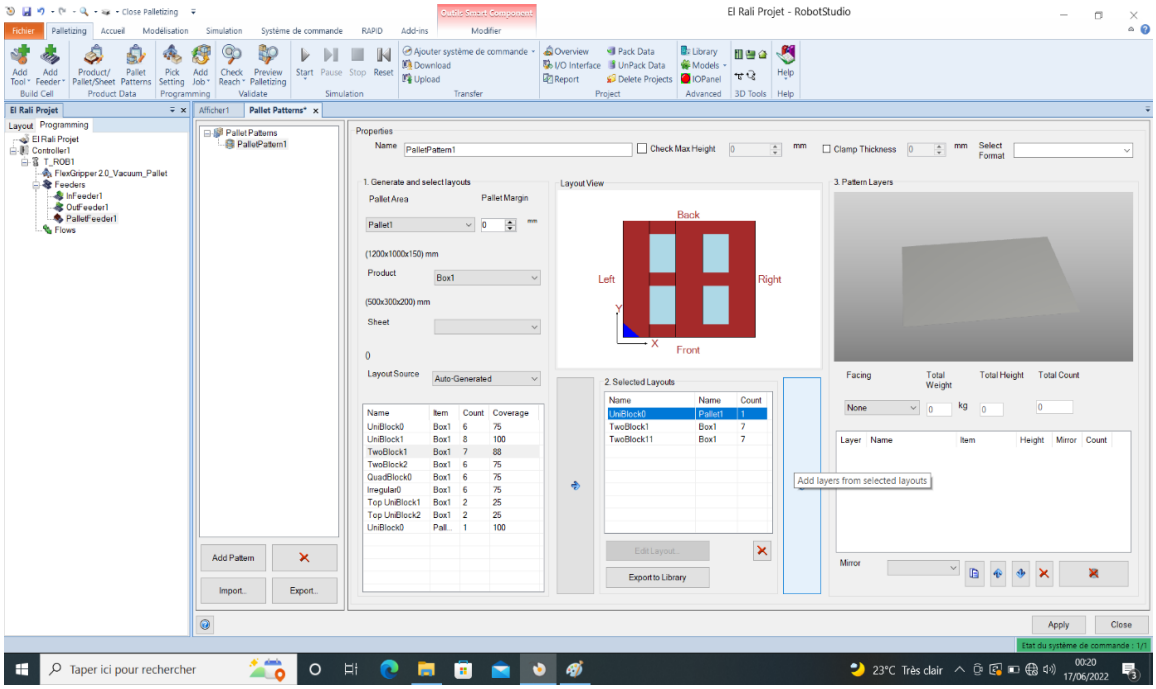


Figure (IV.18) : Ajouter des calques à partir des mises en page sélectionnées

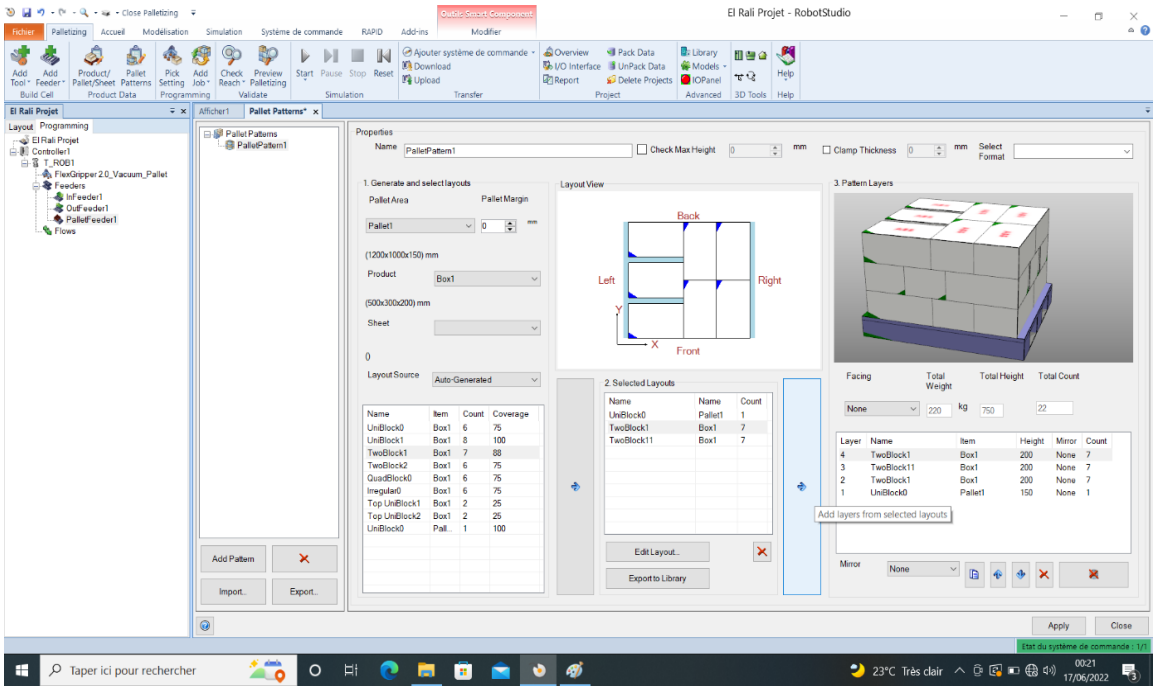


Figure (IV.19) : Terminer l'ajout de calques à partir des mises en page sélectionnées

• Une fois que vous avez terminé d'ajouter des calques à partir des mises en page sélectionnées je clique sur le bouton « Apply »

• On va tout d'abord aller sur Palletizing ► Pick Setting

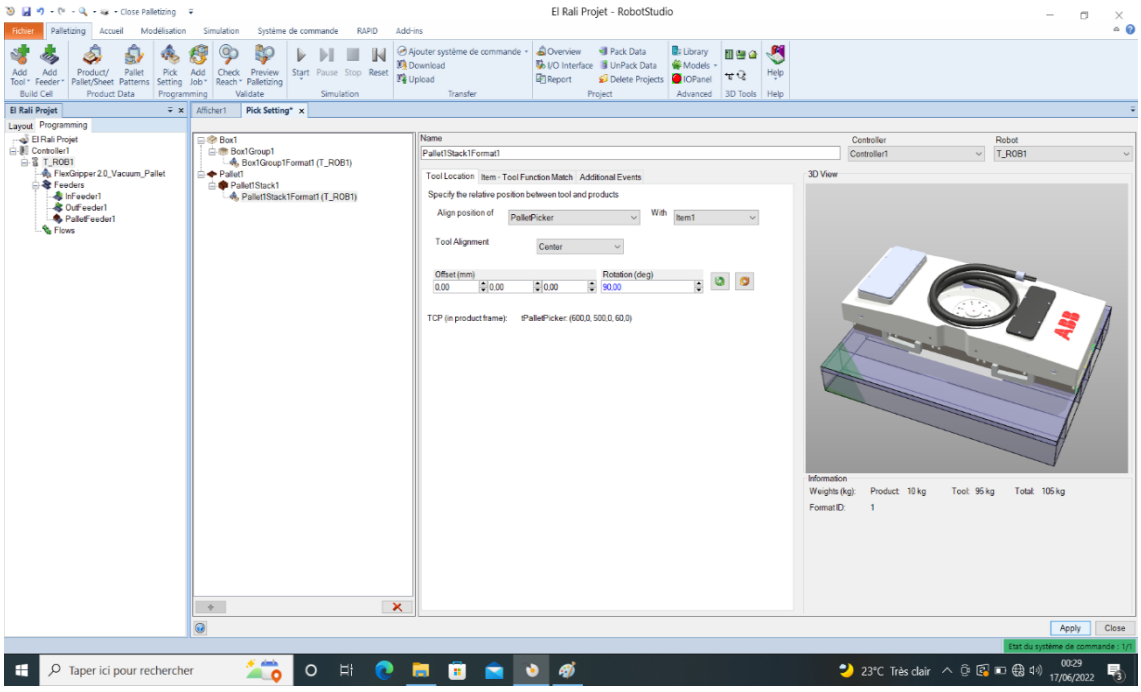


Figure (IV.20) : Ajoute Offset et la rotation.

Pick Setting	Offset			Rotation
Box Group Format (T_ROB1)	0	-75	0	0
Pallet Stack Format (T_ROB1)	0	0	0	90

Tableau (IV.3) : Configuration de la forme des boites et de la palette.

• Et maintenant nous allons sur Palletizing ► Add Job ► Placé les boites et Pallet et clique sur bouton « Next »

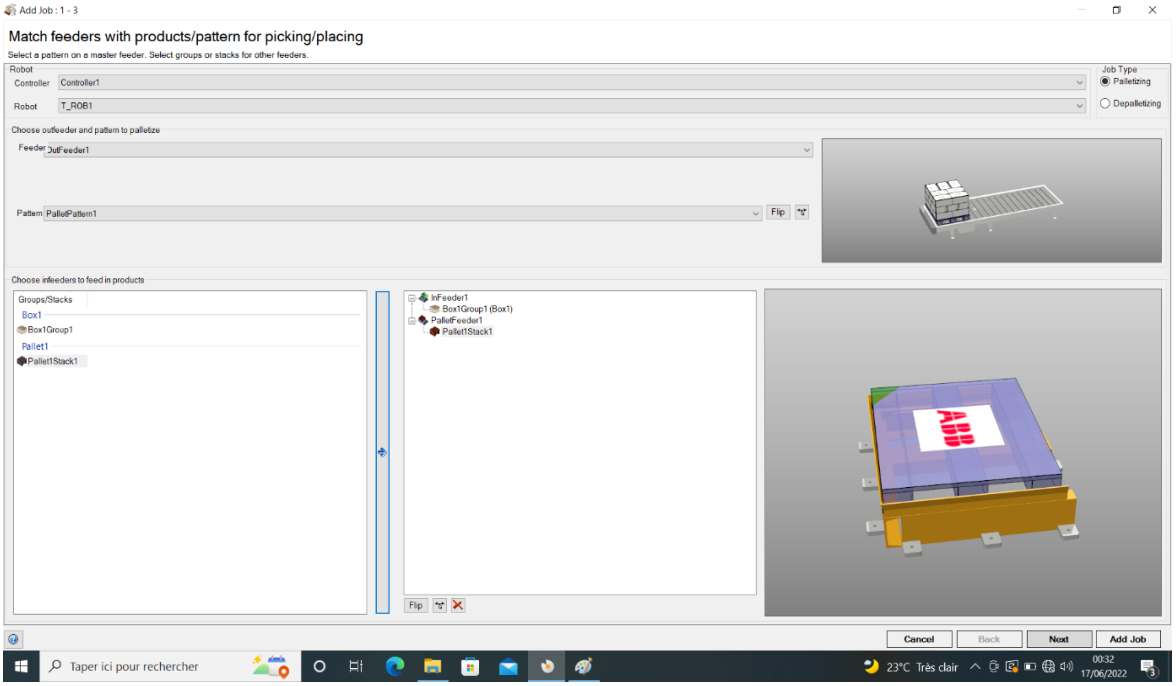


Figure (IV.21) : Placé les « box & Pallet ».

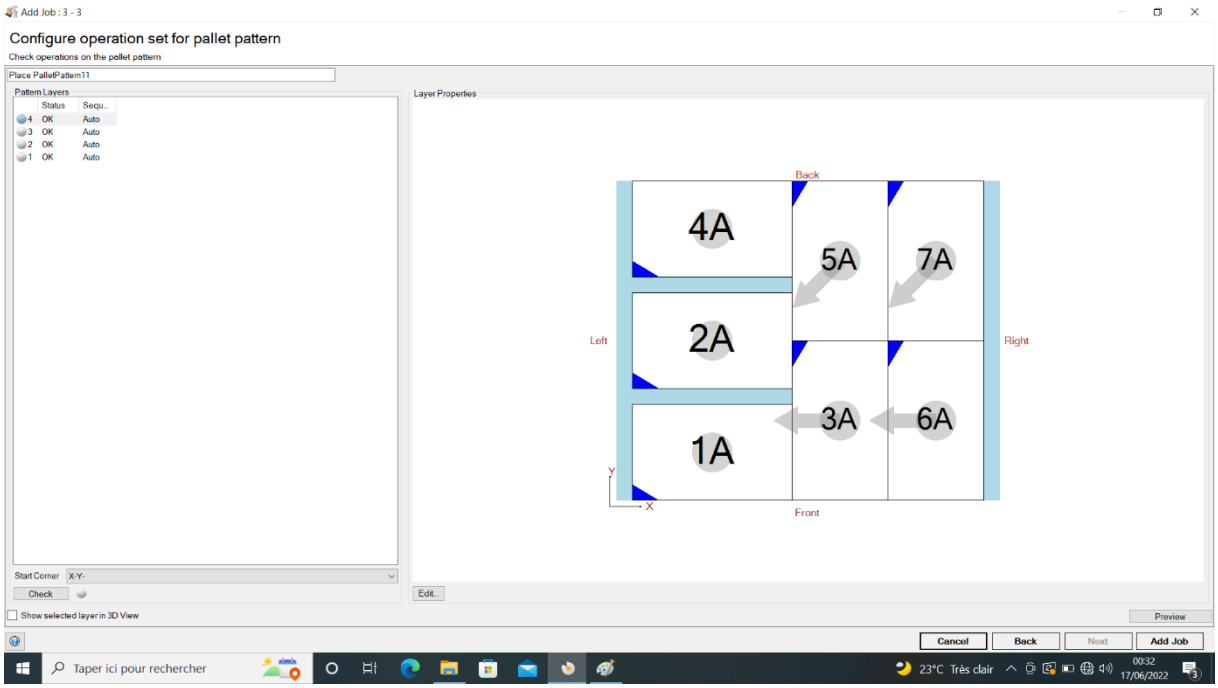


Figure (IV.22) : Configurer le jeu d'opérations pour le modèle de palette.

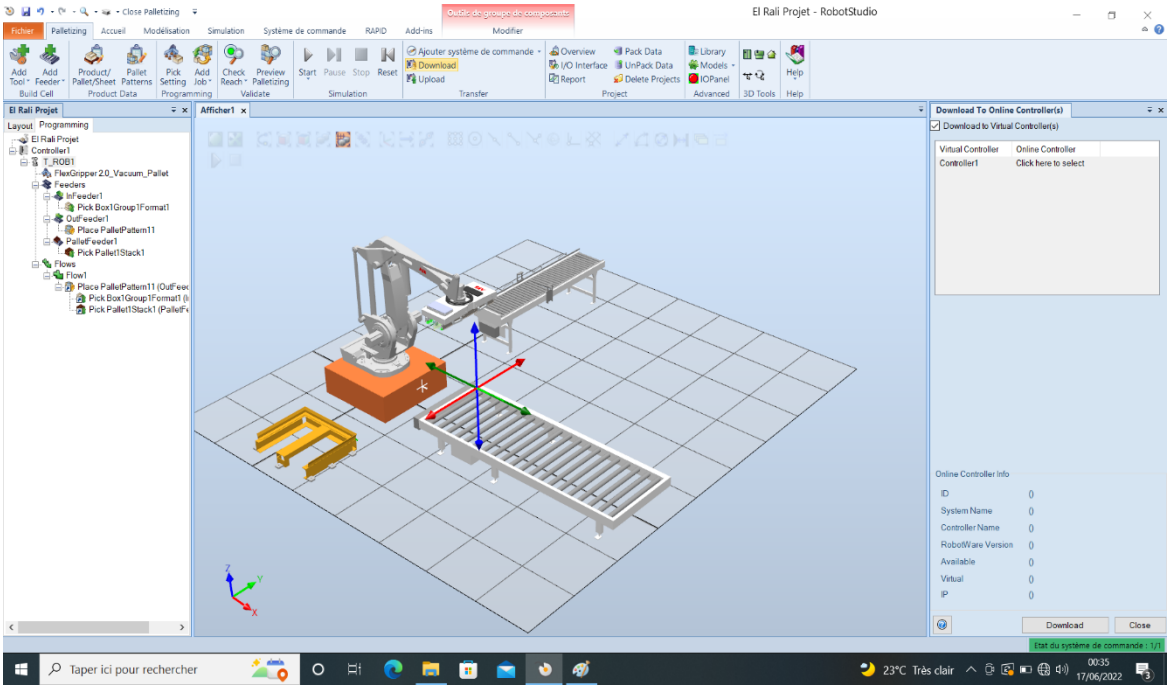


Figure (IV.23) : Télécharger le contrôleur

• Et la fin nous allons sur simulation ► Star ► Quick Star

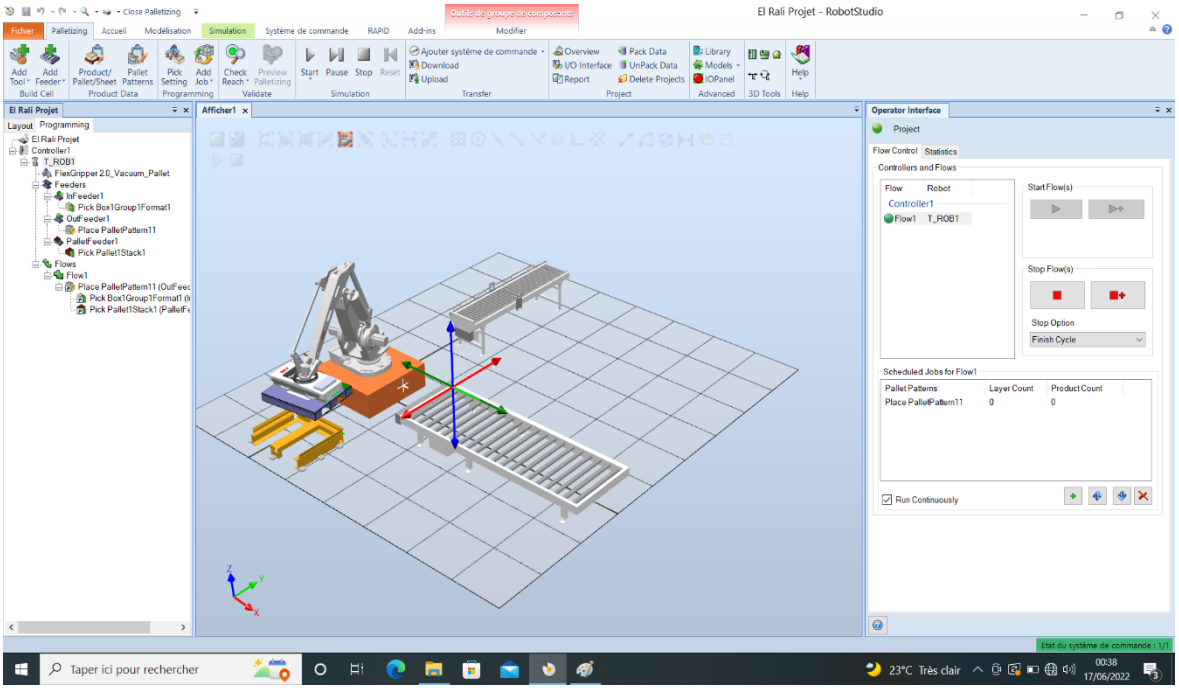


Figure (IV.24) : Le début du processus de simulation.

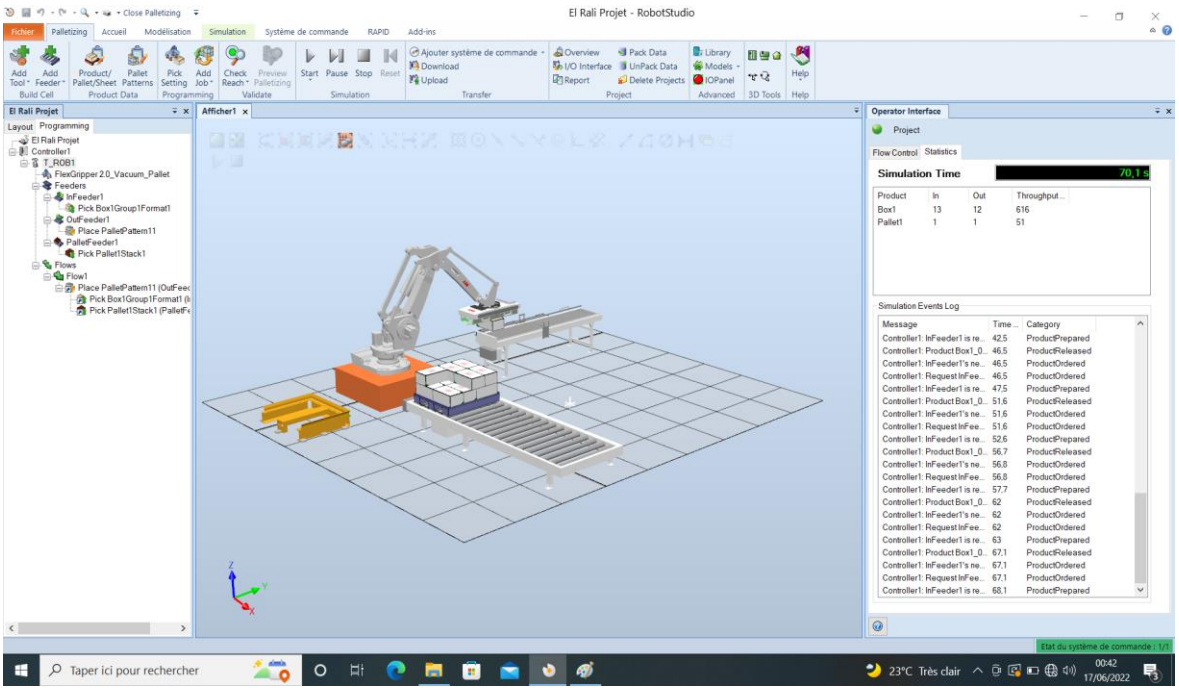


Figure (IV.25) : Fin du processus de simulation.

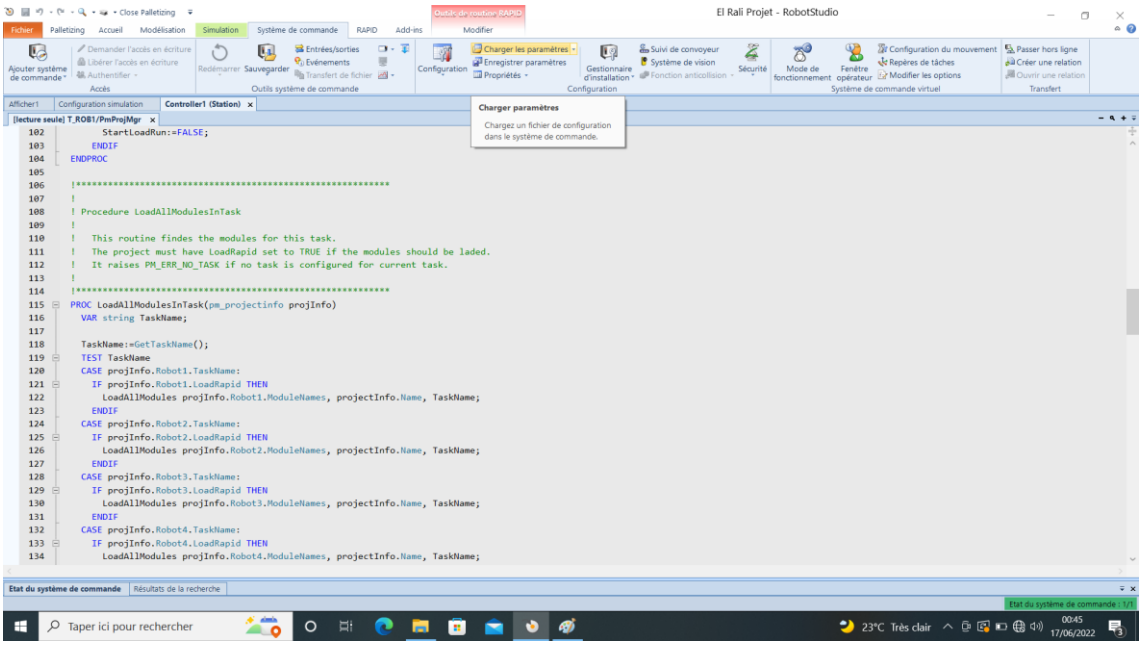


Figure (IV.26) : Le programme avec langage RAPID de la simulation.

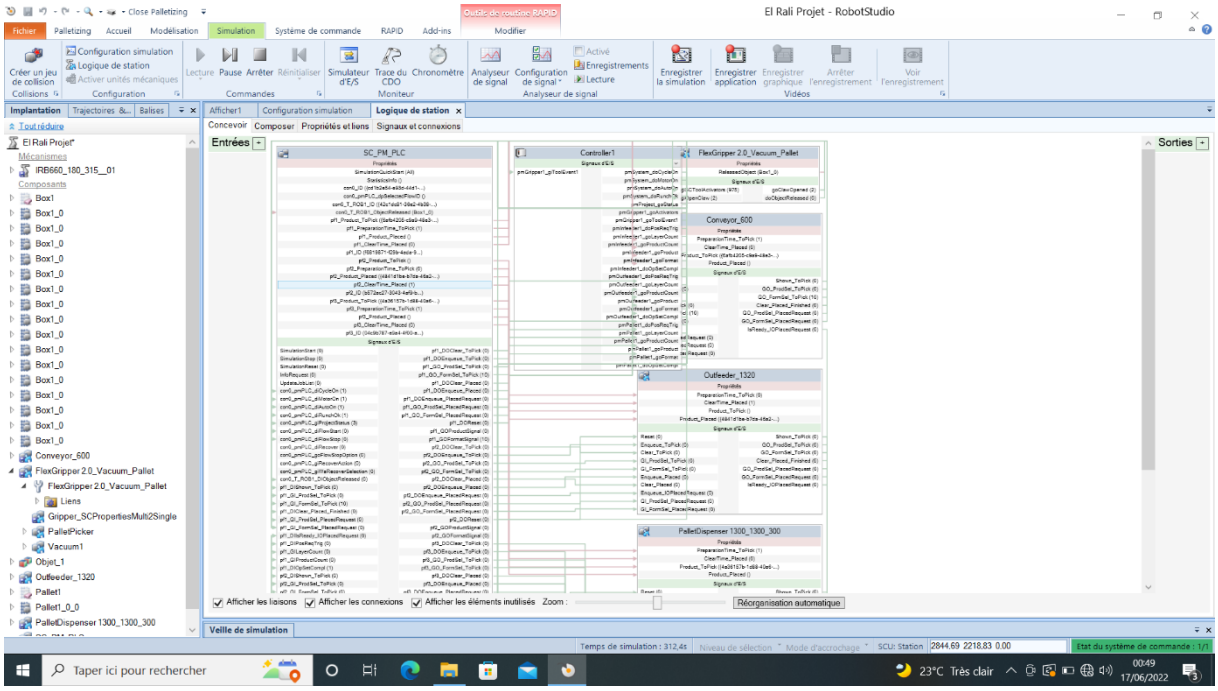


Figure (IV.27) : La logique de la station.

Conclusion

Après un apprentissage approfondi sur l'utilisation de ce logiciel et la liaison avec des différentes bases en automatisme et en informatiques, on a réussi à créer un programme simulant la tâche de palettisation pour déplacer des boites sur la palette dans un convoyeur avec une rentabilité, précision et rapidité remarquable assuré par le logiciel ABB Robot-Studio.

Conclusion Générale

Conclusion générale et perspectives

La robotique industrielle permet l'automatisation de certaines tâches sur votre chaîne de production, vous apportant flexibilité et gain de productivité.
L'intérêt de ce travail porte sur la planification et la programmation des tâches exécutables, aussi de ramener la précision requise et la facilité d'exécution.
Le travail de ce mémoire, c'est inscrit dans une tentative de maitrise de planification et de programmation de ces tâches susceptibles de ramener la précision requise et la facilité d'exécution.
Ce travail nous a montrer l'intérêt de l'outil de programmation CFAO Rob Studio qui offre à son utilisateur la facilité et la richesse de conception et de planification des tâches couramment utilisées en industrie, de la plus simple à la plus complexe, profitant ainsi du repère du superviseur et de la haute qualité graphique en 3D.
Le premier chapitre est consacré à étudier l'état de l'art de la programmation des robots industriels.
Ensuite dans le deuxième chapitre on a essayé de citer les Outils mathématiques pour la planification des taches industrielles.

Dans le troisième chapitre on s'est focalisé sur la programmation graphique des taches avec le logiciel robot Studio.

Enfin dans le quatrième et le dernier chapitre On a déterminé les différentes étapes suivies pour arriver à la simulation d'une cellule robotisée avec le logiciel robot Studio qui offre à son utilisateur la facilité et la richesse de conception et de planification des taches habituellement utilisé dans l'industrie.

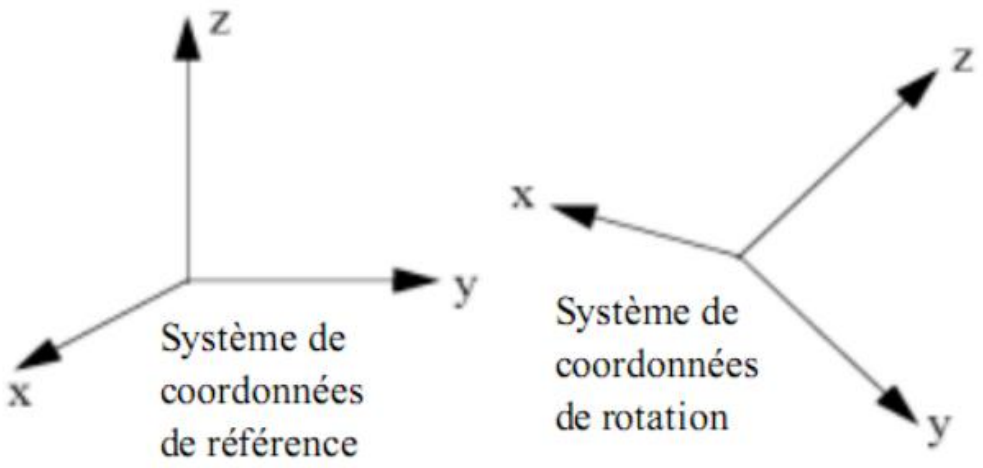
Grâce à ce modeste travail on a encaissé plusieurs connaissances à propos delà robotique, ce projet nous a permis d'enrichir nos propres compétences dans la simulation et le domaine industriel en général.

Comme perspectives nous proposons la poursuite de ce travail avec la création de cellules plus complexes avec plus de robots en coopération, ainsi que l'adjonction de l'intelligence artificielle dans la planification des taches industrielles robotisées.

Annexe

Le quaternion

Dans le système du robot, le paramètre d'orientation est généralement décrit sous forme d'un quaternion qui comporte donc quatre éléments : q1 , q2 , q3, q4. L'orientation d'un système de coordonnées (par exemple celui d'un outil) peut être décrite par une matrice de rotation qui décrit la direction des axes du système de coordonnées par rapport à un système de référence.



Les axes du système de coordonnées rotatif (x,y,z) sont des vecteurs qui peuvent être exprimés dans le système de coordonnées de référence comme suit: $x = (x1, x2, x3)$ (A.1) $y = (y1, y2, y3)$ (A.2) $z = (z1, z2, z3)$ (A.3) Ceci signifie que la composante x du vecteur x dans le système de coordonnées de référence sera x1 , la composante y sera x2 , etc. Ces trois vecteurs peuvent être réunis dans une matrice, une matrice de rotation, où chaque vecteur formeunecolonne :

$$\begin{pmatrix} x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \\ x_3 & y_3 & z_3 \end{pmatrix} \qquad \text{(A.4)}$$

Un quaternion est tout simplement une façon plus concise de décrire cette matrice de rotation; les quaternions sont calculés en se basant sur les éléments de la matrice de rotation :

$$q1 = \frac{\sqrt{x_1+y_2+z_3+1}}{2} \quad (A.5)$$

$$q2 = \frac{\sqrt{x_1-y_2-z_3+1}}{2} \quad (A.6)$$

$$q3 = \frac{\sqrt{y_2-x_1-z_3+1}}{2} \quad (A.7)$$

$$q4 = \frac{\sqrt{z_3-x_1-y_2+1}}{2} \quad (A.8)$$

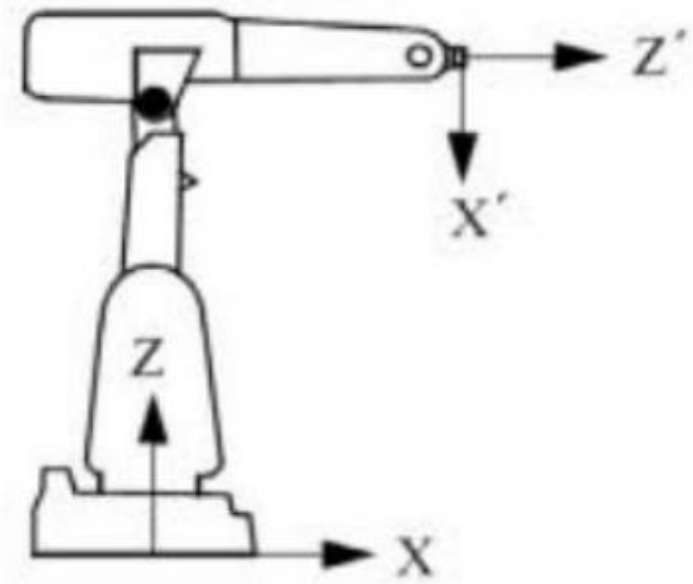
On a :

$$\text{sign}(q_2) = \text{sign}(y_3 - z_2) \quad (A.9)$$

$$\text{sign}(q_3) = \text{sign}(z_1 - x_3) \quad (A.10)$$

$$\text{sign}(q_4) = \text{sign}(x_2 - y_1) \quad (A.11)$$

Un outil est orienté de façon à ce que son axe Z soit tourné directement vers l'avant (dans la même direction que l'axe X du système de coordonnées de base). L'axe Y de l'outil correspond à l'axe Y du système de coordonnées de base.



L'orientation de l'outil dans la position programmée correspond normalement au système de coordonnées de la pièce utilisé. Si les coordonnées de base sont celles du système de coordonnées universelles, l'orientation correspond au système de coordonnées de base et la relation entre les axes sera comme suit :

$$x' = -z = (0,0,-1)$$

$$y' = y = \ (0,1,0)$$

$$z' = x = (1,0,0)$$

Ce qui correspond à la matrice de rotation suivante :

$$\begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ -1 & 0 & 0 \end{pmatrix}$$

La matrice de rotation fournit un quaternion correspondant :

$$q1 = \frac{\sqrt{0+1+0+1}}{2} = \frac{\sqrt{2}}{2} = 0.707$$

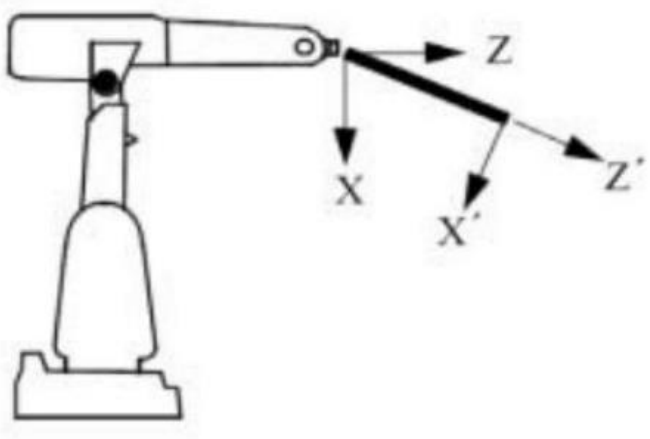
$$q2 = \frac{\sqrt{0-1-0+1}}{2} = 0$$

$$q3 = \frac{\sqrt{-1-0-0+1}}{2} = \frac{\sqrt{2}}{2} = 0.707$$

$$q4 = \frac{\sqrt{0-0-1+1}}{2} = 0$$

$$sign(q3) = sign(1+1) = +$$

Si la direction de l'outil fait l'objet d'une rotation de 30° autour des axes X et Z par rapport au système de coordonnées du poignet, l'orientation de l'outil est définie dans les données de l'outil par la relation suivante entre les axes :



$$x' = (cos30°, 0, -sin30°)$$

$$y' = (O, 1,0)$$

$$z' = (sin30°, 0, cos30°)$$

Ce qui correspond à la matrice de rotation suivante :

$$\begin{pmatrix} \cos 30° & 0 & \sin 30° \\ 0 & 1 & 0 \\ -\sin 30° & 0 & \cos 30° \end{pmatrix}$$

La matrice de rotation fournit un quaternion correspondant :

$$q1 = \frac{\sqrt{cos30° + 1 + cos30° + 1}}{2} = 0.965926$$

$$q2 = \frac{\sqrt{cos30° - 1 - cos30° + 1}}{2} = 0$$

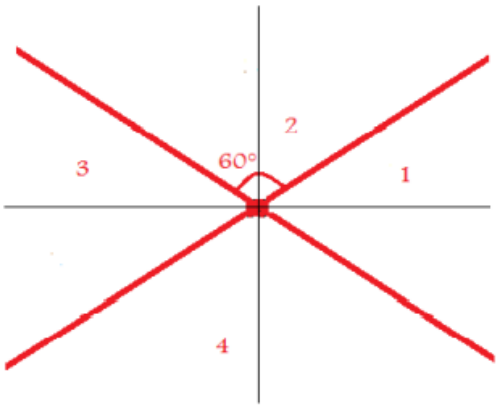
$$q3 = \frac{\sqrt{1 - cos30° - cos30° + 1}}{2} = 0.258819$$

$$q4 = \frac{\sqrt{cos30° - COS30° - 1 + 1}}{2} = 0$$

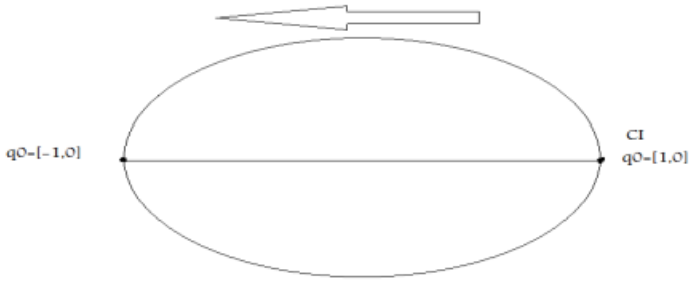
$$sign(q3) = sign(sin30° + sin30°) = +$$

Dans les autres cas de développement, il faut transformer le quaternion à la matrice de rotation pour calculer l'orientation dans un système de référence. La transformation s'établit comme suit :

$$R = \begin{pmatrix} q_0^2 + q_1^2 - q_2^2 - q_3^2 & 2(q_1q_2 - q_0q_3) & 2(q_1q_3 + q_0q_2) \\ 2(q_1q_2 + q_0q_3) & q_0^2 - q_1^2 + q_2^2 - q_3^2 & 2(q_2q_3 - q_0q_1) \\ 2(q_1q_3 - q_0q_2) & 2(q_2q_3 + q_0q_1) & q_0^2 - q_1^2 - q_2^2 + q_3^2 \end{pmatrix} \quad \text{(A.12)}$$



L'inconvénient des représentations (cadrons) (1) et (2) c'est qu'elles peuvent contenir des singularités dans certaines rotation de 90° (cos Π/2 = 0 et sin 0 = 0) ce qui n'est pas le cas pour (3), par contre son inconvénient est la non robustesse numérique (elle présente des problèmes numériques de calcule algorithme instable) (sensibilité au erreurs). Ce n'est pas le cas pour les quaternions, on ne trouve pas des problèmes de singularité et ils montrent une robustesse numérique, leur seul défaut est la non unicité de la condition initial et la présence d'une discontinuité lorsqu'on veut passer d'une condition initiale vers une position qui se trouve de l'autre coté comme le montre la figure suivante :



Un quaternion q = [q_0, q_1, q_2,q_3] , sa moyenne $\|q\| = \sqrt{q_0^2 + q_1^2 + q_2^2 + q_3^2} = 1$

Bibliographie

[1] ISO, Robots et composants robotiques {Vocabulaire, ISO 8373, 2012}.

[2] Université ABOU-BEKR BELKAID - TLEMCEN FACULTE DES SCIENCES DE L'INGENIEUR DEPARTEMENT D'AUTOMATIQUE, Modélisation des robots, Master automatique Mme S. BORSALI.

[3] Notes de cours GPA546 Ilian Bonev, ing. Yanick Noiseux, ing. 21 septembre 2014.

[4] Pollard Jr., W.L.G., < Spray painting machine >, brevet américain No. 2 213 108, déposé le 29 octobre 1934, accepté le 27 aout 1940.

[5] Roselund, H.A., < Means for moving spray guns or other devices through predetermined paths >, brevet américain No. 2 344 108, déposé le 17 aout 1939, accepté le 14 mars 1944.

[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, Informationsf• orlaget, 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] Modélisation, identification et commande des robots WISSAMA Khalil/ Etienne Dombre.

[11] Practical session 5 : rapid programming ARTURO GIL Aparcicio.