Ingénierie des systèmes cyber-physiques

R2.10 - Introduction à la robotique industrielle

Support de cours

Thomas Fromentèze

iut du Limousin
INSTITUT UNIVERSITAIRE DE TECHNOLOGIE

1 Définitions et applications

1.1 Contexte historique

Les robots industriels permettent l'automatisation de tâches de production. Ils fonctionnent grâce au pilotage de parties mécaniques à l'aide de systèmes de commandes programmables associés à des actionneurs et des capteurs.

La seconde guerre mondiale marque le point de départ de la robotique avec la naissance de l'informatique puis le développement des premiers transistors. On trouve malgré tout des exemples de machines programmables bien plus anciennes, utilisant notamment des cartes et rubans perforés depuis le milieu du 18^{e} siècle. Les métiers à tisser Jacquard permettaient par exemple d'automatiser la fabrication de textiles présentant des motifs complexes.

FIGURE 1 – Métier Jacquard au musée des Arts et Métiers de Paris

Les cartes perforées, aussi appliquées à l'automatisation d'instruments de musique, ont par la suite permis de concevoir les premières machines à calculer programmables, ancêtres des ordinateurs.

Depuis la deuxième partie du 20^{e} siècle, les robots ont progressivement été adaptés au domaine industriel, implémentés en premier lieu dans les milieux à risques. Ils ont ensuite été mis en place pour remplacer la main d'œuvre humaine sur de nombreuses tâches fastidieuses et répétitives, impliquant des actions particulièrement physiques ou pouvant engendrer des troubles musculo-squelettiques.

1.2 Applications et marché industriel

Tel qu'illustré en figure 2, les robots sont déployés pour l'automatisation de nombreux procédés industriels, tels que la peinture, le soudage, l'usinage, l'assemblage et la palettisation (aussi appelé le *pick and place* – déplacement et positionnement précis d'objets d'un endroit à un autre pour du conditionnement et du stockage).

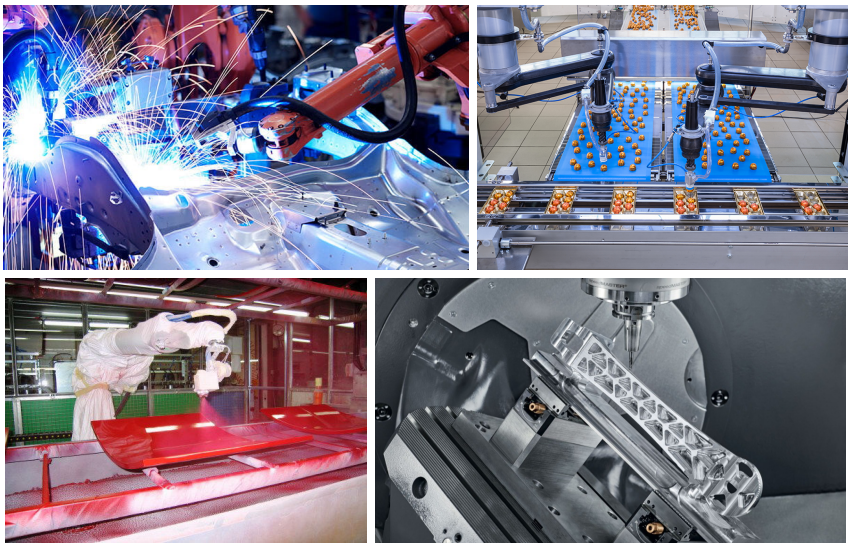


FIGURE 2 – Applications de la robotique à la production industrielle

Parmi les principaux fabricants, ordonnés par chiffre d'affaire en 2020 :

- Fanuc (Japon)
- Yaskawa Electric (Japon)
- ABB (Suisse)
- KUKA (Allemagne)
- Mistubishi Electric (Japon)
- C.R.ONSRUD/Fanuc (USA)
- Epson Robots (Japon)
- Omron Adept (Japon)
- Stäubli (Suisse)
- Kawasaki Robotics (Japon)
- Nachi Robotics (Japon)
- Denso Robotics (Japon)
- (Danemark) Universal Robots
- Siasun (Chine)
- Techman Robot (Taïwan)
- Gudel (Suisse)

Quelques données sur l'évolution de la robotique dans le monde :

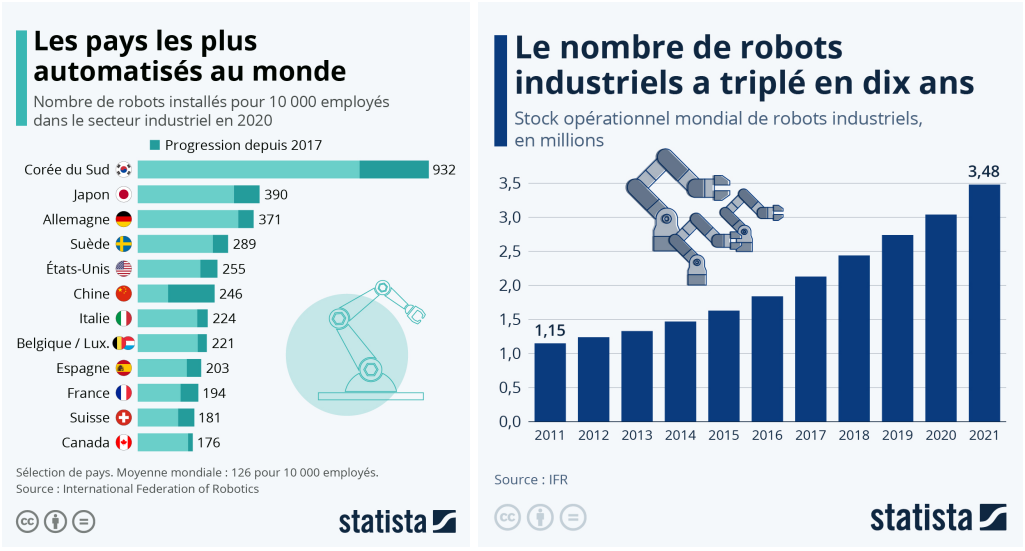


FIGURE 3 – Source : https ://fr.statista.com/

D'après l'*Usine Nouvelle* (média spécialisé), adapté de *Research and Markets* (cabinet de conseil) :

> **2021 : année record pour le marché** - *Le marché affiche une croissance record de 31% selon la Fédération Internationale de Robotique (IFR). 517000 nouveaux robots ont été livrés à travers le monde. En France, l'augmentation du nombre de robots est de 11%. Évalué à 15 milliards de dollars en 2022, le marché de la robotique industrielle pourrait doubler en 5 ans et atteindre 30,8 milliards de dollars en 2027.*

1.3 Composition

Pour assurer le fonctionnement d'un robot, on retrouve généralement les éléments suivants :

- **Un robot multi-axes** : décrit plus en détail dans les parties suivantes.
- **Une armoire de commande** : contient tous les organes de pilotage des actionneurs du robot (on parle généralement de drivers), ainsi que des circuits programmables permettant de contrôler ces derniers et d'interroger des capteurs (aussi appelé contrôleurs).
- **Un boîtier de commande** : interface homme-machine permettant d'envoyer des instructions et de lire des informations.
- **Un ordinateur** : nécessaire au moment du déploiement du robot et pour sa maintenance. Une fois programmé, le contrôleur est autonome.

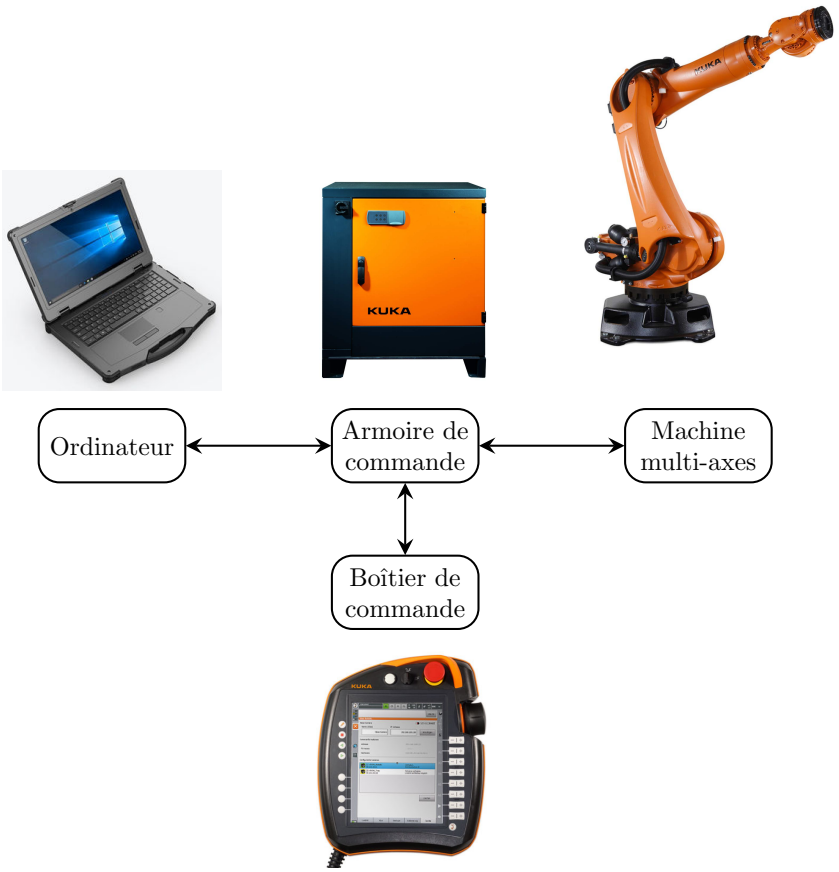


FIGURE 4 – Éléments requis pour la programmation et le fonctionnement d'un robot industriel

Il existe de nombreux types de robots industriels qui seront décrits dans la suite de ce cours. On les étudie généralement à l'aide d'une décomposition en un certain nombre de corps rigides (aussi appelés segments) reliés par des axes (aussi appelés articulations). Dans le cas des robots articulés, on numérote les axes en partant de la base (le socle) qui sert de repère de référence au robot pour son positionnement dans l'espace.

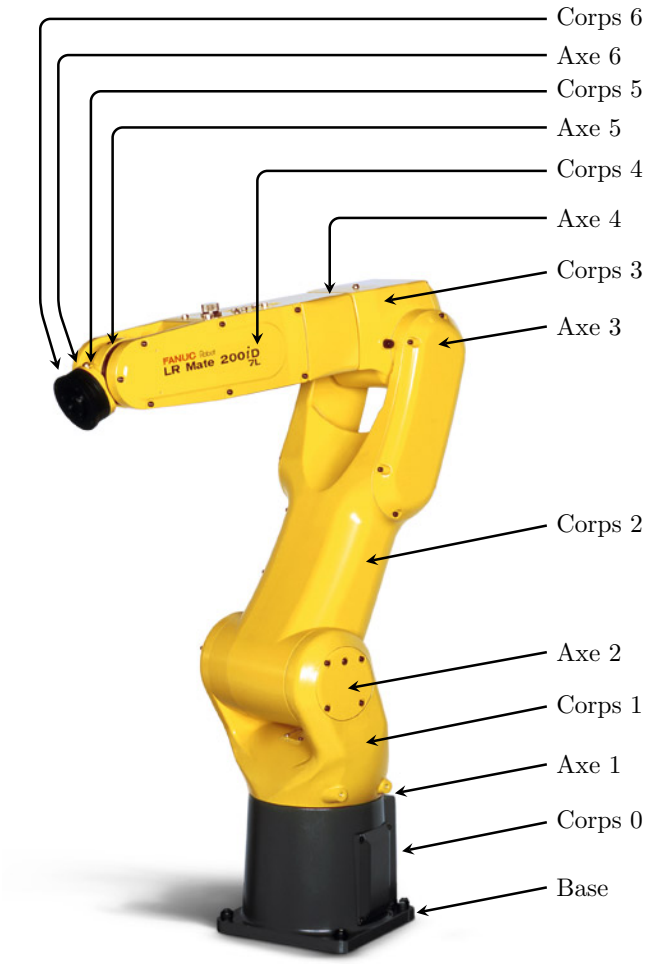
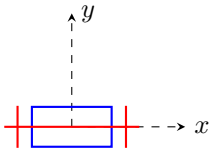


FIGURE 5 – Anatomie d'un robot articulé 6 axes

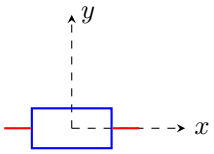
Pour étudier les capacités d'un robot à travailler dans l'espace, il est nécessaire d'aborder plus en détail les types d'articulations qu'il est possible d'exploiter, définissant par ailleurs les degrés de liberté du robot. On retrouve essentiellement deux types d'articulations :

Les articulations rotoïdes : Il s'agit d'une liaison que l'on note **R** de type **pivot** . On définit un axe à l'intersection des segments autour duquel est réalisé une rotation.



	Translation	Rotation
	0	R_x
	0	0
	0	0

Les articulations prismatiques : Il s'agit d'une liaison de type **glissière** que l'on note **P**. On réalise ici la translation d'un corps par rapport à l'autre selon un axe commun.



	Translation	Rotation
	T_x	0
	0	0
	0	0

En pratique, la majorité des articulations sont choisies de type rotoïdes parce que les actionneurs électriques sont essentiellement des moteurs. Il est plus délicat de réaliser des mouvements de translation, pour lesquels on a tendance à convertir des rotations en déplacement linéaire à l'aide de liaisons helicoïdales.

Les articulations rotoïdes ou prismatiques sont associées à un unique degré de mobilité. Pour rappel, cette notion dépend de la liaison entre deux solides indéformables et définit le nombre de paramètres indépendants permettant de décrire lors d'un déplacement la position et l'orientation de l'un par rapport à l'autre.

1.4 Représentation

Pour une étude de nos robots industriels, nous allons avoir besoin de représenter ces derniers de façon simplifiée, en prenant en compte les différents corps et axes les composants. Il est possible d'utiliser pour cela des schémas cinématiques ou des représentations schématiques simplifiées :

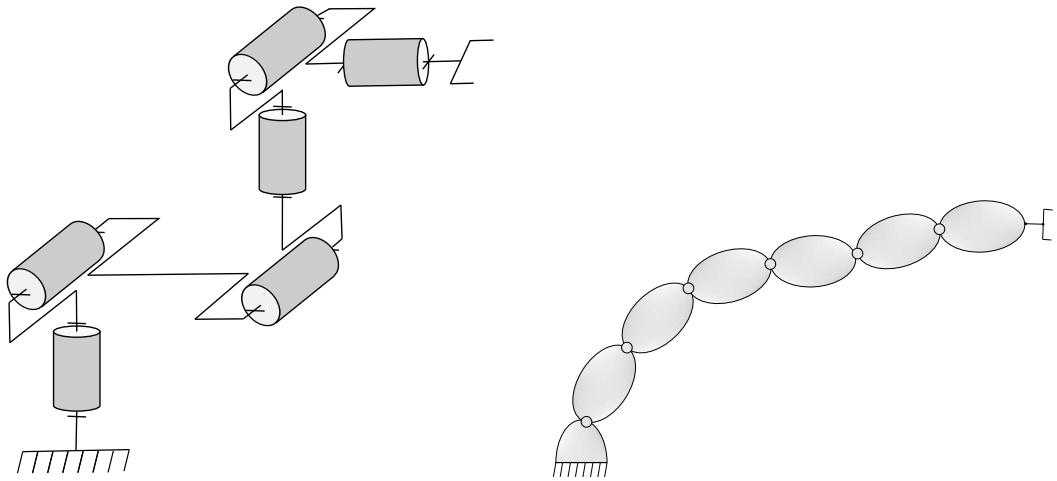


FIGURE 6 – Gauche : Schéma cinématique d'un bras robot - Droite : Représentation simplifiée

1.5 Erreur de positionnement

La problématique du positionnement de l'organe terminal d'un robot (préhenseur, outil ou capteur) est centrale dans ces activités, puisqu'elle influence la qualité des opérations réalisées. La figure suivante permet d'illustrer les notions centrales d'**exactitude** et de **précision** :

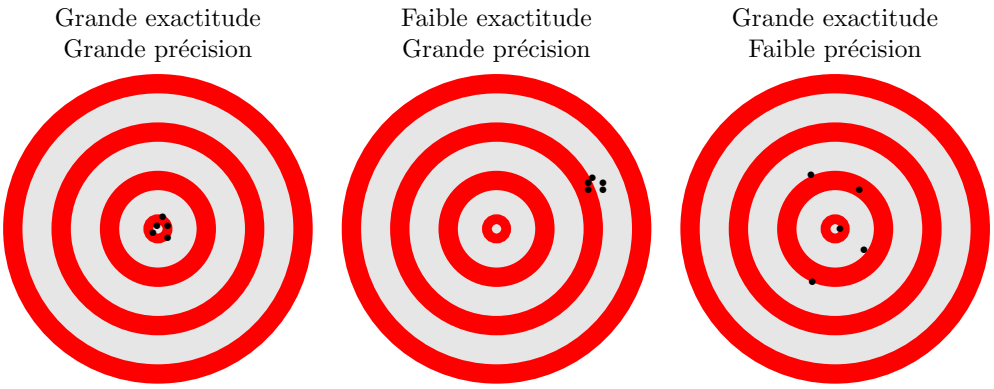


FIGURE 7 – Différentes distributions de points montrant des erreurs de positionnement par rapport à un objectif placé au centre de la cible.

La définition des notions d'exactitude et de répétabilité des robots est aujourd'hui liée à la norme ISO 9283 : "Robots manipulateurs industriels - Critères de performance et méthodes d'essai correspondantes", définie par l'Organisation Internationale de Normalisation. Ces valeurs sont déterminées dans le cadre d'essais composés de 30 positionnement dans 5 configurations, incluant un traitement statistique rigoureux des données mesurées. Une illustration simplifiée de ces notions est présentée dans la figure suivante.

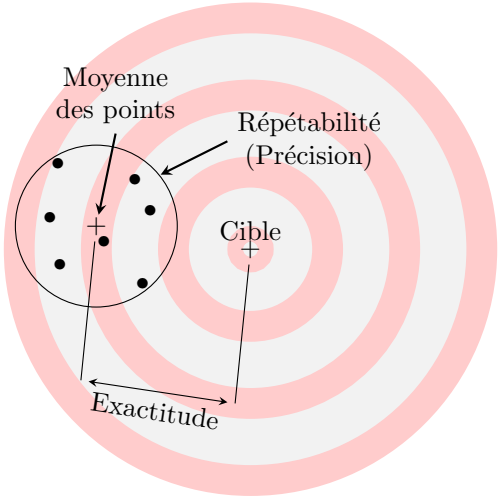


FIGURE 8 – Illustration des notions de précision et de répétabilité

2 Quelques types de robots industriels

Il existe de nombreux types de robots industriels classés en fonction de leurs configurations. Sans avoir pour ambition d'être exhaustif, on étudie dans cette section les types les plus couramment rencontrés.

2.1 Robots cartésiens

Définition :

Robot capable de déplacement d'un outil selon les axes cartésiens.
Constitué d'articulations prismatiques offrant deux ou trois degrés de libertés en translation, on parle de robot PPP (P = Pristmatique).

Avantages

- Commande et modélisation simplifiées
- Grands espaces de travail
- Compatible avec des charges lourdes (>100kg pour certains modèles)
- Vitesse/accélération importantes (pour certains modèles)

Inconvénients

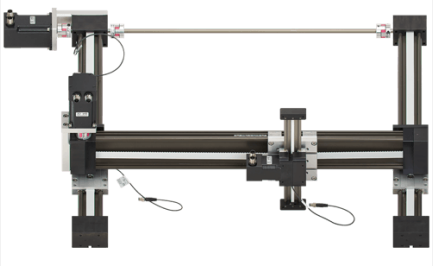
- Encombrement au sol important
- Trois degrés de liberté
- Axes du robot alignés avec ceux de la zone de travail

Exemple de caractéristiques :

Portique de découpe au laser Kuka KR70LP

Charge maximale : 70 kg
Zone de travail max : 30m × 3m × 1m
Répétabilité : 0.02 mm
Vitesse sur chaque axe : 120m/min
Accélération sur chaque axe : 8m/s^2

Positionneur cartésien Igus



Charge maximale : 2.5 kg
Zone de travail : 500mm × 500mm × 100mm
Répétabilité : 0.8 mm
Vitesse sur chaque axe : 0.5m/min
Accélération sur chaque axe : 1.5m/s^2

Exemples d'applications :

- Usinage par commande numérique (ex : fraiseuse CNC)
- Impression 3D
- Palettisation/conditionnement
- Soudure
- Découpage
- Collage

2.2 Robots SCARA

Définition :

Son nom est un acronyme qui signifie *Selective Compliance Assembly/Articulated Robot Arm*. Il est constitué de deux liaisons rotoïdes et d'une liaison prismatique, en faisant un robot de type RRP. Une dernière liaison rotoïde peut être associée en bout de bras pour ajouter un degré de mobilité.

Avantages	Inconvénients
— Bon compromis entre enveloppe de travail et occupation au sol — Déplacements rapides — Parmi les meilleures performances en répétabilité — Architecture simple et peu couteuse	— Cinématique inverse complexe (même si des solutions analytiques existent, voir travaux dirigés) — Limité à de faibles charges utiles — Mobilité limitée (généralement 4 degrés de liberté)

Exemple de caractéristiques :

Epson SCARA T3

Charge normale : 1kg, max 3kg
Zone de travail :
400mm (rayon) × 150mm (hauteur)
Répétabilité : 0.02 mm et 0.02°
Vitesse maximale : entre 1 et 3.7m/s selon les axes et 2600°/s
Accélération sur chaque axe : 8m/s^2
Force de compression : 89N

Fanuc SR-3iA

Charge normale : 3 kg
Zone de travail :
400mm (rayon) × 150mm (hauteur)
Répétabilité : 0.8 mm
Vitesse sur chaque axe : 0.5m/min
Accélération sur chaque axe : 1.5m/s^2

Exemples d'applications :

— Pick and place/palettisation/conditionnement
— Assemblage
— Inspection qualité

2.3 Robots delta

Définition :

C'est un type de robot composé de trois bras parallèles reliées à une plateforme mobile par l'intermédiaire de joints universels (joints de cardan). La rotation de trois moteurs permet le contrôle en translation de la partie mobile.

Avantages	Inconvénients
— Faible inertie de la partie mobile — Vitesse et accélération élevées — Placé au dessus de la zone de travail	— Cinématique inverse complexe — Limité à de faibles charges utiles — Mobilité limitée (généralement 3 translations)

Exemple de caractéristiques :

ABB IRB 360-1 FlexPicker

Charge utile : 1kg
Zone de travail :
1130mm (diamètre) × 250mm (hauteur)
Répétabilité : 0.1mm
Vitesse maximale : 10m/s
Accélération maximale : 150m/s^2

Yaskawa MPP3H

Charge normale : 1 kg
Zone de travail :
420mm (rayon) × 300mm (hauteur)
Répétabilité : 0.1mm
Vitesse maximale : 2m/s, 1400°/s
Vitesse de production : 230 cycles/min

Exemples d'applications :

— Pick and place/palettisation/conditionnement
— Assemblage
— Alternative aux limites des SCARA

2.4 Bras robotisés - robots 6 axes

Définition :

Composé de 6 articulations de type rotoïdes en série, noté RRRRRR. La composition et le schéma cinématique associé à ce type de robot ont été donnés précédemment. La grande mobilité des bras robotisés en fait une solution adaptée face aux limites des autres types de robots industriels.

Avantages	Inconvénients
— 6 degrés de mobilité — Compatible avec des charges lourdes — Compatible avec des processus complexes — Grande versatilité	— Coût élevé — Cinématique inverse complexe — Précision dépendante de l'allonge — Gestion de la sécurité

Exemple de caractéristiques :

Universal Robots UR10e

Charge utile : 12.5kg
Zone de travail : 1300mm (rayon)
Répétabilité : 0.05mm
Vitesse maximale : de 120°/s
à 180°/s par axe

Staubli TX2-160

Charge utile : 40 kg
Zone de travail : 1710mm (rayon)
Répétabilité : 0.05mm
Vitesse maximale : 10.3m/s

Exemples d'applications :

— Assemblage
— Usinage
— Soudage
— Découpage laser
— Métrologie et inspection qualité

3 Effecteurs

Les robots industriels sont associés à un large panel d'organes terminaux (aussi appelés effecteurs) adaptés à de nombreuses applications. Il peut s'agir de systèmes permettant de saisir des objets appelés préhenseurs, ou bien de machines outils permettant par exemple de réaliser des procédés d'assemblage, de découpage ou de finition. On trouve aussi toute une gamme d'organes terminaux permettant d'interroger l'environnement dans lequel se trouve le robot avec de nombreux capteurs ponctuels et des caméras 2D et 3D.

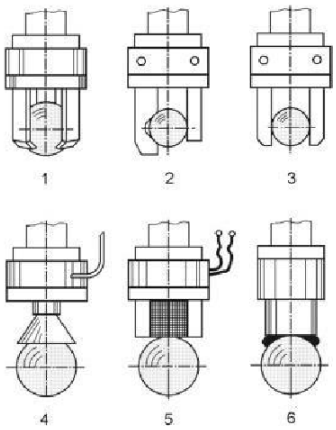
3.1 Les préhenseurs

Il existe une grande variété de préhenseurs adaptables aux bras robotisés. On utilise le plus généralement l'énergie électrique ou pneumatique pour les actionner.

Électrique À effet Venturi Adaptatif

FIGURE 9 – Exemples de préhenseurs adaptables à des bras robotisés développés par OnRobot et Festo.

En pratique, on choisit une technologie de préhenseur en fonction des contraintes imposées par une application.



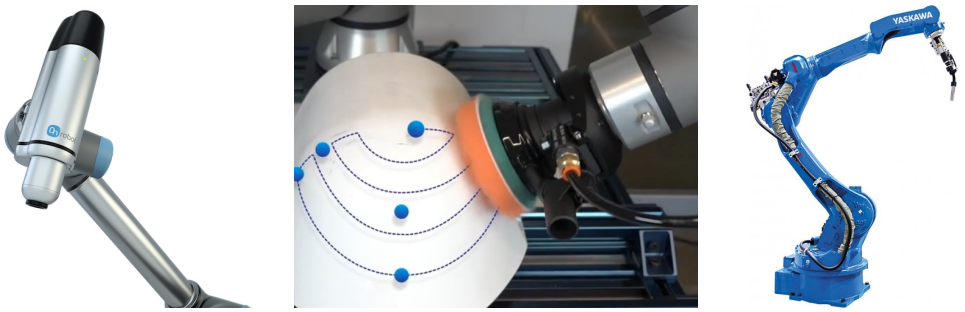
Source : Livre Robot Grippers
G. Monkman - Édition Wiley

Différentes technologies de préhenseurs sont illustrées ci-contre :

1. Fermeture sans serrage
2. Ajustement de forme combiné à une force de serrage
3. Maintien par force de serrage
4. Maintien par vide d'air
5. Rétention par un champ magnétique
6. Rétention par adhésion

3.2 Outillage et procédés

Pour l'ensemble des procédés de fabrication et de finition présentés précédemment, il est nécessaire d'adapter à la partie terminale de chaque robot industriel un outil qui pourrait normalement être exploité par un opérateur humain. Quelques exemples sont présentés dans la figure suivante.



Visseuse Ponceuse-polisseuse Bras de soudage TIG

FIGURE 10 – Exemples d'outils adaptés à des bras robotisés développés par OnRobot et Yaskawa.

Ces illustrations ne sont bien entendues pas exhaustives. Il faut garder à l'esprit que l'objectif pour les fabricants et les industriels est le remplacement de la main d'œuvre humaine. Pour certaines opérations particulièrement délicates, il est difficile de développer des solutions entièrement automatisées. Même lorsque c'est réalisable, l'automatisation de la production doit faire l'objet d'une étude financière avancée, permettant de déterminer si les coûts associés à la conception, au développement et à la fabrication de ces solutions peuvent être amortis par la valeur ajoutée associée à cette production.

3.3 Capteurs

Dans certains cas, la seule définition précise d'un parcours machine peut être suffisant pour l'automatisation des tâches. Dans d'autres scénarios, il peut être nécessaire de disposer d'informations liées à l'environnement du robot et à son interaction avec les pièces manipulées ou usinées. Parmi les solutions exploitées dans ce domaine, on retrouve notamment les capteurs de force qui peuvent être montés en amont des organes terminaux. Certains capteurs permettent ainsi de mesurer jusqu'à trois forces et trois moments selon trois axes orthogonaux, dont le principal est porté par l'orientation du corps sur lequel est monté l'outil. Les robots peuvent ainsi être dotés d'un sens du touché particulièrement sensible, mais beaucoup plus basique que le notre puisque concentré sur un nombre très restreint de capteurs.

Capteur de forces 6 composantes Système de vision

FIGURE 11 – Exemples de capteurs adaptés à des bras robotisés développés par Robotiq et OnRobot.

L'ajout de systèmes de vision est parfois nécessaire, permettant d'imager la scène dans la-

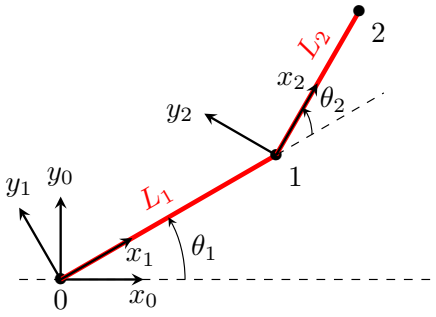
quelle évolue le robot. La vision par ordinateur est un domaine de recherche à part entière, mais des solutions existent déjà dans le commerce et permettent la reconstruction en temps réel de scènes en 3D. Les capteurs peuvent être aussi bien montés sur les bras robotisés où statiques dans son environnement et reposent généralement sur l'exploitation d'ondes électromagnétiques (visibles, infrarouges, microondes) et plus exceptionnellement sur des ondes acoustiques (notamment ultrasons).

4 Cinématique

En lien avec les éléments introduits précédemment, il existe de nombreuses architectures de robot permettant de déplacer des objets ou des outils dans l'espace. La cinématique est donc un aspect central à considérer pour l'étude d'un robot, permettant de comprendre la façon dont son organe terminal se positionne en fonction de tous les degrés de liberté à disposition. On définit ainsi en premier lieu les formalismes impliqués dans la cinématique des robots multi-axes, permettant ensuite de comprendre la notion importante d'enveloppe de travail.

4.1 Formalisation

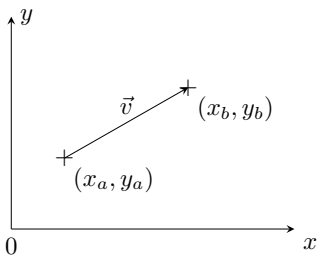
Nous allons nous concentrer ici sur les transformations de coordonnées n'impliquant pas de déformation, aussi appelées **transformations homogènes**. Ces notions seront abordées ici rapidement, puis approfondies dans le cadre d'exercices de travaux dirigés. Les liaisons précédemment étudiées nous permettent de restreindre ces transformations à seulement deux types de mouvements que sont la translation et la rotation. Les explications sont ici données selon seulement deux dimensions.



4.1.1 Translation

Une translation est définie par un vecteur $\vec{v}$ entre deux points. Par décomposition dans un espace cartésien à deux dimensions, la translation s'écrit alors de la façon suivante :

$$\begin{bmatrix} x_b \\ y_b \end{bmatrix} = \begin{bmatrix} x_a \\ y_a \end{bmatrix} + \begin{bmatrix} v_x \\ v_y \end{bmatrix} \tag{1}$$



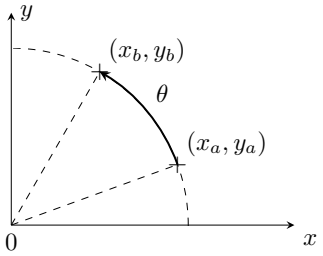
4.1.2 Rotation

Une rotation peut être appliquée à un point après avoir défini un axe de rotation, ici selon l'axe z passant par l'origine, et un angle noté ici θ. La transformation mathématique associée s'écrit de la façon suivante :

$$x_b = x_a \cos(\theta) - y_a \sin(\theta) \tag{2}$$
$$y_b = x_a \sin(\theta) + y_a \cos(\theta) \tag{3}$$

Pour simplifier la programmation des rotations, on préfère utiliser la notation matricielle. On obtient alors l'opération équivalente :

$$\begin{bmatrix} x_b \\ y_b \end{bmatrix} = \underbrace{\begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix}}_{\text{matrice de rotation}} \begin{bmatrix} x_a \\ y_a \end{bmatrix} \tag{4}$$



4.1.3 Généralisation à un formalisme homogène

Nous avons vu que les translations correspondaient en écriture vectorielle à de simples additions, tandis que les rotations pouvaient s'écrire au moyen d'une multiplication par une matrice. Bien que ces notations soient déjà particulièrement compactes, il est possible de proposer une écriture commune aux deux opérations, permettant de chaîner plus facilement une succession de transformations. On introduit ainsi les **coordonnées homogènes**. Dans cette nouvelle notation, les coordonnées d'un point en deux dimensions s'écrivent de la façon suivante :

August Ferdinand Möbius
1790-1868
Développe les coordonnées homogènes en géométrie projective et le ruban de Möbius

$$\begin{bmatrix} x \\ y \\ w \end{bmatrix} \tag{5}$$

où ce qui ressemble à une dimension supplémentaire notée w correspond en fait à un facteur d'échelle. Ce dernier est généralement défini tel que $w = 1$ en robotique.

On retrouve ainsi des notations du type $\begin{bmatrix} x_a \\ y_a \\ 1 \end{bmatrix}$ pour représenter un point de coordonnées (x_a, y_a).

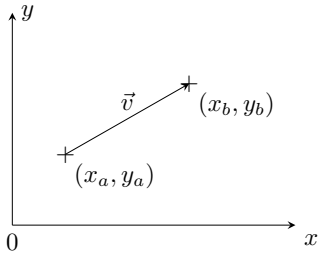
Grâce à ces nouvelles notations, nous allons pouvoir décomposer n'importe quelle transformation à l'aide d'une **matrice de transformation homogène H**, définie de la façon suivante :

$$\begin{bmatrix} x_b \\ y_b \\ 1 \end{bmatrix} = \begin{bmatrix} R_{11} & R_{12} & T_1 \\ R_{21} & R_{22} & T_2 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_a \\ y_a \\ 1 \end{bmatrix} = \mathbf{H} \begin{bmatrix} x_a \\ y_a \\ 1 \end{bmatrix} \tag{6}$$

A l'aide de cette écriture unique, notre point de coordonnées (x_a, y_a) subit une translation par le vecteur $\vec{t} = (T_1, T_2)$, ainsi qu'une rotation par la matrice de rotation $\mathbf{R} = \begin{bmatrix} R_{11} & R_{12} \\ R_{21} & R_{22} \end{bmatrix}$. Pour simplifier nos études, on décomposera toujours nos transformations en une succession de translations et rotations indépendantes.

On retrouve ainsi le cas simplifié d'une **translation** par un vecteur $\vec{v} = (v_x, v_y)$.

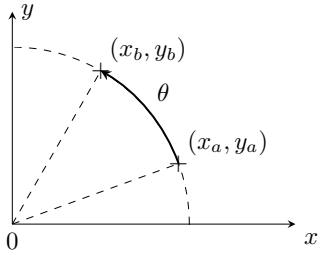
$$\begin{bmatrix} x_b \\ y_b \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & v_x \\ 0 & 1 & v_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_a \\ y_a \\ 1 \end{bmatrix} \tag{7}$$



On remarque que la matrice de rotation est définie telle que $\mathbf{R} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$. Cette dernière correspond à une matrice identité, qui a pour propriété de ne pas modifier un vecteur lorsqu'elle est multipliée à ce dernier.

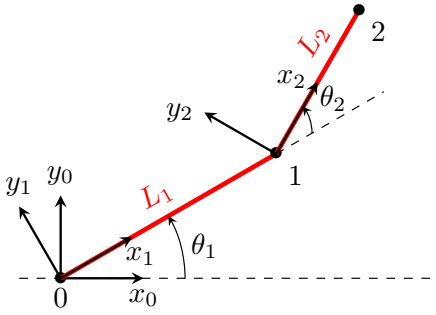
Dans le cas d'une **rotation** d'origine (0,0) et d'angle θ, les coordonnées homogènes sont transformées de la façon suivante :

$$\begin{bmatrix} x_b \\ y_b \\ 1 \end{bmatrix} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_a \\ y_a \\ 1 \end{bmatrix} \tag{8}$$



La matrice de rotation correspond ainsi à celle vue précédemment et le vecteur de translation de la matrice de transformation est nul.

Pour illustrer l'utilité d'un tel formalisme, reprenons notre précédent exemple composé de deux corps liés par des articulations rotoïdes (type robot SCARA). **L'objectif est de définir la position de l'ensemble des axes dans le repère de la base du robot.** Pour faciliter le suivi, il va être nécessaire de définir une notation permettant de différencier les différents repères et les indices des points.



A titre d'exemple, la position du nœud 2 dans le repère $\mathcal{R}_1$ s'écrit ${}^1\vec{p}_2$:

exprimé dans le repère $\mathcal{R}_1$ ← ${}^1\vec{p}_2$ → nœud d'indice 2

Expression du nœud 1 dans le repère $\mathcal{R}_0$:

Pour exprimer la position du nœud 1 dans le repère $\mathcal{R}_0$, on commence par définir la position de ce nœud dans le repère $\mathcal{R}_1$:

$$ {}^1\vec{p}_1 = \begin{bmatrix} L_1 \\ 0 \\ 1 \end{bmatrix} \tag{9} $$

On définit ensuite une **matrice de transformation homogène** du repère $\mathcal{R}_1$ vers le repère $\mathcal{R}_0$ à l'aide de la matrice ${}^0\mathbf{H}_1$:

repère final $\mathcal{R}_0$ ← ${}^0\mathbf{H}_1$ → repère initial $\mathcal{R}_1$

On obtient alors la transformation suivante :

$$ {}^0\vec{p}_1 = {}^0\mathbf{H}_1 \, {}^1\vec{p}_1 \tag{10} $$

Dans notre cas, cette transformation correspond à une simple rotation par un angle θ_1 :

$$ {}^0\mathbf{H}_1 = {}^0\mathbf{R}_1 = \begin{bmatrix} \cos(\theta_1) & -\sin(\theta_1) & 0 \\ \sin(\theta_1) & \cos(\theta_1) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{11} $$

On réduit ici la transformation homogène à une simple rotation ${}^0\mathbf{R}_1$ entre ces deux repères. Nous pouvons essayer de développer le calcul matriciel :

$$ {}^0\vec{p}_1 = {}^0\mathbf{H}_1 \, {}^1\vec{p}_1 \tag{12} $$

$$ {}^0\vec{p}_1 = \begin{bmatrix} \cos(\theta_1) & -\sin(\theta_1) & 0 \\ \sin(\theta_1) & \cos(\theta_1) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} L_1 \\ 0 \\ 1 \end{bmatrix} \tag{13} $$

$$ {}^0\vec{p}_1 = \begin{bmatrix} L_1 \cos(\theta_1) \\ L_1 \sin(\theta_1) \\ 1 \end{bmatrix} \tag{14} $$

On retrouve bien les projection du nœud 1 sur les axes x_0 et y_0 et la conservation de la troisième composante correspondant au facteur d'échelle égal à 1.

Expression du nœud 2 dans le repère $\mathcal{R}_0$:

On peut maintenant tenter d'exprimer la position de l'effecteur du robot au nœud 2 dans le repère de la base $\mathcal{R}_0$. On commence par donner son expression dans le repère $\mathcal{R}_2$:

$$ {}^2\vec{p}_2 = \begin{bmatrix} L_2 \\ 0 \\ 1 \end{bmatrix} \tag{15} $$

L'expression de ce point sur le repère de la base nécessite cette fois d'enchaîner deux transformations, passant du repère $\mathcal{R}_2$ au repère intermédiaire $\mathcal{R}_1$ jusqu'au repère $\mathcal{R}_0$:

$$ {}^0\vec{p}_2 = {}^0\mathbf{H}_1 \, {}^1\mathbf{H}_2 \, {}^2\vec{p}_2 \tag{16} $$

Il reste ainsi à définir la matrice de transformation homogène ${}^1\mathbf{H}_2$. Cette dernière correspond à une rotation d'angle θ_2 puis à une translation de norme L_1 :

$$ {}^1\mathbf{H}_2 = {}^1\mathbf{T}_2 \, {}^1\mathbf{R}_2 \tag{17} $$

$$ {}^1\mathbf{H}_2 = \underbrace{\begin{bmatrix} 1 & 0 & L_1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{\text{Translation}} \underbrace{\begin{bmatrix} \cos(\theta_2) & -\sin(\theta_2) & 0 \\ \sin(\theta_2) & \cos(\theta_2) & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{\text{Rotation}} \tag{18} $$

En exploitant la précédente définition de la transformation entre le repère $\mathcal{R}_1$ et le repère $\mathcal{R}_0$, on peut enfin développer le calcul de l'expression des coordonnées du nœud 2 dans le repère $\mathcal{R}_0$:

$$ {}^0\vec{p}_2 = {}^0\mathbf{H}_1 \, {}^1\mathbf{H}_2 \, {}^2\vec{p}_2 \tag{19} $$

$$ {}^0\vec{p}_2 = {}^0\mathbf{R}_1 \, {}^1\mathbf{T}_2 \, {}^1\mathbf{R}_2 \, {}^2\vec{p}_2 \tag{20} $$

$$ {}^0\vec{p}_2 = \begin{bmatrix} \cos(\theta_1) & -\sin(\theta_1) & 0 \\ \sin(\theta_1) & \cos(\theta_1) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & L_1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos(\theta_2) & -\sin(\theta_2) & 0 \\ \sin(\theta_2) & \cos(\theta_2) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} L_2 \\ 0 \\ 1 \end{bmatrix} \tag{21} $$

On pourra vérifier en travaux dirigés que ce changement de repère correspond bien à nos attentes. Même s'il est possible de réaliser ces calculs à la main pour des robots relativement simples, on préfère généralement programmer ces opérations pour qu'un ordinateur les réalise à notre place. La bibliothèque *numpy* de *Python 3* permet par exemple de réaliser des multiplications matricielles très rapidement et avec une écriture simple. Ces mêmes opérations sont par ailleurs réalisées par tous les logiciels de CAO intégrant des modules cinématiques.

Mise en garde 1 :
L'ordre des éléments dans une multiplication matricielle a une importance ($\mathbf{AB} \neq \mathbf{BA}$), cette opération n'est **pas commutative** contrairement à la multiplication classique. Il faudra donc que l'on démontre en TD que ${}^1\mathbf{T}_2 \, {}^1\mathbf{R}_2$ correspond au bon ordre des opérations dans notre exemple, plutôt que ${}^1\mathbf{R}_2 \, {}^1\mathbf{T}_2$.

Mise en garde 2 :
Il existe malheureusement presque autant de notations différentes pour les transformations homogènes que de bonnes ressources sur le sujet. Vous trouverez des écritures alternatives pour ${}^0\mathbf{H}_1$, notamment $\mathbf{H}_1^0$, ${}_1^0\mathbf{H}$ et $\mathbf{H}_{01}$ (en anglais et français). Notre notation des vecteurs exprimés en coordonnées homogènes, par exemple ${}^1\vec{p}_2$, comporte aussi des variantes dans la littérature.

La notation retenue pour ce cours a pour avantage de laisser l'indice des vecteurs disponible pour la numérotation des nœuds, évite l'utilisation des exposants qui peuvent correspondre à des puissances et facilite l'interprétation visuelle du changement de repère sur les écritures du type ${}^0\vec{p}_1 = {}^0\mathbf{H}_1 \, {}^1\vec{p}_1$.

4.2 Enveloppe de travail

Ayant défini quelques bases de cinématiques, il est maintenant possible d'aborder la notion importante d'enveloppe de travail. On définit ici l'ensemble des lieux que l'organe terminal d'un robot est capable de parcourir. A titre d'exemple, voici l'enveloppe de travail extraite de la documentation d'un robot 6 axes Stäubli TX2-60 :

Characteristics

	TX2-60	TX2-60L
Load capacity	4.5 kg	3.7 kg
Reach (between axis 1 and 6)	670 mm	920 mm
Number of degrees of freedom	6	6
Repeatability - ISO 9283	± 0.02 mm	± 0.03 mm
Weight	52.2 kg	52.9 kg
UL certification	✓	✓
Attachment methods	360°	

Performance

Joint speed -axis 1	435°/s	435°/s
Joint speed -axis 2	410°/s	385°/s
Joint speed -axis 3	540°/s	500°/s
Joint speed -axis 4	995°/s	995°/s
Joint speed -axis 5	1065°/s	1065°/s
Joint speed -axis 6	1445°/s	1445°/s
Maximum speed at load gravity center	8.4 m/s	11.1 m/s
Maximum inertia axis 5	0.325 kg.m²	0.125 kg.m²
Maximum inertia axis 6	0.1 kg.m²	0.032 kg.m²
Brakes	All axes	

Work envelope and range of motion

Maximum reach between axis 1 and 5 (R. M)	600 mm	850 mm
Minimum reach between axis 1 and 5 (R.m1)	190 mm	209 mm
Minimum reach between axis 2 and 5 (R.m2)	189 mm	208 mm
Reach between axis 3 and 5 (R.b)	310 mm	450 mm

Axis 1 (A)	± 180°	± 180°
Axis 2 (B)	± 127.5°	± 127.5°
Axis 3 (C)	± 142.5 °	± 152.5 °
Axis 4 (D)	± 270°	± 270°
Axis 5 (E)	+ 132.5°/-121°	+ 132.5°/-121°
Axis 6 (F)	± 270° (1)	± 270° (1)

(1) Software configurable up to ± 11 250°

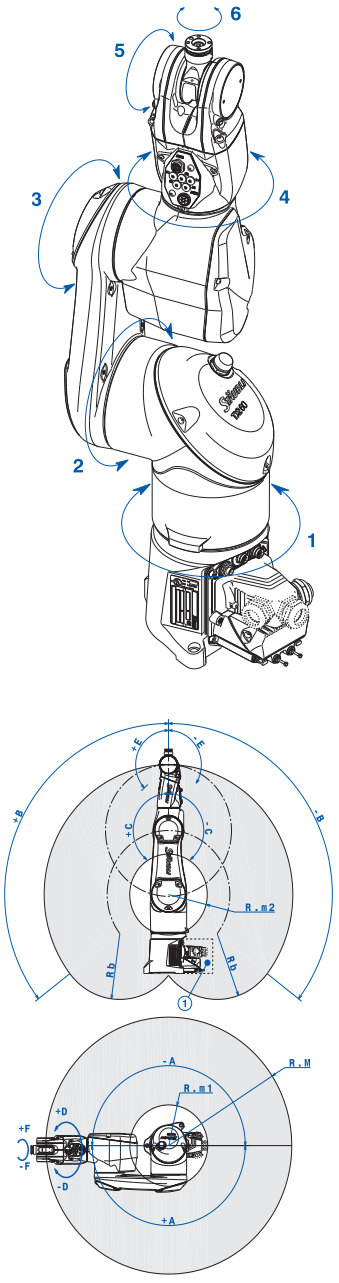


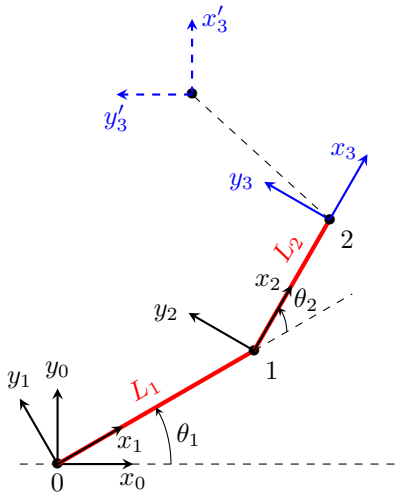
FIGURE 12 – Caractéristiques et enveloppe de travail extraites de la documentation du robot 6 axes Stäubli TX2-60.

La zone grise de cette dernière figure correspond ainsi à l'enveloppe de travail du robot, déterminée en fonction de l'amplitude des rotations des 6 axes du robot. La définition analytique d'une telle enveloppe peut être particulièrement complexe mais il est possible d'utiliser les modules cinématiques de certains logiciels de CAO et de programmation robotique pour reconstruire ces zones de façon numérique.

4.3 Cinématique inverse (le début des vrais problèmes en robotique)

La cinématique est relativement accessible et permet de calculer par une série de transformations la position et l'orientation de l'outil d'un robot par rapport à sa base en connaissant l'état de tous ses degrés de liberté.

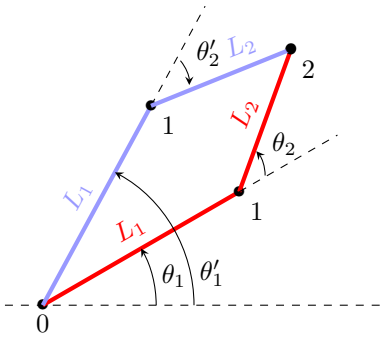
La cinématique inverse s'intéresse quant à elle au problème opposé, où l'on souhaite déterminer la valeurs des degrés de liberté d'un robot permettant d'atteindre une position et une orientation souhaitées.



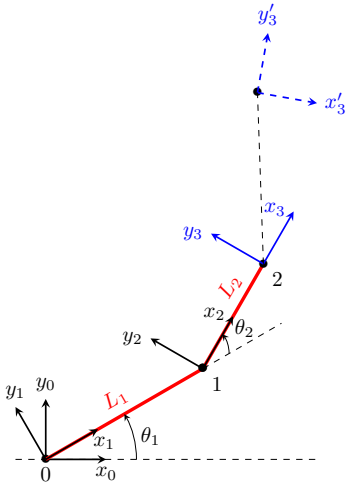
C'est un problème qui est généralement **beaucoup plus complexe à résoudre**. Pour construire une certaine intuition sur le sujet, il est possible d'analyser notre cas précédent. On souhaite déplacer l'organe du robot dans une nouvelle position et une nouvelle orientation. Il est alors nécessaire de déterminer la valeur des angles (θ_1, θ_2) permettant d'atteindre cet objectif (qui n'est pas nécessairement réalisable !).

Il existe un nombre de cas très restreints pour lesquels il est possible de trouver des solutions analytiques (un ensemble d'équations qui donnent directement leurs valeurs attendues). Les robots SCARA en sont un exemple que nous étudierons en travaux dirigés.

De façon générale, la cinématique inverse est plus difficile à calculer parce qu'il existe souvent plusieurs configurations possibles des degrés de liberté d'un robot permettant d'atteindre une position (voire une orientation) souhaitée. **Ce problème est amplifié avec l'augmentation du nombre de degrés de liberté.** Les matrices considérées contiennent aussi les valeurs des paramètres à déterminer (exemple θ_1 et θ_2), rendant la **résolution directe par diverses techniques d'inversion matricielle impossible**.



Il existe aussi de nombreux cas où la position et l'orientation souhaitées ne sont simplement pas atteignables à partir des seuls degrés de liberté à disposition. Dans le cas ci-contre, l'extension des corps du robot ne permet pas d'atteindre le repère objectif, placé en dehors de son **enveloppe de travail**.



Toutes ces notions reposent sur des bases d'algèbre trop avancées pour des BUT1. En termes simples, les techniques de cinématiques inverses reposent sur des méthodes informatiques qui convergent vers les positions et orientations attendues en ajustant progressivement tous les degrés de libertés disponibles, souvent avec du calcul de dérivés sur toutes ces variables.

Les plus motivé(e)s peuvent jeter un coup d'œil aux matrices jacobiennes, aux problèmes de singularité et aux méthodes numériques non-linéaires comme Newton-Raphson (niveau bac+5 pour la prise en main, généralement un peu plus pour vraiment comprendre les mathématiques associées et aller plus loin).

5 Sécurité

La robotique est développée dans des milieux industriels où il est nécessaire de garantir la sécurité des opérateurs impliqués dans la production, mais aussi celle des personnes chargées de l'installation et de la maintenance des équipements. Nous avons vu précédemment que certains robots peuvent atteindre des vitesses importantes. Couplées à la masse des parties mobiles, les collisions peuvent représenter un danger qu'il est nécessaire de considérer.

Pour limiter les risques, les robots peuvent être placés dans des cellules fermées et adaptées à l'enveloppe de travail de la machine.

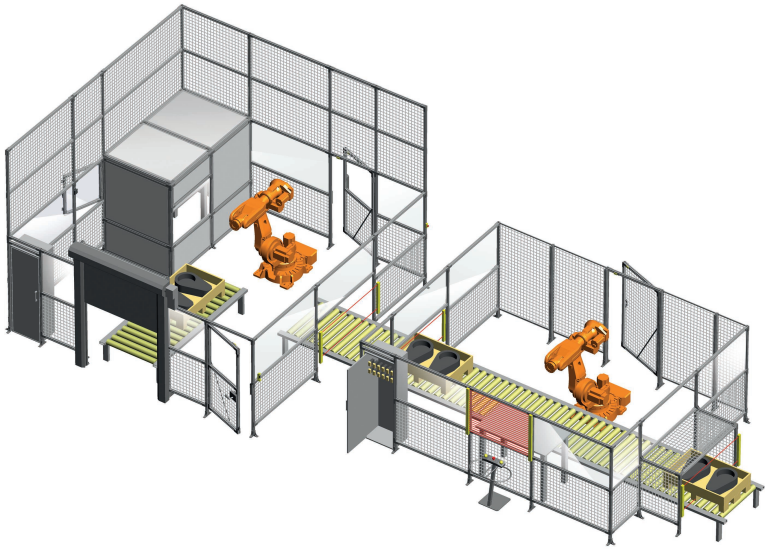


FIGURE 13 – Illustrations de barrières de sécurité autour de deux postes robotisés sur une chaîne de production (société ABB).

Les portes des cellules peuvent être équipées d'interrupteurs qui permettent d'arrêter les robots pendant l'execution de leur programme. Lorsqu'il est nécessaire d'offrir un accès plus facile aux zones de travail du robot, il est possible de mettre en place des barrière "immatérielles", constituées de faisceaux laser dont on mesure la rupture par le passage d'une partie du corps.

FIGURE 14 – Vue d'artiste d'une barrière immatérielle (les faisceaux sont normalement invisibles en l'absence de poussières.)

Par mesure de sécurité, la détection d'une personne doit mener à la mise en pause ou à l'interruption du programme du robot.

Quelques règles de sécurité :

On retrouve dans ce domaine trois règles qu'il est important de mémoriser et d'appliquer :

- **Si le robot ne bouge pas, ne pas présumer qu'il ne bougera pas.**
- **Si le robot répète une séquence, ne pas présumer qu'il va continuer à le faire.**
- **Maintenir du respect pour ce qu'est un robot et ce qu'il peut faire**

Ces règles sont adaptées du cours GPA546 de l'Ecole de Technologie Supérieur de Montréal, par Ilian Bonev et Yanick Noiseux. On les retrouve formulées avec quelques variantes dans de nombreuses ressources en français et anglais donc je n'ai pas la possibilité d'en donner les auteurs avec certitude. Le lien vers le cours (excellente ressource en français) est disponible dans la section Bibliographie.

Enfin, les phases de développement et de test de nouveaux programmes doivent aussi faire l'objet d'une attention particulière. Il est courant qu'un programme informatique comporte des erreurs qui peuvent être corrigées à force d'essais. Pour le développement de programmes de robotique, des erreurs de positionnement peuvent facilement engendrer des dégâts humains et matériels. Il convient d'observer les règles suivantes :

- **Pour les essais de programmation, l'enveloppe de travail du robot doit être dégagée.**
- **Les essais doivent être réalisés à vitesse réduite.**
- **Le bouton d'arrêt de la machine doit toujours être à portée de main, sur l'armoire de commande ou sur le boîtier déporté.**

Un point rapide sur la cobotique :

La cobotique est un domaine qui a suscité beaucoup d'intérêt ces dernières années en raison de son potentiel à faire évoluer notre façon de travailler. Elle est une combinaison des mots anglais "collaborative" et "robotics" et fait référence à l'utilisation de robots qui peuvent travailler aux côtés des humains pour effectuer des tâches. La robotique traditionnelle était principalement conçue pour des tâches dangereuses, répétitives ou nécessitant une grande précision.

La cobotique, quant à elle, vise à accroître la productivité et l'efficacité en tirant parti des forces des robots et des humains. Grâce aux progrès technologiques, la cobotique est de plus en plus utilisée dans des secteurs tels que la fabrication, la logistique et le domaine médical. Pour assurer la sécurité des personnes travaillant avec des cobots, il est nécessaire de développer des systèmes équipés d'une grande quantité de capteurs. Ces robots sont ainsi particulièrement conscients de leur environnement et sont directement programmés pour la gestion des interactions et des collisions.

Nous nous intéressons enfin à différentes méthodes de programmation des robots industriels. Dans chaque cas, l'objectif est de faciliter la génération de codes qui pourront être interprétés par le contrôleur d'un robot.

6 Méthodes de programmation

6.1 Programmation par apprentissage

Cette première approche nécessite de passer le robot en mode "programmation", impliquant qu'il devra arrêter ses tâches automatisées de production durant toute la phase de développement et d'essais. En contre-partie, c'est la méthode la plus simple à mettre en place dans la mesure où les fabricants mettent aujourd'hui à disposition des outils de programmation en ligne qui facilitent l'apprentissage de nouvelles séquences à répéter en boucle. On rencontre généralement deux approches pour la programmation en ligne qui sont décrites dans la suite de cette section.

Les robots sont associés à un boîtier de commande servant d'interface avec le robot.

FIGURE 15 – Boîtier de commande de la marque ABB, aussi appelé FlexPendant.

Il est possible par l'intermédiaire de ces boîtiers d'enregistrer une suite de mouvements et d'ainsi apprendre au robot un nouveau programme sans avoir besoin de connaître de langage de programmation. On parle alors de méthode de programmation graphique, qui a pour avantage d'être particulièrement simple à mettre en œuvre mais qui peut s'avérer limitée pour le développement de programmes complexes.

FIGURE 16 – Enregistrement manuel d'une position d'un robot ABB.

Il est aussi possible dans le cadre de la programmation en ligne de guider manuellement le robot au travers des différentes étapes de son programme, permettant d'enregistrer des positions et des orientations de son organe terminal. Cette méthode a l'avantage d'être particulièrement intuitive mais s'avère moins précise et nécessite souvent des capteurs supplémentaires. Des calculs de cinématique inverse et d'optimisation de parcours sont ensuite réalisés par la machine afin de produire un programme qui pourra être répété dans le cadre d'une production.

Les méthodes de programmation par apprentissage sont ainsi particulièrement simples à utiliser mais ne sont pas adaptées à des programmes trop complexes ou impliquant un très grand nombre d'étapes. On préfèrera dans ce cas utiliser la programmation hors-ligne.

6.2 Programmation hors-ligne

La programmation hors-ligne d'un robot repose sur le développement d'un code sans imposer l'arrêt de fonctionnement prolongé généralement associé aux méthodes de programmation par apprentissage. Certains experts sont capables d'écrire de tels codes dans de simples éditeurs de texte. Ces programmes doivent être rédigés dans des langages propres à chaque robot et peuvent être transférés directement dans la mémoire des contrôleurs des robots pour réaliser des essais. Même si cette méthode permet de faciliter l'optimisation des codes en limitant les intermédiaires entre le concepteur et le processeur de la machine, elle reste particulièrement fastidieuse à mettre en œuvre et implique des phases complexes de débuggage des codes.

```
    MoveL ToPutFix200_60,v300,z20,Gripp411_Lh_TCP\WObj:=V316Fixt2C
    MoveL InPutFix200,v200,fine,Gripp411_Lh_TCP\WObj:=V316Fixt200_
    GripperOpen Gripp411ClampOpn130Y10SR;
    !
    RETURN ;
  ENDPROC

  LOCAL PROC V316PutFix200FdLh_2()
    MoveL FrPutFix200_10,v1000,z0,Gripp411_Lh_TCP\WObj:=V316Fixt2C
    !
    GripperChkNoPart Gripp411PartChk1;
    !
    MoveJ FrPutFix200_20,v4000,z200,Gripp411_Lh_TCP\WObj:=wobj0;
    MoveJ FrPutFix200_30,v4000,z200,Gripp411_Lh_TCP\Wobj:=wobj0;
    MoveJ FrPutFix200_40,v4000,fine,Gripp411_Lh_TCP\Wobj:=wobj0;
    !
    RETURN ;
  ENDPROC

  LOCAL PROC V316PutFix200FdLh_3()
    MoveJ FrPutFix200_50,v4000,z100,Gripp411_Lh_TCP\Wobj:=wobj0;
    MoveJ FrPutFix200_60,v4000,z100,Gripp411_Lh_TCP\Wobj:=wobj0;
    MoveJ FrPutFix200_70,v4000,z100,Gripp411_Lh_TCP\Wobj:=wobj0;
    MoveAbsJ HomeGripp411,v4000,fine,Gripp411_Lh_TCP\Wobj:=wobj0;
    !
    RETURN ;
  ENDPROC

ENDMODULE
```

LOCAL PROC V316GetFix125FdLh_1()
LOCAL PROC V316GetFix125FdLh_2()
LOCAL PROC V316PutFix200FdLh_1()
LOCAL PROC V316PutFix200FdLh_2()
LOCAL PROC V316PutFix200FdLh_3()
LOCAL PROC V316WeldStMa331FdLh()
LOCAL PROC V316WeldStMa332FdLh()
PROC V316SchFdLh()

FIGURE 17 – Exemple de programme en langage RAPID utilisé pour les robots ABB.

Une autre méthode plus simple de programmation hors-ligne repose sur l'utilisation d'un logiciel permettant à la fois de simuler le comportement d'un robot, mais aussi de faciliter sa programmation par des méthodes graphiques et d'extraire un code qui pourra être exporté vers le contrôleur du robot.

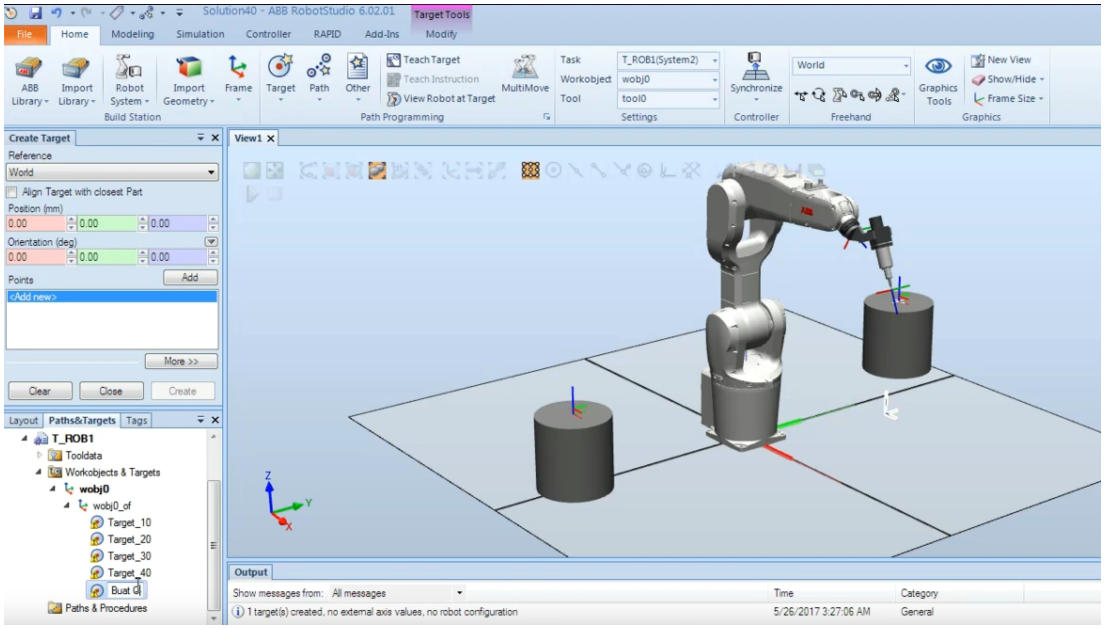


FIGURE 18 – Interface du logiciel Robot Studio d'ABB que nous utiliserons en travaux pratiques.

Dans le cadre de nos travaux pratiques, nous découvrirons l'interface du logiciel Robot Studio et apprendrons à créer une simulation de programmation d'un robot interagissant avec des objets importés.

Bibliographie - Ressources

— https ://www.robotshop.com
— https ://www.creality.com
— https ://www.epson.be
— https ://www.fanuc.eu
— https ://www.erm-automatismes.com
— https ://cours.etsmtl.ca/gol510
— https ://www.directindustry.fr
— https ://www.nederman.com
— https ://metallerie.pmg.be
— https ://robodk.com
— https ://www.directindustry.fr
— https ://www.kuka.com
— http ://www-sop.inria.fr/members/Jean-Pierre.Merlet/Cours/
— https ://www.igus.fr
— https ://www.faure-technologies.com
— https ://www.festo.com
— https ://onrobot.com
— https ://robotiq.com/fr/
— https ://www.lirmm.fr/ chemori/Temp/Wafa/Dombre-Notations.pdf
— https ://www.officiel-prevention.com/dossier/formation/fiches-metier/la-prevention-des-risques-de-la-robotisation-industrielle
— https ://new.abb.com/safety/machine-safety
— https ://www.usinenouvelle.com/expo/barriere-immaterielle-o6247.html
— https ://cours.etsmtl.ca/gpa546/Notes/Manuel.pdf
— https ://www.cobots.ch/fr/avantages-des-cobots-dur-les-cobots-sont-surs-et-collaboratifs/

TD1 - Matrices de transformation homogène

1 Démonstration des éléments de cours

On souhaite dans cette première partie démontrer un certains nombre de notions admises dans le support de cours.

1.1 Matrice identité

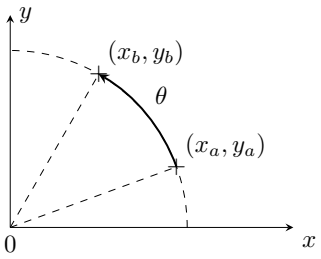
On définit une matrice identité telle que $\mathbf{I}_2 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$.

Démontrer que la multiplication d'un vecteur quelconque (de dimensions adaptées) par cette matrice identité n'a aucun impact sur le vecteur.

1.2 Matrice de rotation

Après avoir développé le calcul matriciel suivant en un système d'équations, démontrer que cette relation décrit bien une rotation.

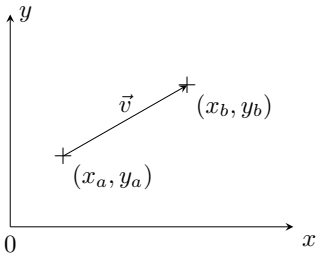
$$\begin{bmatrix} x_b \\ y_b \end{bmatrix} = \underbrace{\begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix}}_{\text{matrice de rotation}} \begin{bmatrix} x_a \\ y_a \end{bmatrix} \qquad (1)$$



1.3 Translation

On définit un vecteur $\vec{v} = \begin{bmatrix} v_x \\ v_y \end{bmatrix}$ et une translation associée définie telle que :

$$\begin{bmatrix} x_b \\ y_b \end{bmatrix} = \begin{bmatrix} x_a \\ y_a \end{bmatrix} + \begin{bmatrix} v_x \\ v_y \end{bmatrix} \qquad (2)$$



1.4 Matrices de transformation homogène

Rappeler la forme générale d'une matrice de transformation homogène dans un espace à deux dimensions.

1.4.1 Translation

Donner ensuite la matrice de transformation homogène permettant de réaliser une translation par le vecteur $\vec{v}$ précédemment défini. On vérifiera par le calcul que la multiplication d'un vecteur par une telle matrice réalise bien une translation.

1.4.2 Rotation

On souhaite maintenant faire le même exercice pour une rotation de centre (0,0) et d'angle θ.

1.5 Analyse d'un robot simplifié à 2 degrés de mobilité

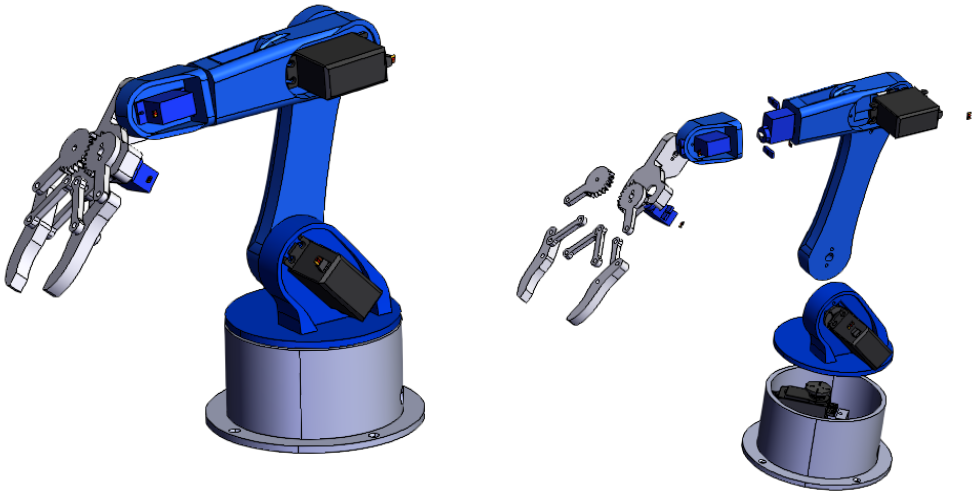
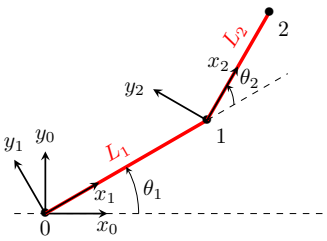


Figure 1: howtomechatronics.com/tutorials/arduino/diy-arduino-robot-arm-with-smartphone-control/

On souhaite maintenant adapter ces formalismes à un robot composé de seulement deux axes rotoïdes.

L'objectif est ici de déterminer la position de l'effecteur du robot situé au nœud 2 en fonction des angles des deux axes, notées respectivement θ_1 et θ_2.



L'expression de la position du nœud 2 dans le repère $\mathcal{R}_0$ de la base a été définie dans le cours de la façon suivante :

$$ {}^0\vec{p}_2 = {}^0\mathbf{H}_1 \, {}^1\mathbf{H}_2 \, {}^2\vec{p}_2 \tag{3} $$

$$ {}^0\vec{p}_2 = {}^0\mathbf{R}_1 \, {}^1\mathbf{T}_2 \, {}^1\mathbf{R}_2 \, {}^2\vec{p}_2 \tag{4} $$

$$ {}^0\vec{p}_2 = \begin{bmatrix} \cos(\theta_1) & -\sin(\theta_1) & 0 \\ \sin(\theta_1) & \cos(\theta_1) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & L_1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos(\theta_2) & -\sin(\theta_2) & 0 \\ \sin(\theta_2) & \cos(\theta_2) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} L_2 \\ 0 \\ 1 \end{bmatrix} \tag{5} $$

1.5.1 Vérification

On souhaite en premier lieu vérifier que ce modèle permet bien d'obtenir la position du point 2 dans le repère $\mathcal{R}_0$.

a. Sans utiliser les matrices de transformation homogène, déterminer les coordonnées cartésiennes du point 2 dans le repère $\mathcal{R}_0$.

b. Développer ensuite le calcul matriciel ci-dessus et vérifier la bonne correspondance des résultats obtenus.

1.5.2 Commutativité

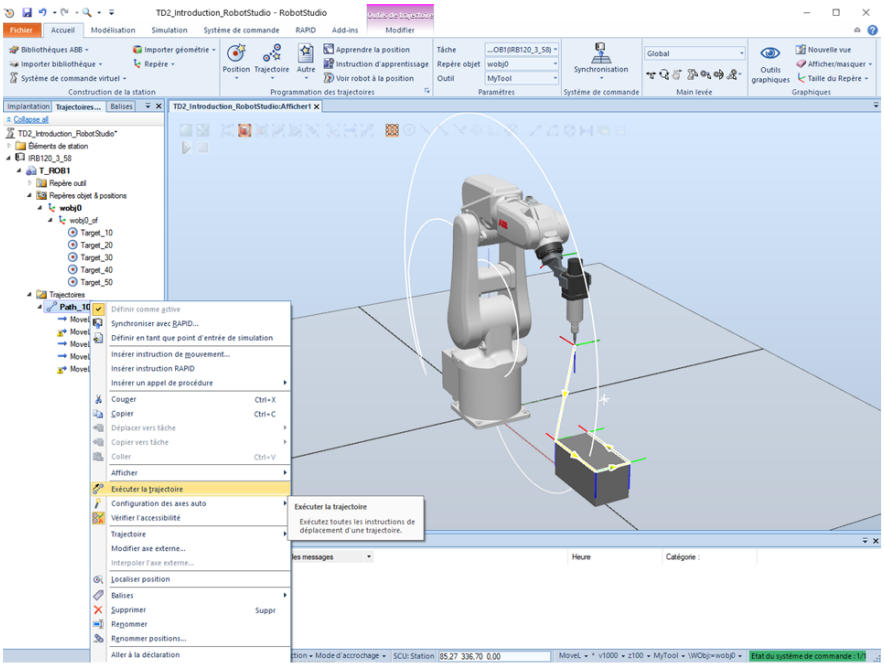
Le calcul matriciel a pour propriété de ne pas être commutatif, impliquant que l'ordre des opérations dans un chaînage de transformations a une importance cruciale.

Démontrer que :

$$ {}^1\mathbf{T}_2 \, {}^1\mathbf{R}_2 \neq {}^1\mathbf{R}_2 \, {}^1\mathbf{T}_2 \tag{6} $$

TD2 - Initiation à la programmation hors-ligne avec *RobotStudio*

Cette séance de travaux dirigés permet la prise en main du logiciel de simulation et de programmation de robots industriels *RobotStudio*, développé par ABB.



Un powerpoint d'une centaine de slides est disponible sur la plateforme communities par le chemin suivant :

Cette séance permet de préparer le module évalué de 4h de travaux pratiques.

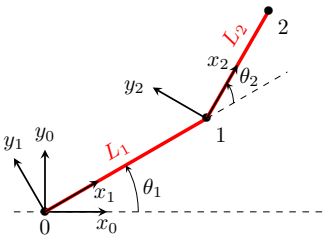
Objectifs :

- Simuler l'implantation et la mise en mouvement d'un robot 6 axes
- Créer un outil et définir un parcours selon le repère associé
- Créer un préhenseur et déplacer des objets

TD3 - Cinématique inverse

Nous avons découvert lors de la première séance de travaux dirigés l'adaptation des formalismes de transformation homogène à un bras robotisé très simple.

Ces modèles ont permis de déterminer les coordonnées d'un effecteur placé en bout de bras en fonction des paramètres géométriques du robot.



Le positionnement est assuré à l'aide des deux degrés de liberté du robot, correspondant aux angles θ_1 et θ_2 de ses deux articulations rotoïdes. Le premier TD a permis de déterminer l'expression des coordonnées du point 2 dans l'espace en fonction de ce couple de paramètres :

$$ {}^0\vec{p}_2 = \begin{bmatrix} {}^0x_2 \\ {}^0y_2 \\ 1 \end{bmatrix} = \begin{bmatrix} L_1\cos(\theta_1) + L_2\cos(\theta_1+\theta_2) \\ L_1\sin(\theta_1) + L_2\sin(\theta_1+\theta_2) \\ 1 \end{bmatrix} \tag{1} $$

On a donc résolu le problème de cinématique directe, consistant à déterminer une position en fonction des positions des axes du robot :

$$ (\theta_1, \theta_2) \rightarrow ({}^0x_2, {}^0y_2) \tag{2} $$

Cette approche peut être utile, mais la situation la plus couramment rencontrée correspond plutôt à un problème de cinématique inverse, souhaitant déterminer les paramètres du robot permettant d'atteindre les coordonnées (et parfois même l'orientation) souhaitées :

$$ ({}^0x_2, {}^0y_2) \xrightarrow{?} (\theta_1, \theta_2) \tag{3} $$

On retrouvera directement ces contraintes dans l'exploitation d'un robot SCARA, où il est nécessaire de positionner la pince aux endroits souhaités. Les formalismes étudiés étant en deux dimensions, on se restreindra à la seule étude des positions angulaires des axes rotoïdes.

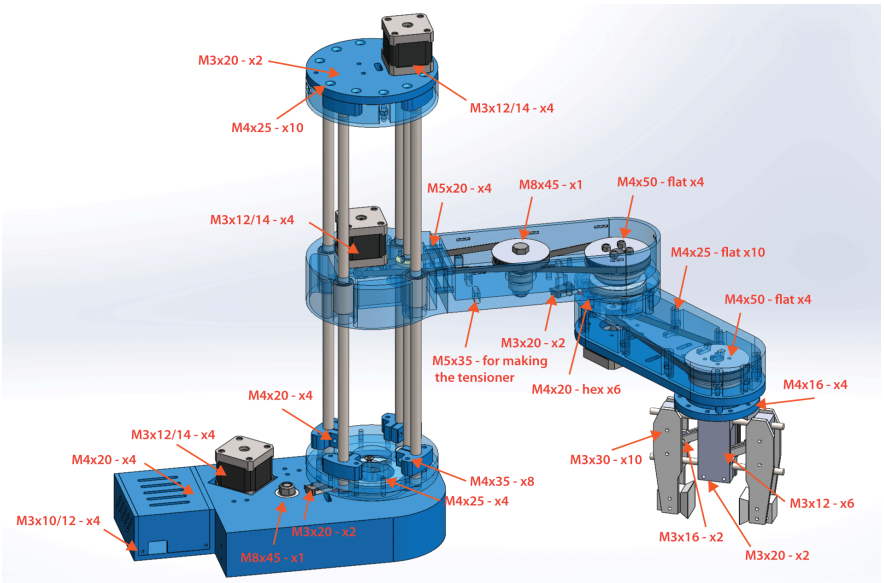


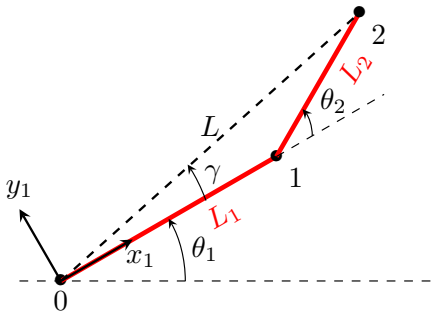
Figure 1: howtomechatronics.com/projects/scara-robot-how-to-build-your-own-arduino-based-robot/

1 Résolution guidée du cas à deux degrés de liberté

Ce problème de cinématique inverse revient donc à déterminer la relation entre les objectifs de positionnement $({}^0x_2, {}^0y_2)$ et les valeurs des deux axes (θ_1, θ_2).

1.1 Calcul de θ_2

Pour simplifier la résolution, il est préférable de déterminer l'expression d'une première inconnue en fonction des objectifs $({}^0x_2, {}^0y_2)$ et des paramètres géométriques du robot.



On souhaite ainsi démontrer premièrement la relation :

$$\theta_2 = \pm \cos^{-1}\left(\frac{{}^0x_2^2 + {}^0y_2^2 - (L_1^2 + L_2^2)}{2L_1L_2}\right) \quad (4)$$

- Rappeler l'expression de ${}^0x_2$ et ${}^0y_2$ en fonction de θ_1 et θ_2.

  On remarque que la longueur L ne dépend pas de θ_1, ce qui permettrait d'imposer une première contrainte sur la valeur de θ_2.

- Déterminer l'expression de L en fonction de ${}^0x_2$ et ${}^0y_2$.
- Déterminer ensuite l'expression de L^2 en fonction de L_1, L_2 et θ_2.
- En déduire l'expression de θ_2 en fonction de L_1, L_2 et les coordonnées ${}^0x_2$ et ${}^0y_2$.
- Pourquoi est-il nécessaire de garantir que $L^2 \leq L_1^2 + L_2^2$.
- Justifier par un schéma l'origine du signe $\pm$ de θ_2.

1.2 Calcul de θ_1

Maintenant que la variable θ_2 est déterminée, on souhaite déduire la valeur de l'angle du premier axe.

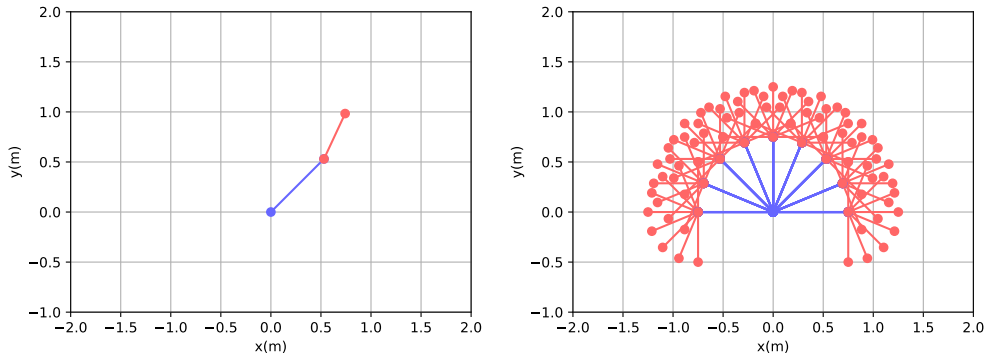
On souhaite ainsi démontrer cette fois que :

$$\theta_1 = \tan^{-1}\left(\frac{{}^0y_2}{{}^0x_2}\right) - \tan^{-1}\left(\frac{{}^1y_2}{{}^1x_2}\right) \quad (5)$$

- Déterminer l'expression des coordonnées ${}^1x_2$ et ${}^1y_2$ en fonction de L_1, L_2 et θ_2.
- Déterminer ensuite l'expression de l'angle γ en fonction de ${}^1x_2$ et ${}^1y_2$.
- Déterminer ensuite la relation entre l'angle $(\theta_1 + \gamma)$ et les coordonnées ${}^0x_2$ et ${}^0y_2$.
- En déduire alors l'expression de θ_1 en fonction de L_1, L_2 et θ_2

TD4 - Programmation Python appliquée à la cinématique

Les précédentes séances de travaux dirigés ont permis de développer un certain nombre de formalismes associés à la cinématique de bras robotisés et de robots SCARA. Dans cette nouvelle séance, l'objectif sera d'implémenter les **transformations homogènes** en langage Python afin de simuler le positionnement dans l'espace d'un bras robotisé. On souhaite ainsi reproduire les deux figures suivantes :



La figure de gauche correspond au positionnement d'un bras robotisé à deux degrés de liberté. Le programme Python doit afficher cette figure pour des valeurs arbitraires d'angles θ_1 et θ_2 et des longueurs de corps L_1 et L_2 entrées au début du code. La figure de droite permet de révéler l'envelope de travail d'un robot. Cette superposition des positions est réalisée pour une distributions de valeurs équi-réparties de θ_1 et θ_2.

On utilise les bibliothèques déjà étudiées en mathématiques pour le calcul matriciel et l'affichage des résultats. Pour harmoniser les notations, on importera ces dernières de la façon suivante :

```
import numpy as np # Pour le calcul matriciel
import matplotlib.pyplot as plt # Pour l'affichage des resultats
plt.figure(dpi=300) # Pour ameliorer la resolution des images dans Spyder
```

Quelques notions utiles :

• Multiplication matricielle avec *Numpy* :

```
M = np.array([[1,2],[3,4]]) # Definition d'une matrice
v1 = np.array([[5,6]]).transpose() # Definition d'un vecteur colonne
v2 = np.dot(M,v1) # Multiplication matricielle
```

• Boucle *for* balayant les valeurs d'un vecteur :

```
Theta1 = np.linspace(0,180,9) # 9 valeurs equireparties entre 0 et 180 degres
for i1 in range(len(Theta1)):  # Pour i1 allant de 0 a (9-1)
  theta1 = Theta1[i1] # Extraction de la valeur du vecteur
  theta1 = np.deg2rad(theta1) # Conversion de degres en radians
```

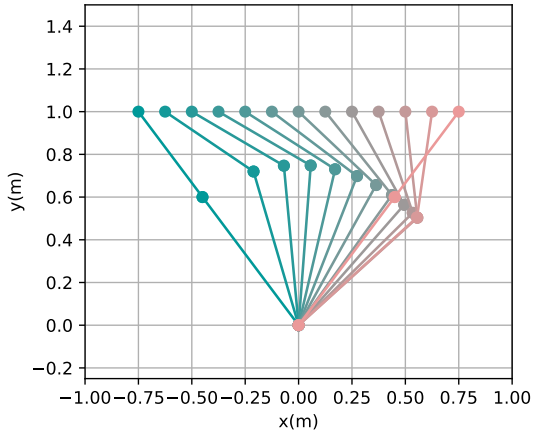
• Affichage d'un segment avec *Matplotlib* :

```
# On precise : coordonnees, type de ligne et de marqueur, couleur RGB
plt.plot([0, v1[0]],[0, v1[1]],'-o', color=(0.4,0.4,1))
plt.xlim(-2,2) # Limites horizontales de la figure
plt.ylim(-1,2) # Limites verticales de la figure
plt.grid(True) # Affiche la grille
#plt.show() # Pour afficher la figure pendant la boucle
plt.gca().set_aspect('equal', adjustable='box') # Axes orthonormes
plt.xlabel('x(m)') # Titre de l'axe horizontal
plt.ylabel('y(m)') # Titre de l'axe vertical
```

TD5 - Programmation Python appliquée à la cinématique inverse

Dans cette dernière séance, on souhaite maintenant résoudre un problème de cinématique inverse pour un bras robotisé composé de deux degrés de liberté.

On implémentera d'abord les calculs développés dans le TD3 avec *Numpy* pour déterminer les angles (θ_1, θ_2) permettant d'atteindre une position $({}^0x_2, {}^0y_2)$ souhaitée. Avec l'aide des éléments développés dans le TD4, on souhaite ensuite reproduire la figure suivante, correspondant au déplacement de l'effecteur du robot selon une ligne horizontale.



Structure du code :

- Une première partie basée sur *Numpy* mais sans matrice de transformation homogène doit permettre de déterminer (θ_1, θ_2)
- Une seconde partie extraite du TD4 doit ensuite permettre de recalculer la position des nœuds du robot et d'afficher son bras complet.
- Il est conseillé d'ajouter la boucle sur les différentes positions $({}^0x_2, {}^0y_2)$ seulement en dernier, une fois l'affichage validé pour un simple couple de paramètres objectifs.

Quelques notions utiles :

La fonction arc tangente $\tan^{-1}\left(\frac{y}{x}\right)$ est connue en méthodes numériques comme problématique pour le calcul d'angles. Des points diamétralement opposés par rapport à l'origine auront toujours le même angle au travers de cette fonction à cause des simplifications de signes sur les fractions $\frac{-x}{y} = \frac{x}{-y}$ et $\frac{-x}{-y} = \frac{x}{y}$.

Les bibliothèques de calcul numérique contiennent souvent une version alternative où il faut fournir les paramètres x et y séparément :

• Arc tangentes implémentées dans *Numpy* :

```
theta = np.arctan(y/x) # plus intuitive mais fiable a 180 degres pres
theta = np.arctan2(y,x) # plus robuste, a utiliser dans ce TD
```

• Fonction arc cosinus

```
# arc cosinus ne pose par contre aucun soucis et sera utile pour le TD
theta = np.arccos(r)
```