

Simulation géométrique des robots manipulateurs avec Matlab

Dépôt légal : 2015MO2856

ISBN: 978-9954-9197-3-6

Imprimerie : OK Copy Rabat

Première édition : Septembre 2015

Lieu d'édition : Rabat

Contact: m.bennani@um5s.net.ma

Simulation géométrique des robots manipulateurs avec Matlab

Mohammed Bennani

Table des matières

Introduction	7
Chapitre 1: Position et orientation des solides	11
Notions de Repère - Objet	11
Matrices Homogènes	12
Propriétés des matrices homogènes	13
Exemples	14
Exercices	16
Représentation de la position et l'orientation	17
Représentation de la position	17
Représentation de l'orientation	19
Représentation des angles d'Euler	19
Angles d'Euler ZYX	20
Angles d'Euler ZYZ	22
Les quaternions	24
Interface graphique des transformations	27
Chapitre 2: Représentation de Denavit Hartenberg	37
Chapitre 3: Représentation graphique des robots	43
Principe de l'animation	43
Exemple	43
Représentation schématique	44
Représentation approximative	47
Bloc prismatique	47
Bloc cylindrique	49
Blocs en mouvements	56
Simulation du robot plan à deux degrés de libertés	59
Représentation avancée	61
Chapitre 4: Espace de travail des robots manipulateurs	69
Principe d'élaboration	69
Exercices	81

Chapitre 5: Les singularités	83
Singularités des robots manipulateurs sériels	85
Singularités du robot anthropomorphique	86
Interface graphique avec des singularités	89
Exercice	93
Chapitre 6: Modélisation géométrique des robots	95
Modèle géométrique direct	96
MGD du robot sphérique	97
MGD du robot Puma à six degrés de liberté	98
Modèle géométrique inverse	101
Approche algébrique	101
MGI du robot Puma	102
Robots aux axes concourants dans le poignet	104
MGI du robot Yasukawa à cinq degrés de liberté	105
Approche géométrique	107
Problème de la Position	107
Problème de l'orientation	108
MGI du robot ABB_IRB_140_6/0.8	110
Problème de l'orientation	114
MGI du robot Puma	116
MGI du Robot Fanuc M_710C/70	119
MGI du robot Standford	124
Bibliographie	129

Introduction

La robotique est une science multidisciplinaire impliquée de plus en plus dans des tâches diverses de la société moderne. Les robots deviennent vitaux dans plusieurs applications industrielles et spatiales, ce qui nécessite des savoirs et des techniques évolués pour aborder la complexité et la diversité de tels systèmes.

La simulation, entre autres, s'avère un outil indispensable pour mener des études d'analyse et de synthèse en robotique concernant les différents aspects de conception, de modélisation et de commande. Ainsi l'objectif principal de cet ouvrage est de faciliter l'accès à cette tâche de simulation robotique dans un environnement évolué et performant de simulation des systèmes qui est Matlab.

On distingue deux aspects importants au cours de six chapitres constituant l'ouvrage: modélisation et simulation. La modélisation géométrique est le noyau de toute étude robotique. Ainsi le premier chapitre s'intéresse aux éléments de base de représentation de la position et de l'orientation d'un système polyarticulé. La notion de matrice homogène est très utile pour ce faire. Une interface graphique utilisateur de Matlab (IGU), en fin de ce chapitre, est présentée pour montrer le passage d'un type de représentation à un autre. Le programme associé à cette interface est long et consistant. En effet la plateforme de programme de l'interface est fourni par Matlab en offrant la possibilité d'insérer les corps des fonctions désirées.

Le deuxième chapitre présente la convention de Denavit Hartenberg sous ses deux formes, en fonction de la définition des paramètres de Denavit Hartenberg. Des fonctions de Matlab sont alors établies pour déduire de manière systématique la position et l'orientation de l'organe terminal par rapport au bâti fixe. Ces

informations sont alors exploitées par la suite dans l'élaboration des modèles géométriques directs et inverses des robots manipulateurs sériels.

Le chapitre trois concerne la représentation graphique des robots. Le principe de l'animation est exposé dans le cas élémentaire d'un point pour être étendu ensuite à d'autres formes géométriques plus complexes. En effet, trois niveaux de simulation sont traités qui couvrent l'ensemble des besoins de simulation en robotique: représentation schématique (schémas et symboles), représentation approximative (prisme, cylindre,...) et représentation avancée (dessins issus d'un environnement CAO). Toutes ces représentations sont illustrées par des exemples et des programmes Matlab associés. L'exemple du robot plan à deux degrés de liberté est fréquemment utilisé pour sa simplicité et sa considération comme un modèle pour des robots évoluant dans l'espace avec des degrés de liberté plus importants.

Le chapitre quatre aborde la notion d'espace de travail des robots manipulateurs sériels, qui est essentielle dans le choix des robots ainsi que dans la planification des tâches. Le principe d'élaboration est exposé avec diverses exemples, incluant des programmes Matlab, montrant la difficulté d'élaboration de ces espaces (formes, coûts de calcul,...).

Le chapitre cinq traite des singularités des robots. Les différents types de singularités pour les robots sont décrits avec une présentation significative des singularités pour les robots manipulateurs sériels. Quelques exemples sont exposés avec des programmes Matlab correspondants. Une interface IGU est présentée enfin de ce chapitre qui montre un problème fréquent en robotique; c'est celui de la détection des singularités d'un robot lors d'un mouvement spécifique de l'organe terminal dans l'espace de travail.

En fin, le chapitre six présente la modélisation géométrique des robots manipulateurs sériels. Le modèle géométrique direct est d'abord exposé avec le principe d'élaboration, des exemples et des programmes Matlab. Ce modèle exige surtout de la rigueur et du soin pour établir ses relations fortement couplées et non linéaires. Les programmes Matlab s'avèrent très performants pour réaliser cet objectif. Cependant, la résolution de telles relations géométriques conduit au modèle géométrique inverse qui ne fait pas disposer, de manière générale, de solutions analytiques. Deux approches sont alors exposées permettant de trouver

des solutions, particulièrement pour les robots dont les axes des trois dernières articulations sont concourants et lorsque les centres de ces articulations sont coïncidents.

L'approche algébrique est classique avec ses procédures d'élaboration des différentes variables articulaires. Quelques exemples des robots à cinq et six degrés de liberté sont traités pour illustrer cette approche avec leurs programmes Matlab.

Un intérêt particulier est donné à l'approche géométrique du modèle géométrique inverse. Elle très efficace dans le calcul des variables articulaires, mais exige une bonne connaissance de la géométrie du robot et de son évolution au cours des mouvements articulaires. Ainsi plusieurs exemples de robots à six degrés de liberté, avec diverses architectures géométriques, sont traités pour montrer le principe de cette approche. Les programmes Matlab, établis dans ces exemples, contribuent à la résolution de ce modèle géométrique inverse.

Cette étude de simulation géométrique des robots manipulateurs sériels peut être étendue à d'autres structures robotiques (arborescentes, parallèles,...). Les programmes Matlab établis dans les différents chapitres doivent être compris et appliqués à plusieurs cas robotiques pour bien maîtriser les aspects de simulation des robots.

L'animation des robots en considérant les effets cinématiques et dynamiques pourrait être conduit de la même manière que celle établie dans le cas d'analyse géométrique. Il faut alors bien maîtriser les concepts théoriques cinématiques et dynamiques des robots et de bien les traduire avec des programmes Matlab pour finaliser encore une simulation plus approfondie et plus réaliste.

Chapitre 1

Position et orientation des solides

Notions de Repère - Objet

Pour localiser le mouvement d'un objet S par rapport à un référentiel R_0 , il suffit de lui attacher un repère R_s . La position et l'orientation de l'objet sont alors déduites de celles du repère R_s par rapport à R_0 .

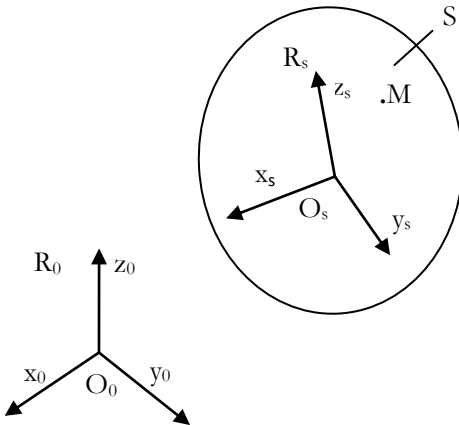


Figure 1.1: *Position et orientation d'un solide.*

La position de tout point M appartenant au solide S est définie comme suit:

$$O_0M = O_0O_s + O_sM \quad (1.1)$$

En projetant la relation (1.1) dans R_0 , on obtient:

$$O_0 M^0 = O_0 O_s^0 + A^{0,s} O_s M^s \quad (1.2)$$

Où $A^{0,s}$ est la matrice d'orientation du repère R_s par rapport à R_0 , définie par les coordonnées des vecteurs unitaires x_s , y_s et z_s dans le repère de base R_0 .

$$A^{0,s} = \begin{pmatrix} a_x & b_x & c_x \\ a_y & b_y & c_y \\ a_z & b_z & c_z \end{pmatrix} \quad (1.3)$$

L'orientation de l'objet S par rapport à R_0 est alors définie entre autres par la matrice d'orientation $A^{0,s}$.

Matrices Homogènes

Pour mener une étude géométrique, cinématique ou dynamique d'un objet, il est très commode de rassembler la position et l'orientation de cet objet dans une seule représentation. La matrice homogène de dimension 4×4 permet alors de réaliser cette tâche.

Si les coordonnées du centre O_s du repère R_s sont définies par:

$$O_0 O_s^0 = \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} \quad (1.4)$$

Et l'orientation est définie par $A^{0,s}$, alors la matrice homogène qui permet la représentation du solide S par rapport à R_0 est donnée par l'expression suivante:

$$T^{0,s} = \begin{pmatrix} A^{0,s} & O_0 O_s^0 \\ 0 & 1 \end{pmatrix} \quad (1.5)$$

D'où en remplaçant $A^{0,s}$ et $O_0 O_s^0$ par leurs expressions, on aboutit à la forme suivante:

$$T^{0,s} = \begin{pmatrix} a_x & b_x & c_x & X \\ a_y & b_y & c_y & Y \\ a_z & b_z & c_z & Z \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad (1.6)$$

Propriétés des matrices homogènes

Lorsqu'on est en présence d'un système multi corps en mouvements les uns par rapport aux autres (comme c'est le cas en robotique), les matrices homogènes disposent de propriétés très utiles pour la représentation de ces objets.

Prenons le cas d'une chaîne cinématique ouverte, constituée de n corps en mouvement les uns par rapport aux autres comme le montre la figure suivante:

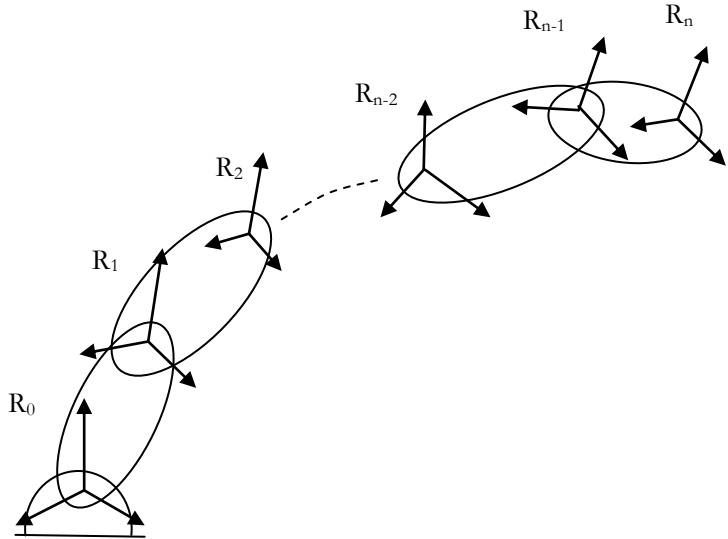


Figure 1.2: Chaîne cinématique ouverte de n corps.

La position et l'orientation du solide S_n par rapport au solide de base S_0 (immobile) est donnée par la matrice homogène:

$$T^{0,n} = T^{0,1} T^{1,2} \dots T^{n-1,n} \quad (1.7)$$

Cette relation est très utile en robotique puisqu'elle permet de définir les expressions géométriques concernant l'organe terminal du robot par rapport à la base fixe.

Une autre propriété intéressante des matrices homogènes est liée à la représentation d'un vecteur V dans différents repères:

$$V^i = A^{i,j} V^j \quad (1.8)$$

Exemples

Soit un mécanisme plan articulé de deux bras S_1 et S_2 . La liaison entre S_1 et S_0 est rotoïde ainsi que celle entre S_1 et S_2 . Les repères R_0 , R_1 et R_2 sont attachés respectivement à S_0 , S_1 et S_2 comme le montre la figure suivante:

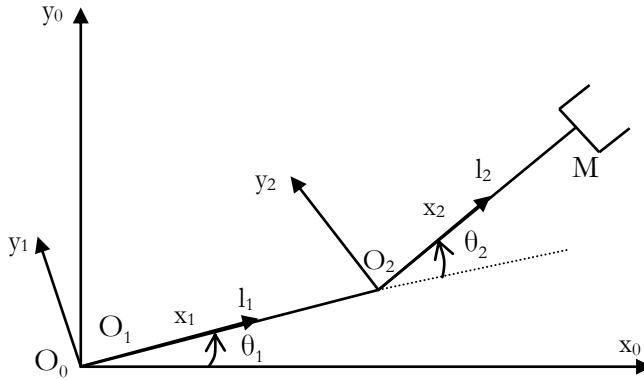


Figure 1.3: Schéma du robot plan à deux degrés de liberté.

Les matrices homogènes $T^{i-1,i}$ $i = 1 \dots 2$ sont définies comme suit:

$$T^{0,1} = \begin{pmatrix} c_1 & -s_1 & 0 & 0 \\ s_1 & c_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad \text{et} \quad T^{1,2} = \begin{pmatrix} c_2 & -s_2 & 0 & l_1 \\ s_2 & c_2 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad (1.9)$$

Avec c_i , s_i , c_{ij} et s_{ij} désignant respectivement $\cos(\theta_i)$, $\sin(\theta_i)$, $\cos(\theta_i + \theta_j)$ et $\sin(\theta_i + \theta_j)$.

La matrice $T^{0,2}$ est alors obtenue en multipliant les deux matrices de la relation (1.9).

$$T^{0,2} = T^{0,1} T^{1,2} \quad (1.10)$$

$$\text{D'où } T^{0,2} = \begin{pmatrix} c_{12} & -s_{12} & 0 & l_1 c_1 \\ s_{12} & c_{12} & 0 & l_1 s_1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad (1.11)$$

La position du point M de coordonnées x, y et z dans R_0 est alors définie par l'expression suivante:

$$O_0 M^0 = O_0 O_2^0 + O_2 M^0 \quad (1.12)$$

$$O_0 M^0 = A^{0,1} O_0 O_2^1 + A^{0,2} O_0 M^2 \quad (1.13)$$

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} c_1 & -s_1 & 0 \\ s_1 & c_1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} l_1 \\ 0 \\ 0 \end{pmatrix} + \begin{pmatrix} c_{12} & -s_{12} & 0 \\ s_{12} & c_{12} & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} l_2 \\ 0 \\ 0 \end{pmatrix} \quad (1.14)$$

En développant la relation (1.14), on obtient:

$$x = l_1 c_1 + l_2 c_{12}$$

$$y = l_1 s_1 + l_2 s_{12} \quad (1.15)$$

$$z = 0$$

Exercices

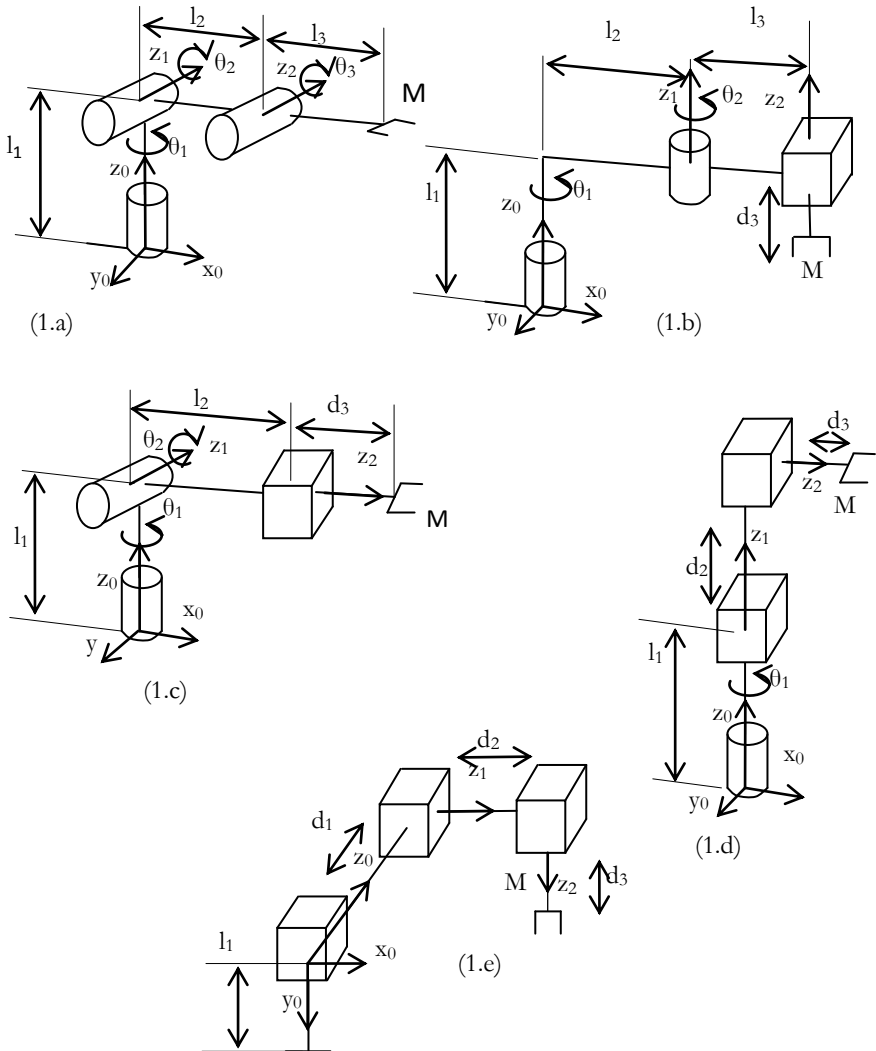


Figure 1.4: Schémas de robots porteurs à 3 ddl.

Considérons les cinq robots porteurs à 3 degrés de liberté (ddl) définis dans la figure ci dessous. Pour chaque robot déterminer:

- La matrice d'orientation $T^{i-1,i}$ $i = 1 \dots 3$.
- La matrice homogène $T^{0,3}$.
- La matrice d'orientation $A^{0,3}$ et le vecteur de position O_0M^0 .

Représentation de la position et l'orientation

Le mode de représentation de la position et de l'orientation d'un solide est important dans le traitement des modèles des robots. Si on adopte généralement la représentation cartésienne en ce qui concerne la position; la représentation de l'orientation reste délicate à élaborer. En effet, la matrice d'orientation comporte neuf paramètres liés entre eux et serait donc encombrante dans le calcul. On préfère alors des formes de représentation minimale telles que la représentation d'Euler ou des quaternions.

Représentation de la position

La position d'un point M appartenant à un solide S est généralement représentée dans un référentiel R_0 par ses coordonnées cartésiennes, cylindriques ou sphériques.

La représentation cartésienne est donnée directement par les coordonnées x, y et z du point M dans R_0 : $M \begin{pmatrix} x \\ y \\ z \end{pmatrix}$.

Si r et θ sont les coordonnées polaires du point M dans le plan $z=0$ et z est la distance du point M au plan $z=0$, alors les coordonnées cylindriques sont : $M(r, \theta, z)$.

Les relations entre les coordonnées cylindriques et les coordonnées cartésiennes se présentent comme suit:

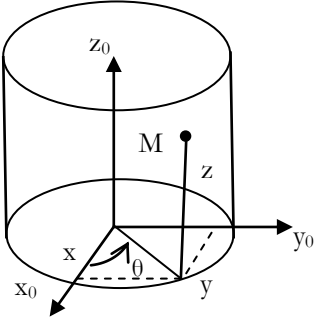
$$\begin{array}{l|l}
 x = r \cos(\theta) & r = \sqrt{x^2 + y^2} \\
 y = r \sin(\theta) & \tan(\theta) = \frac{y}{x} \\
 z = z & z = z
 \end{array} \quad (1.16)$$


Figure 1.5: *Coordonnées cartésiennes.*

En coordonnées sphériques, le point M est représenté par les coordonnées ρ , θ et φ : M (ρ , θ et φ).

Avec ρ : distance du point M à l'origine.

θ : angle formé par l'axe Ox et le segment liant l'origine à la projection de M sur le plan horizontal x_0y_0 .

φ : angle entre l'axe Oz et le segment liant le point M à l'origine.

Les relations entre les coordonnées sphériques et les coordonnées cartésiennes sont déterminées comme suit:

$$\begin{array}{l|l}
 x = \rho \sin(\varphi) \cos(\theta) & \rho = \sqrt{x^2 + y^2 + z^2} \\
 y = \rho \sin(\varphi) \sin(\theta) & \tan(\theta) = \frac{y}{x} \\
 z = \rho \cos(\varphi) & \cos(\varphi) = \frac{z}{\sqrt{x^2 + y^2 + z^2}}
 \end{array} \quad (1.17)$$

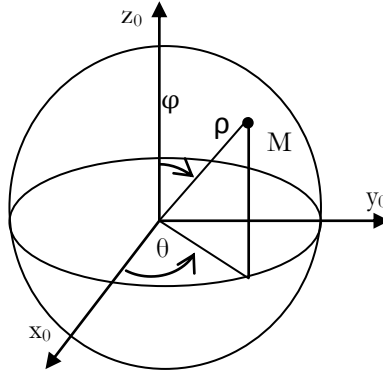


Figure 1.6: Coordonnées sphériques.

Représentation de l'orientation

Dans cette partie, les modes d'orientation par les angles d'Euler et les quaternions sont définies ainsi que leurs relations avec la matrice d'orientation. Des fonctions de Matlab sont établies pour ce faire et sont intégrées dans une interface Graphique d'utilisateur (IGU) pour une meilleure présentation.

Soit la matrice $A^{0,s}$ qui définit l'orientation du repère R_s par rapport au repère R_0 .

$$A^{0,s} = \begin{pmatrix} a_x & b_x & c_x \\ a_y & b_y & c_y \\ a_z & b_z & c_z \end{pmatrix} \quad (1.18)$$

Cette matrice est une première forme pour la représentation de l'orientation du solide S par rapport à S_0 . Elle est redondante en nombre de termes liés entre eux; mais elle reste très significative dans les traitements des modèles utilisés.

Représentation des angles d'Euler

Il existe plusieurs types de représentation par les angles d'Euler suivant les axes de rotation adoptées. On utilisera dans cette partie les angles d'Euler identifiés par angles d'Euler ZYX et angles d'Euler ZYZ. On déterminera par la suite le mode de passage de ces représentations avec la matrice d'orientation.

Angles d'Euler ZYX

Soit à déterminer l'orientation du repère R_s par rapport au repère R_0 .

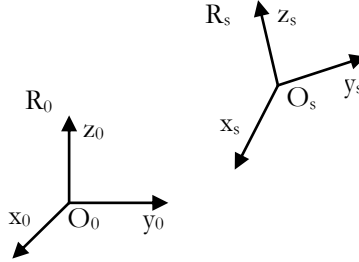


Figure 1.7: Orientation des repères.

Le problème est donc de ramener le repère R_0 coïncidant avec R_s par trois rotations successives.

$$1. \quad R(z_0, \alpha) = R_z = \begin{pmatrix} c\alpha & -s\alpha & 0 \\ s\alpha & c\alpha & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad (1.19)$$

$$2. \quad R(y_1, \beta) = R_y = \begin{pmatrix} c\beta & 0 & s\beta \\ 0 & 1 & 0 \\ -s\beta & 0 & c\beta \end{pmatrix} \quad (1.20)$$

$$3. \quad R(x_2, \gamma) = R_x = \begin{pmatrix} 1 & 0 & 0 \\ 0 & c\gamma & -s\gamma \\ 0 & s\gamma & c\gamma \end{pmatrix} \quad (1.21)$$

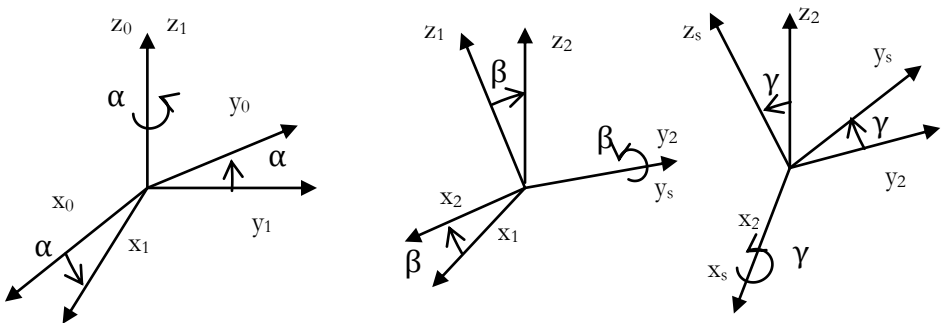


Figure 1.8: Représentation des angles d'Euler ZYX.

L'expression de la matrice des angles d'Euler ZYX est alors définie comme suit:

$$A_{ZYX} = R(z, \alpha) R(y, \beta) R(x, \gamma) \quad (1.22)$$

$$A_{ZYX} = \begin{pmatrix} c\alpha & -s\alpha & 0 \\ s\alpha & c\alpha & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} c\beta & 0 & s\beta \\ 0 & 1 & 0 \\ -s\beta & 0 & c\beta \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & c\gamma & -s\gamma \\ 0 & s\gamma & c\gamma \end{pmatrix} \quad (1.23)$$

$$A_{ZYX} = \begin{pmatrix} c\alpha c\beta & c\alpha s\gamma s\beta - s\alpha c\gamma & c\alpha c\gamma s\beta + s\alpha s\gamma \\ s\alpha c\beta & s\alpha s\gamma s\beta + c\alpha c\gamma & s\alpha c\gamma s\beta - c\alpha s\gamma \\ -s\beta & s\gamma c\beta & c\gamma c\beta \end{pmatrix} \quad (1.24)$$

On peut donc déterminer la matrice d'Euler A_{ZYX} (qui n'est autre que la matrice $A^{0,s}$) à partir des angles α , β et γ .

La fonction en Matab Euler_ZYX.m

```
function R = Euler_ZYX(alpha,beta,gama)

% Permet de déterminer la matrice d'orientation
% A partir des angles d'Euler ZYX
% Calcul de la matrice de rotation

R1 = [cosd(alpha), -sind(alpha), 0;
      sind(alpha), cosd(alpha) 0 ;
      0, 0, 1];
R2 = [cosd(beta), 0, sind(beta);
      0, 1, 0;
      -sind(beta), 0, cosd(beta)];
R3 = [1 0 0;
      0 cosd(gama), -sind(gama);
      0, sind(gama), cosd(gama)];
R = R1*R2*R3;
```

Le problème inverse consiste à trouver les angles d'Euler une fois connue la matrice de d'orientation $A^{0,s}$. En posant l'égalité $A_{ZYX} = A^{0,s}$:

$$\begin{pmatrix} c\alpha c\beta & c\alpha s\gamma s\beta - s\alpha c\gamma & c\alpha c\gamma s\beta + s\alpha s\gamma \\ s\alpha c\beta & s\alpha s\gamma s\beta + c\alpha c\gamma & s\alpha c\gamma s\beta - c\alpha s\gamma \\ -s\beta & s\gamma c\beta & c\gamma c\beta \end{pmatrix} = \begin{pmatrix} a_x & b_x & c_x \\ a_y & b_y & c_y \\ a_z & b_z & c_z \end{pmatrix} \quad (1.25)$$

L'identification des termes adéquats des deux matrices conduit aux résultats suivants:

$$\beta = \text{Atan2}\left(-a_z, \sqrt{a_x^2 + a_y^2}\right) \quad (1.26)$$

$$\alpha = \text{Atan2}\left(a_y/c_\beta, a_x/c_\beta\right) \quad (1.27)$$

$$\gamma = \text{Atan2}\left(b_z/c_\beta, c_z/c_\beta\right) \quad (1.28)$$

La fonction de Matab qui permet de résoudre ce problème inverse est présentée comme suit:

```
function [alpha, beta,gama]=Inv_Euler_ZYX(R)

% Permet de déterminer les angles d'EULER ZYX
% à partir de la matrice d'orientation
% Calcul des angles

beta=atan2d(-R(3,1),sqrt(R(1,1)^2+R(2,1)^2));
alpha= atan2d( R(2,1)/cosd(beta),R(1,1)/cosd(beta));
gama= atan2d(R(3,2)/cosd(beta),R(3,3)/cosd(beta));
```

Angles d'Euler ZYZ

Dans ce cas, les rotations avec les angles α , β et γ se font de la manière suivante:

$$R(z_0, \alpha) = R_z = \begin{pmatrix} c\alpha & -s\alpha & 0 \\ s\alpha & c\alpha & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad (1.29)$$

$$R(y_1, \beta) = R_y = \begin{pmatrix} c\beta & 0 & s\beta \\ 0 & 1 & 0 \\ -s\beta & 0 & c\beta \end{pmatrix} \quad (1.30)$$

$$R(z_2, \gamma) = R_z = \begin{pmatrix} c\gamma & -s\gamma & 0 \\ s\gamma & c\gamma & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad (1.31)$$

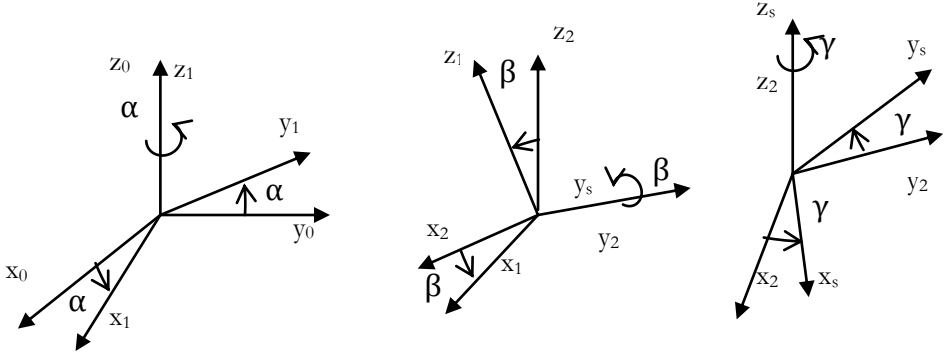


Figure 1.9: Représentation des angles d'Euler ZYZ.

On peut alors déterminer la matrice des angles d'Euler ZYZ comme suit:

$$A_{ZYZ} = R(z, \alpha) R(y, \beta) R(z, \gamma) \quad (1.32)$$

$$A_{ZYZ} = \begin{pmatrix} c\alpha & -s\alpha & 0 \\ s\alpha & c\alpha & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} c\beta & 0 & s\beta \\ 0 & 1 & 0 \\ -s\beta & 0 & c\beta \end{pmatrix} \begin{pmatrix} c\gamma & -s\gamma & 0 \\ s\gamma & c\gamma & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad (1.33)$$

$$A_{ZYZ} = \begin{pmatrix} c\alpha c\gamma c\beta - s\alpha s\gamma & -c\alpha s\gamma c\beta - s\alpha c\gamma & c\alpha s\beta \\ s\alpha c\gamma c\beta + c\alpha c\gamma & -s\alpha s\gamma c\beta + c\alpha c\gamma & s\alpha s\beta \\ -s\beta c\gamma & s\gamma s\beta & c\beta \end{pmatrix} \quad (1.34)$$

La fonction de Matlab Euler_ZYZ .m qui permet de déterminer la matrice d'orientation à partir des angles d'Euler est définie comme suit:

```
function R = Euler_ZYZ(alpha,beta,gama)

% Permet de déterminer la matrice d'orientation
% à partir des angles d'Euler ZYZ
% Calcul de la matrice de Rotation

R1 = [cosd(alpha), -sind(alpha), 0;
      sind(alpha), cosd(alpha) 0 ;
      0, 0, 1];
R2 = [cosd(beta), 0, sind(beta);
      0, 1, 0;
      -sind(beta), 0, cosd(beta)];
R3 = [cosd(gama), -sind(gama), 0;
      sind(gama), cosd(gama), 0;
      0, 0, 1];
R = R1*R2*R3;
```

Le problème inverse est alors élaboré comme suit:

$$\begin{pmatrix} c\alpha c\gamma c\beta - s\alpha s\gamma & -c\alpha s\gamma c\beta - s\alpha c\gamma & c\alpha s\beta \\ s\alpha c\gamma c\beta + c\alpha c\gamma & -s\alpha s\gamma c\beta + c\alpha c\gamma & s\alpha s\beta \\ -s\beta c\gamma & s\gamma s\beta & c\beta \end{pmatrix} = \begin{pmatrix} a_x & b_x & c_x \\ a_y & b_y & c_y \\ a_z & b_z & c_z \end{pmatrix} \quad (1.35)$$

Après identification, on obtient les expressions suivantes:

$$\beta = \text{Atan2}\left(\sqrt{a_z^2 + b_z^2}, c_z\right) \quad (1.36)$$

$$\alpha = \text{Atan2}\left(c_y/s\beta, c_x/s\beta\right) \quad (1.37)$$

$$\gamma = \text{Atan2}\left(b_z/s\beta, -a_z/s\beta\right) \quad (1.38)$$

La fonction Inv_Euler_ZYZ.m permet de trouver la solution du problème inverse.

```
function [alpha, beta,gama]=Inv_Euler_ZYZ(R)

% Permet de déterminer les angles d'EULER ZYZ
% à partir de la matrice d'orientation
% Calcul des angles

beta=atan2d(sqrt(R(3,1)^2+R(3,2)^2),R(3,3));
```



```
alpha= atan2d( R(2,3)/sind(beta),R(1,3)/sind(beta) );
gama= atan2d(R(3,2)/sind(beta),-R(3,1)/sind(beta) );
```

Les quaternions

Les quaternions, appelés aussi les paramètres d'Euler ou les paramètres d'Olinde_Rodrigues, sont définis par 4 paramètres $\lambda_1, \lambda_2, \lambda_3$ et λ_4 . Le principe de l'orientation étant celui de rotation du solide S d'un angle θ autour d'un axe de direction u , pour ramener le repère R_0 coïncidant avec le repère R_s .

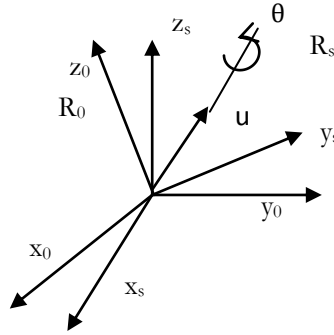


Figure 1.10: Rotation du repère R_s par rapport au repère R_0 .

Les paramètres d'Euler sont alors définis comme suit:

$$\lambda_1 = u_x \sin(\theta/2) \quad (1.39)$$

$$\lambda_2 = u_y \sin(\theta/2) \quad (1.40)$$

$$\lambda_3 = u_z \sin(\theta/2) \quad (1.41)$$

$$\lambda_4 = \cos(\theta/2) \quad (1.42)$$

Avec la relation suivante entre les quatre paramètres:

$$\lambda_1^2 + \lambda_2^2 + \lambda_3^2 + \lambda_4^2 = 1 \quad (1.43)$$

La matrice de rotation correspondant aux paramètres d'Euler est:

$$A_E = \begin{pmatrix} 1 - 2\lambda_2^2 - 2\lambda_3^2 & 2(\lambda_1 \lambda_2 - \lambda_3 \lambda_4) & 2(\lambda_1 \lambda_3 + \lambda_2 \lambda_4) \\ 2(\lambda_1 \lambda_2 + \lambda_3 \lambda_4) & 1 - 2\lambda_1^2 - 2\lambda_3^2 & 2(\lambda_3 \lambda_2 - \lambda_1 \lambda_4) \\ 2(\lambda_1 \lambda_3 - \lambda_2 \lambda_4) & 2(\lambda_3 \lambda_2 + \lambda_1 \lambda_4) & 1 - 2\lambda_1^2 - 2\lambda_2^2 \end{pmatrix} \quad (1.42)$$

Le programme `Parm_quaternions.m` détermine les paramètres d'Euler une fois connus l'angle de rotation t et le vecteur de rotation $K=[K_x \ K_y \ K_z]$.

```
function [a1,a2,a3,a4] = Parm_quaternions(t,Kx,Ky,Kz)

% Permet de déterminer la matrice d'orientation
% à partir des quaternions

K=[Kx Ky Kz];
n=norm (K);
a1=(Kx/n)*sind(t/2);
a2=(Ky/n)*sind(t/2);
a3=(Kz/n)*sind(t/2);
a4=cosd(t/2);
```

Le programme `quaternions.m` de Matlab permet de déterminer la matrice de rotation à partir des quaternions.

```
function R=quaternions(a1,a2,a3,a4)

% Calcul des coefficients de la matrice d'orientation

R(1,1)= 1-2*a2^2-2*a3^2;
R(1,2)= 2*(a1*a2-a3*a4);
R(1,3)= 2*(a1*a3+a2*a4);
R(2,1)= 2*(a1*a2+a3*a4);
R(2,2)= 1-2*a1^2-2*a3^2;
R(2,3)= 2*(a2*a3-a1*a4);
R(3,1)= 2*(a1*a3-a2*a4);
R(3,2)= 2*(a2*a3+a1*a4);
R(3,3)= 1-2*a1^2-2*a2^2;
```

Etant donné la matrice d'orientation $A^{0,s}$, son identification avec la matrice A_E donne:

$$\lambda_1 = \frac{(b_z - c_y)}{4\lambda_4} \quad (1.44)$$

$$\lambda_2 = \frac{(c_x - a_z)}{4\lambda_4} \quad (1.45)$$

$$\lambda_3 = \frac{(a_y - b_x)}{4\lambda_4} \quad (1.46)$$

$$\lambda_4 = 1/2\sqrt{(1 + a_x + b_y + c_z)} \quad (1.47)$$

Ainsi le programme `Inv_Quaternions.m` permet la détermination des quaternions à partir de la matrice de rotation.

```
function [a1,a2,a3,a4]=Inv_Quaternions(R)

% Permet de déterminer les quaternions
% à partir de la matrice d'orientation

a4=0.5*sqrt(1+R(1,1)+R(2,2)+R(3,3));
a1=(R(3,2)-R(2,3))/(4*a4);
a2=(R(1,3)-R(3,1))/(4*a4);
a3=(R(2,1)-R(1,2))/(4*a4);
```

Interface graphique des transformations

L'interface graphique élaborée avec les programmes `transformation.m` et `transformation.fig`, présente les possibilités de passage des angles d'Euler ZYZ, ZYX et les quaternions à la matrice de rotation et vice versa.

```
function varargout = transformation(varargin)
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                  'gui_Singleton',   gui_Singleton, ...
                  'gui_OpeningFcn',   @transformations_OpeningFcn, ...
                  'gui_OutputFcn',    @transformations_OutputFcn, ...
                  'gui_LayoutFcn',    [] , ...
                  'gui_Callback',     []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end
if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State,
                                         varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
```

```

function transformations_OpeningFcn(hObject, eventdata, handles,
varargin)
    handles.output = hObject;
    guidata(hObject, handles);

function varargout = transformations_OutputFcn(hObject,
    eventdata, handles)
    varargout{1} = handles.output;

function r11_matrix_Callback(hObject, eventdata, handles)
input = str2num(get(hObject, 'String'));
    % checks to see if input is empty. if so, default
    % input1_editText to zero
    if (isempty(input))
        set(hObject, 'String', '0')
    end
    guidata(hObject, handles);

function r11_matrix_CreateFcn(hObject, eventdata, handles)
    if ispc && isequal(get(hObject,'BackgroundColor'),
        get(0,'defaultUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
    end

function r12_matrix_Callback(hObject, eventdata, handles)
input = str2num(get(hObject, 'String'));
    if (isempty(input))
        set(hObject, 'String', '0')
    end
    guidata(hObject, handles);

function r12_matrix_CreateFcn(hObject, eventdata, handles)
    if ispc && isequal(get(hObject,'BackgroundColor'),
        get(0,'defaultUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
    end

function r13_matrix_Callback(hObject, eventdata, handles)
input = str2num(get(hObject, 'String'));
    if (isempty(input))
        set(hObject, 'String', '0')
    end
    guidata(hObject, handles);

function r13_matrix_CreateFcn(hObject, eventdata, handles)
    if ispc && isequal(get(hObject,'BackgroundColor'),
        get(0,'defaultUicontrolBackgroundColor'))

```

```

        set(hObject,'BackgroundColor','white');
    end

function r21_matrix_Callback(hObject, eventdata, handles)
    input = str2num(get(hObject, 'String'));
    if (isempty(input))
        set(hObject, 'String', '0')
    end
    guidata(hObject, handles);

function r21_matrix_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
    get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function r22_matrix_Callback(hObject, eventdata, handles)
    input = str2num(get(hObject, 'String'));
    if (isempty(input))
        set(hObject, 'String', '0')
    end
    guidata(hObject, handles);

function r22_matrix_CreateFcn(hObject, eventdata, handles)
    if ispc && isequal(get(hObject,'BackgroundColor'),
        get(0,'defaultUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
    end

function r23_matrix_Callback(hObject, eventdata, handles)
    input = str2num(get(hObject, 'String'));
    if (isempty(input))
        set(hObject, 'String', '0')
    end
    guidata(hObject, handles);

function r23_matrix_CreateFcn(hObject, eventdata, handles)
    if ispc && isequal(get(hObject,'BackgroundColor'),
        get(0,'defaultUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
    end

function r31_matrix_Callback(hObject, eventdata, handles)
    input = str2num(get(hObject, 'String'));
    if (isempty(input))
        set(hObject, 'String', '0')
    end
    guidata(hObject, handles);

function r31_matrix_CreateFcn(hObject, eventdata, handles)

```

```

    if ispc && isequal(get(hObject,'BackgroundColor'),
        get(0,'defaultUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
    end

function r32_matrix_Callback(hObject, eventdata, handles)
    input = str2num(get(hObject, 'String'));
    if (isempty(input))
        set(hObject, 'String', '0')
    end
    guidata(hObject, handles);

function r32_matrix_CreateFcn(hObject, eventdata, handles)
    if ispc && isequal(get(hObject,'BackgroundColor'),
        get(0,'defaultUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
    end

function r33_matrix_Callback(hObject, eventdata, handles)
    input = str2num(get(hObject, 'String'));
    if (isempty(input))
        set(hObject, 'String', '0')
    end
    guidata(hObject, handles);

function r33_matrix_CreateFcn(hObject, eventdata, handles)
    if ispc && isequal(get(hObject,'BackgroundColor'),
        get(0,'defaultUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
    end

function alpha_ZYZ_Callback(hObject, eventdata, handles)
    input = str2num(get(hObject, 'String'));
    if (isempty(input))
        set(hObject, 'String', '0')
    end
    guidata(hObject, handles);

function alpha_ZYZ_CreateFcn(hObject, eventdata, handles)
    if ispc && isequal(get(hObject,'BackgroundColor'),
        get(0,'defaultUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
    end

function beta_ZYZ_Callback(hObject, eventdata, handles)
    input = str2num(get(hObject, 'String'));
    if (isempty(input))
        set(hObject, 'String', '0')
    end

```

```

guidata(hObject, handles);

function beta_ZYZ_CreateFcn(hObject, eventdata, handles)
    if ispc && isequal(get(hObject,'BackgroundColor'),
        get(0,'defaultUiControlBackgroundColor'))
        set(hObject,'BackgroundColor','white');
    end

function gama_ZYZ_Callback(hObject, eventdata, handles)
    input = str2num(get(hObject, 'String'));
    if (isempty(input))
        set(hObject, 'String', '0')
    end
    guidata(hObject, handles);

function gama_ZYZ_CreateFcn(hObject, eventdata, handles)
    if ispc && isequal(get(hObject,'BackgroundColor'),
        get(0,'defaultUiControlBackgroundColor'))
        set(hObject,'BackgroundColor','white');
    end

function R_ZYZ_Callback(hObject, eventdata, handles)
    a11 = get(handles.r11_matrix, 'String'); R(1,1)= str2num(a11);
    a12 = get(handles.r12_matrix, 'String'); R(1,2)= str2num(a12);
    a13 = get(handles.r13_matrix, 'String'); R(1,3)= str2num(a13);
    a21 = get(handles.r21_matrix, 'String'); R(2,1)= str2num(a21);
    a22 = get(handles.r22_matrix, 'String'); R(2,2)= str2num(a22);
    a23 = get(handles.r23_matrix, 'String'); R(2,3)= str2num(a23);
    a31 = get(handles.r31_matrix, 'String'); R(3,1)= str2num(a31);
    a32 = get(handles.r32_matrix, 'String'); R(3,2)= str2num(a32);
    a33 = get(handles.r33_matrix, 'String'); R(3,3)= str2num(a33);

    [alp, bet, gam]=Inv_Euler_ZYZ(R);
    alpha = num2str(alp); beta= num2str(bet); gama = num2str(gam);
    set(handles.alpha_ZYZ, 'String', alpha);
    set(handles.beta_ZYZ, 'String', beta);
    set(handles.gama_ZYZ, 'String', gama);

function ZYZ_R_Callback(hObject, eventdata, handles)
    a = get(handles.alpha_ZYZ, 'String'); A(1)= str2num(a);
    b = get(handles.beta_ZYZ, 'String'); A(2)= str2num(b); c =
    get(handles.gama_ZYZ, 'String'); A(3)= str2num(c);
    R = Euler_ZYZ(A);
    r11 = num2str(R(1,1)); r12 = num2str(R(1,2)); r13 =
    num2str(R(1,3));
    r21 = num2str(R(2,1)); r22 = num2str(R(2,2)); r23 =
    num2str(R(2,3));
    r31 = num2str(R(3,1)); r32 = num2str(R(3,2)); r33 =
    num2str(R(3,3));
    set(handles.r11_matrix, 'String', r11); set(handles.r12_matrix,

```

```

    'String',r12);set(handles.r13_matrix,'String',r13);
    set(handles.r21_matrix,'String',r21);set(handles.r22_matrix,
    'String',r22);set(handles.r23_matrix,'String',r23);
    set(handles.r31_matrix,'String',r31);set(handles.r32_matrix,
    'String',r32);set(handles.r33_matrix,'String',r33);
    guidata(hObject, handles);

```

```

function reset_Callback(hObject, eventdata, handles)
    set(handles.r11_matrix,'String',0);set(handles.r12_matrix,
    'String',0);set(handles.r13_matrix,'String',0);
    set(handles.r21_matrix,'String',0);
    set(handles.r22_matrix,'String',0);set(handles.r23_matrix,
    'String',0);
    set(handles.r31_matrix,'String',0);set(handles.r32_matrix,
    'String',0);set(handles.r33_matrix,'String',0);
    set(handles.alpha_ZYZ,'String',0);
    set(handles.beta_ZYZ,'String',0);
    set(handles.gama_ZYZ,'String',0);
    set(handles.alpha_ZYX,'String',0);
    set(handles.beta_ZYX,'String',0);
    set(handles.gama_ZYX,'String',0);
    set(handles.a1_Quat,'String',0);
    set(handles.a2_Quat,'String',0);
    set(handles.a3_Quat,'String',0);
    set(handles.a4_Quat,'String',0);

```

```

function alpha_ZYX_Callback(hObject, eventdata, handles)
    input = str2num(get(hObject,'String'));
    if (isempty(input))
        set(hObject,'String','0')
    end
    guidata(hObject, handles);

```

```

function alpha_ZYX_CreateFcn(hObject, eventdata, handles)
    if ispc && isequal(get(hObject,'BackgroundColor'),
        set(hObject,'BackgroundColor','white'));
    end

```

```

function beta_ZYX_Callback(hObject, eventdata, handles)
    input = str2num(get(hObject,'String'));
    if (isempty(input))
        set(hObject,'String','0')
    end
    guidata(hObject, handles);

```

```

function beta_ZYX_CreateFcn(hObject, eventdata, handles)
    if ispc && isequal(get(hObject,'BackgroundColor'),
        get(0,'defaultUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
    end

```



```

end

function gama_ZYX_Callback(hObject, eventdata, handles)

    input = str2num(get(hObject, 'String'));
    if (isempty(input))
        set(hObject, 'String', '0')
    end
    guidata(hObject, handles);function gama_ZYX_CreateFcn(hObject,
    eventdata, handles)
    if ispc && isequal(get(hObject,'BackgroundColor'),
        get(0,'defaultUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
    end

function R_ZYX_Callback(hObject, eventdata, handles)

    a11 = get(handles.r11_matrix,'String');R(1,1)= str2num(a11);
    a12 = get(handles.r12_matrix,'String');R(1,2)= str2num(a12);
    a13 = get(handles.r13_matrix,'String');R(1,3)= str2num(a13);
    a21 = get(handles.r21_matrix,'String');R(2,1)= str2num(a21);
    a22 = get(handles.r22_matrix,'String');R(2,2)= str2num(a22);
    a23 = get(handles.r23_matrix,'String');R(2,3)= str2num(a23);
    a31 = get(handles.r31_matrix,'String');R(3,1)= str2num(a31);
    a32 = get(handles.r32_matrix,'String');R(3,2)= str2num(a32);
    a33 = get(handles.r33_matrix,'String');R(3,3)= str2num(a33);

    [alp, bet,gam]=Inv_Euler_ZYX(R);
    alpha = num2str(alp);beta= num2str(bet);gama = num2str(gam);
    set(handles.alpha_ZYX,'String',alpha);
    set(handles.beta_ZYX,'String',beta);
    set(handles.gama_ZYX,'String',gama);

function ZYX_R_Callback(hObject, eventdata, handles)
    a = get(handles.alpha_ZYX,'String');A(1)= str2num(a);
    b = get(handles.beta_ZYX,'String');A(2)= str2num(b);c =
    get(handles.gama_ZYX,'String');A(3)= str2num(c);
    R = Euler_ZYX(A);
    r11 = num2str(R(1,1));r12 = num2str(R(1,2));r13 =
    num2str(R(1,3));
    r21 = num2str(R(2,1));r22 = num2str(R(2,2));r23 =
    num2str(R(2,3));
    r31 = num2str(R(3,1));r32 = num2str(R(3,2));r33 =
    num2str(R(3,3));
    set(handles.r11_matrix,'String',r11);set(handles.r12_matrix,
    'String',r12);set(handles.r13_matrix,'String',r13);
    set(handles.r21_matrix,'String',r21);set(handles.r22_matrix,
    'String',r22);set(handles.r23_matrix,'String',r23);
    set(handles.r31_matrix,'String',r31);set(handles.r32_matrix,
    'String',r32);set(handles.r33_matrix,'String',r33);

```

```

guidata(hObject, handles);

function a1_Quat_Callback(hObject, eventdata, handles)
    input = str2num(get(hObject, 'String'));
    if (isempty(input))
        set(hObject, 'String', '0')
    end
    guidata(hObject, handles);

function a1_Quat_CreateFcn(hObject, eventdata, handles)
    if ispc && isequal(get(hObject, 'BackgroundColor'),
        get(0, 'defaultUicontrolBackgroundColor'))
        set(hObject, 'BackgroundColor', 'white');
    end

function a2_Quat_Callback(hObject, eventdata, handles)
    input = str2num(get(hObject, 'String'));
    if (isempty(input))
        set(hObject, 'String', '0')
    end
    guidata(hObject, handles);

function a2_Quat_CreateFcn(hObject, eventdata, handles)
    if ispc && isequal(get(hObject, 'BackgroundColor'),
        get(0, 'defaultUicontrolBackgroundColor'))
        set(hObject, 'BackgroundColor', 'white');
    end

function a3_Quat_Callback(hObject, eventdata, handles)
    input = str2num(get(hObject, 'String'));
    if (isempty(input))
        set(hObject, 'String', '0')
    end
    guidata(hObject, handles);

function a3_Quat_CreateFcn(hObject, eventdata, handles)
    if ispc && isequal(get(hObject, 'BackgroundColor'),
        get(0, 'defaultUicontrolBackgroundColor'))
        set(hObject, 'BackgroundColor', 'white');
    end

function a4_Quat_Callback(hObject, eventdata, handles)
    input = str2num(get(hObject, 'String'));
    if (isempty(input))
        set(hObject, 'String', '0')
    end
    guidata(hObject, handles);

function a4_Quat_CreateFcn(hObject, eventdata, handles)

```

```

    if ispc && isequal(get(hObject,'BackgroundColor'),
        get(0,'defaultUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
    end

```

```

function R_Quat_Callback(hObject, eventdata, handles)
    a11 = get(handles.r11_matrix,'String');R(1,1)= str2double(a11);
    a12 = get(handles.r12_matrix,'String');R(1,2)= str2double(a12);
    a13 = get(handles.r13_matrix,'String');R(1,3)= str2double(a13);
    a21 = get(handles.r21_matrix,'String');R(2,1)= str2double(a21);
    a22 = get(handles.r22_matrix,'String');R(2,2)= str2double(a22);
    a23 = get(handles.r23_matrix,'String');R(2,3)= str2double(a23);
    a31 = get(handles.r31_matrix,'String');R(3,1)= str2double(a31);
    a32 = get(handles.r32_matrix,'String');R(3,2)= str2double(a32);
    a33 = get(handles.r33_matrix,'String');R(3,3)= str2double(a33);
    [a1,a2,a3,a4]=Inv_Quaternions(R);
    b1 = num2str(a1);b2 = num2str(a2);b3 = num2str(a3);b4 =
    num2str(a4);
    set(handles.a1_Quat,'String',b1);
    set(handles.a2_Quat,'String',b2);
    set(handles.a3_Quat,'String',b3);
    set(handles.a4_Quat,'String',b4);

```

```

function Quat_R_Callback(hObject, eventdata, handles)
    a = get(handles.a1_Quat,'String');a1= str2num(a);
    b = get(handles.a2_Quat,'String');a2= str2num(b);
    c = get(handles.a3_Quat,'String');a3= str2num(c);
    d = get(handles.a4_Quat,'String');a4= str2num(d);
    R = quaternions(a1,a2,a3,a4);
    r11 = num2str(R(1,1));r12 = num2str(R(1,2));r13 =
    num2str(R(1,3));
    r21 = num2str(R(2,1));r22 = num2str(R(2,2));r23 =
    num2str(R(2,3));
    r31 = num2str(R(3,1));r32 = num2str(R(3,2));r33 =
    num2str(R(3,3));
    set(handles.r11_matrix,'String',r11);set(handles.r12_matrix,
    'String',r12);set(handles.r13_matrix,'String',r13);
    set(handles.r21_matrix,'String',r21);set(handles.r22_matrix,
    'String',r22);set(handles.r23_matrix,'String',r23);
    set(handles.r31_matrix,'String',r31);set(handles.r32_matrix,
    'String',r32);set(handles.r33_matrix,'String',r33);
    guidata(hObject, handles);

```



Figure 1.11: *Interface graphique utilisateur des transformations des modes d'orientation.*

Chapitre 2

Représentation de Denavit Hartenberg

Cette représentation est bien adaptée à la modélisation géométrique des robots manipulateurs sériels. Ces derniers sont constitués de bras articulés entre eux successivement entre une base fixe et l'organe terminal. Les articulations peuvent être rotoïdes ou prismatiques.

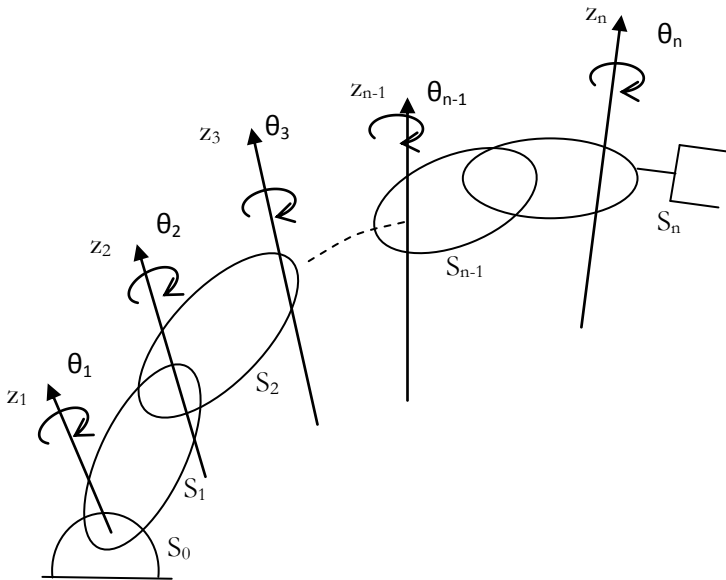


Figure 2.1: Chaîne cinématique d'un robot sériel.

Le principe de la convention de Denavit Hartenberg (DH) est d'attacher les axes z aux axes des articulations et de construire les axes x en perpendiculaires communes aux axes z comme le montre la figure suivante:

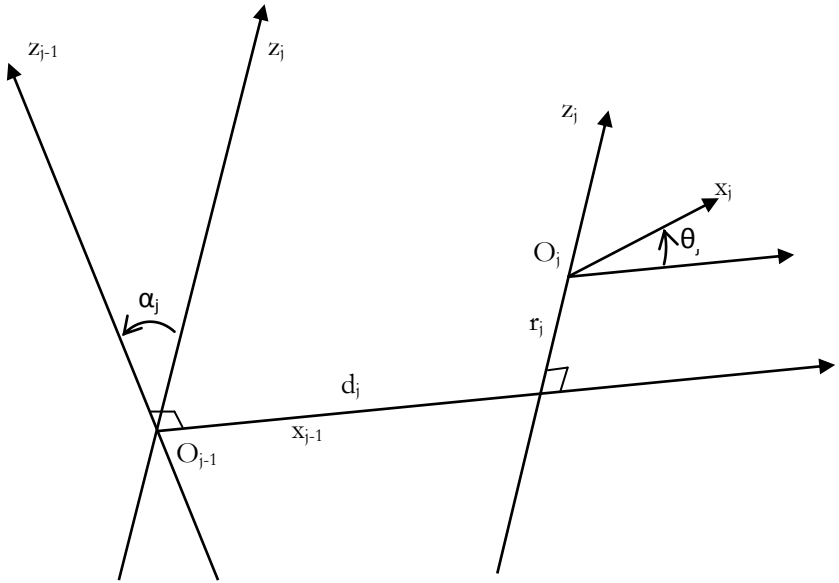


Figure 2.2 : Représentation de Denavit Hartenberg.

d_j : distance entre z_{j-1} et z_j mesurée le long de x_{j-1} .

α_j : angle entre z_{j-1} et z_j mesuré autour de x_{j-1} .

r_j : distance entre x_{j-1} et x_j mesurée le long de z_j .

θ_j : angle entre x_{j-1} et x_j mesuré autour de z_j .

d_j , α_j , r_j et θ_j sont appelés les paramètres de Denavit Hartenberg. Les transformations qui permettent le passage du repère R_{j-1} au repère R_j sont exprimées par la relation suivante:

$$T^{j-1,j} = \text{Rot}(x, \alpha_j) \text{Trans}(x, d_j) \text{Rot}(z, \theta_j) \text{Trans}(z, r_j) \quad (2.1)$$

Une fois que les matrices sont déterminées, la matrice homogène $T^{j-1,j}$ devient égale à :

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & c\alpha_j & -s\alpha_j & 0 \\ 0 & s\alpha_j & c\alpha_j & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & d_j \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} c\theta_j & -s\theta_j & 0 & 0 \\ s\theta_j & c\theta_j & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & r_j \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$T^{j-1,j} = \begin{pmatrix} c\theta_j & -s\theta_j & 0 & d_j \\ c\alpha_j s\theta_j & c\alpha_j c\theta_j & -s\alpha_j & -r_j s\alpha_j \\ s\alpha_j s\theta_j & s\alpha_j c\theta_j & c\alpha_j & r_j c\alpha_j \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad (2.2)$$

Dans certains cas de manipulateurs, les paramètres de DH sont déterminés de la manière suivante:

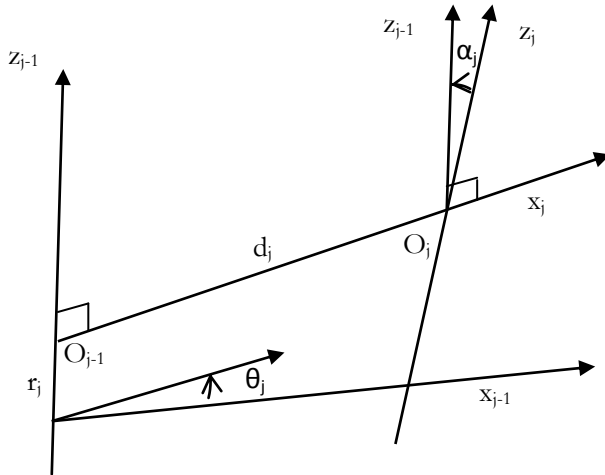


Figure 2.3: Représentation de Denavit Hartenberg.

d_j : distance entre z_{j-1} et z_j mesurée le long de x_j .

α_j : angle entre z_{j-1} et z_j mesurée autour de x_j .

r_j : distance entre x_{j-1} et x_j mesurée le long de z_{j-1} .

θ_j : angle entre x_{j-1} et x_j mesurée autour de z_{j-1} .

$$T^{j-1,j} = Rot(z, \theta_j) Trans(z, r_j) Trans(x, d_j) Rot(x, \alpha_j) \quad (2.3)$$

$$T^{j-1,j} = \begin{pmatrix} c\theta_j & -s\theta_j & 0 & 0 \\ s\theta_j & c\theta_j & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & r_j \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & d_j \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & c\alpha_j & -s\alpha_j & 0 \\ 0 & s\alpha_j & c\alpha_j & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$T^{j-1,j} = \begin{pmatrix} c\theta_j & -c\alpha_j s\theta_j & s\alpha_j s\theta_j & d_j c\theta_j \\ s\theta_j & c\alpha_j c\theta_j & -s\alpha_j c\theta_j & d_j s\theta_j \\ 0 & s\alpha_j & c\alpha_j & r_j \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad (2.4)$$

La fonction de Matlab Denavit1.m permet de déterminer la matrice $T^{j-1,j}$ suivant la première représentation. Tandis que la fonction Denavit2.m est associée à la deuxième représentation.

```
function T = Denavit1(a,d,t,r)

digits(2);
a=sym(a);
T=[cos(t) -sin(t) 0 d;cos(a)*sin(t) cos(a)*cos(t) -sin(a)
   - r*sin(a);...
   sin(a)*sin(t) sin(a)*cos(t) cos(a) r*cos(a);...
   0 0 0 1];
T=vpa(T);
```



```

function T = Denavit2(a,d,t,r)

a=sym(a);
digits(2);
T=[cos(t) ,-cos(a)*sin(t), sin(a)*sin(t) ,d*cos(t);sin(t),
    cos(a)*cos(t), -sin(a)*cos(t), d*sin(t);...
    0 , sin(a), cos(a), r;...
    0 ,0, 0, 1];
T=vpa(T);

```

Remarque: Lorsqu'on écrit $T = \text{Denavit1}(\alpha, d, \text{teta}, a)$, les paramètres α et d sont introduits numériquement. Alors que teta et a peuvent être symboliques (`sym('alpha')`, `sym('a')`) ou numériques suivant la nature de la liaison (prismatique ou rotoïde).

La position et l'orientation de l'organe terminal en fonction des différentes variables articulaires sont rassemblées dans la matrice de transformation homogène suivante:

$$T^{0,n} = \begin{pmatrix} A^{0,n} & O_0 O_n^0 \\ 0 & 1 \end{pmatrix} \quad (2.5)$$

Il est donc intéressant de construire des fonctions de Matlab qui permettent d'obtenir les différentes matrices homogènes ainsi que la matrice $T^{0,n}$.

La fonction `matrices_TH1.m` élabore les matrices de transformation homogène (TH) et la matrice $T^{0,n}$ pour la première représentation de DH.

```

function [A,P,L]= matrices_TH1(alpha,a,teta,r)

n=size(alpha);
n=n(1,2);
H=eye(4,4);
for i=1:n
    T=Denavit1(alpha(i),a(i),teta(i),r(i));
    L(:, :, i)=T;
    H=H*T;
end
A=H(1:3,1:3);
P=H(1:3,4);

```

L'autre fonction `matrices_TH2.m` est destinée pour la deuxième représentation de DH.

```

function [A,P,L]= matrices_TH2(alpha,a,teta,r)

n=size(alpha);
n=n(1,2);
H=eye(4,4);
for i=1:n
    T=Denavit2(alpha(i),a(i),teta(i),r(i));
    L(:,i)=T;
    H=H*T;
end
A=H(1:3,1:3);
P=H(1:3,4);

```

La syntaxe de ces fonctions s'écrit, dans le premier cas, `matrices_TH1(alpha,a,teta,r)`. Les paramètres `alpha`, `d`, `teta` et `a` sont introduits comme des vecteurs relevés à partir de la table de DH représentant le robot. `A` est la matrice d'orientation $A^{0,n}$, `P` est le vecteur de position $O_0O_n^0$ et `L` représente une matrice regroupant toutes les matrices de TH lorsque l'indice de l'articulation varie de 1 jusqu'à `n`. Par exemple `L(:,1)` correspond à la matrice $T^{0,1}$ et `L(:,3)` à $T^{2,3}$.

Chapitre 3

Représentation graphique des robots

La simulation des robots se base principalement sur un mode graphique de représentation. La précision de ce mode dépend de la tâche affectée au robot ainsi que des performances que l'on désire réaliser. On peut alors distinguer trois types de représentations: schématique, approximative et avancée.

Principe de l'animation

L'animation graphique avec Matlab d'un robot passe généralement par les étapes suivantes:

- Initialisation: toutes les données disponibles dans l'espace de travail de Matlab sont effacées et les fenêtres graphiques sont fermées.
- Création des données: les objets définis avant et après les effets des mouvements.
- Tracé initial: une configuration initiale est tracée ce qui permet de créer les objets graphiques utiles pour l'animation.
- Propriétés de l'objet graphique: toutes les propriétés intéressantes des objets (couleur, mode d'effacement,...) sont déterminées.
- Arrangement de la figure: les axes limites de la figure sont définis, les titres, le zoom, le type de vue 2D ou 3D, etc.
- Boucle d'animation: les données des objets sont récupérées pour créer l'animation.

Exemple : cet exemple simple montre les différentes étapes d'animation d'un point se déplaçant suivant une trajectoire axiale sinusoïdale (animation_point.m).

```

% Initialisation

clear; close all;

% création de données

t = 0:0.001:1; % temps
x = sin(2*pi*t); % Position du point

% Tracé initial

figure(1);
set(gcf,'Renderer','OpenGL');
h = plot(x(1),0,'o');

% Propriétés de l'objet graphique

set(h,'Color','g','EraseMode','none',...
    'MarkerSize',10,'MarkerFaceColor','b' );

% Arrangement de la figure

xlim([-1.5,1.5]);
ylim([-1.5,1.5]);

% Boucle d'animation

for i=1:length(x)
    set(h,'XData',x(i));
    drawnow;
end

```

Représentation schématique

Le robot peut être représenté par un ensemble de segments correspondants aux différents corps du mécanisme. Dans la partie de création des données, les informations utiles concernent la trajectoire de l'organe terminal et l'évolution des variables articulaires. Le modèle géométrique direct et inverse permettent d'assurer cette tâche.

Dans le tracé initial, un segment quelconque (passant par deux point $P_1(x_1, y_1, z_1)$ et $P_2(x_2, y_2, z_2)$ peut être défini par l'instruction suivante: `hLine1=line('XData', [x1,x2], 'YData', [y1,y2], 'ZData', [z1,z2])`

Le programme suivant (bar2.m) présente l'animation du robot plan à deux degrés de liberté avec une représentation schématique.

```
%Initialisation

clear all;clf
ERASEMODE = 'normal';
RENDERER = 'painters';

% Création de données

t = 0:0.01:1;
teta= 360*t;
x=400*cosd(teta);y=400*sind(teta); %trajectoire circulaire du
point terminal

% Calcul des angles par le modèle géométrique inverse

[t1,t2]=mgi(x,y);
xa=150*cosd(t1);ya=150*sind(t1); %définition du bras 1
xb=xa+350*cosd(t1+t2);yb=ya+350*sind(t1+t2);%définition du bras 2

% Tracé initial

hLine1 = line('XData',[ 0 xa(1)], 'YData',[0 ya(1)]);
hLine2 = line('XData',[ xa(1) xb(1)], 'YData',[ya(1) yb(1)]);
hold on;
plt=plot(0,400,'r'); % point de la trajectoire circulaire

% Propriétés des objets graphiques

set(hLine1, 'EraseMode',ERASEMODE , 'Color','b'); %premier bras
set(hLine1, 'EraseMode',ERASEMODE , 'Color','b'); %deuxième bras

% Arrangement de la figure

set(gcf,'renderer','zbuffer');
grid on
view(52,30)
title('Trajectoire circulaire du robot plan');
xlabel('X')
ylabel('Y')
zlabel('Z')
axis('equal')
axis('square')
axis([-500 500 -500 500 ]);
rotate3d on
```

```

%Boucle d'animation
for i=1:numel(t)
    set(plt,'xdata',x(1:i),'ydata',y(1:i),...
        'Marker','.', 'MarkerSize',2);
    set(hLine1, 'XData',[ 0 xa(i)], 'YData',
        [0 ya(i)], 'Color','b')
    set(hLine2, 'XData',[ xa(i) xb(i)],
        'YData',[ya(i) yb(i)], 'Color','b')
    drawnow
    pause(.1) % pause dans l'animation
end

```

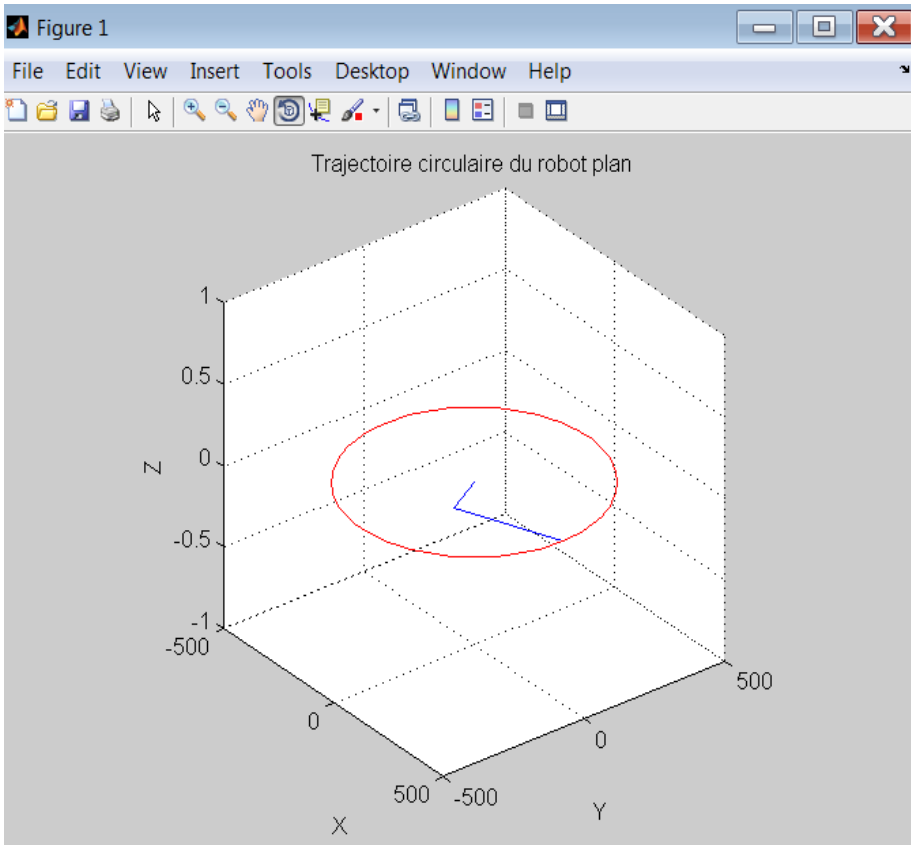


Figure 3.1: Animation schématique du robot plan à deux degrés de liberté.

Représentation approximative

Dans cette représentation, les corps des robots sont définis par l'intermédiaire des formes cylindriques et prismatiques. Ces formes sont alors des primitives de base qui permettent de construire d'autres formes.

Bloc prismatique

Soit un bloc prismatique défini par six faces, huit sommets et les dimensions D_x , D_y et D_z tels que le montre la figure suivante:

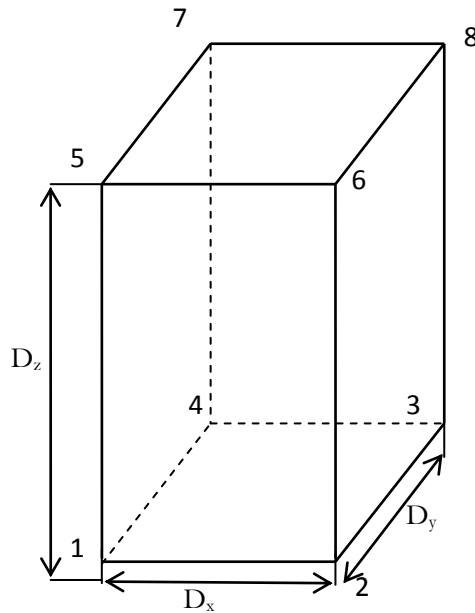


Figure 3.2: *Bloc prismatique.*

Le programme `box_make.m` permet de construire un prisme, une fois que ses dimensions D_x , D_y et D_z sont connues.

```
function [X,Y,Z]=box_make(Dx,Dy,Dz)

% Définition des sommets des six faces

A=[0 Dx Dx 0 0 Dx 0 Dx;0 0 Dy Dy 0 0 Dy Dy;0 0 0 0 Dz Dz Dz Dz];
```

```

X(:,1)=[ A(1,1);A(1,2);A(1,6);A(1,5) ];
X(:,2)=[ A(1,2);A(1,3);A(1,8);A(1,6) ];
X(:,3)=[ A(1,5);A(1,6);A(1,8);A(1,7) ];
X(:,4)=[ A(1,1);A(1,4);A(1,7);A(1,5) ];
X(:,5)=[ A(1,1);A(1,2);A(1,3);A(1,4) ];
X(:,6)=[ A(1,4);A(1,3);A(1,8);A(1,7) ];

Y(:,1)=[ A(2,1);A(2,2);A(2,6);A(2,5) ];
Y(:,2)=[ A(2,2);A(2,3);A(2,8);A(2,6) ];
Y(:,3)=[ A(2,5);A(2,6);A(2,8);A(2,7) ];
Y(:,4)=[ A(2,1);A(2,4);A(2,7);A(2,5) ];
Y(:,5)=[ A(2,1);A(2,2);A(2,3);A(2,4) ];
Y(:,6)=[ A(2,4);A(2,3);A(2,8);A(2,7) ];

Z(:,1)=[ A(3,1);A(3,2);A(3,6);A(3,5) ];
Z(:,2)=[ A(3,2);A(3,3);A(3,8);A(3,6) ];
Z(:,3)=[ A(3,5);A(3,6);A(3,8);A(3,7) ];
Z(:,4)=[ A(3,1);A(3,4);A(3,7);A(3,5) ];
Z(:,5)=[ A(3,1);A(3,2);A(3,3);A(3,4) ];
Z(:,6)=[ A(3,4);A(3,3);A(3,8);A(3,7) ];

```

Le programme `test_prisme.m` permet alors d'afficher le prisme construit précédemment.

```

% Dimensions du prisme

Dx=10;Dy=20;Dz=40;

% Création du prisme
[X,Y,Z]=box_make(Dx,Dy,Dz);

% Affichage du prisme

figure(1);
h = patch(X,Y,Z,'y');
set(h,'FaceLighting','phong','EdgeLighting','phong');
set(h,'EraseMode','normal');

% Propriétés de la figure

xlabel('x','FontSize',14);
ylabel('y','FontSize',14);
zlabel('z','FontSize',14);
set(gca,'FontSize',14);
view([-37.5,30]);
camlight;
grid on;

```



```

xlim([-30,30]);
ylim([-30,30]);
zlim([-10,70]);

```

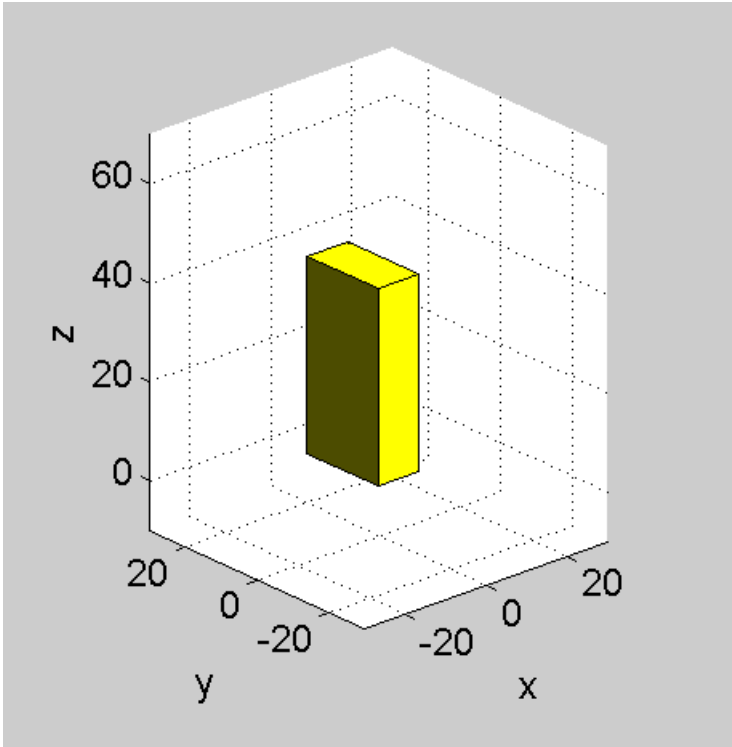


Figure 3.3: *Affichage du prisme.*

Remarque: De la même manière, les solides de différents profils géométriques peuvent être tracés. Il faut alors déterminer tous les sommets et les faces. L'instruction `patch(X,Y,Z,'c')` permet alors de dessiner les contours de toutes les faces.

Bloc cylindrique

Le bloc cylindrique est constitué d'une paroi de plusieurs facettes (n) de hauteur h et de deux faces circulaires aux extrémités. L'axe du cylindre peut être vertical (voir `cyliz.m`), frontal (`cylx.m`) ou de profil (`cyly.m`).

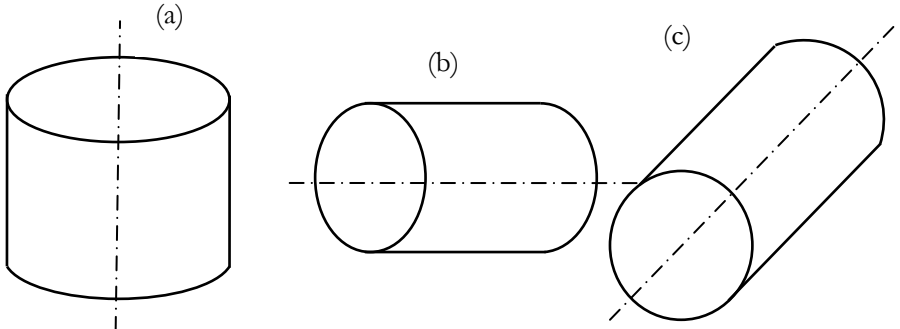


Figure 3.4: *Cylindre vertical (a), cylindre frontal (b) et cylindre de profil (c).*

Les fonctions Matlab associées aux trois types de cylindres sont présentées comme suit:

```
function [PatX,PatY,PatZ]=cylx(h,n)

% Profil circulaire dals le plan yz
i=0;
for t=0:0.1:2*pi
    i=i+1;
    y(i)=10*cos(t);z(i)=10*sin(t);x(i)=0;
end
X(:,1)=x';Y(:,1)=y';Z(:,1)=z';
t=0:0.1:2*pi;
index=numel(t);
X(:,2)=X(:,1)+h;Y(:,2)=Y(:,1);Z(:,2)=Z(:,1);
view(3)
p=round(index/n);
k=0;

% Tracé de parroi cylindrique

for i=1:p:index-p
    k=i;
    PatchX(:,i)=[X(i,1);X(i+p,1);X(i+p,2);X(i,2)];
    PatchY(:,i)=[Y(i,1);Y(i+p,1);Y(i+p,2);Y(i,2)];
    PatchZ(:,i)=[Z(i,1);Z(i+p,1);Z(i+p,2);Z(i,2)];
end
L=k;
```

```

patch(PatchX,PatchY,PatchZ,'y');

% Tracé des faces

for i=1:p:index-p
    PatchX1(:,i)=[X(i,1);X(i+p,1);0;0];
    PatchX2(:,i)=[X(i,2);X(i+p,2);h;h];
    PatchY1(:,i)=[Y(i,1);Y(i+p,1);0;0];
    PatchY2(:,i)=[Y(i,2);Y(i+p,2);0;0];
    PatchZ1(:,i)=[Z(i,1);Z(i+p,1);0;0];
    PatchZ2(:,i)=[Z(i,2);Z(i+p,2);0;0];
end

% Groupement des données

ClosX=[X(1,1);X(L+p,1);X(L+p,2);X(1,2)];
ClosX1=[X(1,1);X(L+p,1);0;0];ClosX2=[X(1,2);X(L+p,2);h;h];
ClosY=[Y(1,1);Y(L+p,1);Y(L+p,2);Y(1,2)];
ClosY1=[Y(1,1);Y(L+p,1);0;0];ClosY2=[Y(1,2);Y(L+p,2);0;0];
ClosZ=[Z(1,1);Z(L+p,1);Z(L+p,2);Z(1,2)];
ClosZ1=[Z(1,1);Z(L+p,1);0;0];ClosZ2=[Z(1,2);Z(L+p,2);0;0];
PatchX=[PatchX ClosX];PatchY=[PatchY ClosY];
PatchZ=[PatchZ ClosZ];
PatchX1=[PatchX1 ClosX1];
PatchY1=[PatchY1 ClosY1];PatchZ1=[PatchZ1 ClosZ1];
PatchX2=[PatchX2 ClosX2];
PatchY2=[PatchY2 ClosY2];PatchZ2=[PatchZ2 ClosZ2];
PatX=[PatchX PatchX1 PatchX2];
PatY=[PatchY PatchY1 PatchY2];
PatZ=[PatchZ PatchZ1 PatchZ2];
patch(PatX,PatY,PatZ,'y');

```

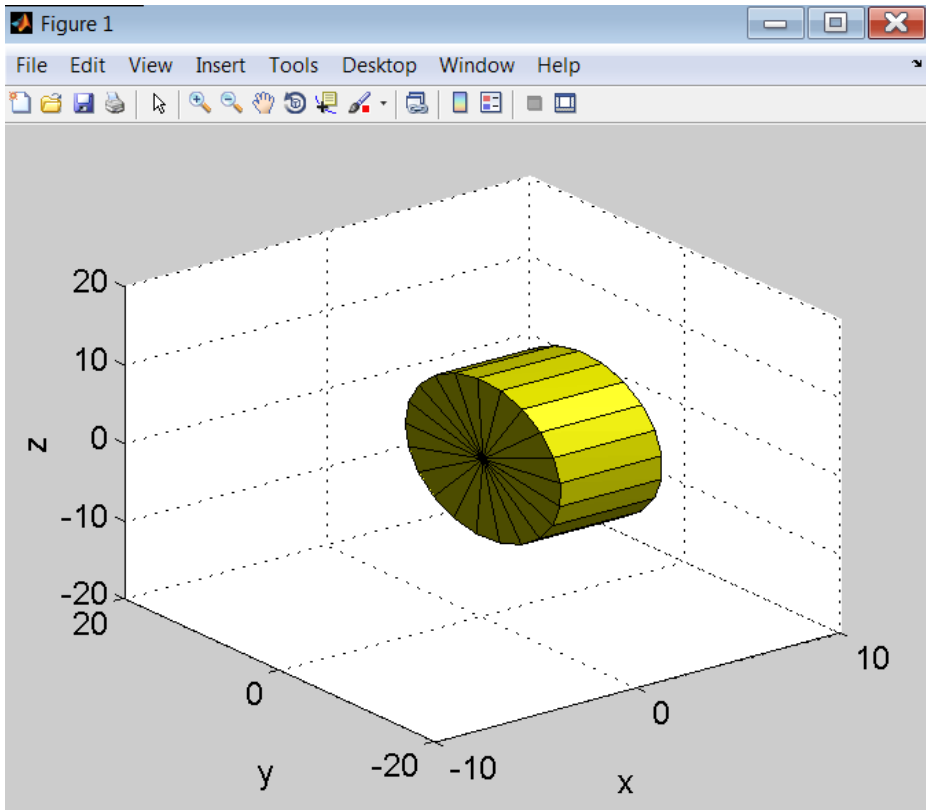


Figure 3.5: *Affichage du cylindre frontal.*

```
function [PatX,PatY,PatZ]=cyl(h,n)

% Profil circulaire dals le plan xz

i=0;
for t=0:0.1:2*pi
    i=i+1;
    x(i)=10*cos(t);z(i)=10*sin(t);y(i)=0;
end
X(:,1)=x';Y(:,1)=y';Z(:,1)=z';
t=0:0.1:2*pi;
index=numel(t);
X(:,2)=X(:,1);Y(:,2)=Y(:,1)+h;Z(:,2)=Z(:,1);
view(3)
p=round(index/n);
```

```

k=0;

% Tracé de parroi cylindrique

for i=1:p:index-p
    k=i;
    PatchX(:,i)=[X(i,1);X(i+p,1);X(i+p,2);X(i,2)];
    PatchY(:,i)=[Y(i,1);Y(i+p,1);Y(i+p,2);Y(i,2)];
    PatchZ(:,i)=[Z(i,1);Z(i+p,1);Z(i+p,2);Z(i,2)];
end
L=k;
patch(PatchX,PatchY,PatchZ,'y');

% Tracé des faces

for i=1:p:index-p
    PatchX1(:,i)=[X(i,1);X(i+p,1);0;0];
    PatchX2(:,i)=[X(i,2);X(i+p,2);0;0];
    PatchY1(:,i)=[Y(i,1);Y(i+p,1);0;0];
    PatchY2(:,i)=[Y(i,2);Y(i+p,2);h;h];
    PatchZ1(:,i)=[Z(i,1);Z(i+p,1);0;0];
    PatchZ2(:,i)=[Z(i,2);Z(i+p,2);0;0];
end

% Groupement des données

ClosX=[X(1,1);X(L+p,1);X(L+p,2);X(1,2)];
ClosX1=[X(1,1);X(L+p,1);0;0];
ClosX2=[X(1,2);X(L+p,2);0;0];
ClosY=[Y(1,1);Y(L+p,1);Y(L+p,2);Y(1,2)];
ClosY1=[Y(1,1);Y(L+p,1);0;0];
ClosY2=[Y(1,2);Y(L+p,2);h;h];
ClosZ=[Z(1,1);Z(L+p,1);Z(L+p,2);Z(1,2)];
ClosZ1=[Z(1,1);Z(L+p,1);0;0];
ClosZ2=[Z(1,2);Z(L+p,2);0;0];
PatchX=[PatchX ClosX];PatchY=[PatchY ClosY];
PatchZ=[PatchZ ClosZ];
PatchX1=[PatchX1 ClosX1];
PatchY1=[PatchY1 ClosY1];PatchZ1=[PatchZ1 ClosZ1];
PatchX2=[PatchX2 ClosX2];
PatchY2=[PatchY2 ClosY2];PatchZ2=[PatchZ2 ClosZ2];
PatX=[PatchX PatchX1 PatchX2];
PatY=[PatchY PatchY1 PatchY2];
PatZ=[PatchZ PatchZ1 PatchZ2];
patch(PatX,PatY,PatZ,'y');

```

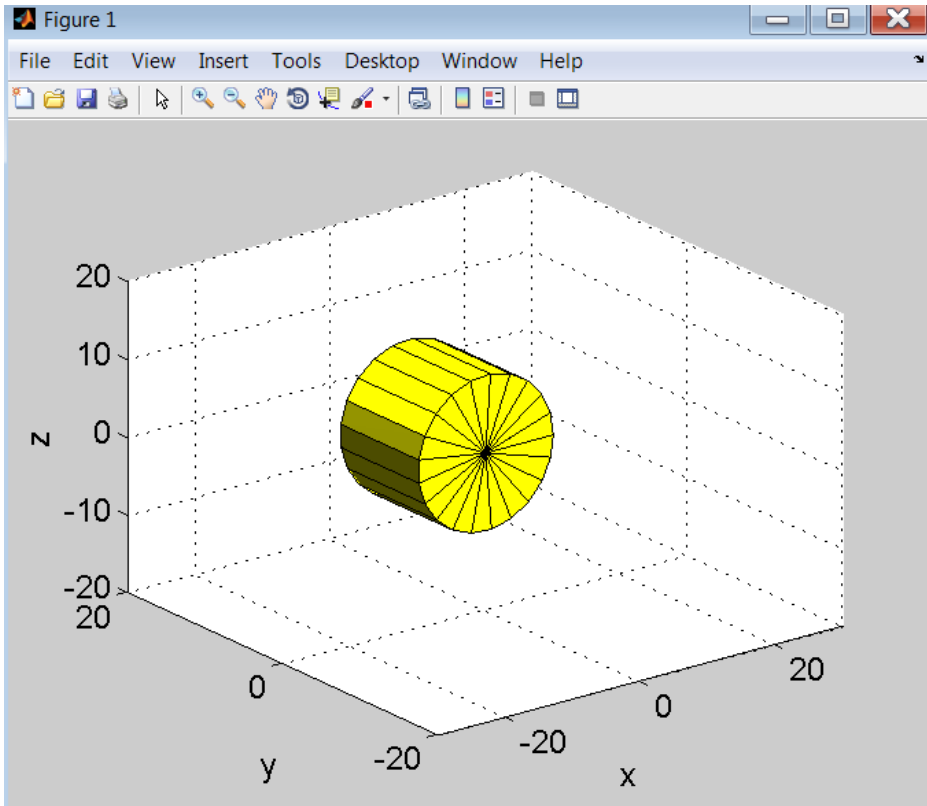


Figure 3.6: *Affichage du cylindre de profil.*

```
function [PatX,PatY,PatZ]=cylz(h,n)

% Profil circulaire dals le plan xy

i=0;
for t=0:0.1:2*pi
    i=i+1;
    x(i)=10*cos(t);y(i)=10*sin(t);z(i)=0;
end
X(:,1)=x';Y(:,1)=y';Z(:,1)=z';
t=0:0.1:2*pi;
index=numel(t);
X(:,2)=X(:,1);Y(:,2)=Y(:,1);Z(:,2)=Z(:,1)+h;
view(3)
```

```

p=round(index/n);
k=0;

% Tracé de parroi cylindrique

for i=1:p:index-p
    k=i;
    PatchX(:,i)=[X(i,1);X(i+p,1);X(i+p,2);X(i,2)];
    PatchY(:,i)=[Y(i,1);Y(i+p,1);Y(i+p,2);Y(i,2)];
    PatchZ(:,i)=[Z(i,1);Z(i+p,1);Z(i+p,2);Z(i,2)];
end
L=k;

% Tracé des faces

for i=1:p:index-p

    PatchX1(:,i)=[X(i,1);X(i+p,1);0;0];
    PatchX2(:,i)=[X(i,2);X(i+p,2);0;0];
    PatchY1(:,i)=[Y(i,1);Y(i+p,1);0;0];
    PatchY2(:,i)=[Y(i,2);Y(i+p,2);0;0];
    PatchZ1(:,i)=[Z(i,1);Z(i+p,1);0;0];
    PatchZ2(:,i)=[Z(i,2);Z(i+p,2);h;h];
end

% Groupement des données

ClosX=[X(1,1);X(L+p,1);X(L+p,2);X(1,2)];
ClosX1=[X(1,1);X(L+p,1);0;0];
ClosX2=[X(1,2);X(L+p,2);0;0];
ClosY=[Y(1,1);Y(L+p,1);Y(L+p,2);Y(1,2)];
ClosY1=[Y(1,1);Y(L+p,1);0;0];ClosY2=[Y(1,2);Y(L+p,2);0;0];
ClosZ=[Z(1,1);Z(L+p,1);Z(L+p,2);Z(1,2)];
ClosZ1=[Z(1,1);Z(L+p,1);0;0];ClosZ2=[Z(1,2);Z(L+p,2);h;h];
PatchX=[PatchX ClosX];
PatchY=[PatchY ClosY];PatchZ=[PatchZ ClosZ];
PatchX1=[PatchX1 ClosX1];
PatchY1=[PatchY1 ClosY1];PatchZ1=[PatchZ1 ClosZ1];
PatchX2=[PatchX2 ClosX2];
PatchY2=[PatchY2 ClosY2];PatchZ2=[PatchZ2 ClosZ2];
PatX=[PatchX PatchX1 PatchX2];
PatY=[PatchY PatchY1 PatchY2];
PatZ=[PatchZ PatchZ1 PatchZ2];
patch(PatX,PatY,PatZ,'b');

```

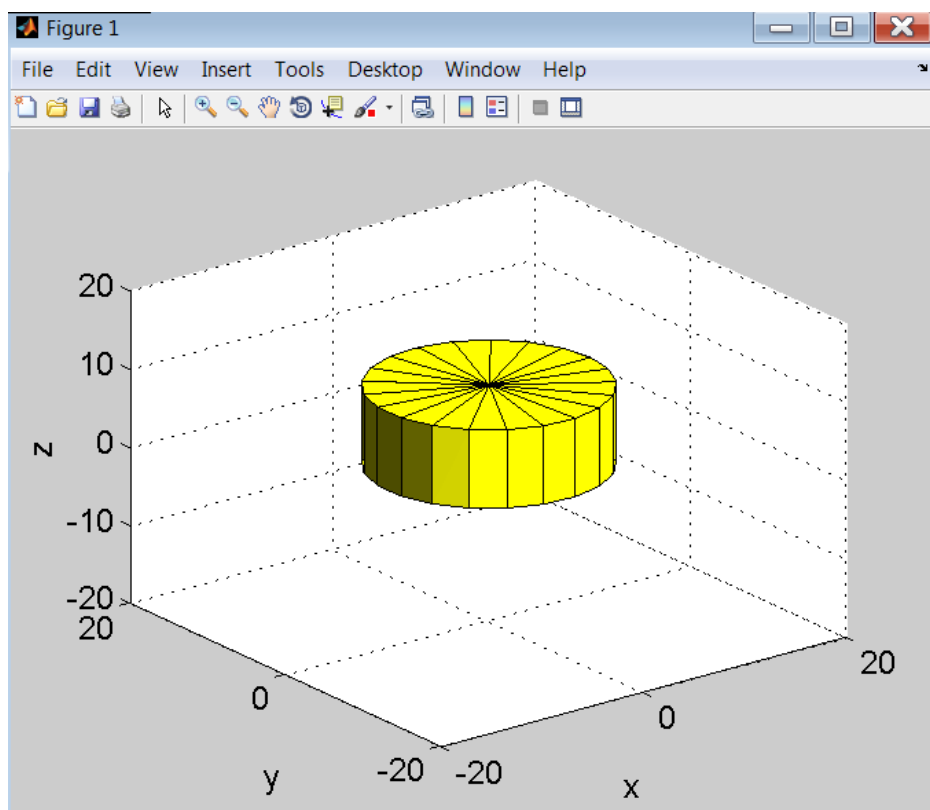


Figure 3.7: *Affichage du cylindre vertical.*

Blocs en mouvements

Le mouvement du bloc cylindrique ou prismatique est entièrement déterminé par la donnée de son déplacement (vecteur D) et de sa rotation (matrice de rotation R).

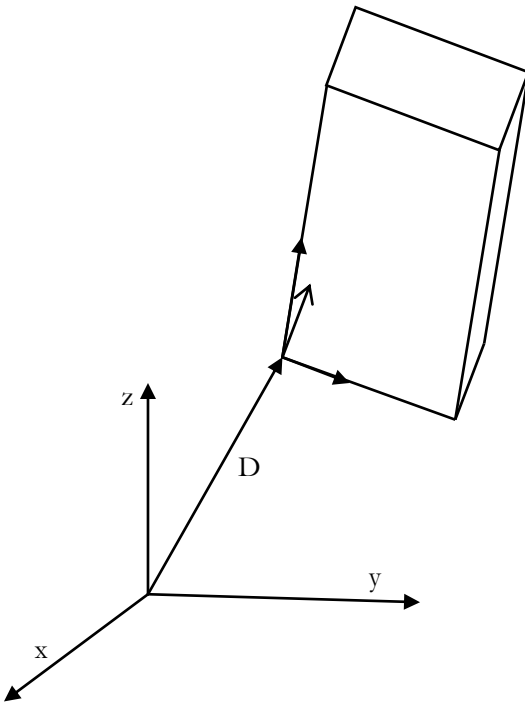


Figure 3.8: Bloc en mouvement.

Les fonctions `moving_box.m` et `cylz_mov.m` permettent d'obtenir respectivement le mouvement d'un prisme et celui d'un cylindre.

```
function [Xm,Ym,Zm]= moving_box(X,Y,Z,D,R)

u=size(X);n=u(1);m=u(2);
for i=1:n
    for j=1:m
        H=R*[X(i,j);Y(i,j);Z(i,j)]+D;
        Xm(i,j)=H(1);
        Ym(i,j)=H(2);
        Zm(i,j)=H(3);
    end
end
```

```

function [Xt,Yt,Zt]=cylz_move(h,n,R,D)

[X,Y,Z]=cylz(h,n);
p=size(X);nj=p(1);ni=p(2);
Xt=zeros(nj,ni);Yt=zeros(nj,ni);Zt=zeros(nj,ni);
for i=1:ni
    for j=1:nj
        H=R*[X(j,i);Y(j,i);Z(j,i)]+D;
        Xt(j,i)=H(1); Yt(j,i)=H(2); Zt(j,i)=H(3);
    end
end
end

```

Les programmes mouvement_box.m et mouvement_cyl.m sont réalisés pour faire l'affichage de ces corps en mouvement.

- mouvement_box.m

```

[X,Y,Z]=box_make(10,20,8);
D=[10;20;30];
R=[1 0 0;0 cosd(45) -sind(45); 0 sind(45) cosd(45)];
[PatchDatam_X,PatchDatam_Y,PatchDatam_Z]= move_box(X,Y,Z,D,R);
patch(PatchDatam_X,PatchDatam_Y,PatchDatam_Z,'b');
xlabel('x','FontSize',14);
ylabel('y','FontSize',14);
zlabel('z','FontSize',14);
set(gca,'FontSize',14);
axis vis3d equal;
view([-37.5,30]);
camlight;
grid on;

```

- mouvement_cyl.m

```

h=10;n=30;
D=[10;20;30];
R=[1 0 0;0 cosd(45) -sind(45); 0 sind(45) cosd(45)];
[Xt,Yt,Zt]=cylz_move(h,n,R,D);
patch(Xt,Yt,Zt,'b');
xlabel('x','FontSize',14);
ylabel('y','FontSize',14);
zlabel('z','FontSize',14);
set(gca,'FontSize',14);
axis vis3d equal;
view([-37.5,30]);
camlight;
grid on;

```

Simulation du robot plan à deux degrés de liberté

Le programme bar2_bloc permet de simuler les mouvements du robot plan à deux degrés de liberté pour une trajectoire circulaire de l'organe terminal.

```
% Initialisation

clear all; close all;

% Spécification de trajectoire
t = [0:0.005:1]';
teta= 360*t;
x=400*cosd(teta);y=400*sind(teta);
[t1,t2]=mgi(x,y);

% Création des blocks

Lx1 = 150; Lx2 = 350;
Ly1 = 10;  Ly2 = 10;
Lz1 = 10;  Lz2 = 10;
n=length(t);
[X1,Y1,Z1]=box_make(Lx1,Ly1,Lz1);
[X2,Y2,Z2]=box_make(Lx2,Ly2,Lz2);

% Mise en mouvement

X1m=zeros(4,6,201);Y1m=zeros(4,6,201);Z1m=zeros(4,6,201);
X2m=zeros(4,6,201);Y2m=zeros(4,6,201);Z2m=zeros(4,6,201);
D1=[0;0;0];
for i=1:n
    D2=[150*cosd(t1(i));150*sind(t1(i));0];
    R1=[cosd(t1(i)) -sind(t1(i)) 0;sind(t1(i)) cosd(t1(i)) 0;
        0 0 1];
    R2=[cosd(t1(i)+t2(i)) -sind(t1(i)+t2(i)) 0;
        sind(t1(i)+t2(i)) cosd(t1(i)+t2(i)) 0;0 0 1];
    [A1,B1,C1]= moving_box(X1,Y1,Z1,D1,R1);
    X1m(:,:,i)= A1; Y1m(:,:,i)= B1; Z1m(:,:,i)= C1;
    [A2,B2,C2]= moving_box(X2,Y2,Z2,D2,R2);
    X2m(:,:,i)= A2; Y2m(:,:,i)= B2; Z2m(:,:,i)= C2;
end

% Tracé initial

figure(1);
h1 = patch(X1m(:,:,1),Y1m(:,:,1),Z1m(:,:,1),'y');
set(h1,'FaceLighting','phong','EdgeLighting','phong');
set(h1,'EraseMode','normal');
hold on;
```

```

h2 = patch(X2m(:,:,1),Y2m(:,:,1),Z2m(:,:,1),'y');hold on
set(h2,'FaceLighting','phong','EdgeLighting','phong');
set(h2,'EraseMode','normal');
plt=plot(0,400,'r');

% Arrangement de figure

xlabel('x','FontSize',14);
ylabel('y','FontSize',14);
zlabel('z','FontSize',14);
set(gca,'FontSize',14);
view([-37.5,30]);
camlight;
grid on;
xlim([-500,500]);
ylim([-500,500]);
zlim([-500,500]);

% Boucle d'animation

for i=1:n
    set(plt,'xdata',x(1:i),'ydata',y(1:i),...
        'Marker','.', 'MarkerSize',2);
    set(h1,'XData',X1m(:,:,i),'YData',Y1m(:,:,i),'ZData',Z1m(:,:,i));
    set(h1,'FaceLighting','phong','EdgeLighting','phong');
    set(h1,'EraseMode','normal');
    set(h2,'XData',X2m(:,:,i),'YData',Y2m(:,:,i),'ZData',Z2m(:,:,i));
    set(h2,'FaceLighting','phong','EdgeLighting','phong');
    set(h2,'EraseMode','normal');
    drawnow;
    pause(0.01);
end

```

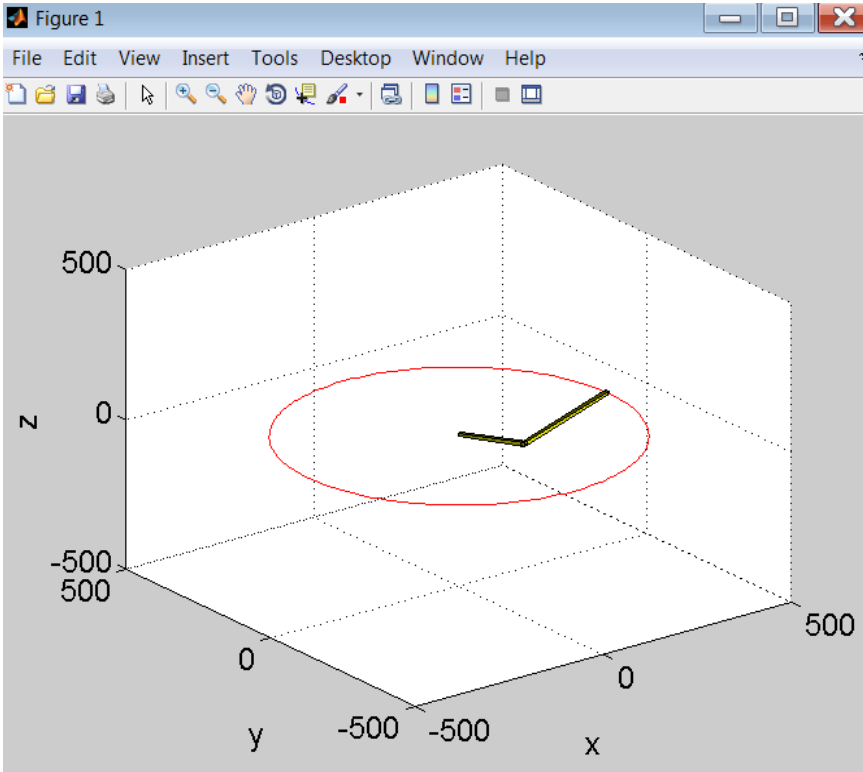


Figure 3.9: Animation avec une représentation approximative du robot.

Représentation avancée

Le robot dessiné dans un environnement CAO est plus réaliste dans ses aspects de visualisation. On peut alors exporter le fichier stl format Ascii du robot à Matlab qui peut l'insérer à l'aide d'un programme de conversion (`rndread.m`)¹. Quand au programme de correction `patchslim.m`², il est destiné à éliminer les sommets et les faces redondants dans le fichier converti en Matlab.

- `rndread.m`

¹ <http://www.mathworks.com/matlabcentral/fileexchange/3642-cad2matdemo-m>

² <http://www.mathworks.com/matlabcentral/fileexchange/29986-patch-slim--patchslim-m>

```

function [fout, vout, cout] = rndread(filename)

% Reads CAD STL ASCII files, which most CAD programs can export.
% Used to create Matlab patches of CAD 3D data.
% Returns a vertex list and face list, for Matlab patch command.
% filename = 'hook.stl'; % Example file.
fid=fopen(filename, 'r'); %Open the file, assumes STL ASCII
format.
if fid == -1
    error('File could not be opened, check name or path.')
end
% Render files take the form:
%solid BLOCK
% color 1.000 1.000 1.000
% facet
%     normal 0.000000e+00 0.000000e+00 -1.000000e+00
%     normal 0.000000e+00 0.000000e+00 -1.000000e+00
%     normal 0.000000e+00 0.000000e+00 -1.000000e+00
%     outer loop
%     vertex 5.000000e-01 -5.000000e-01 -5.000000e-01
%     vertex -5.000000e-01 -5.000000e-01 -5.000000e-01
%     vertex -5.000000e-01 5.000000e-01 -5.000000e-01
%     endloop
%     endfacet
% The first line is object name, then comes multiple facet and
vertex lines.
% A color specifier is next, followed by those faces of that
color, until
% next color line.
CAD_object_name = sscanf(fgetl(fid), '%s %s'); %CAD object
name, if needed.
%Some STLs have it, some don't.
vnum=0; %Vertex number counter.
report_num=0; %Report the status as we go.
VColor = 0;
while feof(fid) == 0 % test for end of file,
if not then do stuff
    tline = fgetl(fid); % reads a line of data
from file.
    fword = sscanf(tline, '%s '); % make the line a
character string
% Check for color
    if strcmpi(fword, 'c',1) == 1; % Checking if a "C"olor line,
as "C" is 1st char.
        VColor = sscanf(tline, '%s %f %f %f');
% & if a C, get the RGB color data of theface.
    end % Keep this color, until the next color is used.
    if strcmpi(fword, 'v',1) == 1;
% Checking if a "V"ertex line, as "V" is 1st char.
        vnum = vnum + 1; % If a V we count the # of V's

```

```

        report_num = report_num + 1;    % Report a counter, so
long files show status
        if report_num > 249;
            disp(sprintf('Reading vertex num: %d.',vnum));
            report_num = 0;
        end
        v(:,vnum) = sscanf(tline, '%*s %f %f %f'); % & if a V, get
the XYZ data of it.
        c(:,vnum) = VColor;
        % A color for each vertex, which will color the faces.
    end
    % we "%s" skip the name "color" and get the data.
end
% Build face list; The vertices are in order, so just number
them.
fnum = vnum/3;          %Number of faces, vnum is number of vertices.
STL is triangles.
flist = 1:vnum;         %Face list of vertices, all in order.
F = reshape(flist, 3,fnum); %Make a "3 by fnum" matrix of face
list data.
% Return the faces and vertexs.
fout = F'; %Orients the array for direct use in patch.
vout = v'; % "
cout = c';%
fclose(fid);

```

- patchslim.m

```

function [vnew, fnew]=patchslim(v, f)

% PATCHSLIM removes duplicate vertices in surface meshes.
% This function finds and removes duplicate vertices.
% USAGE: [v, f]=patchslim(v, f)
% Where v is the vertex list and f is the face list specifying
vertex
% connectivity.
% v contains the vertices for all triangles [3*n x 3].
% f contains the vertex lists defining each triangle face [n x
3].
% This will reduce the size of typical v matrix by about a factor
of 6.
% For more information see:
% http://www.esmonde-white.com/home/diversions/matlab-program-
for-loading-stl-files
% Francis Esmonde-White, May 2010

if ~exist('v','var')
    error('The vertex list (v) must be specified.');
```

```

if ~exist('f','var')
    error('The vertex connectivity of the triangle faces (f) must
be specified. ');
end
[vnew, indexm, indexn] = unique(v, 'rows');
fnew = indexn(f);

```

Dans l'exemple suivant (test_CAO.m) un prisme est dessiné en CAO puis exporté en fichier 'prisme1.stl' (format Ascii) vers Matlab. La procédure de correction a été réalisée par l'instruction `p=patch('faces',fnew,'vertices',vnew)`.

```

clear; close all;
[fout, vout, cout] = rndread('prisme1.stl');
[vnew, fnew]=patchslim(vout, fout);
p = patch('faces', fnew, 'vertices', vnew);
set(p, 'facec', 'r'); % couleur de la face
set(p, 'FaceVertexCData', cout);

% Couleurs à partir du fichier

set(p, 'EdgeColor','none'); % couleur des arrêtes
light % ajout de lumière par défaut
daspect([1 1 1]) % aspect ratio
view(3) % vue isométrique
xlabel('X'),ylabel('Y'),zlabel('Z')

```

Pour traiter le cas d'un robot quelconque dessiné dans un environnement CAO, il faut alors suivre les étapes suivantes:

- Construire les fichiers stl associés à chaque pièce du robot.
- Rassembler ces informations dans un fichier de données Matlab avec une extension.mat.
- Eliminer les données redondantes dans les sommets et les faces élaborés par chaque fichier stl (procédure patchslim.m).
- Initialiser les repères attachés à chaque solide pour pouvoir appliquer les transformations géométriques adéquates du type: $O_0M^0 = O_0O_i^0 + A^{0,i} O_iM^i$.
- Appliquer la procédure d'animation (initialisation-crédation de données-tracé initial-propriétés de l'objet-arrangement de la figure et boucle d'animation).

Après avoir appliqué toutes ces étapes au robot à deux degrés de libertés on aboutit au programme suivant:

```
clear; close all;
load('Link.mat','s1','s2','s3');
v1=s1.V0;f1=s1.F0;[v1, f1]=patchslim(v1, f1);
v2=s2.V0;f2=s2.F0;[v2, f2]=patchslim(v2, f2);
v3=s3.V0;f3=s3.F0;[v3, f3]=patchslim(v3, f3);
s=size(v2);
a1= ones(s(1),1)*0;a2=ones(s(1),1)*0;a3=ones(s(1),1)*20;
u2=[a1 a2 a3];
v2=v2-u2;
s=size(v3);
a1= ones(s(1),1)*40;a2=ones(s(1),1)*0;a3=ones(s(1),1)*26;
u3=[a1 a2 a3];
v3=v3-u3;
t = [0:0.005:1]'; n=length(t);
teta= 360*t;
x=75*cosd(teta);y=50*sind(teta);z=29*ones(n,1);
[t1,t2]=mgi(x,y);

% Mise en mouvement

a=size(v2);
for i=1:n
    R1=[cosd(t1(i)) -sind(t1(i)) 0; sind(t1(i)) cosd(t1(i)) 0;
        0 0 1];
    D1=R1*[0;0;20];
for j=1:a(1)
    H=R1*[v2(j,1);v2(j,2);v2(j,3)]+D1;
    vm2(j,1,i)= H(1);
    vm2(j,2,i)=H(2);
    vm2(j,3,i)=H(3);
end
end
a=size(v3);
for i=1:n
    R2=[cosd(t1(i)+t2(i)) -sind(t1(i)+t2(i)) 0;
        sind(t1(i)+t2(i)) cosd(t1(i)+t2(i))
        0;0 0 1];
    D2=[40*cosd(t1(i));40*sind(t1(i));26];
for j=1:a(1)
    H=R2*[v3(j,1);v3(j,2);v3(j,3)]+D2;
    vm3(j,1,i)= H(1);
    vm3(j,2,i)=H(2);
    vm3(j,3,i)=H(3);
end
end
```

```

% Tracé initial

figure(1);
h2 = patch('faces', f2, 'vertices' ,vm2(:,:,1));
hold on;
h3 = patch('faces', f3, 'vertices' ,vm3(:,:,1));
plt=plot3(0,43.96,16.2,'r');

% Arrangement de figure

xlabel('x','FontSize',14);
ylabel('y','FontSize',14);
zlabel('z','FontSize',14);
set(gca,'FontSize',14);
view(3);
camlight;daspect([1 1 1])
grid on;
xlim([-100,100]);
ylim([-100,100]);
zlim([-30,30]);
p1 = patch('faces', f1, 'vertices' ,v1);
set(p1, 'facec', 'g'); % couleur de la face
set(p1, 'EdgeColor','none'); % couleur des arrêtes
set(p1,'FaceLighting','gouraud ')
set(p1,'EdgeLighting','gouraud ')

% Boucle d'animation

for i=1:n
    set(plt,'xdata',x(1:i),'ydata',y(1:i),...
        'zdata',z(1:i),'Marker','.', 'MarkerSize',2);
    set(h2,'faces', f2,'vertices',vm2(:,:,i));
    set(h2, 'facec', 'r'); % couleur de la face
    set(h2, 'EdgeColor','none'); % couleur des arrêtes
    set(h2,'FaceLighting','gouraud ')
    set(h2,'EdgeLighting','gouraud ')
    set(h3,'faces', f3,'vertices',vm3(:,:,i));
    set(h3,'facec', 'r'); % couleur de la face
    set(h3,'EdgeColor','none'); % couleur des arrêtes
    set(h3,'FaceLighting','gouraud ')
    set(h3,'EdgeLighting','gouraud ')
    drawnow;
    pause(0.01);
end

```

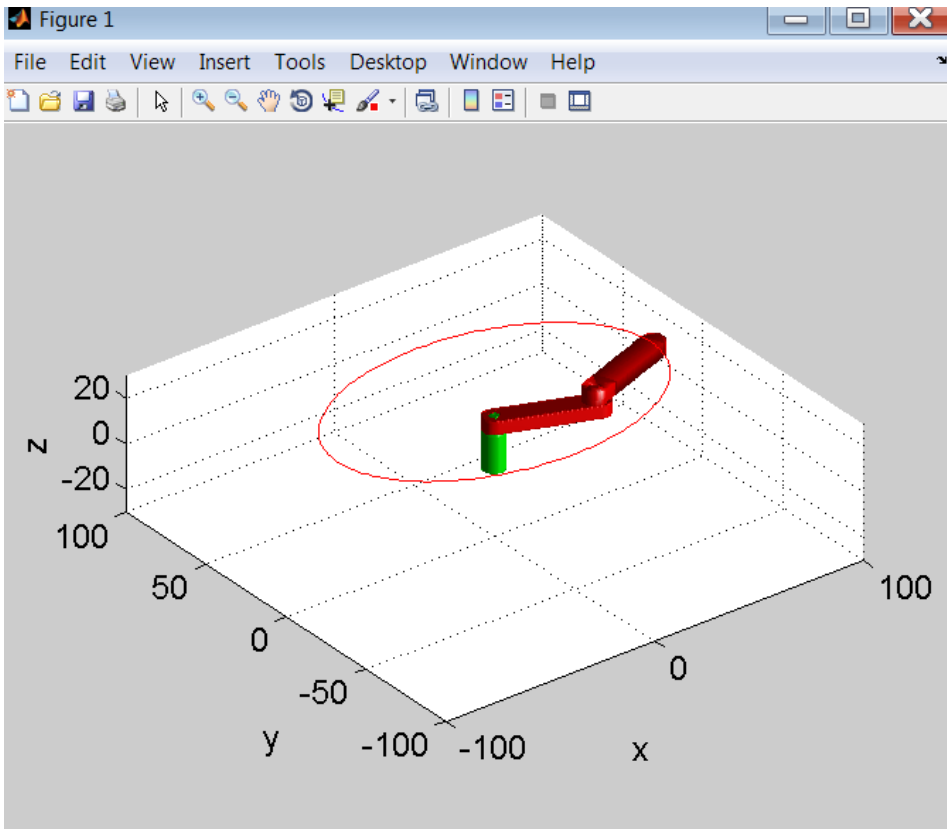


Figure 3.10: *Animation du Robot plan.*

Chapitre 4

Espace de travail des robots manipulateurs

L'espace de travail (ET) est une des caractéristiques importantes que possède un robot. Il est attaché aux possibilités de mouvement de l'organe terminal pour accomplir une tâche spécifique. Les robots sériels disposent généralement d'un espace de travail plus large que les robots parallèles vu la disposition successive des bras depuis la base jusqu'à l'organe terminal.

Définition: L'espace de travail d'un robot est l'ensemble de positions de l'organe terminal lorsque toutes les articulations motorisées sont actives.

Principe d'élaboration

L'espace de travail d'un robot est élaboré en partant de l'espace décrit par la dernière articulation, lorsque les autres sont immobiles, jusqu'à l'espace engendré par la première articulation suivant la démarche suivante:

- E_n : espace décrit par un point de l'organe terminal si seulement la variable articulaire θ_n est active.
- E_{n-1} : espace décrit par l'espace E_n si θ_{n-1} est active.
- .
- .
- E_1 : espace décrit par l'espace E_2 si θ_1 est active.

Remarque: L'espace de travail global du robot n'est autre que l'espace E_1 qui regroupe tous les autres espaces préalables. C'est un espace nécessitant trop de calcul si le nombre de degré de liberté est élevé.

Pour l'exemple du robot plan à deux degrés de liberté, l'espace de travail est déterminé par le programme `Vt_bar2.m`.

```
clear; close all;

% Activation de la deuxième articulation

i=0;
for t2=0:0.3:2*pi
    i=i+1;
    R2=[cos(t2) -sin(t2) 0; sin(t2) cos(t2) 0;0 0 1];
    D2=[150;0 ;0];
    H=R2*[350;0;0]+D2;
    x2(i)=H(1);y2(i)=H(2);z2(i)=H(3);
end
n2=size(x2);
axis([-600 600 -600 600 -20 20])
view(2)

% Activation de la première articulation

i=0;
for t1=0:0.3:2*pi
    i=i+1;
    for j=1:n2(2)
        R1=[cos(t1) -sin(t1) 0; sin(t1) cos(t1) 0;0 0 1];
        H=R1*[x2(j);y2(j);z2(j)];
        x1(j,i)=H(1);y1(j,i)=H(2);z1(j,i)=H(3);
    end
end
surface(x1,y1,z1,z1);hold on;
```

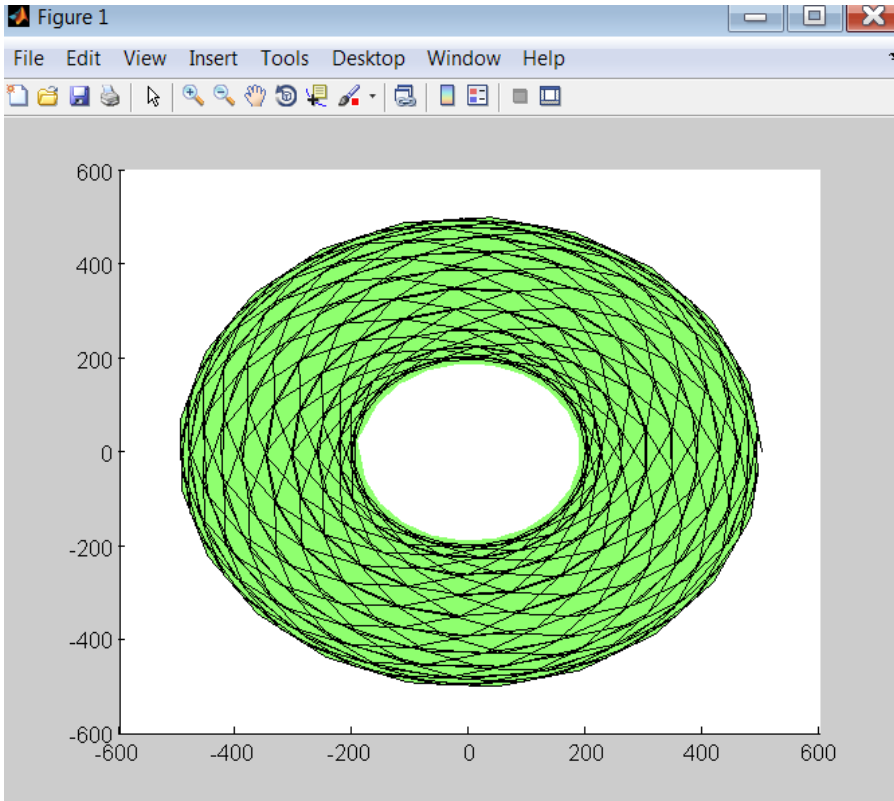


Figure 4.1: *Espace de travail du robot plan à deux ddl.*

Dans le programme `bar2VT_bloc.m`, l'espace de travail est déterminé ainsi que la trajectoire décrite par l'organe terminal. Cette trajectoire est définie par des points prédéfinis (récupérées dans un fichier texte) avec une interpolation polynomiale pour disposer d'une trajectoire continue passant par tous ces points. L'animation est ensuite appliquée avec une représentation approximative.

```
% Initialisation

clear all; close all;

% Espace travail
i=0;
```

```

for t2=-pi/2:0.3:pi/2
    i=i+1;
    R2=[cos(t2) -sin(t2) 0; sin(t2) cos(t2) 0;0 0 1];
    D2=[150;0 ;0];
    H=R2*[350;0;0]+D2;
    x2(i)=H(1);y2(i)=H(2);z2(i)=H(3);
end
n2=size(x2);
axis([-600 600 -600 600 -20 20])
view(2)
=0;
for t1=0:0.3:2*pi
    i=i+1;
    for j=1:n2(2)

        R1=[cos(t1) -sin(t1) 0; sin(t1) cos(t1) 0;0 0 1];
        H=R1*[x2(j);y2(j);z2(j)];
        x1(j,i)=H(1);y1(j,i)=H(2);z1(j,i)=H(3);
    end
end
surface(x1,y1,z1,z1);

% Fin espace travail
% Spécification de trajectoire

load('point.txt')
pt=10*point;
pt(11,2)=10.0479;
a=size(pt);
t=1:a(1); t=t';
px=polyfit(t,pt(:,1),6);
py=polyfit(t,pt(:,2),6);
t=0:0.1:a(1);t=t';
x=polyval(px,t); %interpolation polynomiale
y=polyval(py,t);

% Modèle géométrique inverse

[t1,t2]= mgi(x,y);

% Création des blocks

Lx1 = 150; Lx2 = 350;
Ly1 = 10;  Ly2 = 10;
Lz1 = 10;  Lz2 = 10;
n=length(t);
[X1,Y1,Z1]=box_make(Lx1,Ly1,Lz1);
[X2,Y2,Z2]=box_make(Lx2,Ly2,Lz2);

```



```
% Mise en mouvement
```

```
X1m=zeros(4,6,201);Y1m=zeros(4,6,201);Z1m=zeros(4,6,201);
X2m=zeros(4,6,201);Y2m=zeros(4,6,201);Z2m=zeros(4,6,201);
D1=[0;0;0];
for i=1:n
    D2=[150*cosd(t1(i));150*sind(t1(i));0];
    R1=[cosd(t1(i)) -sind(t1(i)) 0;sind(t1(i)) cosd(t1(i)) 0;0 0 1];
    R2=[cosd(t1(i)+t2(i)) -sind(t1(i)+t2(i)) 0;sind(t1(i)+t2(i))
        cosd(t1(i)+t2(i)) 0;0 0 1];
    [A1,B1,C1]= moving_box(X1,Y1,Z1,D1,R1);
    X1m(:,:,i)= A1; Y1m(:,:,i)= B1; Z1m(:,:,i)= C1;
    [A2,B2,C2]= moving_box(X2,Y2,Z2,D2,R2);
    X2m(:,:,i)= A2; Y2m(:,:,i)= B2; Z2m(:,:,i)= C2;
end
```

```
% Tracé initial
```

```
figure(1);
h1 = patch(X1m(:,:,1),Y1m(:,:,1),Z1m(:,:,1),'y');
set(h1,'FaceLighting','phong','EdgeLighting','phong');
set(h1,'EraseMode','normal');
hold on;
h2 = patch(X2m(:,:,1),Y2m(:,:,1),Z2m(:,:,1),'y');hold on
set(h2,'FaceLighting','phong','EdgeLighting','phong');
set(h2,'EraseMode','normal');
plt=plot(0,400,'r');
```

```
% Arrangement de figure
```

```
xlabel('x','FontSize',14);
ylabel('y','FontSize',14);
zlabel('z','FontSize',14);
set(gca,'FontSize',14);
%axis vis3d equal;
view([-37.5,30]);
camlight;
grid on;
xlim([-500,500]);
ylim([-500,500]);
zlim([-500,500]);
% Boucle d'animation
for i=1:n
    set(plt,'xdata',x(1:i),'ydata',y(1:i),...
        'Marker','.', 'MarkerSize',4);
    set(h1,'XData',X1m(:,:,i),'YData',Y1m(:,:,i),'ZData',Z1m(:,:,i));
    set(h1,'FaceLighting','phong','EdgeLighting','phong');
```

```

set(h1,'EraseMode','normal');
set(h2,'XData',X2m(:,:,i),'YData',Y2m(:,:,i),'ZData',Z2m(:,:,i));
set(h2,'FaceLighting','phong','EdgeLighting','phong');
set(h2,'EraseMode','normal');
drawnow;
pause(0.01);
end

```

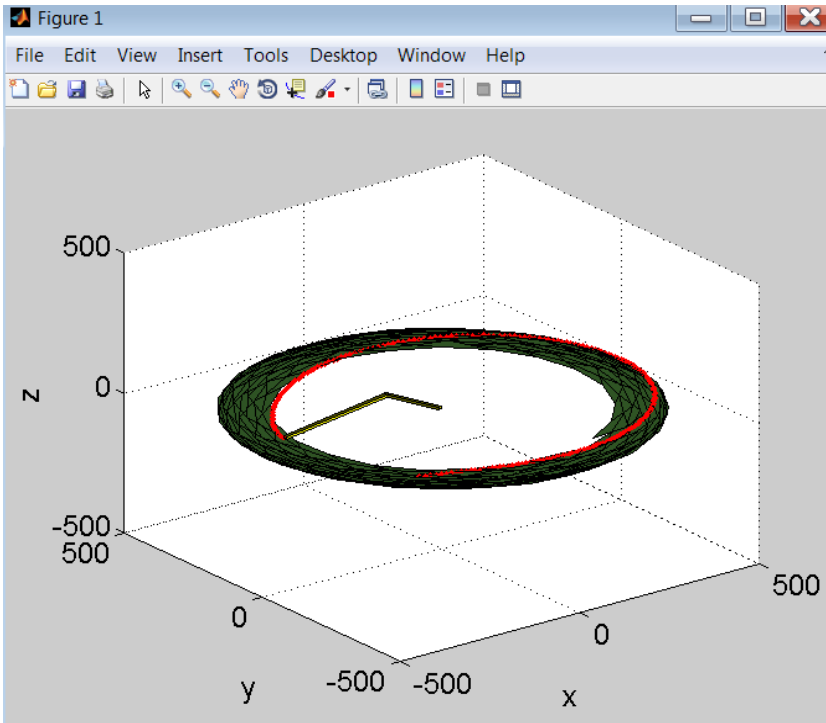


Figure 4.2: Animation avec une trajectoire inscrite dans l'espace de travail.

Remarque: Dans certains cas, il est intéressant d'alléger la représentation de l'espace de travail d'un robot en choisissant le mode de représentation filaire. La fonction `wireframe.m`³ ci-dessous permet la réalisation de ce type d'affichage.

³ <http://www.mathworks.com/matlabcentral/fileexchange/?utf8=%E2%9C%93&term=wireframe.m>

```

function L=wireframe(X,Y,Z,N)

% Default example...
if nargin==0
    f=figure; hold on; axis off; set(f,'color',[0 0 0]);
    N=25; % Undersampling factor...
    [X,Y,Z]=sphere(250);
end
if isscalar(N)
    N= repmat(N,[1 2]);
end
if (isvector(X) & length(X)==size(Z,2)) & (isvector(Y) &
length(Y)==size(Z,1))
    [X,Y]=meshgrid(X,Y);
elseif ~isequal(size(X),size(Y)) | ~isequal(size(Y),size(Z))
    error('Input arguments don't have the right dimensions');
end
XS1=X([1:N(1):end end],:); XS1(:,end+1)=NaN; % Force inclusion
% of last data row & tag with NaN for line breaks

YS1=Y([1:N(1):end end],:); YS1(:,end+1)=NaN;
ZS1=Z([1:N(1):end end],:); ZS1(:,end+1)=NaN;

XS2=X(:,[1:N(2):end end]); XS2(end+1,:)=NaN; % Force inclusion
of last data column & tag with NaN for line breaks
YS2=Y(:,[1:N(2):end end]); YS2(end+1,:)=NaN;
ZS2=Z(:,[1:N(2):end end]); ZS2(end+1,:)=NaN;
set(gcf,'NextPlot','Add'); % Add surf lines to current plot...
L(1)=line(lin(XS1'),lin(YS1'),lin(ZS1')); % Plot the latitudinal
% lines
set(L(1),'Color',[0 0 0]);
L(2)=line(XS2(:),YS2(:),ZS2(:)); % Plot the longitudinal lines
set(L(2),'Color',[0 0 0]);

% Continue with the default example...

if nargin==0
    S=surf(X,Y,Z);
    set(S,'Edgecolor','None','BackfaceLighting','Lit');
    view(45,45);
    l=light;
    axis equal
    camzoom(1.5);
    colormap(jet(1024));
end
return;

% This function linearizes a matrix (m) of any dimension (eg
% M=m(:)).

```

```

% If an index vector (ind) is given then the the ind entries of
% m(:) are
% returned.
% SYNTAX: m=lin(m);
%          m=lin(m,ind);
% DBE 2003/12/22
function m=lin(m,ind);
m=m(:);
if nargin==2
    m=m(ind);
end
return

```

Soit à déterminer l'espace de travail d'un robot sphérique décrit dans le schéma suivant:

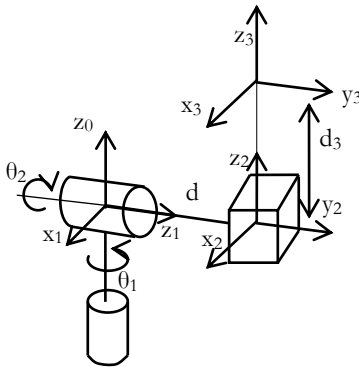


Figure 4.3: Robot sphérique à trois ddl.

Le programme VT_spherical.m permet d'obtenir l'espace de travail du robot sphérique. Celui ci se présente alors sous une forme cylindrique.

```

clear; close all;

% Paramètres du robot

d3i=10;%distance initiale de la liaison prismatique
a1=30;a2=50;

% Activation de la liaison prismatique

i=0;

```

```

for d3=0:15
    i=i+1;
    x3(i)=0;y3(i)=a2;z3(i)=a1+d3i+d3;
end
n3=size(x3);
xlabel('X')
ylabel('Y')
zlabel('Z')
axis([-60 60 -60 60 -10 80]);
rotate3d on
zoom

% Activation de la deuxième articulation

i=0;
for tt2=0:0.1:2*pi
    i=i+1;
    for j=1:n3(2)
        D2=[0;a2;a1];
        R2=[cos(tt2) 0 sin(tt2);0 1 0;-sin(tt2) 0 cos(tt2)];
        H=R2*[x3(j);y3(j)-a2;z3(j)-a1]+D2;
        x2(j,i)=H(1);y2(j,i)=H(2);z2(j,i)=H(3);
    end
end
n2=size(x2); nl=n2(1);nc=n2(2);

% Activation de la première articulation

i=0;
for tt1=0:0.1:2*pi
    i=i+1;
    for j=1:nl
        for k=1:nc
            R1=[cos(tt1) -sin(tt1) 0; sin(tt1) cos(tt1) 0;0 0 1];
            H=R1*[x2(j,k);y2(j,k);z2(j,k)];
            x1(j,k,i)=H(1);y1(j,k,i)=H(2);z1(j,k,i)=H(3);
        end
    end
end
for i=1:nc

    h=surface(x1(:, :, i),y1(:, :, i),z1(:, :, i),z1(:, :, i));
end

```

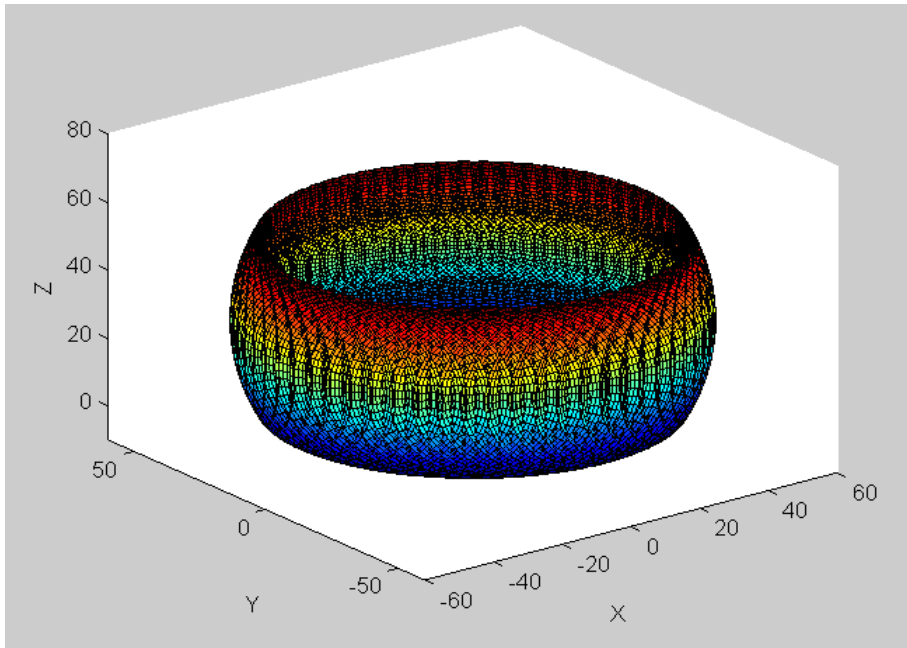


Figure 4.4: *Espace de travail du robot sphérique.*

Quand au robot Stanford à cinq degrés de liberté, le programme ET_Stanford.m montre la difficulté d'élaborer l'espace de travail en terme du nombre important des opérations à effectuer. Mais la structure du programme reste la même que pour les cas précédents avec des simplifications dans les pas de calcul pour les intervalles d'évolution des différentes variables articulaires.

```
clear all;close all;

% Paramètres du robot

d1=1.3;d2=1.4;d3i=0.5;d6=0.4;

% Activation de la cinquième articulation

i=0;
for t5=-100:10:100
    i=i+1;
    R5=[1 0 0;0 cosd(t5) -sind(t5) ;0 sind(t5) cosd(t5) ];
    D5=[0;d2 ;d1+d3i];
    H=R5*[0;d6;0]+D5;
```

```

        x5(i)=H(1);y5(i)=H(2);z5(i)=H(3);
    end
    n5=size(x5);

    % Activation de la quatrième articulaion

    i=0;
    for t4=-150:10:150
        i=i+1;
        for j=1:n5(2)
            D4=D5;
            R4=[cosd(t4) -sind(t4) 0; sind(t4) cosd(t4) 0;0 0 1];
            H=R4*[x5(j);y5(j)-d2;z5(j)-(d1+d3i)]+D4;
            x4(j,i)=H(1);y4(j,i)=H(2);z4(j,i)=H(3);
        end
    end

    n4=size(x4); n1=n4(1);nc=n4(2);

    % Activation de la troisième articulaion
    i=0;
    for d3=0:0.05:0.8
        i=i+1;
        for j=1:n1
            for k=1:nc
                x3(j,k,i)=x4(j,k);y3(j,k,i)=y4(j,k);z3(j,k,i)=z4(j,k)+d3;
            end
        end
    end
    n3=size(x3); n1=n3(1);nc=n3(2);na=n3(3);
    % Activation de la deuxième articulaion
    i=0;
    for t2=-180:20:70
        i=i+1;
        for j=1:n1
            for k=1:nc
                for l=1:na
                    D2=[0;0;d1];
                    R2=[cosd(t2) 0 sind(t2) ; 0 1 0;-sind(t2) 0
                        cosd(t2) 1];
                    H=R2*[x3(j,k,l);y3(j,k,l);z3(j,k,l)-d1]+D2;

                    x2(j,k,l,i)=H(1);y2(j,k,l,i)=H(2);z2(j,k,l,i)=H(3);
                end
            end
        end
    end
    n2=size(x2); n1=n2(1);nc=n2(2);na=n2(3);nb=n2(4);
    for i=1:na
        for j=1:nb

```

```

%surface(x2(:,:,i,j),y2(:,:,i,j),z2(:,:,i,j),z2(:,:,i,j))
    end
end

% Activation de la première articulation
i=0;
for t1=-160:40:160
    i=i+1;
    for j=1:n1
        for k=1:nc
            for l=1:na
                for m=1:nb
                    R1=[cosd(t1) -sind(t1) 0; sind(t1) cosd(t1) 0;
                        0 0 1];
                    H=R1*[x2(j,k,l,m);y2(j,k,l,m);z2(j,k,l,m)];
                    x1(j,k,l,m,i)=H(1);
                    y1(j,k,l,m,i)=H(2);
                    z1(j,k,l,m,i)=H(3);
                end
            end
        end
    end
end
n1=size(x1); n1=n1(1);nc=n1(2);na=n1(3);nb=n1(4);nd=n1(5);
for i=1:na
    for j=1:nb
        for k=1:nd
            surface(x1(:,:,i,j,k),y1(:,:,i,j,k),z1(:,:,i,j,k),z1(:,:,i,j,k))
        end
    end
end
end

```

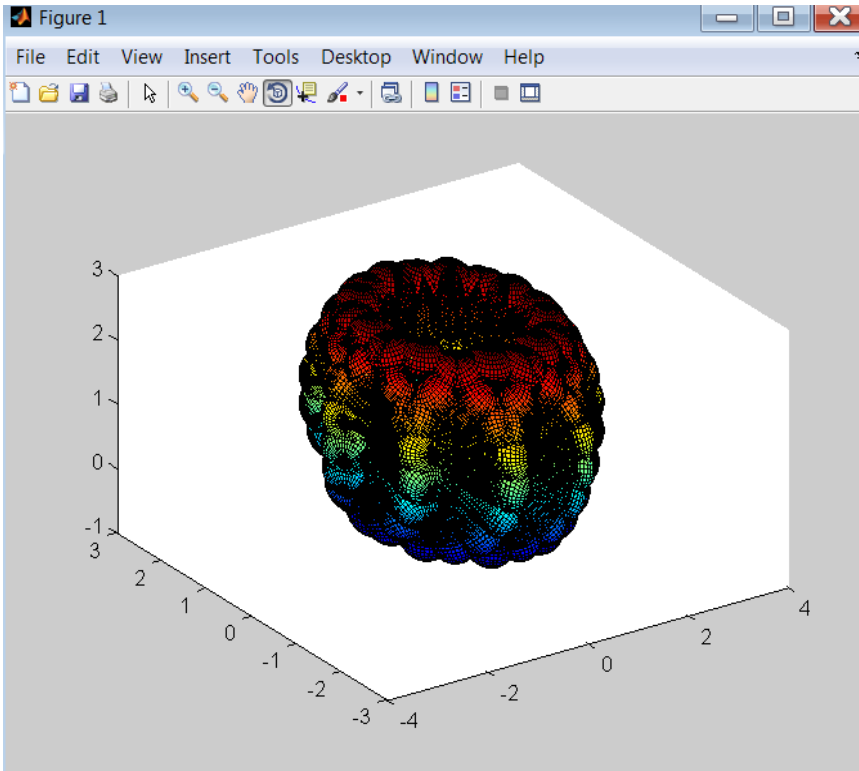



Figure 4.5: *Espace de travail du robot Stanford.*

Exercices

- Elaborer les espaces de travail des robots Puma et Scara.
- Construire une interface graphique qui permet d'afficher l'espace de travail du robot anthropomorphique ainsi que de son animation.

Chapitre 5

Les singularités

Dans certaines régions de l'espace de travail d'un robot manipulateur, le contrôle, la mobilité et la rigidité du robot sont affectés. Cela correspond à des configurations géométriques particulières du robot que l'on appelle singularités. Il faut alors éviter ces singularités lors de la planification des tâches du robot.

Pour un robot à n degrés de liberté, soient θ et X désignant respectivement le vecteur des variables articulaires et le vecteur des coordonnées cartésiennes:

Pour un robot à n degrés de liberté, soient θ et X désignant respectivement le vecteur des variables articulaires et le vecteur des coordonnées cartésiennes:

$$\theta = (\theta_1 \dots \dots \theta_n)^T \quad (5.1)$$

$$X = (x_1 \dots \dots x_n)^T \quad (5.2)$$

La description géométrique du robot permet alors de décrire la relation suivante:

$$f(\theta, X) = 0 \quad (5.3)$$

C'est cette relation qui aboutit aux deux modèles classiques du robot: modèle géométrique direct et modèle géométrique inverse.

En dérivant la relation (5.3) par rapport au temps, on obtient:

$$A\dot{X} + B\dot{\theta} = 0 \quad (5.4)$$

$$\text{Avec } A = \frac{\partial f(\theta, X)}{\partial x} \quad (5.5) \text{ et } B = \frac{\partial f(\theta, X)}{\partial \theta} \quad (5.6)$$

A et B sont appelées matrices jacobienues.

Trois types de singularités [3 Safrioui] sont rencontrées pour les robots manipulateurs suivant l'état des matrices jacobienues A et B.

- *singularités de type I*

La matrice B est singulière, son déterminant est alors nul:

$$\det(B)=0 \quad (5.7)$$

Ce type de singularité est observé surtout pour les robots manipulateurs sériels lorsque l'organe terminal est dans une limite externe ou interne de l'espace de travail (intersection de deux branches du modèle géométrique inverse). Bien que les moteurs sont actifs ($\dot{\theta} \neq 0$), on ne peut pas se déplacer dans certaines directions ($\dot{X} = 0$). On perd alors de mobilité dans ces directions et l'organe terminal peut résister à des couples ou des forces très importants dans ces directions.

- *Singularités de type II*

Ce type de singularité se produit essentiellement pour les robots parallèles avec une matrice jacobienne qui est singulière:

$$\det(A) = 0 \quad (5.8)$$

Ces singularités se produisent à l'intérieur de l'espace de travail. On peut alors trouver dans certaines directions des déplacements possibles ($\dot{X} \neq 0$) bien que les moteurs sont bloqués ($\dot{\theta} = 0$). On gagne cette fois des degrés de liberté et le robot devient très fragile dans ces directions.

- *Singularités de type III*

Ce type de singularité se présente lorsque pour un robot, les matrices jacobienues A et B peuvent être singulières. C'est l'architecture du robot qui est responsable de cette situation avec des conceptions particulières des ses composantes.

Singularités des robots manipulateurs sériels

C'est donc le type I de singularités qui se produit pour ces robots. Identifier ces singularités consiste à trouver les configurations géométriques qui rendent la matrice jacobienne singulière.

Si J est la matrice jacobienne du robot alors son expression est:

$$J = \frac{\partial X}{\partial \theta} \quad (5.9)$$

$$J \text{ est singulière si et seulement si } \det(J)=0 \quad (5.10)$$

Dans le cas du robot plan à deux degrés de liberté, le modèle géométrique direct s'écrit sous la forme:

$$x = l_1 c_1 + l_2 c_{12}$$

$$y = l_1 s_1 + l_2 s_{12} \quad (5.10)$$

Alors J devient :

$$J = \begin{bmatrix} -l_1 c_1 - l_2 s_{12} & -l_2 s_{12} \\ l_1 c_1 + l_2 c_{12} & l_2 c_{12} \end{bmatrix} \quad (5.11)$$

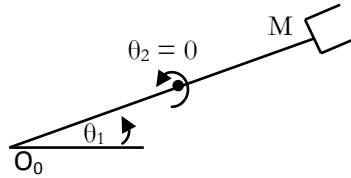
Le déterminant de J , après calcul et simplifications devient:

$$\det(J) = l_1 l_2 s_2 \quad (5.12)$$

Les singularités se produisent lorsque le déterminant de J est nul:

$\det(J) = 0 \iff \theta_2 = 0$ (la solution $\theta_2 = \pi$ est écartée si un tel débattement articulaire est interdit).

Cette configuration se produit lorsque les deux bras du robot sont tendus. La direction de singularité est donc celle du vecteur $O_0 M$ définie dans la figure suivante:

Figure 5.1: *Singularité du robot plan.*

En Matlab, l'instruction `jacobian` permet directement de déterminer la matrice jacobienne du robot à partir du modèle géométrique. Le programme `sing_bar2.m` décrit la démarche que l'on peut appliquer pour tout robot. Il suffit alors de relever l'expression symbolique du déterminant de J et d'identifier la configuration géométrique correspondante de singularité.

– `sing_bar2.m`

```
syms t1 t2 x y % définition symbolique

% Modèle géométrique

x=150*cos(t1)+350*cos(t1+t2);
y=150*sin(t1)+350*sin(t1+t2);

% Détermination de la jacobienne

J=jacobian([x;y],[t1 t2]);

% Déterminant de J

d=det(J);

% Simplification

d=simplify(d);
```

Singularités du robot anthropomorphique

Soit à établir les singularités du robot anthropomorphique dont les paramètres de DH (suivant la deuxième représentation) sont établis dans le tableau suivant:

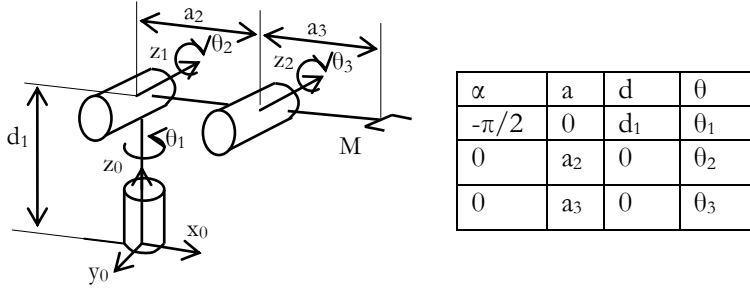


Figure 5.2: Robot anthropomorphe et Tableau des paramètres de DH.

- sing_antropom.m

```

alph=[-pi/2 0 0];
a=[0 sym('a2') sym('a3')];
d=[sym('d1') 0 0];
t=[sym('t1') sym('t2') sym('t3')];
[A,P,L]= mg_direct2(alph,a,t,d);
P=simplify(P);
J=jacobian(P,[t(1) t(2) t(3)]);
J=simplify(J);
D=det(J);
D=simplify(D);

```

Les résultats sont alors établis comme suit:

$$P = O_0 O_3 = \begin{bmatrix} c_1(a_2 c_2 + a_3 c_{23}) & s_1(a_2 c_2 + a_3 c_{23}) & (d_1 - a_2 s_2 - a_3 s_{23}) \end{bmatrix}^T \quad (5.13)$$

$$J = \begin{pmatrix} -s_1(a_2 c_2 + a_3 c_{23}) & -a_2 c_1 s_2 - a_3 s_{23} c_1 & -a_3 s_{23} c_1 \\ c_1(a_2 c_2 + a_3 c_{23}) & -a_2 s_1 s_2 - a_3 s_{23} s_1 & -a_3 s_{23} s_1 \\ 0 & -a_2 c_2 - a_3 c_{23} & -a_3 c_{23} \end{pmatrix} \quad (5.14)$$

$$\det(J) = (a_2 c_2 + a_3 c_{23}) a_3 a_2 s_3 \quad (5.15)$$

Les configurations singulières se produisent alors aux cas suivants:

- $s_3 = 0$ (bras 2 et 3 sont tendus)

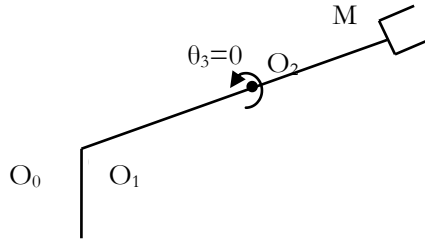


Figure 5.3: *Première configuration singulière.*

- $a_2 c_2 + a_3 c_{23} = 0 \Rightarrow a_2 c_2 = -a_3 c_{23}$

Cette singularité correspond à la configuration géométrique suivante:

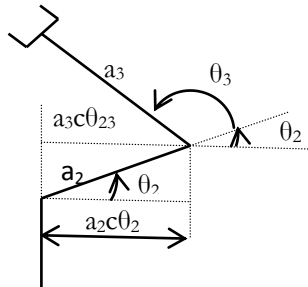


Figure 5.4: *Deuxième configuration singulière.*

Interface graphique avec des singularités

L'interface graphique élaboré dans `gui_bar2.m` permet de réaliser les tâches suivantes pour le robot plan à deux degrés de liberté:

- Détermination de l'espace de travail.
- Spécification d'une trajectoire dans l'espace de travail.
- Animation du robot.
- Analyse de singularité.
- Affichage des variables articulaires.

Remarques:

- La trajectoire du robot peut être importée avec des formats spécifiques de Matlab. Dans ce cas, un fichier texte est élaboré regroupant un ensemble de points par lesquels passe la courbe. L'interpolation polynomiale est ensuite appliquée pour assurer une continuité de la courbe.

- Une fois que l'animation est active, lorsque le robot atteint une configuration singulière, un message d'alerte est affiché dans une fenêtre de l'interface graphique.

```
function varargout = gui_bar2(varargin)

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                  'gui_Singleton',   gui_Singleton, ...
                  'gui_OpeningFcn', @gui_bar2_OpeningFcn, ...
                  'gui_OutputFcn',  @gui_bar2_OutputFcn, ...
                  'gui_LayoutFcn',   [] , ...
                  'gui_Callback',    []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargin
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end

% End initialization code - DO NOT EDIT
```

```

% --- Executes just before gui_bar2 is made visible.

function gui_bar2_OpeningFcn(hObject, eventdata, handles,
varargin)
% Choose default command line output for gui_bar2
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% --- Outputs from this function are returned to the command
line.

function varargout = gui_bar2_OutputFcn(hObject, eventdata,
handles)
varargout{1} = handles.output;

function ET_Callback(hObject, eventdata, handles)
axes(handles. axes1 )

%Activation de la deuxième articulation

i=0;
for t2=0:0.3:2*pi
    i=i+1;
    R2=[cos(t2) -sin(t2) 0; sin(t2) cos(t2) 0;0 0 1];
    D2=[150;0 ;0];
    H=R2*[350;0;0]+D2;
    x2(i)=H(1);y2(i)=H(2);z2(i)=H(3);
end
n2=size(x2);
axis([-600 600 -600 600 -20 20])
view(3)

%Activation de la première articulation

i=0;
for t1=0:0.3:2*pi
    i=i+1;
    for j=1:n2(2)

        R1=[cos(t1) -sin(t1) 0; sin(t1) cos(t1) 0;0 0 1];
        H=R1*[x2(j);y2(j);z2(j)];
        x1(j,i)=H(1);y1(j,i)=H(2);z1(j,i)=H(3);
    end
end
surface(x1,y1,z1,z1);hold on;

```

```

% --- Executes on button press in trajectory.

function trajectory_Callback(hObject, eventdata, handles)
axes(handles.axes1)

load('point.txt')
pt=10*point;
pt(11,2)=10.0479;
a=size(pt);
t=1:a(1); t=t';
px=polyfit(t,pt(:,1),6);
py=polyfit(t,pt(:,2),6);
t=0:0.1:a(1);t=t';
x=polyval(px,t);
y=polyval(py,t);
plot(x,y,'r');

% Title('trajectoire du robot')

function animation_Callback(hObject, eventdata, handles)

ERASEMODE = 'normal';
RENDERER = 'painters';

axes(handles.axes1)
load('point.txt')
pt=10*point;
pt(11,2)=10.0479;
a=size(pt);
t=1:a(1); t=t';
px=polyfit(t,pt(:,1),6);
py=polyfit(t,pt(:,2),6);
t=0:0.1:a(1);t=t';
x=polyval(px,t);
y=polyval(py,t);

% Modèle géométrique inverse [t1,t2]=mgi(x,y);

xa=150*cosd(t1);ya=150*sind(t1); %définition du bras 1
xb=xa+350*cosd(t1+t2);yb=ya+350*sind(t1+t2);%définition du bras 2

% Tracé initial

hLine1 = line('XData',[ 0 xa(1)], 'YData',[0 ya(1)]);
hLine2 = line('XData',[ xa(1) xb(1)], 'YData',[ya(1) yb(1)]);
hold on;
plt=plot(0,400,'r'); % point de la trajectoire circulaire

% Propriétés des objets graphiques

```

```

set(hLine1, 'EraseMode',ERASEMODE , 'Color','b'); %premier bras
set(hLine1, 'EraseMode',ERASEMODE , 'Color','b'); %deuxième bras

%Arrangement de la figure
set(gcf,'renderer','zbuffer');
grid on
view(52,30)
title('Trajectoire circulaire du robot plan');
xlabel('X')
ylabel('Y')
zlabel('Z')
axis('equal')
axis('square')
axis([-500 500 -500 500 ]);
rotate3d on

%Boucle d'animation

for i=1:numel(t)
    if round(t2(i))== 5
        t0=t(i);t20=t2(i);
        c='singularité à teta2=0';
        set(handles.sing,'String',c);
        guidata(hObject, handles);
    else
        end
    set(plt,'xdata',x(1:i),'ydata',y(1:i),...
        'Marker','.', 'MarkerSize',6);
    set(hLine1, 'XData',[ 0 xa(i)], 'YData',[0
ya(i)], 'Color','b')
    set(hLine2, 'XData',[ xa(i) xb(i)], 'YData',[ya(i)
yb(i)], 'Color','b')
    drawnow
    pause(.1) % pause dans l'animation
end

axes(handles. axes2 )
plot(t,t1,'g',t,t2,'b');hold on;
plot(t0,t20,'*r')
legend('teta1','teta2');

```

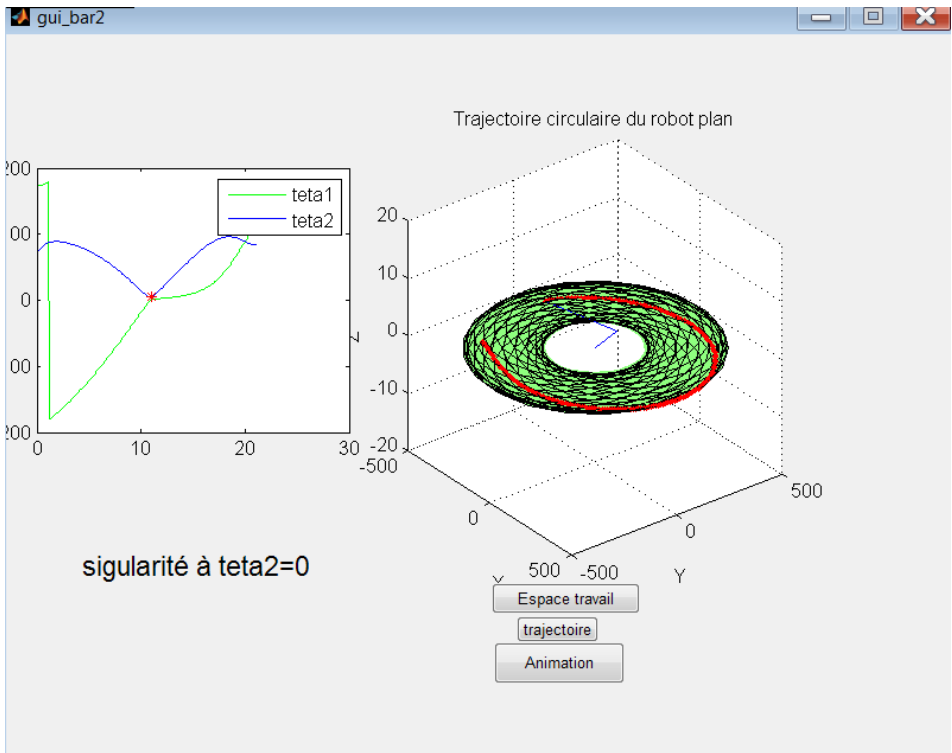


Figure 5.5: Interface graphique du robot plan à deux degrés de liberté.

Exercice

Refaire cette interface graphique avec le robot anthropomorphique et le robot sphérique.

Chapitre 6

Modélisation géométrique des robots

La modélisation géométrique est d'une grande importance pour ses différentes finalités: analyse et amélioration de la conception, planification des tâches, simulation du contrôle et bien d'autres. Le modèle géométrique exprime la relation entre les variables articulaires liées aux actionneurs (θ) et les coordonnées opérationnelles (X) suivant l'expression suivante:

$$f\left(\begin{matrix} m \times 1 & n \times 1 \\ \tilde{\theta} & , \tilde{X} \end{matrix}\right) = 0 \quad (6.1)$$

Avec $n \geq m$ (le cas $n > m$ est réservé aux cas des robots redondants)

On distingue alors deux types des modèles:

- Le modèle géométrique direct (MGD) fait disposer les coordonnées opérationnelles X une fois connues les variables articulaires θ . C'est un modèle non linéaire et couplé mais il est abordable, pour les robots manipulateurs sériels, à partir de la description géométrique du robot:

$$X = f(\theta) \quad (6.2)$$

- Le modèle géométrique inverse(MGI) permet d'identifier les variables articulaires θ lorsqu'on dispose des coordonnées opérationnelles X . Pour les robots manipulateurs sériels, ce modèle est élaboré à partir du MGD avec des solutions analytiques pour des architectures particulières des robots. Autrement dit, les méthodes numériques sont applicables pour la résolution des équations non linéaires présentées par ce modèle :

$$\theta = g(X) \quad (6.3)$$

Modèle géométrique direct

La description géométrique des robots plans permet généralement de manière aisée d'élaborer ce modèle comme c'est le cas du robot plan à deux degrés de liberté. Dans le cas général où l'architecture du robot est spatiale à plusieurs degrés de liberté, la représentation de DH (dans ses deux formes) reste un outil efficace pour ce faire. En effet avec cette représentation, on peut aboutir de manière systématique aux matrices homogènes attachées aux différents corps du robot. La matrice $T^{0,n}$ (où S_0 est le bâti et S_n constitue l'organe terminal) permet de mettre en forme les équations du MGD:

$$T^{0,n} = \begin{pmatrix} n & O_0 O_n^0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad (6.4)$$

L'orientation et la position de l'organe terminal par rapport au bâti sont respectivement déterminées par $A^{0,n}$ et $O_0 O_n^0$. C'est ce qui donne finalement les expressions du MGD comme ça va être montré dans les exemples suivants.

Exemple du robot plan à trois degrés de liberté

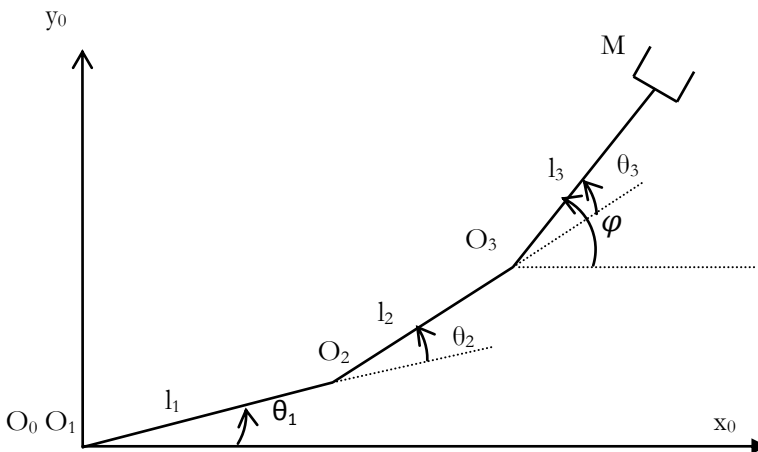


Figure 6.1: Robot Plan à trois degrés de liberté

La position du point M de l'organe terminal est exprimée par le vecteur $O_0M^0(x, y)$, tandis que l'orientation est définie par l'angle φ . On peut alors constituer le vecteur des variables articulaires θ et le vecteur des coordonnées opérationnelles X comme suit:

$$\theta = (\theta_1 \quad \theta_2 \quad \theta_3)^T \quad (6.5)$$

$$X = (x \quad y \quad \varphi)^T \quad (6.6)$$

L'analyse géométrique simple du mécanisme présentée dans la figure 6.1 conduit au MGD suivant:

$$x = l_1 c1 + l_2 c12 + l_3 c123 \quad (6.7)$$

$$y = l_1 s1 + l_2 s12 + l_3 s123 \quad (6.8)$$

$$\varphi = \theta_1 + \theta_2 + \theta_3 \quad (6.9)$$

MGD du robot sphérique

Reprenons le robot sphérique à trois degrés de liberté évoluant dans l'espace avec deux articulations rotoïdes et une troisième articulation prismatique. Le schéma du robot ainsi que la table des paramètres de DH sont présentés dans la figure suivante:

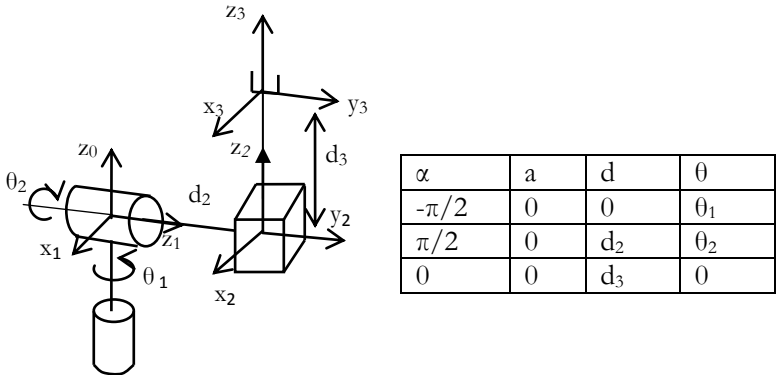


Figure 6.2: Robot sphérique à trois degrés de liberté.

Le programme `robot_spheric.m` permet de définir toutes les matrices de transformations homogène nécessaires à l'élaboration du MGD.

- `robot_spheric.m`

```
alph=[-pi/2 pi/2 0];
a=[0 0 0];
d=[0 sym('d2') sym('d3')];
t=[sym('t1') sym('t2') 0];
[A,P,L]= mg_direct2(alph,a,t,d);
```

Les résultats établis par ce programme se présentent comme suit:

$$T^{0,1} = \begin{pmatrix} c\theta_1 & 0 & -s\theta_1 & 0 \\ s\theta_1 & 0 & c\theta_1 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad T^{1,2} = \begin{pmatrix} c\theta_2 & 0 & s\theta_2 & 0 \\ s\theta_2 & 0 & -c\theta_2 & 0 \\ 0 & 1 & 0 & d_2 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$T^{2,3} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_3 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad \text{et} \quad T^{0,3} = \begin{pmatrix} c_1 c_2 & -s_1 & c_1 s_2 & c_1 s_2 d_3 - s_1 d_2 \\ s_1 c_2 & c_1 & s_1 s_2 & s_1 s_2 d_3 + c_1 d_2 \\ -s_2 & 0 & c_2 & c_2 d_3 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad (6.10)$$

Si x, y et z sont les coordonnées du vecteur position $O_0 M^0$, alors le MGD s'écrit sous la forme suivante:

$$x = s_2 c_1 d_3 - s_1 d_2 \quad (6.11)$$

$$y = s_2 s_1 d_3 + c_1 d_2 \quad (6.12)$$

$$z = c_2 d_3 \quad (6.13)$$

MGD du robot Puma à six degrés de liberté

Le robot Puma 560 dispose de six degrés de liberté avec six articulations rotoïdes disposées dans l'architecture suivante:

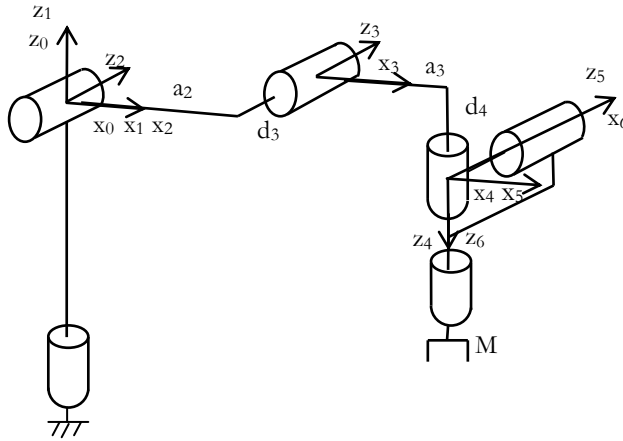


Figure 6.3: Robot Puma 560.

Le tableau rassemblant les paramètres de DH du robot Puma est présenté ci dessous:

α	a	d	θ
0	0	0	θ_1
$-\pi/2$	0	0	θ_2
0	a_2	d_3	θ_3
$-\pi/2$	a_3	d_4	θ_4
$\pi/2$	0	0	θ_5
$-\pi/2$	0	0	θ_6

Tableau 6.1 : Paramètres de DH du robot Puma.

Le programme mgd_puma.m permet d'élaborer le MGD du robot Puma 560.

- mgd_puma.m

```
alph=[0 -pi/2 0 -pi/2 pi/2 -pi/2];
```

```
digits=4;
```

```
a=[0 0 sym('a2') sym('a3') 0 0];
```

```
d=[0 0 sym('d3') sym('d4') 0 0];
```

```
t=[sym('t1') sym('t2') sym('t3') sym('t4') sym('t5') sym('t6')];
```

```
[A,P,L]= mg_direct1(alph,a,t,d);
P=simplify(P);
A=simplify(A);
```

L'exécution de ce programme permet alors d'aboutir au MGD avec les résultats suivants.

Si l'orientation du robot et la position de son organe terminal sont définies comme suit:

$$A^{0,6} = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{23} & a_{33} \end{pmatrix} \text{ et } O_0 O_6 = \begin{pmatrix} x \\ y \\ z \end{pmatrix} \text{ alors on obtient:}$$

$$\begin{aligned} a_{11} &= c_1(c_{23}(c_4c_5c_6 - s_4s_6) - s_{23}s_5c_5) + s_1(s_4c_5c_6 + c_4s_6) \\ a_{21} &= s_1(c_{23}(c_4c_5c_6 - s_4s_6) - s_{23}s_5s_6) - c_1(s_4c_5c_6 + c_4s_6) \\ a_{31} &= -s_{23}(c_4c_5c_6 - s_4s_6) - c_{23}s_5s_6 \\ a_{12} &= c_1(c_{23}(-c_4c_5c_6 - s_4c_6) + s_{23}s_5s_6) + s_1(c_4c_6 - s_4c_5s_6) \\ a_{22} &= s_1(c_{23}(-c_4c_5c_6 - s_4c_6) + s_{23}s_5s_6) - c_1(c_4c_6 - s_4c_5s_6) \\ a_{23} &= -s_{23}(-c_4c_5c_6 - s_4c_6) + c_{23}s_5s_6 \end{aligned} \quad (6.14)$$

$$\begin{aligned} a_{31} &= -c_1(c_{23}c_4s_5 + s_{23}c_5) - s_1s_4s_5 \\ a_{32} &= -s_1(c_{23}c_4s_5 + s_{23}c_5) + c_1s_4s_5 \\ a_{33} &= s_{23}c_4s_5 - c_{23}c_5 \end{aligned}$$

$$\begin{aligned} x &= c_1(a_2c_2 + a_3c_{23} - d_4s_{23}) - d_3s_1 \\ y &= s_1(a_2c_2 + a_3c_{23} - d_4s_{23}) + d_3c_1 \\ z &= -a_3s_{23} - a_2s_2 - d_4c_{23} \end{aligned}$$

L'orientation du robot est déterminée par les 9 paramètres de la matrice $A^{0,6}$. Une dépendance s'établit en fait entre ces paramètres, ce qui produit une redondance dans la description du MGD.

Pour avoir une représentation minimale avec trois équations seulement pour l'orientation, il est préférable d'utiliser un mode adéquat de représentation: les angles d'Euler ZYZ par exemple.

La fonction `Inv_Euler_ZYZ.m` peut être alors appliquée pour déterminer les trois angles d'Euler φ , θ et Ψ comme suit:

$$\varphi = \text{Atan2}\left(\sqrt{a_{31}^2 + a_{32}^2}, a_{33}\right) \quad (6.15)$$

$$\theta = \text{Atan2}\left(\sqrt{a_{23}^2/s\varphi}, a_{13}/s\varphi\right) \quad (6.16)$$

$$\Psi = \text{Atan2}(a_{32}/s\varphi - a_{31}/s\varphi) \quad (6.17)$$

Modèle géométrique inverse

Le modèle géométrique inverse (MGI) permet d'obtenir les variables articulaires θ à partir des coordonnées opérationnelles X . Il n'y a pas de solutions analytiques de manière générale pour ce modèle sauf pour quelques architectures des robots, particulièrement pour ceux dont les axes des trois dernières articulations sont concourants. On peut alors considérer deux approches pour la résolution de ce problème: algébriques et géométriques.

Approche algébrique

Le principe de cette approche est de progresser dans la connaissance des variables articulaires à partir des opérations particulières sur la matrice connue de la position et de l'orientation de l'organe terminal par rapport au bâti et des différentes matrices de transformation homogène. Raisonnons pour le cas général d'un robot à six degrés de liberté sachant que c'est le même principe de résolution qui peut être appliqué aux autres types de robots.

$$\text{Notons } H_0 \text{ cette matrice bien établie } H_0 = \begin{pmatrix} A^{0,6} & O_0 O_6^0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad (6.18)$$

En Posant à droite de chaque terme les paramètres dont il dépend entre parenthèse, on peut alors écrire:

$$\begin{aligned} \bullet \quad H_0 &= T^{0,6}(\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6). \\ \bullet \quad H_1(\theta_1) &= T^{1,0} H_0 = T^{1,0} T^{0,6} = T^{1,6}(\theta_2, \theta_3, \theta_4, \theta_5, \theta_6). \\ \bullet \quad H_2(\theta_2) &= T^{2,1} H_1 = T^{2,6}(\theta_3, \theta_4, \theta_5, \theta_6). \\ \bullet \quad H_3(\theta_3) &= T^{3,2} H_2 = T^{3,6}(\theta_4, \theta_5, \theta_6). \\ \bullet \quad H_4(\theta_4) &= T^{4,3} H_3 = T^{4,6}(\theta_5, \theta_6). \\ \bullet \quad H_5(\theta_5) &= T^{5,4} H_4 = T^{5,6}(\theta_6). \end{aligned} \quad (6.19)$$

L'identification des matrices de gauche et de droite des relations précédentes permet de déterminer toutes les variables articulaires. Certaines relations peuvent conduire à plusieurs des variables. L'ordre de ces relations ne conduit pas nécessairement à une détermination successive des variables articulaires.

L'identification et la résolution des équations pertinentes dépendent des choix bien intuitifs. Cependant les relations suivantes sont très fréquentes pour lesquelles les solutions sont bien établies.

$$a\theta + bc\theta = c \quad \theta = \text{Atan2}(a, b) \pm \text{Atan2}(\sqrt{a^2 + b^2 - c^2}, c) \quad (6.20)$$

$$a\theta + bc\theta = 0 \quad \theta = \text{Atan2}(-b, a) \text{ ou } \theta = \text{Atan2}(b, -a) \quad (6.21)$$

$$c\theta = a, s\theta = b \quad \theta = \text{Atan2}(b, a) \quad (6.22)$$

$$c\theta = a \quad \theta = \pm \text{Atan2}(\sqrt{1 - a^2}, a) \quad (6.23)$$

$$s\theta = a \quad \theta = \text{Atan2}(a, \pm \sqrt{1 - a^2}) \quad (6.24)$$

MGI du robot Puma

Toutes les équations (6.19) sont considérées dans le programme `mg_i_puma.m`. Le vecteur $\theta = (0.1, 0.2, 0.3, 0.4, 0.5, 0.6)^T$ est introduit dans le MGD du robot ce qui donne la matrice H_0 .

`mg_i_puma.m`

```
alph=[0 -pi/2 0 -pi/2 pi/2 -pi/2];
a2=0.4318;a3=0.02;d3=0.1244;d4=0.4318;
digits=4;
a=[0 0 a2 a3 0 0];
d=[0 0 d3 d4 0 0];
t=[sym('t1') sym('t2') sym('t3') sym('t4') sym('t5') sym('t6')];

[A,P,L]= mg_direct1(alph,a,t,d);
```

```
% Formules pour le calcul des teta
T06=[A P;0 0 0 1];T06=simplify(T06);
H0=subs(T06,{sym('t1'),sym('t2'),sym('t3'),sym('t4'),sym('t5'),sym('t6')},{0.1,0.2,0.3,0.4,0.5,0.6});
H0=double(H0);
T01=L(:, :, 1);T10=inv(T01);
T12=L(:, :, 2);T21=inv(T12);
T23=L(:, :, 3);T32=inv(T23);
T34=L(:, :, 4);T43=inv(T34);
T45=L(:, :, 5);T54=inv(T45);
T56=L(:, :, 6);T65=inv(T56);
H1=T10*H0;
H2=T21*H1;H2=subs(H2,sym('t1'),0.1);
H3=T32*H2;H3=subs(H3,{sym('t2'),sym('t3')},{0.2,0.3});
H4=T43*H3;%relation inutilisée
H5=T54*H4;%relation inutilisée
T16=L(:, :, 2)*L(:, :, 3)*L(:, :, 4)*L(:, :, 5)*L(:, :, 6);
T16=simplify(T16);
T26=L(:, :, 3)*L(:, :, 4)*L(:, :, 5)*L(:, :, 6);T26=simplify(T26);
T36=L(:, :, 4)*L(:, :, 5)*L(:, :, 6);T36=simplify(T36);
T46=L(:, :, 5)*L(:, :, 6);T46=simplify(T46);
T56=L(:, :, 6);
```

A partir des résultats établis par le programme , la résolution passe par les étapes suivantes:

1. En identifiant les termes de $H_1(2,4)$ et $T_{16}(2,4)$, on obtient l'équation:

$$0.1471c_1 - 0.2201s_1 = 0.1244 \quad (6.25)$$

Cette équation est du type (6.20) dans les relations fréquentes d'où la résolution donne $\theta_1 = 0.1$

$$(6.26)$$

2. En considérant les termes (1,4), (2,4) de H_2 et T_{26} et en effectuant la somme au carré de ces termes, on arrive à l'équation

$$0.01727c_3 - 0.3729s_3 = -0.0937 \quad (6.27)$$

$$\text{d'où } \theta_3 = 0.3 \quad (6.28)$$

3. En identifiant les termes (2,4) dans H_2 et T_{26} , on obtient:

$$0.4743c_2 - 0.2337s_2 = 0.4318c_3 + 0.02s_3 \quad (6.29)$$

ce qui donne $\theta_2 = 0.2$ puisque θ_3 est connue dans le second membre de l'égalité.

4. En identifiant les termes (2,3) de H_3 et T_{36} , on trouve :

$$c_5 = 0.8776 \text{ d'ou } \theta_5 = 0.5 \quad (6.30)$$

5. En identifiant les termes (3,3) de H_3 et T_{36} on obtient :

$$s_4 s_5 = 0.1867 \text{ d'ou } \theta_4 = 0.4 \quad (6.31)$$

6. Et finalement, l'identification des termes (2,1) de H_3 et T_{36} , on obtient:

$$c_6 s_5 = 0.3957 \text{ d'ou } \theta_6 = 0.6 \quad (6.32)$$

Robots aux axes concourants dans le poignet

Lorsque les axes des trois dernières articulations sont concourantes, on dispose alors d'un découplage entre la position et l'orientation de l'organe terminal du robot. Cette dernière n'affecte pas en fait la position du point concourant du poignet. Le vecteur de position $O_0 O_4^0$ s'exprime alors seulement en fonction des variables articulaires θ_1, θ_2 et θ_3 .

Puisque les centres des articulations du poignet coïncident (O_4, O_5 et O_6 *confondus*) on peut alors établir les relations suivantes qui permettent de déterminer les variables articulaires liées à la position (θ_1, θ_2 et θ_3):

$$\begin{aligned}
 \bullet \quad U_0 &= O_0 O_6^0 = O_0 O_4^0 = T^{0,4} \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} (\theta_1, \theta_2, \theta_3) \\
 \bullet \quad U_1 &= T^{1,0} U_0(\theta_1) = T^{1,0} T^{0,4} \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} = T^{1,4} \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} (\theta_2, \theta_3) \\
 \bullet \quad U_2 &= T^{2,1} U_1(\theta_2) = T^{2,4} \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} (\theta_3) \\
 \bullet \quad U_3 &= T^{3,2} U_2(\theta_3) = T^{3,4} \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}
 \end{aligned} \quad (6.33)$$

L'identification des termes spécifiques des matrices de gauche (connus) et ceux de droite (inconnus) des équations (6.33) permet alors d'obtenir θ_1 , θ_2 et θ_3 .

Pour la détermination des variables articulaires liées à l'orientation θ_4 , θ_5 et θ_6 . Si S_0 est la matrice connue de rotation de l'organe terminal par rapport au bâti, on peut écrire les relations suivantes:

$$\begin{aligned}
 \bullet \quad S_0 &= A^{0,6} \\
 \bullet \quad S_1 &= A^{3,0} S_0 = A^{3,6} \\
 \bullet \quad S_2 &= A^{4,3} S_1 = A^{4,6} \\
 \bullet \quad S_3 &= A^{5,4} S_2 = A^{5,6}
 \end{aligned} \tag{6.34}$$

L'identification des termes adéquats des relations (6.34) permet alors de trouver les variables articulaires θ_4 , θ_5 et θ_6 .

MGI du robot Yasukawa à cinq degrés de liberté

Le robot Yasukawa dispose de cinq degrés de liberté avec des liaisons qui sont toutes rotoïdes. Les deux dernières articulations sont concourantes avec des centres coïncidents. On peut ainsi utiliser les équations (6.33) pour déterminer le MGI. le schéma du robot ainsi que les paramètres de DH sont présentés ci dessous:

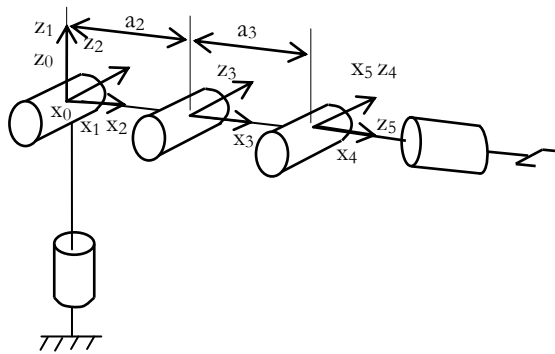


Figure 6.4: Robot Yasukawa et Tableau des paramètres de DH.

α	a	d	θ
0	0	0	θ_1
$-\pi/2$	0	0	θ_2
0	a_2	0	θ_3
0	a_3	0	θ_4
$\pi/2$	0	0	θ_5

Tableau 6.2 : Paramètres de DH du robot Yasukawa.

Le programme `mg_i_yasukawa.m` permet délaborer toutes les étapes nécessaires à l'obtention des variables articulaires.

- `mg_i_yasukawa.m`

```

alph=[0 -pi/2 0 0 pi/2 ];

digits=4;a2=200;a3=150;
a=[0 0 a2 a3 0 ];
d=[0 0 0 0 0];
t=[sym('t1') sym('t2') sym('t3') sym('t4') sym('t5') ];
[A,P,L]= mg_direct1(alph,a,t,d);

U0=subs(P,{sym('t1'),sym('t2'),sym('t3')},{0.1,0.2,0.3});
U0=double(U0);
S0=subs(A,{sym('t1'),sym('t2'),sym('t3'),sym('t4'),sym('t5')},{0.1,0.2,0.3,0.4,0.5});S0=double(S0);
T10=inv(L(:, :, 1));T21=inv(L(:, :, 2));T32=inv(L(:, :, 3));
U1=T10(1:3,1:3)*U0;
U2=T21(1:3,1:3)*U1;U2=subs(U2,sym('t1'),0.1);
T14=L(:, :, 2)*L(:, :, 3)*L(:, :, 4);T14=simplify(T14(1:3,4));
T24=L(:, :, 3)*L(:, :, 4);T24(1:3,1:3);T24=simplify(T214(1:3,4));
A03=L(:, :, 1)*L(:, :, 2)*L(:, :, 3);A03=A03(1:3,1:3);
A30=inv(A03);A30=subs(A30,{sym('t1'),sym('t2'),sym('t3')},{0.1,0.2,0.3});
A36=L(:, :, 4)*L(:, :, 5);A36=A36(1:3,1:3);
S1=A30*S0;

```

Une fois que le programme est exécuté, on peut alors faire la résolution du MGI du robot en suivant les étapes suivantes:

- U_0 est le vecteur position de l'organe terminal pour $\theta_1 = 0.1$, $\theta_2 = 0.2$ et $\theta_3 = 0.3$. 6.35)

- $U_1 = T^{1,4} \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$. En identifiant le deuxième terme des deux vecteurs on aboutit la relation

$$32.71c_1 - 326s_1 = 0 \text{ ce qui donne } \theta_1 = 0.1 \quad (6.36)$$

- $U_2 = T^{2,1} U_1 = T^{2,4} \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$. En ajoutant le carré des deux premières composantes des vecteurs de droite et de gauche et après identification, on obtient :

$$60000c_3 + 62500 = 1.198 e5 \text{ ce qui donne } \theta_3 = 0.3 \quad (6.37)$$

L'identification du deuxième terme des deux vecteurs conduit à la relation:

$$150 \sin(0.3) = 111.6c_2 - 327.s_2 \text{ ainsi } \theta_2 = 0.2 \quad (6.38)$$

- $S_1 = A^{3,0} S_0 = A^{3,6}$. L'identification des termes (1,3), (3,2) des deux matrices de droites et de gauche permet d'écrire les deux relations:

$$\sin(\theta_4) = 0.3894 \text{ et } \cos(\theta_5) = 0.8775. \text{ ainsi } \theta_4 = 0.4 \text{ et } \theta_5 = 0.5 \quad (6.39)$$

Approche géométrique

Cette approche est toujours valable lorsque les axes des trois dernières articulations (poignet) sont concourants. On peut alors isoler la position et déterminer géométriquement les trois premières variables articulaires. Par la suite l'orientation est considérée pour déterminer le reste des autres variables articulaires. Le principe adopté est résumé dans la démarche suivante:

Problème de la Position

Les données du MGI sont la position P et la matrice d'orientation A de l'organe terminal par rapport au bâti et qui sont exprimés comme suit:

$$P = O_0 O_6^0 = \begin{pmatrix} x \\ y \\ z \end{pmatrix}, A = A^{0,6} = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{23} & a_{33} \end{pmatrix} \quad (6.40)$$

- Si les centres des trois dernières articulations coïncident en un seul point O_4 alors peut écrire $O_0 O_4^0 = O_0 O_6^0$ (6.41). Dans certains cas le point O_6 est écarté des points coïncidents O_4 et O_5 , le vecteur de position est alors déterminé par l'expression:

$$O_0 O_4^0 = O_0 O_6^0 - A^{0,6} O_4 O_6^0 \quad (6.42)$$

- Le vecteur de position est décomposé comme suit:

$$O_0 O_4 = O_0 O_1 + O_1 O_2 + O_2 O_3 + O_3 O_4 \quad (6.43)$$

- On choisit un plan de projection pour la relation (6.43) ou évoluent les variables articulaires recherchées (par exemple le plan $x_0 y_0$ pour déterminer θ_1 et le plan $x_1 y_1$ pour θ_2).
- Quelque soit le plan de projection, il faut bien respecter la géométrie du robot élaboré dans le tableau des paramètres de DH et considérer la donnée du vecteur $O_0 O_4^0 = (P_x \ P_y \ P_z)^T$ (6.44)

Problème de l'orientation

On traite le cas du centre O_6 écarté des points coïncidents O_4 et O_5 (dans le cas contraire le point M de l'organe terminal remplacera le point O_6 dans les formules qui suivent).

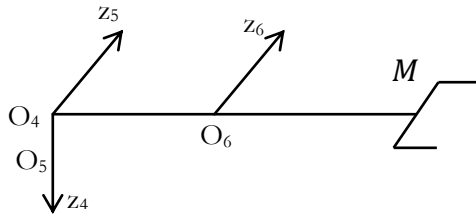


Figure 6.5 : Position des centres des articulations dans le poignet.

L'orientation du poignet est déterminée par le vecteur $O_4 O_6^0$ (ou $O_4 M^0$) tel que:

$$O_4 O_6^0 = A^{0,6} O_4 O_6^6 \quad (6.45)$$

Les coordonnées du vecteur $O_4 O_6^0$ seront connues à travers la relation (6.45) et spécifiées par $O_4 O_6^0 = \begin{pmatrix} U_x \\ U_y \\ U_z \end{pmatrix}$ (6.46)

On peut alors réécrire ce vecteur dans le repère R_3 :

$$O_4 O_6^3 = A^{3,0} O_4 O_6^0 \quad (6.47)$$

La matrice $A^{3,0}$ est la matrice inverse de $A^{0,3}$:

$$A^{0,3} = A^{0,1} A^{1,2} A^{2,3} \quad (6.48)$$

(matrices connues puisque θ_1, θ_2 et θ_3 le sont aussi).

Donc le vecteur $O_4 O_6^3 = \begin{pmatrix} V_x \\ V_y \\ V_z \end{pmatrix}$ est bien défini. (6.49)

$$\text{D'autre part, le vecteur } O_4 O_6^3 = A^{3,6} O_4 O_6^6 \quad (6.50)$$

Le terme de droite est exprimé en fonction de θ_4, θ_5 et θ_6 . On dispose alors de trois relations et d'une configuration géométrique particulière de R_6 par rapport à R_3 . Ce qui permet généralement de résoudre le problème. Dans certains cas, on a besoin d'introduire d'autres relations de produit scalaire ou vectoriel des vecteurs unitaires des repères R_4, R_5 et R_6 . Par exemple, pour la détermination de θ_6 , on utilise la relation suivante:

$$x_5 \cdot x_6 = \cos(\theta_6) \quad (6.51)$$

MGI du robot ABB_IRB_140_6/0.8

Le robot ABB_IRB possède 6 ddl avec toutes les articulations qui sont rotoïdes. Les axes des trois dernières articulations sont concourants. Toutefois le centre de la sixième articulation est décalée d'une distance d_6 par rapport au centres O_4 et O_5 qui coïncident comme le montre la figure suivante:

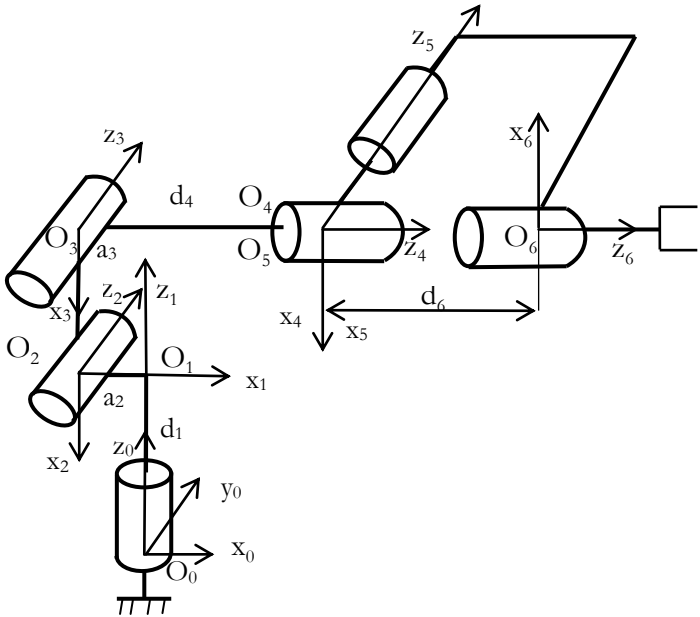


Figure 6.6: Robot ABB_IRB_140_6/0.8 à 6ddl.

Les paramètres de DH du Robot sont établis dans le tableau suivant:

α	a	d	θ
0	0	352	θ_1
$-\pi/2$	70	0	$\theta_2 - \pi/2$
0	360	0	θ_3
$-\pi/2$	0	380	θ_4
$\pi/2$	0	0	θ_5
$-\pi/2$	0	65	$\theta_6 + \pi$

Tableau 6.3: Paramètres de DH du robot ABB_IRB.

Le programme `mgi_abb6.m` permet de déterminer le MGI du robot ABB à six degrés de liberté.

- `mgi_abb6.m`

```

alph=[0 -pi/2 0 -pi/2 pi/2 -pi/2];
a=[0 70 360 0 0 0];
d=[352 0 0 380 0 65];
a2=70;a3=360;d1=352;d4=380;
t=[sym('t1') sym('t2')-pi/2 sym('t3') sym('t4')
sym('t5')sym('t6')+pi];

[A,P,L]= mg_direct1(alph,a,t,d);
P=subs(P,{sym('t1'),sym('t2'),sym('t3'),sym('t4'),sym('t5'),sym('t6')},{0.1,0.2,0.3,0.4,0.5,0.6});
P=double(P);
A=subs(A,{sym('t1'),sym('t2'),sym('t3'),sym('t4'),sym('t5'),sym('t6')},{0.1,0.2,0.3,0.4,0.5,0.6});
A=double(A);

% Détermination de teta1 (t1)

P040=P-A*[0;0;65];
Px= P040(1);Py= P040(2);Pz= P040(3);
t1=atan2(Py,Px);

% Détermination de teta2 (t2) et teta3(t3)
% Projection dans le plan x1z1

A01=[cos(0.1) -sin(0.1) 0;sin(0.1) cos(0.1) 0; 0 0 1];
A10=inv(A01);
P141=A10*([0;0;-352]+P040);
P241=P141-[70;0;0];
pz=P241(3);px= P241(1);

% Détermination de t3

m=(px^2+pz^2-a3^2-d4^2)/(2*a3*d4);
t3=asin(m);

% Détermination de t2

a=-a3-d4*sin(t3);
b=d4*cos(t3);c=px;
t2= atan2(a,b)+atan2(sqrt(a^2+b^2-c^2),c);

% Orientation
% Détermination de teta4(t4) et teta5(t5)

```

```

P460=A*[0;0;65];
A03=L(:, :, 1)*L(:, :, 2)*L(:, :, 3); A03=A03(1:3, 1:3);
A03=subs(A03, {sym('t1'), sym('t2'), sym('t3')}, {0.1, 0.2, 0.3});
A03=double(A03);
A30=inv(A03);
P463=A30*P460; x463=P463(1); y463=P463(2); z463=P463(3);
A36=L(:, :, 4)*L(:, :, 5)*L(:, :, 6); A36=A36(1:3, 1:3);
P463=A36*[0;0;65];
t5=acos(y463/65);
%t5=acos(57.0426/65);
%t4=atan(12.1346/28.7070);
t4=atan(-z463/x463);

% Détermination de teta6(t6)

A05=L(:, :, 1)*L(:, :, 2)*L(:, :, 3)*L(:, :, 4)*L(:, :, 5); A05=A05(1:3, 1:3)
A05=subs(A05, {sym('t1'), sym('t2'), sym('t3'), sym('t4'), sym('t5')},
{0.1, 0.2, 0.3, 0.4, 0.5})
A05=double(A05);
A50=inv(A05);
x60=A(:, 1);
x65=A50*x60;
m=dot([1;0;0], x65);
t6=acos(-m);

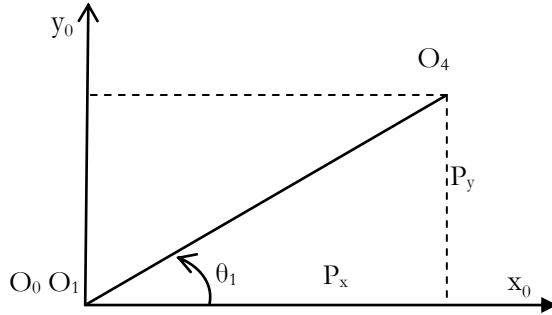
```

En premier lieu, le vecteur position P et la matrice d'orientation A sont déterminés pour des valeurs des variables articulaires (0.1, 0.2, 0.3, 0.4, 0.5, 0.6).

- Le vecteur position $O_0O_4^0$ est calculé dans l'expression:

$$P_{040} = \begin{pmatrix} P_x \\ P_y \\ P_z \end{pmatrix} = \begin{pmatrix} 472.5950 \\ 47.4225 \\ 522.6395 \end{pmatrix} \quad (6.52)$$

La relation (6.43) est projetée dans le plan x_0y_0 (comme le montre la figure suivante) pour déterminer θ_1 .

Figure 6.7: *Projection dans le plan x_0y_0 .*

On peut simplement déduire θ_1 D'après la figure avec l'expression suivante:

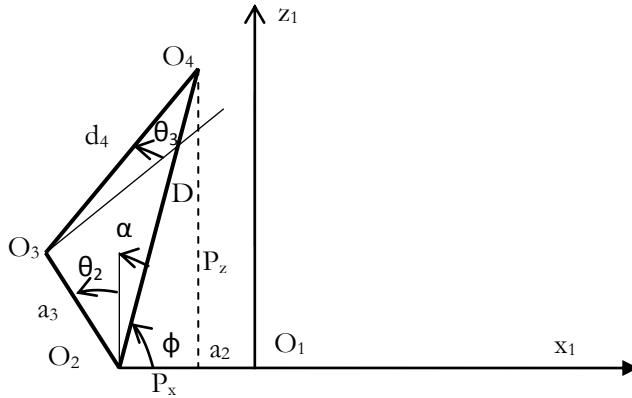
$$\theta_1 = \text{Atan2}(P_y, P_x) \quad (6.53)$$

- La relation (6.43) est ensuite projetée dans le plan x_1z_1 pour déterminer les variables θ_2 et θ_3 :

$$O_1O_4^1 = A^{1,0} (O_1O_0^0 + O_0O_4^0) \quad (6.54)$$

$$\text{On déduit alors } O_2O_4^1 = O_1O_4^1 - O_1O_2^1 \quad (6.55)$$

$$O_2O_4^1 = (P_x \ P_y \ P_z)^T = (404.9683 \ 0.0048 \ 170.6395)^T \quad (6.66)$$

Figure 6.8: *Projection dans le plan x_1z_1 .*

Considérons le triangle (O_2, O_3, O_4) . La relation triangulaire qui dispose l'angle $(\theta_2 + \alpha)$ se présente comme suit:

$$\cos(\theta_2 + \alpha) = \frac{a_3^2 + D^2 - d_4^2}{2 a_3 D} = M \quad (6.67)$$

$$\text{Avec } D = \sqrt{P_x^2 + P_z^2}, \alpha = \frac{\pi}{2} - \emptyset \quad \text{et } \text{tg}(\emptyset) = \frac{P_z}{P_x} \quad (6.68)$$

$$\text{D'où } \theta_2 = \arccos(M) - \alpha = -0.2 \quad (6.69)$$

D'autre part, l'angle au sommet O_4 du triangle (O_2, O_3, O_4) est déterminé par la relation suivante:

$$\cos\left(\frac{\pi}{2} + \theta_3\right) = \frac{d_4^2 - D^2 + a_3^2}{2 d_4 a_3} = N \quad (6.70)$$

$$\text{D'où } \theta_3 = \arccos(N) - \frac{\pi}{2} = -0.3 \quad (6.71)$$

Remarque: Les angles θ_2 et θ_3 sont négatifs puisque les rotations dans le plan $x_1 z_1$ sont effectuées dans le sens négatifs.

Problème de l'orientation

Le vecteur $O_4 O_6^0 = A^{0,6} O_4 O_6^6$ a pour coordonnées:

$$O_4 O_6^0 = \begin{pmatrix} U_x \\ U_y \\ U_z \end{pmatrix} = \begin{pmatrix} 34.9050 \\ 15.6975 \\ -52.5395 \end{pmatrix} \quad (6.72)$$

La relation (6.47) permet d'élaborer $O_4 O_6^3$ comme suit:

$$P463 = \begin{pmatrix} -65 c_4 c_5 \\ 65 c_5 \\ 65 s_4 s_5 \end{pmatrix} \text{ avec } P463 = \begin{pmatrix} -28.7070 \\ 57.0426 \\ 121346 \end{pmatrix} \quad (6.73)$$

Ce qui correspond à la configuration géométrique suivante:

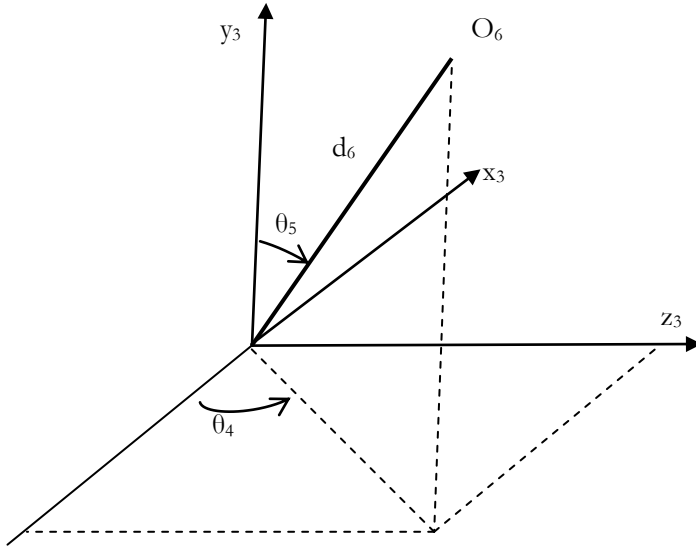


Figure 6.9: *Repérage de l'orientation.*

On peut alors déduire $\theta_5 = \arccos\left(\frac{57.042}{65}\right) = 0.5$ et $\theta_4 = \arctan\left(\frac{12.0426}{65}\right) = 0.4$.

θ_6 peut être enfin déduite à partir du produit scalaire:

$$x_5^5 \cdot x_6^5 = \cos(\theta_6 + \pi) \quad (6.74)$$

$$\text{Avec } x_5^5 = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} \text{ et } x_6^5 = A^{5,0} x_6^0 \quad (6.75)$$

$$x_6^0 = A(:,1) \text{ et } A^{5,0} = \text{inv } A^{0,5} \quad (6.76)$$

$$\text{D'où } \theta_6 = 0.6 \quad (6.77)$$

MGI du robot Puma

Le programme mgi_puma.m conduit aux résultats désirés avec les mêmes procédures établies auparavant.

- mgi_puma.m

```

alph=[0 -pi/2 0 -pi/2 pi/2 -pi/2];

a2=0.4318;a3=0.02;d3=0.1244;d4=0.4318;d6=0.5;
digits=4;
a=[0 0 a2 a3 0 0];
d=[0 0 d3 d4 0 0];
t=[sym('t1') sym('t2') sym('t3') sym('t4') sym('t5') sym('t6')];

[A,P,L]= mg_direct1(alph,a,t,d);

P=subs(P,{sym('t1'),sym('t2'),sym('t3'),sym('t4'),sym('t5'),sym('t6')},{0.1,0.2,0.3,0.4,0.5,0.6});
P=double(P);
A=subs(A,{sym('t1'),sym('t2'),sym('t3'),sym('t4'),sym('t5'),sym('t6')},{0.1,0.2,0.3,0.4,0.5,0.6});
A=double(A);

% Détermination de teta1 (t1)

P040=P;Px=P040(1);Py=P040(2);
D=sqrt(Px^2+Py^2);m=d3/D;phi=atan(Py/Px);
t1=phi-asin(m);

% Détermination de teta2 (t2) et teta3(t3)

A01=[cos(0.1) -sin(0.1) 0;sin(0.1) cos(0.1) 0; 0 0 1];
A10=inv(A01);
P141=A10*P040;px=P141(1);pz=P141(3);
D=sqrt(px^2+pz^2);u=sqrt(a3^2+d4^2);ph=atan(pz/px);
m=(a2^2+D^2-u^2)/(2*D*a2);
t2=ph+acos(m);
m1=(u^2+D^2-a2^2)/(2*u*D);alpha=acos(m1);
m2=(u^2+d4^2-a3^2)/(2*u*d4);beta=acos(m2);
gama=pi-(-ph+alpha+beta);
a=a3*cos(0.2);b=a3*sin(0.2);
c=abs(pz)-(d4*sin(gama)+a2*sin(0.2));
t3=atan2(a,b)-atan2(sqrt(a^2+b^2-c^2),c);

% Orientation
% Détermination de teta4(t4) et teta5(t5)

```

```

P4M0=A*[0;0;d6];
A03=L(:,:,1)*L(:,:,2)*L(:,:,3);A03=A03(1:3,1:3);
A03=subs(A03,{sym('t1'),sym('t2'),sym('t3')},{0.1,0.2,0.3});
A03=double(A03);
A30=inv(A03);
P4M3=A30*P4M0;x4M3=P4M3(1);y4M3=P4M3(2);z4M3=P4M3(3);
A36=L(:,:,4)*L(:,:,5)*L(:,:,6);A36=A36(1:3,1:3);
P4M3=A36*P4M3;
t5=acos(y4M3/d6);
t4=atan(-z4M3/x4M3);

% Détermination de teta6(t6)

A05=L(:,:,1)*L(:,:,2)*L(:,:,3)*L(:,:,4)*L(:,:,5);
A05=A05(1:3,1:3);A05=subs(A05,{sym('t1'),sym('t2'),sym('t3'),sym('t4'),sym('t5')},{0.1,0.2,0.3,0.4,0.5});
A05=double(A05);
A50=inv(A05);
xM0=A(:,1);
xM5=A50*xM0;
m=dot([1;0;0],xM5);
t6=acos(m);

```

Les figures de base qui ont permis la détermination de θ_1 , θ_2 et θ_3 se présentent comme suit:

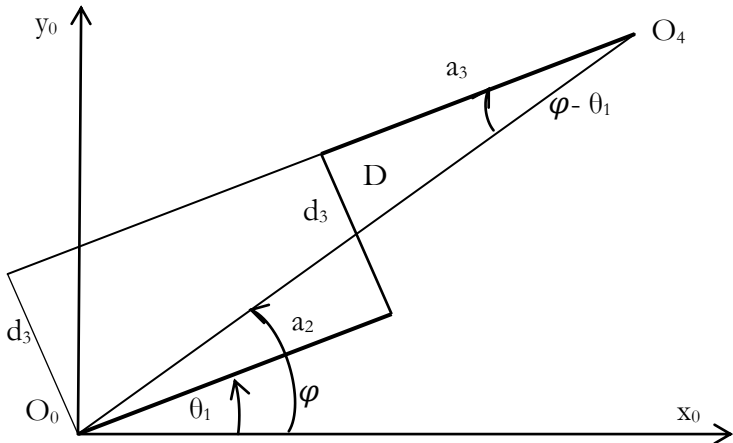


Figure 6.10: Projection du vecteur O_0O_4 dans le plan x_0y_0 .

D'après la figure ci dessus, on peut écrire:

$$s(\alpha) = \frac{d_3}{D} \quad \text{avec } \alpha = (\varphi - \theta_1) \quad (6.78)$$

$$D = \sqrt{P_x^2 + P_y^2} \quad \text{et} \quad O_0 O_4^0 = \begin{pmatrix} P_x \\ P_y \\ P_z \end{pmatrix} \quad (6.79)$$

L'angle φ est déterminé comme suit:

$$\varphi = \text{atan} \left(\frac{P_y}{P_x} \right) \quad \text{avec } \theta_1 = (\varphi - \alpha) \quad \text{d'ou } \theta_1 = 0.1.$$

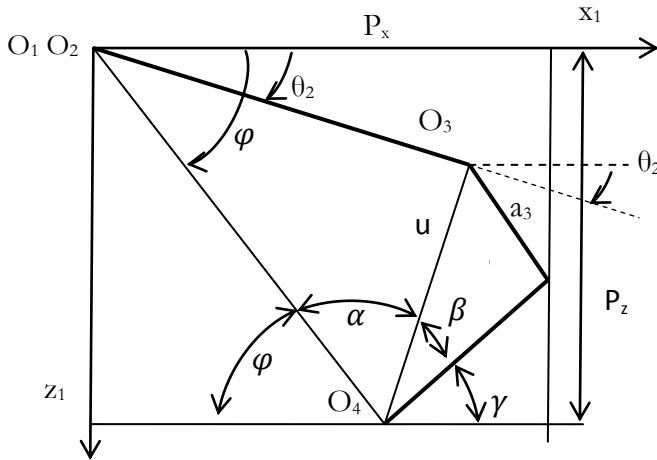


Figure 6.11: Projection du vecteur O_1O_4 dans le plan x_1z_1 .

A partir du triangle $O_1O_3O_4$, on peut écrire la relation suivante:

$$c(\varphi - \theta_2) = \frac{a_2^2 + D^2 - u^2}{2 a_2 D} = m \quad (6.80)$$

$$\theta_2 = \varphi + \text{acos}(m) = -0.2 \quad (6.81)$$

D'autre part, les angles α et β sont définis respectivement à partir des relations trigonométriques sur les triangles $O_4O_3O_1$ et O_4O_3H :

$$\cos(\alpha) = \frac{u^2 + D^2 - a_2^2}{2 u D} \text{ et } \cos(\beta) = \frac{u^2 + d_4^2 - a_3^2}{2 u d_4} \quad (6.82)$$

Tandis que l'angle γ s'exprime en fonction des autres angles de la manière suivante:

$$\gamma = \pi - (-\varphi + \alpha + \beta) \quad (6.83)$$

On peut alors écrire:

$$d_4 s\varphi + a_3 s_{23} + a_2 s_2 = P_z \quad (6.84)$$

$$a_3 s_{23} = |P_z| - d_4 s\varphi - a_2 s_2 \quad (6.85)$$

En posant $a = a_3 \cos(0.2)$, $b = a_3 \sin(0.2)$ et $c = |P_z| - d_4 s\varphi - a_2 \sin(0.2)$

$$\text{Ce qui donne } \theta_3 = 0.3 \quad (6.86)$$

Le même raisonnement est appliqué pour déterminer les variables articulaires liées à l'orientation θ_4 , θ_5 et θ_6 du robot Puma en comparaison au robot ABB (voir le programme `mg_i_puma.m`). On a simplement remplacé le point O_6 par le point M, ce qui a donné ainsi les valeurs attendues pour ces variables ($\theta_4 = 0.4$, $\theta_5 = 0.5$ et $\theta_6 = 0.6$).

MGI du Robot Fanuc M_710C/70

Le robot Fanuc M_710C/70 est aussi un robot à 6 ddl dont l'architecture ressemble au robot ABB étudié précédemment. Cependant au niveau du centre du poignet, la géométrie change et l'analyse géométrique est menée différemment. Les paramètres de DH ainsi que la représentation schématique sont présentés dans ce qui suit:

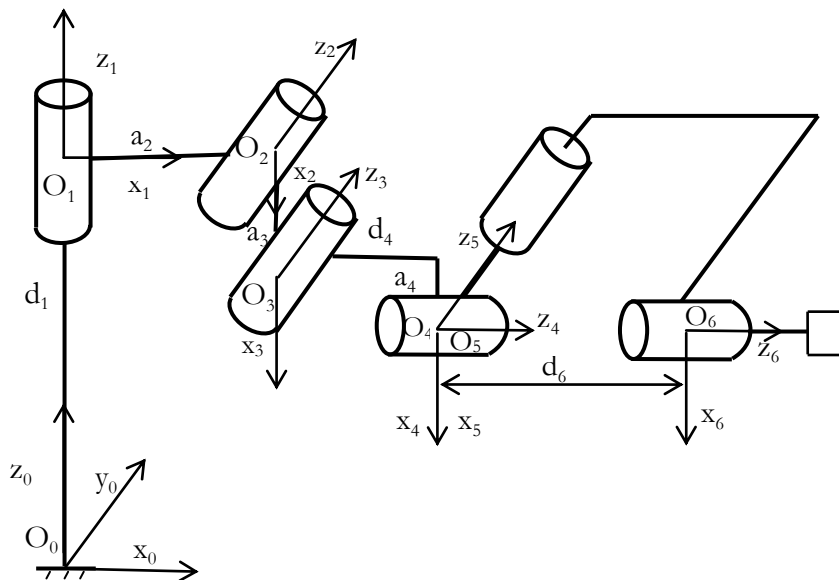


Figure 6.12: Représentation schématique du robot Fanuc.

α	a	d	θ
0	0	570	θ_1
$-\pi/2$	150	0	$\theta_2 - \pi/2$
0	870	0	θ_3
$-\pi/2$	170	1016	θ_4
$\pi/2$	0	0	θ_5
$-\pi/2$	0	175	θ_6

Tableau 6.4: Paramètres de DH du robot Fanuc.

Le programme mgi_fanuc.m permet la résolution du problème du MGI.

```
- mgi_fanuc.m

clear;
alph=[0 -pi/2 0 -pi/2 pi/2 -pi/2];
```



```

a=[0 150 870 170 0 0];
d=[570 0 0 1016 0 175];
a2=150;a3=870;a4=170;d1=570;d4=1016;d6=175;
t=[sym('t1') sym('t2')-pi/2 sym('t3') sym('t4') sym('t5')
sym('t6')];

[A,P,L]= mg_direct1(alph,a,t,d);
P=subs(P,{sym('t1'),sym('t2'),sym('t3'),sym('t4'),sym('t5'),sym('t6')},{0.1,0.2,0.3,0.4,0.5,0.6});
P=double(P);
A=subs(A,{sym('t1'),sym('t2'),sym('t3'),sym('t4'),sym('t5'),sym('t6')},{0.1,0.2,0.3,0.4,0.5,0.6});
A=double(A);

% Détermination de teta1 (t1)

P040=P-A*[0;0;d6];
Px= P040(1);Py= P040(2);Pz= P040(3);
t1=atan(Py/Px);

% Détermination de teta2 (t2) et teta3(t3)
% Projection dans le plan x1z1

A01=[cos(0.1) -sin(0.1) 0;sin(0.1) cos(0.1) 0; 0 0 1]
A10=inv(A01)
P141=A10*([0;0;-d1]+P040);
P241=P141-[a2;0;0];
pz=P241(3);px= P241(1);
alpha=atan(a4/d4);D=sqrt(px^2+pz^2);H=sqrt(d4^2+a4^2);
m=(D^2-H^2-a3^2)/(2*H*a3);
t3=asin(m)-alpha;

% Détermination de t2

a=-a3-H*sin(t3+alpha);
b=H*cos(t3+alpha);c=px;
t2= atan2(a,b)+atan2(sqrt(a^2+b^2-c^2),c);

% Orientation
% Détermination de teta4(t4) et teta5(t5)

P460=A*[0;0;d6];
A03=L(:, :, 1)*L(:, :, 2)*L(:, :, 3);A03=A03(1:3,1:3);
A03=subs(A03,{sym('t1'),sym('t2'),sym('t3')},{0.1,0.2,0.3});
A03=double(A03);
A30=inv(A03);
P463=A30*P460;x463=P463(1);y463=P463(2);z463=P463(3);
A36=L(:, :, 4)*L(:, :, 5)*L(:, :, 6);A36=A36(1:3,1:3);
P463=A36*[0;0;d6];
t5=acos(y463/d6);

```

```

t4=atan(-z463/x463);

% Détermination de teta6(t6)

A05= L(:, :, 1)*L(:, :, 2)*L(:, :, 3)*L(:, :, 4)*L(:, :, 5);
A05=A05(1:3, 1:3)
A05=subs(A05, {sym('t1'), sym('t2'), sym('t3'), sym('t4'), sym('t5')},
{0.1, 0.2, 0.3, 0.4, 0.5})
A05=double(A05);
A50=inv(A05);
x60=A(:, 1);
x65=A50*x60;
m=dot([1; 0; 0], x65);
t6=acos(m);

```

L'exécution du programme permet l'élaboration de toutes les variables articulaires. Les trois dernières variables liées à l'orientation sont déterminées exactement comme les autres cas de robots étudiés précédemment. Tandis que le problème de la position est formulé tout à fait autrement.

La variable θ_1 est obtenue en considérant la projection du vecteur position O_0O_4 dans le plan x_0y_0 comme le montre la figure suivante

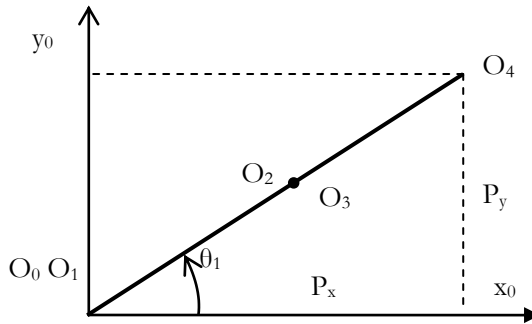


Figure 6.13: Projection sur le plan x_0y_0 .

La variable θ_1 est alors déterminée comme suit:

$$\theta_1 = \text{atan}\left(\frac{P_y}{P_x}\right) = 0.1 \quad (6.87)$$

Les variables θ_2 et θ_3 sont obtenues lors de la projection du vecteur O_1O_4 dans le plan x_1z_1 .

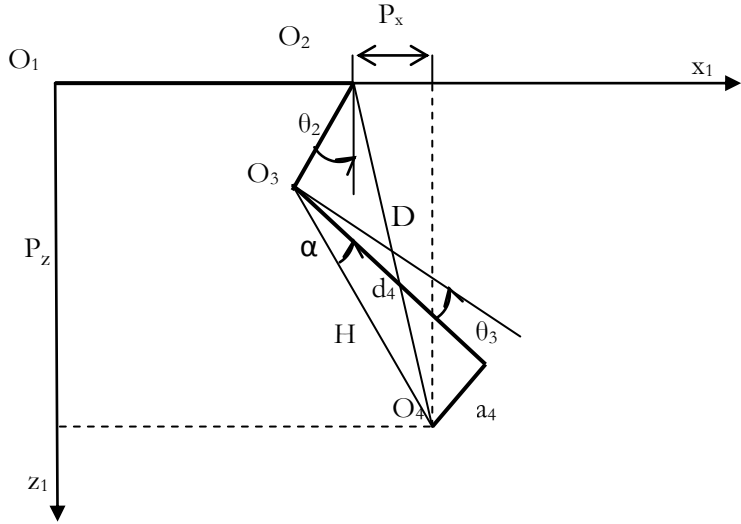


Figure 6.14: *Projection dans le plan x_1z_1 .*

Le programme permet d'une part de déterminer le vecteur $O_2O_4^1 = \begin{pmatrix} P_x \\ P_z \end{pmatrix}$. Ainsi la distance D dans la figure peut être définie comme suit:

$$D = \sqrt{P_x^2 + P_z^2} \quad (6.88)$$

La distance H est aussi importante pour la détermination des différents angles de la figure:

$$H = \sqrt{d_4^2 + a_4^2} \quad (6.89)$$

$$\text{Posons l'angle } \beta = \alpha + \theta_3 + \pi/2 \text{ avec } \alpha = \text{atan}\left(\frac{a_4}{d_4}\right) \quad (6.90)$$

On peut alors considérer la triangle $O_2O_3O_4$ pour écrire la relation trigonométrique:

$$\cos(\alpha + \theta_3 + \pi/2) = \frac{H^2 + a_3^2 - D^2}{2 H a_3} = m \text{ d'ou } \alpha + \theta_3 = \arcsin(-m) \\ \text{et } \theta_3 = \arcsin(-m) - \alpha = 0.3. \quad (6.91)$$

Pour définir θ_2 , on peut utiliser la relation suivante:

$$H \cos(\alpha + \theta_2 + \theta_3) - a_3 \sin(\theta_2) = P_x \quad (6.92)$$

En posant $\delta = \alpha + \theta_3$ la relation (6.92) devient:

$$H(c_2 c \delta - s_2 s \delta) - a_3 s_2 = P_x \quad (6.93)$$

En développant la relation (6.93), on obtient:

$$s_2(-a_3 - H s \delta) + c_2(H c \delta) = P_x \quad (6.94)$$

En posant $a = (-a_3 - H s \delta)$, $b = H c \delta$ et $c = P_x$, la solution de la relation devient $\theta_2 = 0.2$.

MGI du robot Stanford

Le robot Stanford possède également 6 ddl avec la troisième articulation qui est prismatique. Les trois dernières articulations constituent un poignet dont les centres coïncident au même point comme le montre la figure qui suit:

α	a	d	θ
0	0	d_1	θ_1
$\pi/2$	0	d_2	θ_2
$-\pi/2$	0	d_3	0
0	0	0	θ_4
$\pi/2$	0	0	θ_5
$-\pi/2$	0	0	θ_6

Tableau 6.5: Paramètres de DH du robot Stanford.

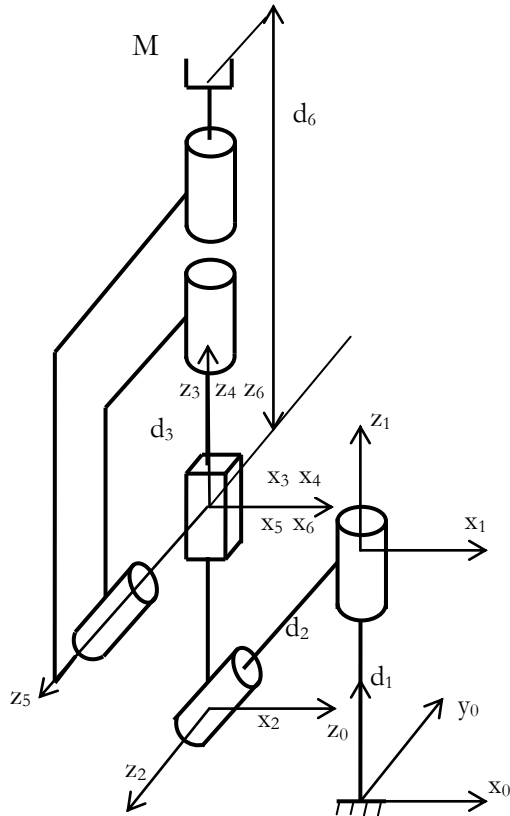


Figure 6.15: Robot Stanford à 6ddl.

Le programme `mg_i_stand` permet de déterminer dans le cas de la position les variables articulaires θ_1 , θ_2 et θ_3 . Il se base pour ce faire sur la projection du vecteur positions O_0O_4 sur le plan x_0y_0 et celle du vecteur O_1O_4 dans le plan x_1z_1 . Le cas de l'orientation est élaboré de la même manière que celui du robot Puma.

```

alph=[0 pi/2 -pi/2 0 pi/2 -pi/2];
a=[0 0 0 0 0 0];
d=[200 150 sym('d3') 0 0 0];
d1=200;d2=150;d6=80;

```

```

t=[sym('t1') sym('t2') 0 sym('t4') sym('t5') sym('t6')];

[A,P,L]= mg_direct1(alph,a,t,d);

P=subs(P,{sym('t1'),sym('t2'),sym('d3'),sym('t4'),sym('t5'),sym('t6')},{0.1,0.2,50,0.4,0.5,0.6});
P=double(P);
A=subs(A,{sym('t1'),sym('t2'),sym('d3'),sym('t4'),sym('t5'),sym('t6')},{0.1,0.2,50,0.4,0.5,0.6});
A=double(A);

% Détermination de teta1 (t1)

A01=[cos(0.1) -sin(0.1) 0;sin(0.1) cos(0.1) 0; 0 0 1];
A10=inv(A01);
P040=P;Px=P040(1);Py=P040(2);D=sqrt(Px^2+Py^2);
t1=acos((Py^2+D^2-Px^2)/(-2*Py*D))+acos(d2/D);
%détermination de teta2 (t2) et teta3(t3)
P141=A10*(P040-[0;0;d1]);px=P141(1);pz=P141(3);
t2= atan(-px/pz);
d3=sqrt(px^2+pz^2);

% Orientation
% Détermination de teta4(t4) et teta5(t5)

P4M0=A*[0;0;d6];
A03=L(:, :, 1)*L(:, :, 2)*L(:, :, 3);A03=A03(1:3,1:3);
A03=subs(A03,{sym('t1'),sym('t2'),sym('d3')},{0.1,0.2,50});
A03=double(A03);
A30=inv(A03);
P4M3=A30*P4M0;x4M3=P4M3(1);y4M3=P4M3(2);z4M3=P4M3(3);
A36=L(:, :, 4)*L(:, :, 5)*L(:, :, 6);A36=A36(1:3,1:3);
P4M3=A36*[0;0;d6];
t4=atan(y4M3/x4M3);
t5=acos(z4M3/d6);

% Détermination de teta6(t6)

A05= L(:, :, 1)*L(:, :, 2)*L(:, :, 3)*L(:, :, 4)*L(:, :, 5);
A05=A05(1:3,1:3)
A05=subs(A05,{sym('t1'),sym('t2'),sym('t3'),sym('t4'),sym('t5')},{0.1,0.2,0.3,0.4,0.5})
A05=double(A05);
A50=inv(A05);
xM0=A(:, 1);
xM5=A50*xM0;
m=dot([1;0;0],xM5);
t6=acos(m);

```

En considérant les projections dans les plans x_0y_0 et x_1z_1 comme le montre les figures suivantes et en exploitant les résultats élaborés par ce programme, on peut alors déterminer les variables articulaires θ_1 , θ_2 et d_3

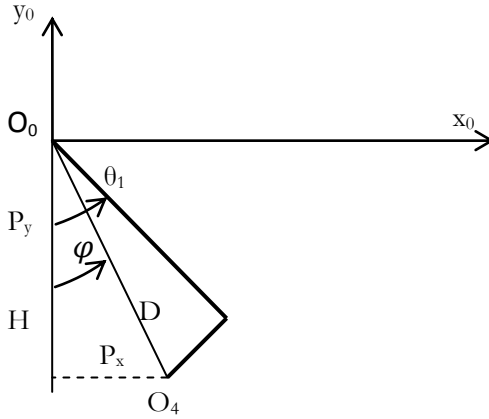


Figure 6.16: *Projection du vecteur O_0O_4 sur le plan x_0y_0 .*

En considérant les coordonnées du vecteur $O_0O_4^0 = \begin{pmatrix} P_x \\ P_y \\ P_z \end{pmatrix}$, on peut écrire

L'expression de la distance D comme suit:

$$D = \sqrt{P_x^2 + P_y^2} \quad (6.95)$$

La relation trigonométrique du triangle O_0O_4H se présente sous la forme:

$$\cos(\varphi) = \frac{P_y^2 + D^2 - P_x^2}{2 P_y D} \quad (6.96)$$

En posant

$$\alpha = \theta_1 - \varphi, \text{ on a } \cos(\alpha) = \frac{d_2}{D} \quad (6.97)$$

$$\text{D'ou } \theta_1 = \alpha + \varphi = 0.1 \quad (6.98)$$

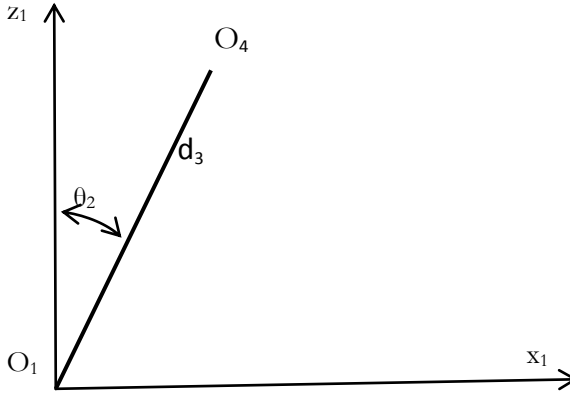


Figure 6.17: *Projection du vecteur O_1O_4 sur le plan x_1z_1 .*

Les coordonnées P_x et P_z du vecteur $O_1O_4^1$ sont obtenus à partir de la relation vectorielle suivante:

$$O_1O_4^1 = A^{1,0} (O_0O_4^0 - (0 \quad 0 \quad d_6)^T) \quad (6.99)$$

Les variables θ_2 et d_3 s'expriment alors simplement comme suit:

$$\theta_2 = \text{atg} \left(\frac{-P_x}{P_z} \right) = 0.2 \quad (6.100)$$

$$\text{et } d_3 = \sqrt{P_x^2 + P_z^2} = 50 \quad (6.101)$$

Bibliographie

- [1] Craig, j, " Introduction To Robotics, Mechanics and Control", Pearson Education International, 2005.
- [2] Wisama, K et Dombre, E, " Modélisation, identification et commande des robots", Hermes, Paris, 1999.
- [3] Safrioui, J, " Problème géométrique et lieux de singularités des manipulateurs parallèles", Thèse de l'université de Laval, 1992.
- [4] Asada, H et Slotine, J, " Robot analysis and control", John Wiley & Sons, New York, 1986.
- [5] Beji L, " Modélisation, identification et commande d'un robot parallèle", Thèse de doctorat, Université d'Evry-Val d'Essone, juin 1997.
- [6] Bennis, F, " Contribution à la modélisation géométrique et dynamique des robots à chaînes simple et complexe" Thèse de doctorat ENSM Nantes, 1991.
- [7] Delignières, S, " Choix morphologie de robots", Thèse de docteur-ingénieur, ENSM Nantes, 1987.
- [8] Gosselin, C et Angeles, J, " Singularities analysis of closed-loop kinematic chains", IEEE, on Robotics and Manufacturing, Vol. RA_6(3), p 281-290, 1990.
- [9] Kholi, D et Spanos, J, " Workspace analysis of mechanical manipulators using polyhomial discriminants", J of Mechanics, Transmissions and Automation in design, Vol. 107, Juin 1985, p 209-215.

- [10] Libre, M et Mampey, R et Chrétien J.P, " Simulation de la dynamique des robots manipulateurs motorisés", Congrès AFCET, Productique et Robotique intelligente, Besançon, nov 1983, p 197-207.
- [11] Mavroidis, C, " Résolution du problème géométrique inverse pour les manipulateurs série à six degrés de liberté", Thèse de doctorat, Université Pierre et Marie Curie, Paris, mai 1993.
- [12] Merlet , J. P, " Les robots parallèles", Hermès, Paris, 1997.

