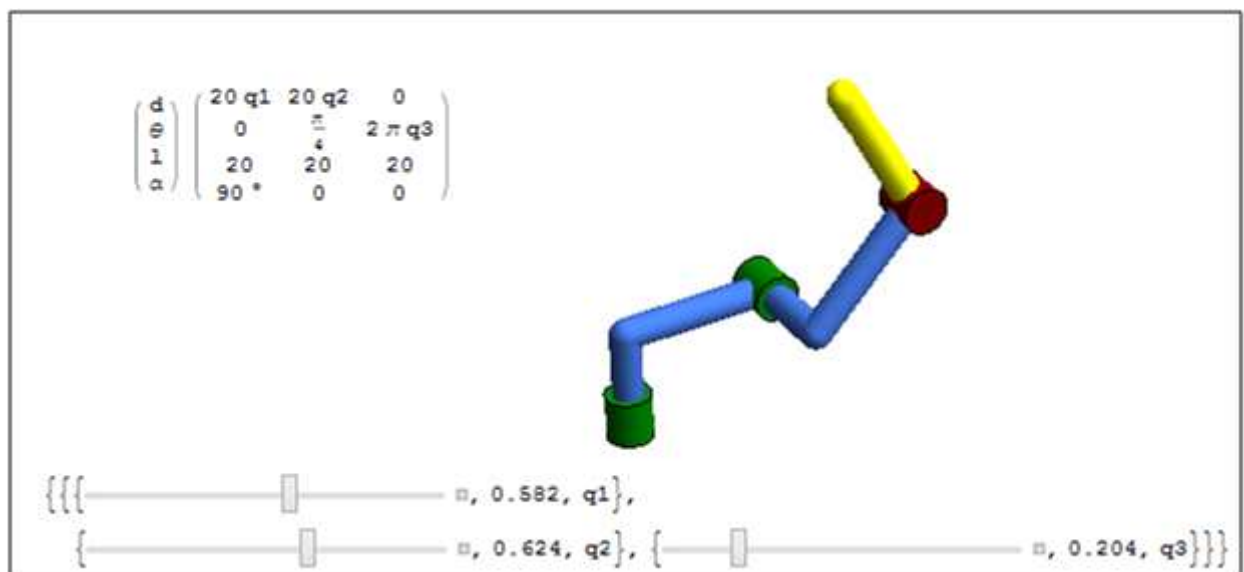


Programme de génération automatique de robots sous Wolfram Mathematica 9



Wolfram *Mathematica* 9

Seamlessly Flow Ideas to Results:
Compute. Develop. Deploy the *Mathematica* Way



Par Adrien PAJON

Université de Rennes 1
9 Rue Jean Macé
35700 Rennes

Sommaire

| | |
|--|----|
| 1- Introduction..... | 5 |
| 2- Définition et modèle en Robotique..... | 6 |
| 2-1. Définition des robots..... | 6 |
| 2-2. Modélisation des robots | 7 |
| 2-2.1. Modèle géométrique | 7 |
| 2-2.2. Convention de Denavit-Hartenberg | 8 |
| 3- Programme de génération automatique de robots | 9 |
| 3-1. Le projet | 9 |
| 3-2. Le programme | 10 |
| 3-2.1. Organisation du programme..... | 10 |
| 3-2.2. Quelques règles avant de commencer | 10 |
| 3-2.3. Génération de la structure du robot..... | 11 |
| 3-2.4. Génération du robot seul | 12 |
| 3-2.5. Génération de l'ellipsoïde de manipulabilité..... | 13 |
| 3-2.6. Génération du repère de chaque pièce | 14 |
| 3-2.7. Fonctions automatiques..... | 16 |
| 3-2.8. Générations de robots série à arborescence | 16 |
| 4- Quelques chiffres et poursuite du projet..... | 17 |
| 5- conclusion | 19 |
| 6- Bibliographie | 20 |

Table des illustrations

| | |
|--|----|
| Figure 1. : Exemple de robot modélisé sous mathematica..... | 5 |
| Figure 2. : Eléments d'un robot série. | 6 |
| Figure 3. : Liaison glissière (à gauche). | 7 |
| Figure 4. : Liaison pivot (à droite). | 7 |
| Figure 5. : Schéma des transformations de Denavit-Hartenberg entre i et $i+1$ | 9 |
| Figure 6. : Schéma des transformations de Denavit-Hartenberg entre i et $i+1$ avec A_i , B_i et C_i | 12 |
| Figure 7. : Exemple de code de génération de robot seul. | 13 |
| Figure 8. : Résultat de l'exemple de code de génération de robot seul. | 13 |
| Figure 9. : Exemple de code de génération de robot+ellipsoïde de manipulabilité..... | 14 |
| Figure 10. : Résultat de l'exemple de code de génération de robot+ellipsoïde de manipulabilité. ... | 14 |

| | |
|--|----|
| Figure 11. : Exemple de code de génération de robot+repères des pièces..... | 15 |
| Figure 12. : Résultat de l'exemple de code de génération de robot+repères des pièces. | 15 |
| Figure 13. : exemple de création de robot seul. | 16 |
| Figure 14. : Exemple de code de génération de robot à arborescence. | 17 |
| Figure 15. : Résultat de l'exemple de code de génération de robot à arborescence. | 17 |
| Figure 16. : GANT prévisionnel et réel du projet. | 18 |

Programme de génération automatique de robots sous Mathematica

1- Introduction

L'industrie moderne utilise et conçoit toujours plus de robots tous les jours. Ils s'avèrent nécessaires dans de nombreuses applications industrielles, telles que la manutention, la peinture, la soudure, le contrôle et l'assemblage mécanique.

Ce sont des systèmes mécatroniques à part entière et c'est pourquoi il est tout naturel que l'enseignement de la robotique prenne une place importante dans la scolarité des élèves de ces filières.

Ce projet est parti du constat que l'on pouvait modéliser graphiquement et commander des robots tout en affichant en instantané les différentes matrices homogènes et géométriques. Mais les modèles que l'on peut trouver ne sont pas modulables et doivent être repensés pour chaque type de robot.

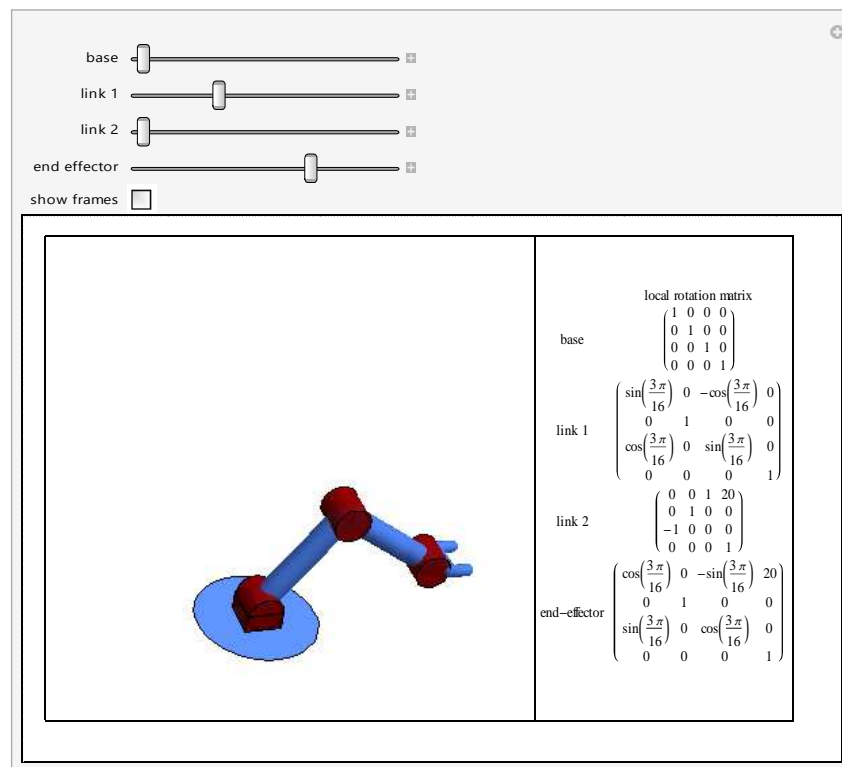


Figure 1. : Exemple de robot modélisé sous mathematica.

Le but de ce projet est de réaliser un programme de génération automatique de robot à but éducatif. Ce programme utilise la convention de Devanit-Hartenberg pour des robots séries qui seront modélisés et commandés en temps réel. Ainsi les élèves pourront facilement comprendre et tester les différents paramètres de Devanit-Hartenberg. Ensuite viendront s'ajouter des options tels

que l'affichage et le calcul de la matrice homogène de l'effecteur, l'affichage des repères des différentes liaisons et enfin le tracé des ellipses de manipulabilité.

2- Définition et modèle en Robotique

2-1. Définition des robots

Les robots sont des appareils où le mécanisme, appelé outil ou effecteur, réalise une tâche en étant guidé par les différentes articulations qui constituent son corps. Par analogie avec le bras humain, ces articulations sont donc appelées : épaule, coude ou poignet.

Un robot série est constitué de deux sous-ensembles distincts :

- Un organe terminal destiné à tenir l'effecteur qui va réaliser une ou plusieurs tâches
- Une structure mécanique articulée constituée de plusieurs chaînes de corps rigides liées par des articulations

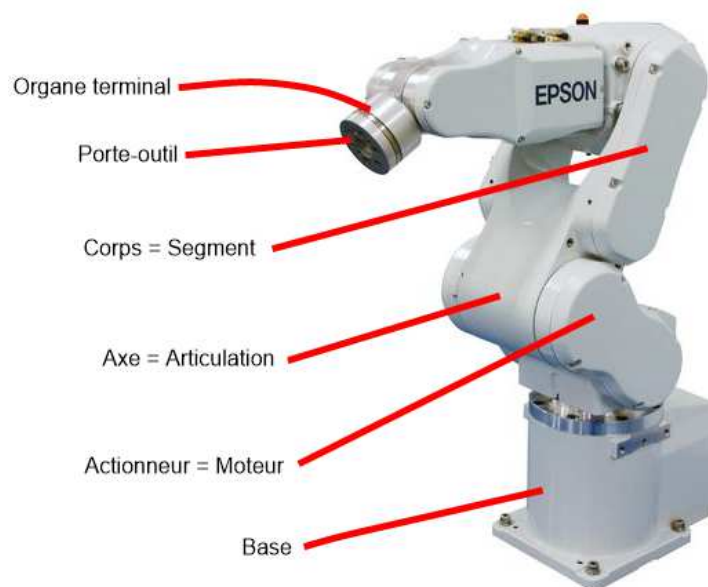


Figure 2. : *Éléments d'un robot série.*

Les chaînes peuvent être soit ouvertes (c'est souvent l'organe terminal) ou en série. On distingue deux liaisons en robotique pour représenter les articulations :

- liaison glissière ou prismatique : consiste en une translation le long d'un axe commun
- liaison pivot ou rotoïde : consiste en une rotation autour d'un axe commun

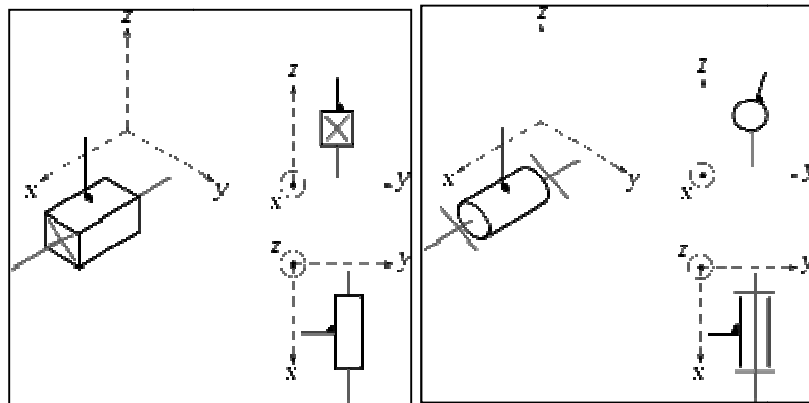


Figure 3. : Liaison glissière (à gauche).

Figure 4. : Liaison pivot (à droite).

La morphologie d'un robot peut être diverse et variée. Mais tous les robots peuvent être mis dans deux catégories : les robots série ou parallèle :

- série : chaîne cinématique ouverte dont l'organe terminal est relié à la base par une seule série de chaînes cinématiques.
- parallèle : chaîne cinématique fermée dont l'organe terminal est relié à la base par plusieurs chaînes cinématiques indépendantes

2-2. Modélisation des robots

2-2.1. Modèle géométrique

La modélisation d'un système mécanique articulé consiste à représenter son comportement par des équations algébriques. Pour ce projet, nous ne nous intéresserons qu'au modèle géométrique qui caractérise les positions. Mais il faut savoir qu'il existe aussi un modèle cinématique pour les vitesses et un modèle dynamique pour les accélérations.

L'aspect théorique de ces modèles intéresse le concepteur de robots, car il s'avère nécessaire pour réaliser des systèmes de commande performants, de faire le choix des éléments mécaniques et des actionneurs et d'étudier l'influence des différents paramètres sur les performances. Pour nous ils vont servir à paramétrer le modèle graphique du robot.

Un robot est constitué d'un ensemble de solides liés par des liaisons mécaniques. Pour chacun de ces solides, les positions et orientations relatives des liaisons sont définies par des longueurs et angles qui correspondent à des paramètres de construction du robot. Ce sont des grandeurs constantes pour un robot donné et caractérisent sa géométrie, on les appelle paramètres géométriques du robot.

On distingue le modèle géométrique direct et inverse :

- Directe : trouver l'attitude de l'organe terminal par rapport à la base, en connaissant les différentes positions articulaires.
- Inverse : trouver l'ensemble des positions articulaires qui permettent de générer une attitude connue de l'organe terminal par rapport à la base.

2-2.2. Convention de Denavit-Hartenberg

La convention de Denavit-Hartenberg est une méthode destinée à systématiser la modélisation de n'importe quel type de robots série.

Le principe est de fixer des repères à chaque corps du robot, de calculer les matrices homogènes entre chaque corps et enfin de calculer la matrice homogène entre la base et l'organe terminal.

Elle consiste en la décomposition en 4 transformations élémentaires :

- Translation le long de l'axe z_i d'une longueur d_{i+1} , donnant la matrice homogène de

$$\text{passage } T(z_i, d_{i+1}) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_{i+1} \\ 0 & 0 & 0 & 1 \end{bmatrix}_i$$

- Rotation autour de z_i d'un angle θ_{i+1} , donnant la matrice homogène de passage

$$R(z_i, \theta_{i+1}) = \begin{bmatrix} \cos \theta_{i+1} & -\sin \theta_{i+1} & 0 & 0 \\ \sin \theta_{i+1} & \cos \theta_{i+1} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}_i$$

- Translation le long de l'axe x_{i+1} d'une longueur r_{i+1} , donnant la matrice homogène

$$\text{de passage } T(x_{i+1}, r_{i+1}) = \begin{bmatrix} 1 & 0 & 0 & r_{i+1} \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}_{i+1}$$

- Rotation autour de x_{i+1} d'un angle α_{i+1} , donnant la matrice homogène de passage

$$R(x_{i+1}, \alpha_{i+1}) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos \alpha_{i+1} & -\sin \alpha_{i+1} & 0 \\ 0 & \sin \alpha_{i+1} & \cos \alpha_{i+1} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}_{i+1}$$

Ces transformations étant faites par rapport au repère i , la matrice de transformation DH_i est la matrice homogène entre i et $i+1$ que l'on obtient avec la formule :

$$DH_i = T(z_i, d_{i+1}) \cdot R(z_i, \theta_{i+1}) \cdot T(x_{i+1}, r_{i+1}) \cdot R(x_{i+1}, \alpha_{i+1})$$

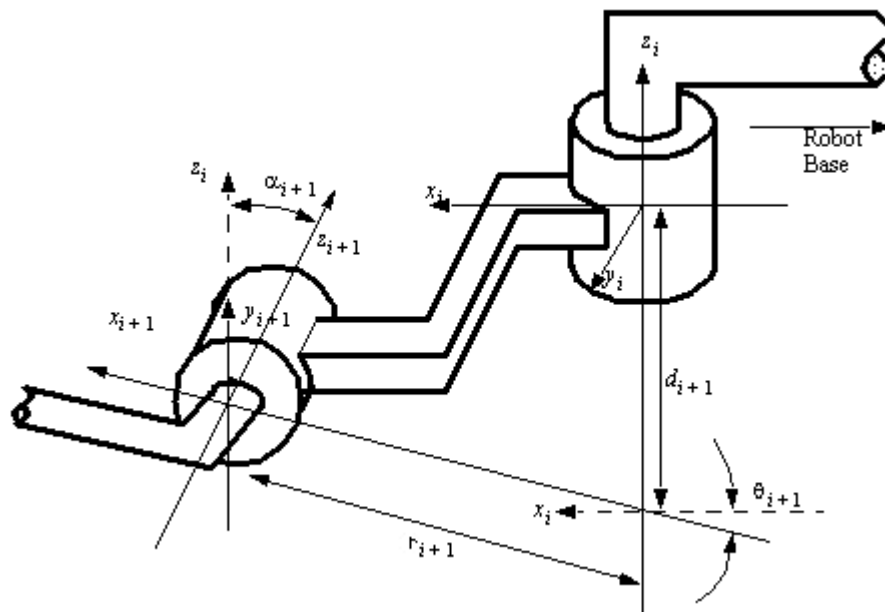


Figure 5. : Schéma des transformations de Denavit-Hartenberg entre i et $i+1$.

Les termes d_i , θ_i , r_i et α_i sont appelés les paramètres de Denavit-Hartenberg. Ils sont uniques à chaque structure de robot. Ils se présentent sous la forme d'un tableau :

$$\begin{bmatrix} d \\ \theta \\ r \\ \alpha \end{bmatrix} \begin{bmatrix} d_1 & \dots & d_i \\ \theta_1 & \dots & \theta_i \\ r_1 & \dots & r_i \\ \alpha_1 & \dots & \alpha_i \end{bmatrix}$$

La matrice Homogène quant à elle permet d'obtenir facilement la partie relative à la position de l'organe terminal en prenant la partie vectorielle de cette matrice.

$$DH_i = \begin{bmatrix} ? & ? & ? & A \\ ? & ? & ? & B \\ ? & ? & ? & C \\ 0 & 0 & 0 & 1 \end{bmatrix}_i \Rightarrow \begin{bmatrix} x = A \\ y = B \\ z = C \end{bmatrix}_i$$

3- Programme de génération automatique de robots

3-1. Le projet

Comme énoncé précédemment, le but de ce projet est de réaliser un programme sous Mathematica qui permet automatiquement de générer la structure d'un robot et de la commander avec un but éducatif pour la compréhension de la Convention de Denavit-Hartenberg. Ce programme respecte quelques règles. Il doit :

- pouvoir générer le robot à partir du tableau des paramètres de Denavit-Hartenberg seuls,
- permettre de bouger le robot en direct,
- afficher la matrice homogène calculée en direct en fonction de la position du robot,

- pouvoir afficher l'ellipsoïde de manipulabilité calculée pour chaque position du robot,
- pouvoir afficher les repères liés à chaque pièce du robot ainsi que de l'organe terminal.

3-2. Le programme

3-2.1. Organisation du programme

Le programme qui a été réalisé s'organise autour de 4 fichiers qui servent de bibliothèques :

- 01_General.nb : appelle et exécute les fichiers suivants
- 02_MatriceGeom.nb : rassemble les fonctions relatives à la transformation du tableau de Denavit-Hartenberg en matrice homogène, géométrique et au calcul de la jacobienne
- 03_FormulesRobotAuto.nb : rassemble les fonctions relatives à la création des différents volumes de la structure du robot, l'ellipsoïde de manipulabilité, les repères des pièces et la création des boutons de contrôle
- 04_RobotAutoTotal.nb : rassemble les fonctions relatives à la création automatique d'un robot série

Ces fichiers sont détaillés en ANNEXE.

Dans la suite, nous allons voir comment utiliser ces fonctions pour générer des robots.

3-2.2. Quelques règles avant de commencer

Dans les paramètres de Denavit-Hartenberg, d_i et θ_i sont les seuls paramètres à pouvoir être des variables, en sachant qu'un seul peut l'être à la fois.

Les noms des fonctions sous Mathematica commencent par une majuscule (ex : `Ma fonction[]`, `MAFONCTION[]`, `MaFonCTION[]`) et les variables par une minuscule (ex : `monparamètre`, `mONPARAMETRE`, `monParaMETRE`). Cela permet d'éviter toute confusion si on veut donner le même nom à une fonction et à une variable (ex : `"MatriceDH"` pour une fonction créant une matrice de Denavit-Hartenberg et `"matriceDH"` comme variable de stockage pour cette matrice).

Le tableau de Denavit-Hartenberg s'écrit sous Mathematica :

$$\text{matriceDH} = \{\{d_1, \dots, d_i\}, \{\theta_1, \dots, \theta_i\}, \{r_1, \dots, r_i\}, \{\alpha_1, \dots, \alpha_i\}\};$$

Les paramètres variables seront nommés q_1, \dots, q_i les rendant plus facile à réinitialiser avec :

`Clear["q * "]`

Mais attention à ne pas créer de fonction ou de variable en commençant par la lettre "q" sinon elles seront également effacées.

Si lors de la création d'un tableau de Denavit-Hartenberg, certains termes sont écrits avec des variables, il faudra penser à leur donner une valeur avant que le robot ne soit généré pour éviter des erreurs.

Ensuite les boutons ne permettant de faire varier les variables que de 0 à 1, il faudra penser à adapter les q_i en fonction des plages que l'on veut (ex : si on veut que q_1 translate entre 10 et 30, il faudra écrire " $20*q_1+10$ ", et si on veut que q_2 tourne entre 2π et 3π ce sera " $\pi*q_2+2\pi$ ")

Les couleurs du robot généré représentent :

- tube bleu = corps des pièces
- cylindre rouge = liaison pivot
- cylindre vert = liaison glissière
- tube jaune = corps de l'organe terminal

Enfin pour lancer le programme, il faut utiliser le fichier "01_General.nb", et l'évaluer complètement. Puis soit créer un nouveau notebook ou utiliser un notebook déjà prérempli en utilisant les fonctions que nous allons voir par la suite.

3-2.3. Génération de la structure du robot

Le but est de créer automatiquement les différents volumes du robot à partir de son tableau de Denavit-Hartenberg étendu aux liaisons. Pour ce faire on va balayer tout le tableau pour créer chaque sous-partie du robot comprenant :

- un tube qui représente le corps du robot caractérisé par 3 points : début du tube, point d'inflexion, fin du tube
- un cylindre qui représente la liaison : centré au début du tube et dont la couleur est fonction du type de liaison

Nous allons voir maintenant comment sont calculés les points caractéristiques du corps. Reprenons le schéma représentant les transformations de Denavit-Hartenberg et rajoutant des points A_i , B_i et C_i :

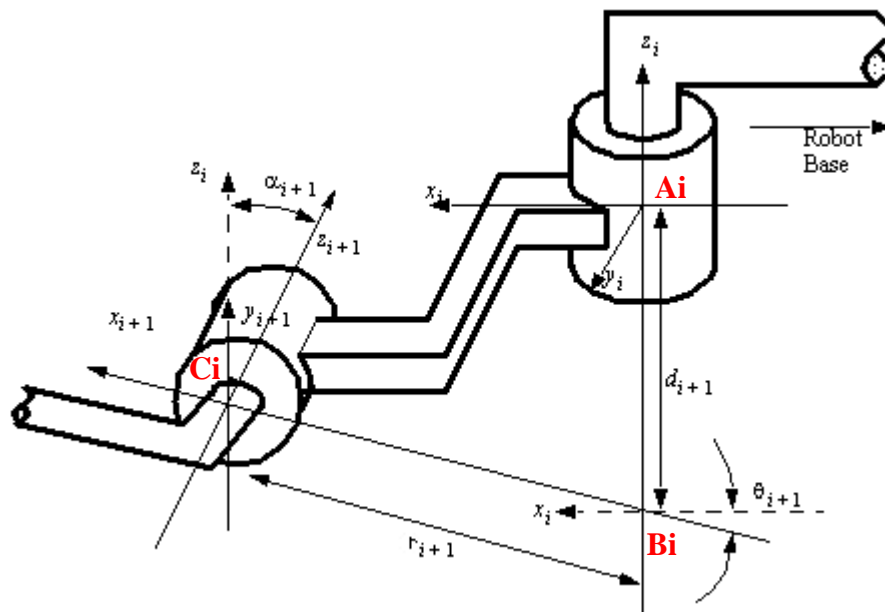


Figure 6. : Schéma des transformations de Denavit-Hartenberg entre i et $i+1$ avec A_i , B_i et C_i .

Le point A_i représente le point de début du corps, B_i le point d'inflexion et C_i la fin du corps de la pièce i .

On constate que $A_i = C_{i-1}$. Donc pour obtenir la position de A_i , il suffit de prendre la partie vectorielle de $[\prod_{s=0}^{i-1} DH_s]$.

B_i quant à lui est le point A_i ayant juste subi la première translation. Donc on obtient B_i en prenant la partie vectorielle de $[\prod_{s=0}^{i-1} DH_s] \cdot T(z_i, d_i)$.

Enfin C_i est obtenu en prenant la partie vectorielle de $[\prod_{s=0}^i DH_s]$.

3-2.4. Génération du robot seul

La marche à suivre est :

- 1) Effacement des éventuelles variables " q_i " qui aurait pu être créées avant
- 2) Création d'une matrice qui représente le tableau de Denavit-Hartenberg
- 3) Affichage de ce tableau
- 4) Affichage de la matrice Homogène du robot
- 5) Création d'un tableau de Denavit-Hartenberg étendu en spécifiant si la liaison est une pivot ou une glissière
- 6) Création des différents volumes de la structure du robot
- 7) Affichage des volumes du robot dans une même fenêtre
- 8) Création des boutons de contrôle

```

Clear["q*"]
matriceDH = {{q1, q2, 0} * 20, {0, 1/8, q3} * 2 * Pi, {20, 20, 20}, {90°, 0, 0}};
MatrixForm[{{d}, {θ}, {r}, {α}}] MatrixForm[matriceDH]
MatrixForm[MatriceHomo[matriceDH, 0, Length[Transpose[matriceDH]]]] // Dynamic
matriceDHLiaisons = CreateLiaisons[matriceDH];
robot := CreateRobot[matriceDHLiaisons];
Show[{robot},
  ImageSize -> {650, 475}, PlotRange -> {{-50, 50}, {-50, 50}, {-50, 50}}, ViewAngle -> Pi/15,
  Lighting -> "Neutral"
] // Dynamic
Boutons[matriceDH]

```

Figure 7. : Exemple de code de génération de robot seul.

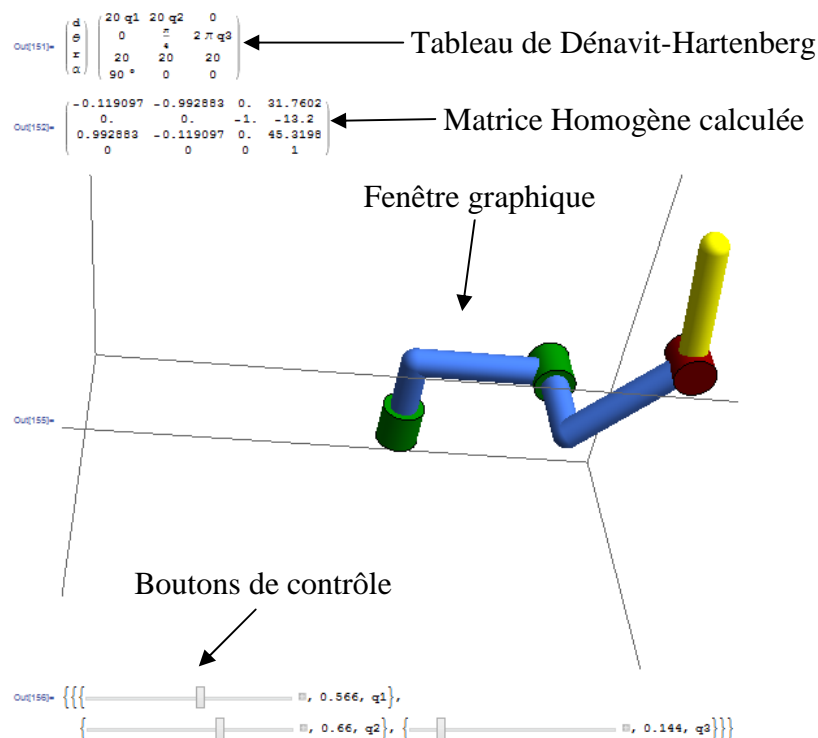


Figure 8. : Résultat de l'exemple de code de génération de robot seul.

3-2.5. Génération de l'ellipsoïde de manipulabilité

La marche à suivre est identique à la précédente, en rajoutant les étapes suivantes entre les étapes 6 et 7 de la méthode qui génère le robot seul :

- 1) Création de la matrice Homogène du robot
- 2) Création de la matrice Géométrique à partir de la matrice Homogène
- 3) Création de la matrice Jacobienne tirée de la matrice Géométrique
- 4) Réduction de la Jacobienne aux termes liés à la translation
- 5) Appel du package "MultivariateStatistics" qui permet de créer des ellipsoïdes
- 6) Création de l'ellipsoïde de manipulabilité
- 7) Rajout de l'ellipsoïde de manipulabilité à l'affichage du robot

```

In[481]:= Clear["q*"]
matriceDH = {{q1, q2, 0} * 20, {0, 1/8, q3} * 2 * Pi, {20, 20, 20}, {90°, 0, 0}};
MatrixForm[{{d}, {θ}, {τ}, {α}}] MatrixForm[matriceDH]
MatrixForm[MatriceHomo[matriceDH, 0, Length[Transpose[matriceDH]]]] // Dynamic
matriceDHliaisons = CreateLiaisons[matriceDH];
robot := CreateRobot[matriceDHliaisons];

mHomo = MatriceHomo[matriceDH, 0, Length[Transpose[matriceDH]]];
mGeom = MatriceGeom[mHomo];
mJacobienne = Jacobienne[mGeom, matriceDH];
mJacobienneTranslations := Transpose[Transpose[(mJacobienne)[[1 ;; 3]]][[1 ;; 3]]];
Needs["MultivariateStatistics`"]
ellipsoide :=
Graphics3D[{RGBColor[0, .49, .49], Opacity[0.5],
  Ellipsoid[VecteurFinal[matriceDH, 0, Length[Transpose[matriceDH]]][[1]],
    Sequence@@Eigensystem[mJacobienneTranslations]]}],

Show[{robot, ellipsoide},
  ImageSize -> {650, 475}, PlotRange -> {{-100, 100}, {-100, 100}, {-100, 100}},
  ViewAngle -> Pi / 15, Lighting -> "Neutral"
] // Dynamic
Boutons[matriceDH]

```

code de création de l'ellipsoïde

Figure 9. : Exemple de code de génération de robot+ellipsoïde de manipulabilité.

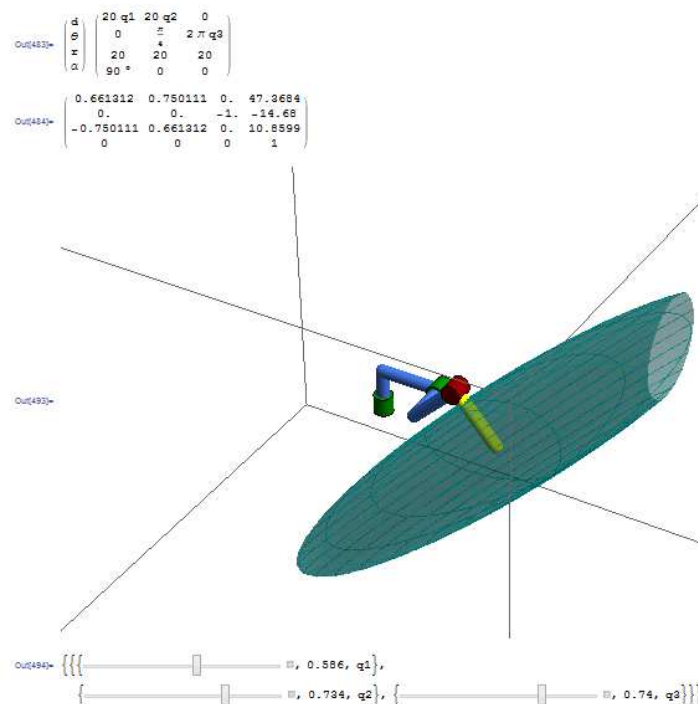


Figure 10. : Résultat de l'exemple de code de génération de robot+ellipsoïde de manipulabilité.

3-2.6. Génération du repère de chaque pièce

La marche à suivre est comme la précédente, sauf qu'il n'y a qu'une seule étape à rajouter entre les étapes 6 et 7 de la méthode pour générer le robot seul :

- 1) Création de l'ensemble des repères des pièces du robot
- 2) Rajout des repères des pièces à l'affichage du robot

Le système de couleur utilisé pour les repères est la norme :

- rouge = axe X
- vert = axe Y
- bleu = axe Z

```
Clear["q*"]
matriceDH = {{q1, q2, 0} * 20, {0, 1/8, q3} * 2 * Pi, {20, 20, 20}, {90°, 0, 0}};
MatrixForm[{{d}, {θ}, {r}, {α}}] MatrixForm[matriceDH]
MatrixForm[MatriceHomo[matriceDH, 0, Length[Transpose[matriceDH]]]] // Dynamic
matriceDHLiaisons = CreateLiaisons[matriceDH];
robot := CreateRobot[matriceDHLiaisons];

reperes := CreateReperes[matriceDHLiaisons, 20]; code de création des repères pièces

Show[{robot, reperes},
  ImageSize -> {650, 475}, PlotRange -> {{-50, 50}, {-50, 50}, {-50, 50}}, ViewAngle -> Pi/15,
  Lighting -> "Neutral"
] // Dynamic
Boutons[matriceDH]
```

Figure 11. : Exemple de code de génération de robot+repères des pièces.

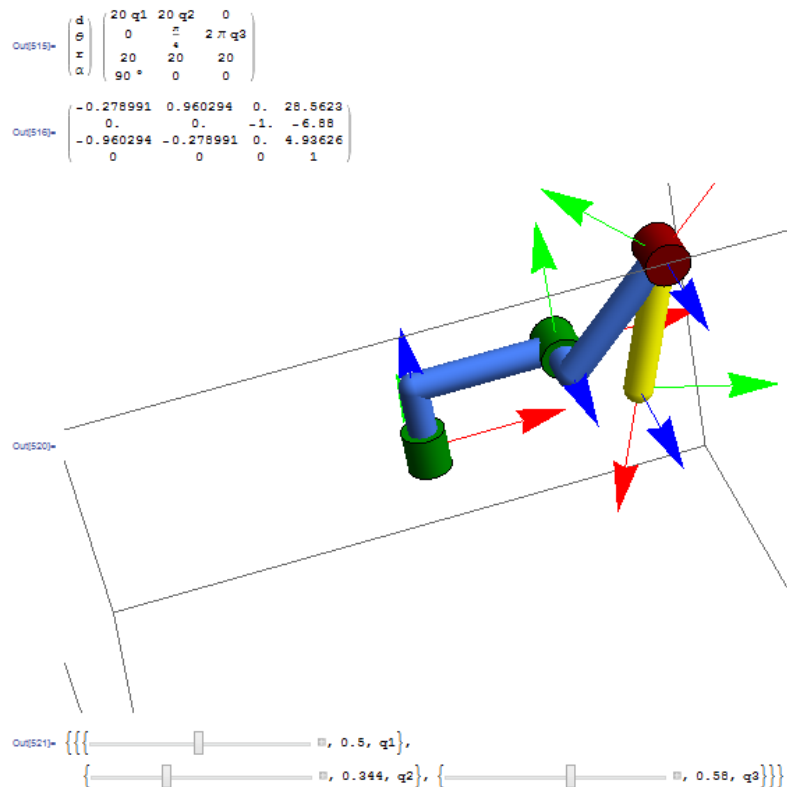


Figure 12. : Résultat de l'exemple de code de génération de robot+repères des pièces.

3-2.7. Fonctions automatiques

Pour plus de simplicité les méthodes ont été réunies dans des fonctions uniques qui sont :

- CreateRobotTotal[] : création du robot seul
- CreateRobotEllipse[] : création du robot avec son ellipsoïde de manipulabilité
- CreateRobotReperes[] : création du robot avec les repères liés aux différentes pièces

La marche à suivre est :

- Effacement des éventuelles variables "qi" qui auraient put être créées avant
- Création d'une matrice qui est le tableau de Denavit-Hartenberg
- Insertion du tableau dans la fonction adéquat en n'oubliant pas les paramètres de taille de fenêtre graphique

```

n[32]= Clear["q*"]
matriceDH = {{q1, q2, 0} * 20, {0, 1 / 8, q3} * 2 * Pi, {20, 20, 20}, {90°, 0, 0}};
CreateRobotTotal[matriceDH, 50, -50, 50, -50, 50, -50]

```

code de génération automatique d'un robot série

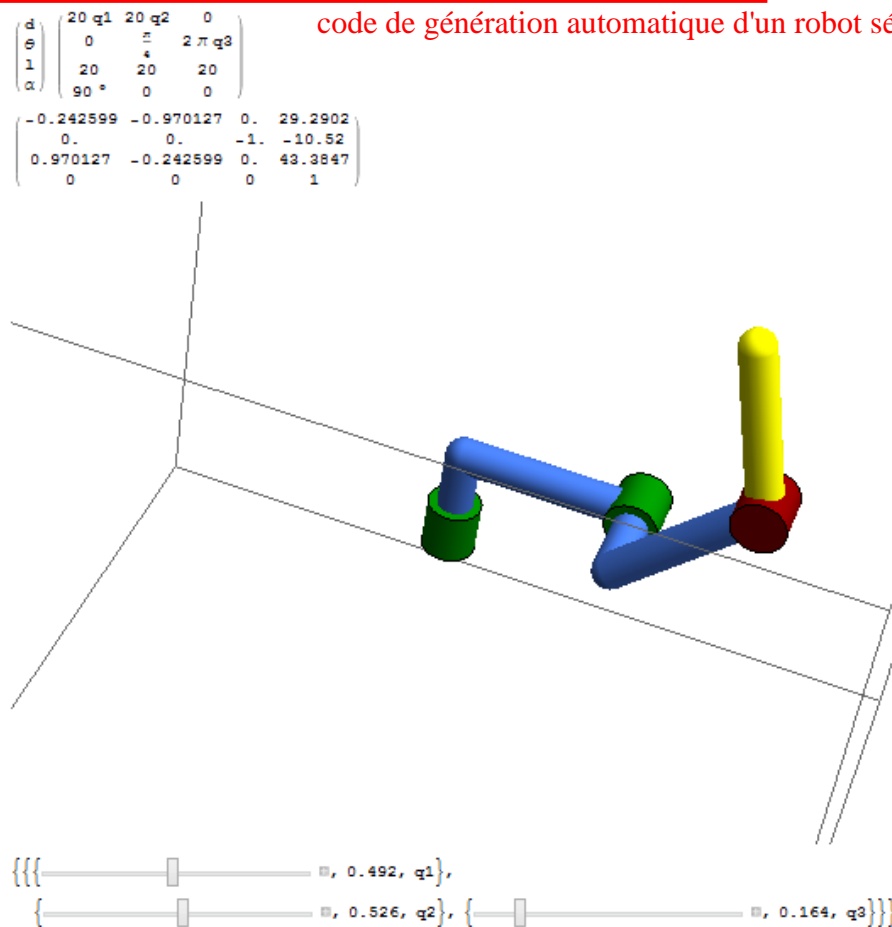


Figure 13. : exemple de création de robot seul.

3-2.8. Générations de robots série à arborescence

Ici pas de possibilité d'utiliser les fonctions automatiques de créations de robot. Il faut donc générer un robot série pour chaque arborescence en repartant de la base.

```

In[571]:= Clear["q*"]

matriceDH1 = {{q1, q2, 0} * 20, {0, 1/8, q3} * 2 * Pi, {20, 20, 20}, {90°, 0, 0}};
MatrixForm[{{d}, {θ}, {x}, {α}}] MatrixForm[matriceDH1]
MatrixForm[MatriceHomo[matriceDH1, 0, Length[Transpose[matriceDH1]]]] // Dynamic
matriceDHLiaisons1 = CreateLiaisons[matriceDH1];
robot1 := CreateRobot[matriceDHLiaisons1]; code de la partie 1 du robot

matriceDH2 = {{q1, q2, 0, 0, 0} * 20, {0, 1/8, q4, q5, q6} * 2 * Pi, {20, 20, 20, 20, 20},
{90°, 0, 0, 0, 0}};
MatrixForm[{{d}, {θ}, {x}, {α}}] MatrixForm[matriceDH2]
MatrixForm[MatriceHomo[matriceDH2, 0, Length[Transpose[matriceDH2]]]] // Dynamic
matriceDHLiaisons2 = CreateLiaisons[matriceDH2];
robot2 := CreateRobot[matriceDHLiaisons2]; code de la partie 2 du robot

Show[{robot1, robot2},
ImageSize -> {650, 475}, PlotRange -> {{-50, 100}, {-50, 100}, {-50, 100}},
ViewAngle -> Pi / 15, Lighting -> "Neutral"
] // Dynamic
Boutons[matriceDH1]
Boutons[matriceDH2]

```

Figure 14. : Exemple de code de génération de robot à arborescence.

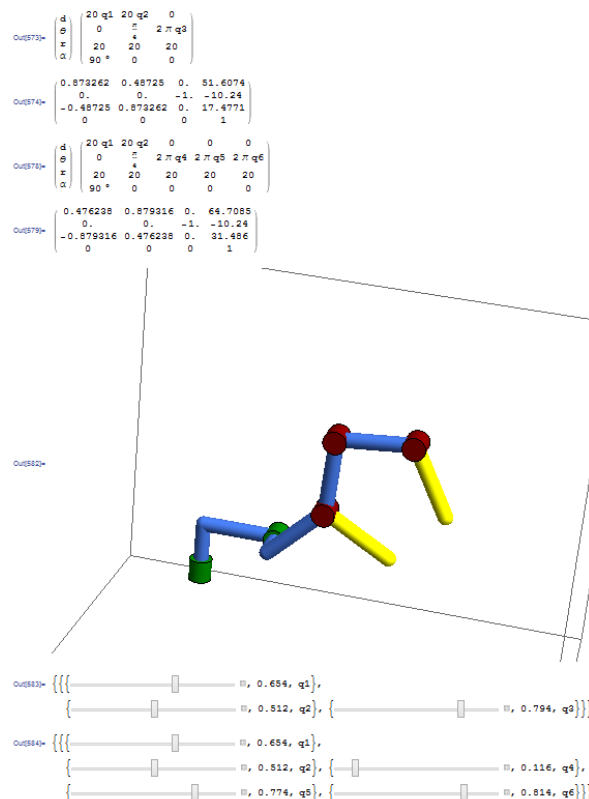


Figure 15. : Résultat de l'exemple de code de génération de robot à arborescence.

4- Quelques chiffres et poursuite du projet

Au final ce projet s'est déroulé sur 17 semaines dont voici le GANT de répartition du travail :

| année | 2013 | | | | | | | | | | | 2014 | | | | | |
|---|------|----|----|----|----|----|----|----|----|----|----|------|---|---|---|---|---|
| semaines | 41 | 42 | 43 | 44 | 45 | 46 | 47 | 48 | 49 | 50 | 51 | 1 | 2 | 3 | 4 | 5 | 6 |
| compréhension du sujet | | | | | | | | | | | | | | | | | |
| compréhension de Mathematica | | | | | | | | | | | | | | | | | |
| réalisation des fonctions de travail sur DH | | | | | | | | | | | | | | | | | |
| génération de la structure du robot | | | | | | | | | | | | | | | | | |
| génération de robot en automatique | | | | | | | | | | | | | | | | | |
| amélioration du code + rajout d'options | | | | | | | | | | | | | | | | | |
| rapport | | | | | | | | | | | | | | | | | |

| | | | | | | | | | | | | | | | | | |
|---|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|
| compréhension du sujet | | | | | | | | | | | | | | | | | |
| compréhension de Mathematica | | | | | | | | | | | | | | | | | |
| réalisation des fonctions de travail sur DH | | | | | | | | | | | | | | | | | |
| génération de la structure du robot | | | | | | | | | | | | | | | | | |
| génération de robot en automatique | | | | | | | | | | | | | | | | | |
| amélioration du code + rajout d'options | | | | | | | | | | | | | | | | | |
| rapport | | | | | | | | | | | | | | | | | |

au plus tard
 au plus tôt
 réel

Figure 16. : GANT prévisionnel et réel du projet.

Le temps nécessaire à la réalisation du projet a été de 132h. Ce qui d'après une estimation par une entreprise de prestation en développement de programmes informatiques revient à 16,5 jours hommes travaillés soit un coût estimé minimal de 5000€ juste pour le développement.

Pour un tel programme à vue éducatif, l'idée serait de le vendre en tant que licence en faisant une marge de 25% sur les 50 premières vendues. Ce qui reviendrait à vendre chaque licence à 125€/licence. Le département mécatronique de l'université Rennes 1 n'ayant que 10 licences Mathematica, le coût total estimé pour ce projet aurait été de 1250€ s'il avait été réalisé à l'extérieur.

Pour la suite du projet, plusieurs points seraient à améliorer ou des fonctionnalités à rajouter :

- Améliorer la fonction `CreatRobot[]` pour qu'elle soit plus optimisée, notamment pour la génération de robot par arborescence. Actuellement les pièces communes aux différentes parties sont recrées et réaffichées ce qui pourrait rallonger en temps de calcul si l'on crée un robot avec beaucoup d'arborescences,
- Contrôler en cinématique et en dynamique le robot en plus du contrôle en position,
- Rajouter une fonctionnalité pour prendre des objets et les déplacer dans la fenêtre graphique,
- Rajouter des fonctionnalités pour prendre en compte les inerties du robot et ainsi connaître les efforts mis en jeu dans les différentes articulations,
- Rajouter des liens pour que le modèle généré par Mathematica qui représenterait le squelette d'un robot puisse contrôler un vrai robot modélisé par un modelleur 3D.

5- conclusion

La génération automatique de robot est une problématique qui intéresse aussi bien l'éducation que les industriels.

J'ai trouvé ce projet très intéressant car d'une part il m'a permis de bien comprendre la convention de Denavit-Hartenberg et comment l'utiliser, et d'autre part il m'a permis de me réconcilier avec Mathematica qui m'avait laissé un mauvais souvenir suite à l'utilisation que j'en avais eu en école d'ingénieur. En effet, j'avais comme enseignant des membres du GROUMF (GROUpe d'Utilisateur de Mathematica Francophones) qui nous demandait de créer des programmes avec une telle rigueur et en considérant que le fonctionnement de Mathematica était évident alors qu'il n'en ai rien.

Mais ayant muri et étant maintenant très intéressé par la programmation en règle générale, j'ai dû pour ce projet passer de nombreuses heures à comprendre la logique de fonctionnement de Mathematica et j'ai presque dû aller jusqu'à son code source pour comprendre le système d'affectation de variables dynamiques et pouvoir ainsi générer et commander des robots.

6- Bibliographie

[1] : Modélisation Dynamique de Robots en Chaînes Arborescentes : Calcul formel et Programmation sous MAPLE - Fatima DJAAFRIA - rapport de projet, février 2012

[2] : Etude et réalisation d'un programme de calcul automatique d'un modèle dynamique optimisé pour un robot avec des bouclages cinématiques - François-Xavier ROCHE - rapport de PFE, juin 1991

[3] : Introduction au calcul formel. Application à la modélisation dynamique des mécanismes robotiques en chaînes arborescentes - Christophe GONZALES et Christophe NEGRIER - rapport de projet

[4] : Exploitation de la redondance pour la commande coordonnée d'un manipulateur mobile d'assistance aux personnes handicapées - Khiair NAIT CHABANE - mémoire de thèse, novembre 2006

Nétographie

[w1] : Site de Wolfram <http://demonstrations.wolfram.com/ForwardKinematics/>, visité en octobre 2013.

[w2] : Site d'aide en ligne de Mathematica
<http://reference.wolfram.com/mathematica/guide/Mathematica.html>, visité en janvier 2014.

[w3] : <http://fr.wikipedia.org/wiki/Denavit-Hartenberg>, visité en juin 2013

Sommaire des annexes

| | |
|-------------------------------|----|
| 01_General.nb | 22 |
| 02_MatriceGeom.nb | 23 |
| 03_FormulesRobotAuto.nb | 24 |
| 04_RobotAutoTotal.nb | 26 |
| 99_Robottest.nb | 29 |

01_General.nb

```
(*Initialisation de mathematica*)
(*Suppression de toutes les variables en mémoires*)
Clear["Global`*"]
(*Definition du répertoire contenant ce notebook comme étant le répertoire principal*)
SetDirectory[NotebookDirectory[]]
C:\Users\Lytha\Desktop\projet M2\projet m2_simplifié_final_commenté
(*Utilisation du fichier 02_MatriceGeom.nb comme un package*)
(*Contient toutes les fonction de transformations matricielles d'un tableau de Denavit-Hartenberg*)
(*ouverture du notebook 02_MatriceGeom.nb*)
NotebookOpen[ToFileName[Directory[], "02_MatriceGeom.nb"]];
(*selection du notebook qui vient d'etre ouvert comme notebook principal*)
SelectedNotebook[];
(*evaluation du notebook principal*)
NotebookEvaluate[%];
(*fermeture du notebook principal*)
NotebookClose[InputNotebook[]]
(*selection du notebook restant comme notebook principal, ici 01_general.nb*)
SelectedNotebook[];
(*Utilisation du fichier 03_FormulesRobotAuto.nb comme un package*)
(*Contient toutes les fonction de génération des volumes representant un robot ainsi que pour son contrôle*)
NotebookOpen[ToFileName[Directory[], "03_FormulesRobotAuto.nb"]];
SelectedNotebook[];
NotebookEvaluate[%];
NotebookClose[InputNotebook[]]
SelectedNotebook[];
(*Utilisation du fichier 04_RobotAutoTotal.nb comme un package*)
(*Contient toutes les fonction de génération automatique d'un robot*)
NotebookOpen[ToFileName[Directory[], "04_RobotAutoTotal.nb"]];
SelectedNotebook[];
NotebookEvaluate[%];
NotebookClose[InputNotebook[]]
SelectedNotebook[];
(*ouverture du fichier de test de génération de robots*)
NotebookOpen[ToFileName[Directory[], "99_Robottest.nb"]];
```

02_MatriceGeom.nb

```
(*Fonction qui renvoie un vecteur avec toutes les variables d'un matrice de Denavit-
Hartenberg (mDH)*)
VariablesDH[mDH_] := Variables[Take[mDH, 4, All]]
(*Fonctions renvoyant les matrices de rotation selon  $\psi$  et selon  $\theta$ *)
B $\psi$ [x_] := {{Cos[x], -Sin[x], 0}, {Sin[x], Cos[x], 0}, {0, 0, 1}};
B $\theta$ [x_] := {{1, 0, 0}, {0, Cos[x], -Sin[x]}, {0, Sin[x], Cos[x]}};
(*Fonction qui renvoie la matrice de passage entre le repère de la pièce
initiale (pinit) et la pièce finale (pfin) à partir de la matrice de Denavit-
Hartenberg (mDH)*)
(*si la pièce initiale est le bati, pinit=0*)
(*si la pièce final est la dernière pièce de la matrice de Denavit-Hartenberg,
pfin=Length[Transpose[matriceDH]]*)
MatriceRepere[mDH_, pinit_, pfin_] :=
If[pinit == pfin, IdentityMatrix[3],
Simplify[If[pinit < pfin - 1, B $\psi$ [mDH[[2, pinit + 1]]].B $\theta$ [mDH[[4, pinit + 1]]].
MatriceRepere[mDH, pinit + 1, pfin], B $\psi$ [mDH[[2, pfin]]].B $\theta$ [mDH[[4, pfin]]]]]]
(*Fonction qui renvoie la matrice de passage entre le repère de la pièce
initiale (pinit) et la première matrice de rotation de la pièce finale
(pfin) à partir de la matrice de Denavit-Hartenberg (mDH)*)
MatriceRepere $\psi$ [mDH_, pinit_, pfin_] :=
MatriceRepere[mDH, pinit, pfin - 1].B $\psi$ [mDH[[2, pfin]]]
(*Fonction qui renvoie la position de la fin de la pièce pfin et dans le
repère de la pièce pinit à partir de la matrice de Denavit-Hartenberg (mDH)*)
VecteurFinal[mDH_, pinit_, pfin_] := If[pinit == pfin,
{{0, 0, 0}},
{Simplify[Sum[
MatriceRepere[mDH, 0, i - 1].{0, 0, mDH[[1, i]]}
+ MatriceRepere $\psi$ [mDH, 0, i].{mDH[[3, i]], 0, 0},
{i, pinit + 1, pfin}]]
}}]
(*Fonction qui renvoie la position de l'inflexion de la pièce pfin et dans
le repère de la pièce pinit à partir de la matrice de Denavit-Hartenberg (mDH)*)
VecteurIntermediaire[mDH_, pinit_, pfin_] :=
VecteurFinal[mDH, pinit, pfin - 1] +
{MatriceRepere[mDH, 0, pfin - 1].{0, 0, mDH[[1, pfin]]}};
(*Fonction qui renvoie la matrice homogène entre la pièce initial (pinit) et
la pièce finale (pfin) de la matrice de Denavit-Hartenberg (mDH)*)
MatriceHomo[mDH_, pinit_, pfin_] :=
Join[Join[MatriceRepere[mDH, pinit, pfin], Transpose[VecteurFinal[mDH, pinit, pfin]], 2],
{{0, 0, 0, 1}}]
(*Algorithme qui renvoie la matrice géométrique à partir de la matrice de Denavit-
Hartenberg (mDH)*)
X[mDH_] := mDH[[1, 4]]
Y[mDH_] := mDH[[2, 4]]
Z[mDH_] := mDH[[3, 4]]
 $\psi$ [mDH_] := If[mDH[[2, 3]]  $\neq$  0, If[mDH[[1, 3]]  $\neq$  0, ArcTan[-mDH[[1, 3]]/mDH[[2, 3]]],  $\infty$ ,
false]
tf[mDH_] := Sin[ $\psi$ [mDH]] * mDH[[1, 3]] - Cos[ $\psi$ [mDH]] * mDH[[2, 3]];
 $\theta$ [mDH_] := If[mDH[[3, 3]]  $\neq$  0, ArcTan[tf[mDH]/mDH[[3, 3]],  $\pi/2$ ]
ta[mDH_] := -Cos[ $\psi$ [mDH]] * mDH[[1, 2]] - Sin[ $\psi$ [mDH]] * mDH[[2, 2]];
tc[mDH_] := Cos[ $\psi$ [mDH]] * mDH[[1, 1]] + Sin[ $\psi$ [mDH]] * mDH[[2, 1]];
```

```

 $\phi[mDH] := \text{If}[(\psi[mDH] \neq \text{false} \ \&\& \ \psi[mDH] \neq \infty),$ 
   $\text{If}[\text{tc}[mDH] \neq 0, \text{ArcTan}[\text{ta}[mDH] / \text{tc}[mDH]], \pi / 2], \text{false}]$ 
MatriceGeom[mDH_] :=
  Simplify[PowerExpand[{X[mDH], Y[mDH], Z[mDH],  $\psi[mDH]$ ,  $\theta[mDH]$ ,  $\phi[mDH]$ }]
(*Fonction qui renvoie la Jacobienne de la matrice géométrique (mgeom) en
  fonction des variables de la matrice de Denavit-Hartenberg (mDH)*)
Jacobienne[mgeom_, mDH_] := Simplify[D[mgeom, {VariablesDH[mDH]}]];

```

03_FormulesRobotAuto.nb

```

(*Fonction qui génère le corps de la pièce a de la matrice de Denavit-
Hartenberg (matriceDH)*)
(*le dernier chiffre (ici 2) est la largeur du tube*)
CreateCorps[matriceDH_, a_] :=
  Graphics3D[{RGBColor[.25, .43, .82],
    Tube[{VecteurFinal[matriceDH, 0, a-1][[1]],
      VecteurIntermediaire[matriceDH, 0, a][[1]],
      VecteurFinal[matriceDH, 0, a][[1]]}, 2]};
(*Fonction qui génère la pivot de la pièce a de la matrice de Denavit-
Hartenberg (matriceDH)*)
(*le dernier chiffre (ici 3) est la largeur du cylindre*)
CreatePivot[matriceDH_, a_] :=
  Graphics3D[{RGBColor[.49, 0, 0],
    Cylinder[{VecteurFinal[matriceDH, 0, a-1][[1]] +
      MatriceRepere[matriceDH, 0, a-1].{0, 0, -3},
      VecteurFinal[matriceDH, 0, a-1][[1]] +
      MatriceRepere[matriceDH, 0, a-1].{0, 0, +3}}, 3]};
(*Fonction qui génère la glissière de la pièce a de la matrice de Denavit-
Hartenberg (matriceDH)*)
(*le dernier chiffre (ici 3) est la largeur du cylindre*)
CreateGlissiere[matriceDH_, a_] :=
  Graphics3D[{RGBColor[0, .49, 0],
    Cylinder[{VecteurFinal[matriceDH, 0, a-1][[1]] +
      MatriceRepere[matriceDH, 0, a-1].{0, 0, -3},
      VecteurFinal[matriceDH, 0, a-1][[1]] +
      MatriceRepere[matriceDH, 0, a-1].{0, 0, +3}}, 3]};
(*Fonction qui génère le corps de l'effecteur a de la matrice de Denavit-
Hartenberg (matriceDH)*)
(*le dernier chiffre (ici 2) est la largeur du tube*)
CreateFinal[matriceDH_, a_] :=
  Graphics3D[{RGBColor[49, 49, 0], JoinForm["Round"],
    Tube[{VecteurFinal[matriceDH, 0, a-1][[1]],
      VecteurIntermediaire[matriceDH, 0, a][[1]],
      VecteurFinal[matriceDH, 0, a][[1]]}, 2]};

```



```
(*Fonction qui génère automatiquement les volumes de toutes les pièce
et liaisons du robot à partir de la matrice de Denavit-
Hartenberg étendu aux liaisons (matriceDHetendue*)
CreateRobot[matriceDHetendue_] :=
Table [
  If[i == Length[Transpose[matriceDHetendue[[2]]]],
    If[matriceDHetendue[[1, i]],
      {CreateFinal[matriceDHetendue[[2]], i],
       CreateGlissiere[matriceDHetendue[[2]], i]},
      {CreateFinal[matriceDHetendue[[2]], i],
       CreatePivot[matriceDHetendue[[2]], i]}],
    If[matriceDHetendue[[1, i]],
      {CreateCorps[matriceDHetendue[[2]], i],
       CreateGlissiere[matriceDHetendue[[2]], i]},
      {CreateCorps[matriceDHetendue[[2]], i],
       CreatePivot[matriceDHetendue[[2]], i]}]
  ],
  {i, Length[Transpose[matriceDHetendue[[2]]]]};
(*Fonction qui renvoie un vecteur avec le type de liaison dans la
matrice de Denavit-Hartenberg (matriceDH*)
(*1=liaison glissière*)
(*0=liaison pivot*)
RechercheGlissiere[matriceDH_] :=
Table[Variables[matriceDH[[1, i]]] == {VariablesDH[matriceDH][[i]]},
  {i, Length[Transpose[matriceDH]]};
(*Fonction qui renvoie la matrice de Denavit-
Hartenberg etendue aux liaisons à partir de a matrice de Denavit-
Hartenberg (matriceDH*)
CreateLiaisons[matriceDH_] := {RechercheGlissiere[matriceDH], matriceDH}
(*Fonction qui crée un bouton lié à la variable (m) et qui réaffiche
le nom de cette variable*)
CreationBouton[m_] := {Manipulator[Dynamic[m]], Dynamic[m], m}
(*Fonction qui crée un bouton lié à la variable numéro (a) du vecteur
(m) ne comprenant que des variables*)
BoutonDepuisVecteur[m_, a_] := CreationBouton[m[[a]]]
(*Fonction qui crée une série de boutons à partir d'un vecteur (m) ne
comprenant que des variables*)
Boutons[m_] :=
{Clear["q*"]; Table[BoutonDepuisVecteur[VariablesDH[m], i],
  {i, Length[VariablesDH[m]]}}
(*Fonctions qui génèrent une flèche de longueur (taille) qui représente l'axe X,
Y ou Z du bout de la pièce (a) de la matrice de Denavit-Hartenberg (matriceDH*)
AxeX[matriceDH_, a_, taille_] :=
Graphics3D[
  {Red, Arrow[{VecteurFinal[matriceDH, 0, a][[1]],
    VecteurFinal[matriceDH, 0, a][[1]] +
    MatriceRepere[matriceDH, 0, a].{taille, 0, 0}}, 0.05]};
```

```

AxeY[matriceDH_, a_, taille_] :=
Graphics3D[
  {Green, Arrow[{VecteurFinal[matriceDH, 0, a][[1]],
    VecteurFinal[matriceDH, 0, a][[1]] +
    MatriceRepere[matriceDH, 0, a].{0, taille, 0}}, 0.05]}}];
AxeZ[matriceDH_, a_, taille_] :=
Graphics3D[
  {Blue, Arrow[{VecteurFinal[matriceDH, 0, a][[1]],
    VecteurFinal[matriceDH, 0, a][[1]] +
    MatriceRepere[matriceDH, 0, a].{0, 0, taille}}, 0.05]}}];
(*Fonction qui génère un repère XYZ de longueur (taille) du bout de
la pièce (a) de la matrice de Denavit-Hartenberg (matriceDH)*)
CreateRepere[matriceDH_, a_, taille_] :=
  {AxeX[matriceDH, a, taille], AxeY[matriceDH, a, taille],
  AxeZ[matriceDH, a, taille]};
(*Fonction qui génère le repère XYZ de longueur (taille) du bout de
toutes les pièces de la matrice de Denavit-Hartenberg (matriceDH)*)
CreateReperes[matriceDH_, taille_] :=
Table[CreateRepere[matriceDH[[2]], i - 1, taille],
  {i, Length[Transpose[matriceDH[[2]]]] + 1}]

```

04_RobotAutoTotal.nb

```

(*Fonction qui permet d'afficher la matrice de Denavit-
Hartenberg (matriceDH) puis de calculer et d'afficher sa matrice homogène en direct*)
PrintMatriceRobot[matriceDH_] := (
  Print[MatrixForm[{{d}, {0}, {1}, {a}}] MatrixForm[matriceDH]];
  Print[Dynamic[MatrixForm[MatriceHomo[matriceDH, 0, Length[Transpose[matriceDH]]]]]];
);
(*Fonction qui permet de générer la modélisation du robot seul à partir de la matrice de Denavit-
Hartenberg (matriceDH)*)
(*les valeur xpos, xneg representent respectivement la valeur max et min de la fenêtre
graphique suivant x*)
(*les valeur ypos, yneg representent respectivement la valeur max et min de la fenêtre
graphique suivant y*)
(*les valeur zpos, zneg representent respectivement la valeur max et min de la fenêtre
graphique suivant z*)
CreateRobotTotal[matriceDH_, xpos_, xneg_, ypos_, yneg_, zpos_, zneg_] := (
  (*affiche la matrice de Denavit-Hartenberg et la matrice géométrique*)
  PrintMatriceRobot[matriceDH];
  (*creation de la matrice de Denavit-Hartenberg étendue aux liaisons*)
  mliaisonCRT = CreateLiaisons[matriceDH];
  (*affichage du robot*)
  Print[Dynamic[Show[CreateRobot[mliaisonCRT], ImageSize -> {650, 475},
    PlotRange -> {{xneg, xpos}, {yneg, ypos}, {zneg, zpos}}, ViewAngle -> Pi/15,
    Lighting -> "Neutral"]]];
  (*creation des boutons de commande*)
  Print[Boutons[matriceDH]];
);

```

```
(*Ouverture du package MultivariateStatistics comprenant la creation d'ellipsoïdes*)
Needs["MultivariateStatistics`"];
(*Fonction qui génère et calcul l'ellipsoïde de manipulabilité à partir de la matrice de Denavit-
Hartenberg (matriceDH)*)
(*Cette ellipsoïde est stockée dans la fonction EllipseCER*)
(*L'ellipsoïde est calculé à partir de la jacobienne de la matrice géométrique du robot
dont on prend la base de de vecteur propre et ses valeurs propres*)
CreateEllipseRobot[matriceDH_] := (
  (*calcul de la jacobienne*)
  mjCER = Jacobienne[MatriceGeom[MatriceHomo[matriceDH, 0, Length[Transpose[matriceDH]]]],
    matriceDH];

  (*extraction des paramètres linéaires de la jacobienne*)
  MbaseCER := Transpose[Transpose[(mjCER)[[1 ;; 3]]][[1 ;; 3]]];

  (*génération de l'ellipsoïde*)
  EllipseCER :=
    Graphics3D[{RGBColor[0, .49, .49], Opacity[0.5],
      Ellipsoid[VecteurFinal[matriceDH, 0, Length[Transpose[matriceDH]]][[1]],
        Sequence @@ Eigensystem[MbaseCER]]}],
  );

(*Fonction qui permet de générer la modélisation du robot avec son ellipsoïde de
manipulabilité à partir de la matrice de Denavit-Hartenberg (matriceDH)*)
(*les valeur xpos, xneg représentent respectivement la valeur max et min de la fenêtre
graphique suivant x*)
(*les valeur ypos, yneg représentent respectivement la valeur max et min de la fenêtre
graphique suivant y*)
(*les valeur zpos, zneg représentent respectivement la valeur max et min de la fenêtre
graphique suivant z*)
CreateRobotEllipse[matriceDH_, xpos_, xneg_, ypos_, yneg_, zpos_, zneg_] := (
  (*affiche la matrice de Denavit-Hartenberg et la matrice géométrique*)
  PrintMatriceRobot[matriceDH];
  (*creation de la matrice de Denavit-Hartenberg étendue aux liaisons*)
  mliaisonCRE = CreateLiaisons[matriceDH];
  (*creation de l'ellipsoïde de manipulabilité du robot*)
  CreateEllipseRobot[matriceDH];
  (*affichage du robot et de son ellipsoïde de manipulabilité*)
  Print[Dynamic[Show[{CreateRobot[mliaisonCRE], EllipseCER}, ImageSize -> {650, 475},
    PlotRange -> {{xneg, xpos}, {yneg, ypos}, {zneg, zpos}}, ViewAngle -> Pi/15,
    Lighting -> "Neutral"]]];
  (*creation des boutons de commande*)
  Print[Boutons[matriceDH]];
  );
```

```
(*Fonction qui permet de générer la modélisation du robot avec les repères attachés à
chaque liaisons à partir de la matrice de Denavit-Hartenberg (matriceDH)*)
(*les valeur xpos, xneg representent respectivement la valeur max et min de la fenêtre
graphique suivant x*)
(*les valeur ypos, yneg representent respectivement la valeur max et min de la fenêtre
graphique suivant y*)
(*les valeur zpos, zneg representent respectivement la valeur max et min de la fenêtre
graphique suivant z*)
CreateRobotReperes[matriceDH_, xpos_, xneg_, ypos_, yneg_, zpos_, zneg_, tailrep_] := (
  (*affiche la matrice de Denavit-Hartenberg et la matrice géométrique*)
  PrintMatriceRobot[matriceDH];
  (*creation de la matrice de Denavit-Hartenberg étendue aux liaisons*)
  mliaisonCRR = CreateLiaisons[matriceDH];
  (*affichage du robot et des repères attachés a chacune de ses liaisons*)
  Print[Dynamic[Show[{CreateRobot[mliaisonCRR], CreateReperes[mliaisonCRR, tailrep]},
    ImageSize -> {650, 475}, PlotRange -> {{xneg, xpos}, {yneg, ypos}, {zneg, zpos}},
    ViewAngle -> Pi/15, Lighting -> "Neutral"}]]];
  (*creation des boutons de commande*)
  Print[Boutons[matriceDH]];
);
```

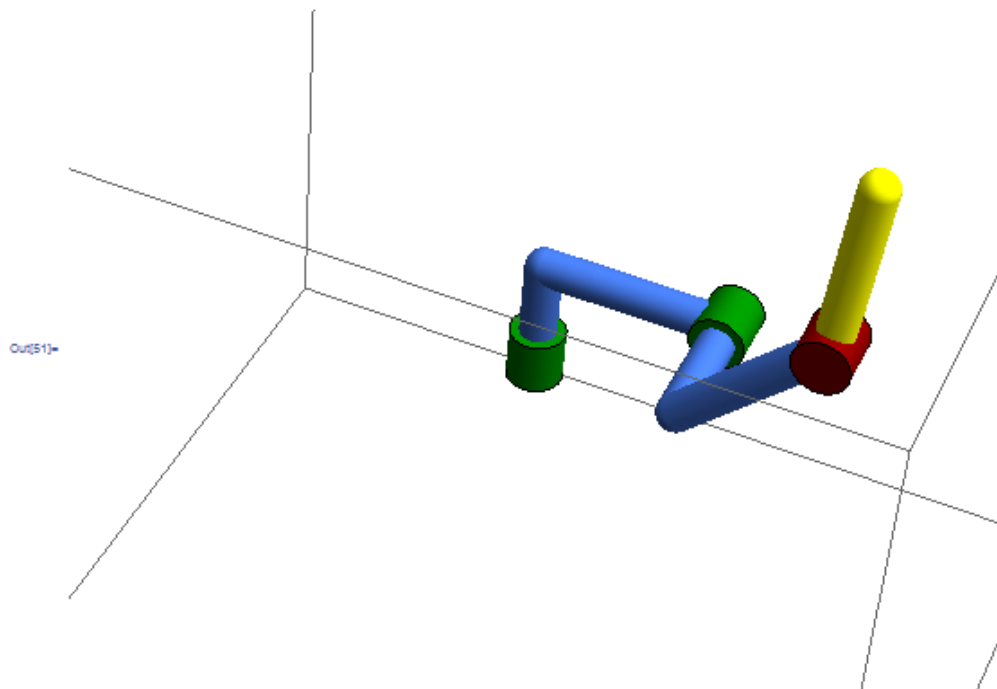
99_Robottest.nb

Les 3 exemples suivant montrent comment générer des robots série.

```
(*Algorithme de creation de robot seul*)
(*effacement de toutes les variables et fonctions commençant par q*)
Clear["q*"]
(*exemple de matrice de Denavit-Hartenberg*)
matriceDH = {{q1, q2, 0} * 20, {0, 1/8, q3} * 2 * Pi, {20, 20, 20}, {90°, 0, 0}};
(*affichage de la matrice de Denavit-Hartenberg*)
MatrixForm[{{d}, {θ}, {r}, {α}}] MatrixForm[matriceDH]
(*affichage de la matrice homogène de la matrice de Denavit-Hartenberg*)
MatrixForm[MatriceHomo[matriceDH, 0, Length[Transpose[matriceDH]]]] // Dynamic
(*creation de la matrice de la matrice de Denavit-Hartenberg étendue aux liaisons*)
matriceDHLiaisons = CreateLiaisons[matriceDH];
(*generation des volumes du robot*)
robot := CreateRobot[matriceDHLiaisons];
(*affichage des volumes du robot dans une seule fenêtre*)
Show[{robot},
  ImageSize -> {650, 475}, PlotRange -> {{-50, 50}, {-50, 50}, {-50, 50}}, ViewAngle -> Pi/15, Lighting -> "Neutral"
] // Dynamic
(*creation des boutons de commande du robot*)
Boutons[matriceDH]
```

$$\text{Out[47]=} \begin{pmatrix} d \\ \theta \\ r \\ \alpha \end{pmatrix} \begin{pmatrix} 20 q_1 & 20 q_2 & 0 \\ 0 & \frac{\pi}{4} & 2 \pi q_3 \\ 20 & 20 & 20 \\ 90^\circ & 0 & 0 \end{pmatrix}$$

$$\text{Out[48]=} \begin{pmatrix} 0.0941083 & -0.995562 & 0. & 36.0243 \\ 0. & 0. & -1. & -14.16 \\ 0.995562 & 0.0941083 & 0. & 45.6534 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$



$$\text{Out[51]=} \left\{ \left\{ \left\{ \text{Slider}, 0.58, q_1 \right\}, \left\{ \text{Slider}, 0.708, q_2 \right\}, \left\{ \text{Slider}, 0.11, q_3 \right\} \right\} \right\}$$

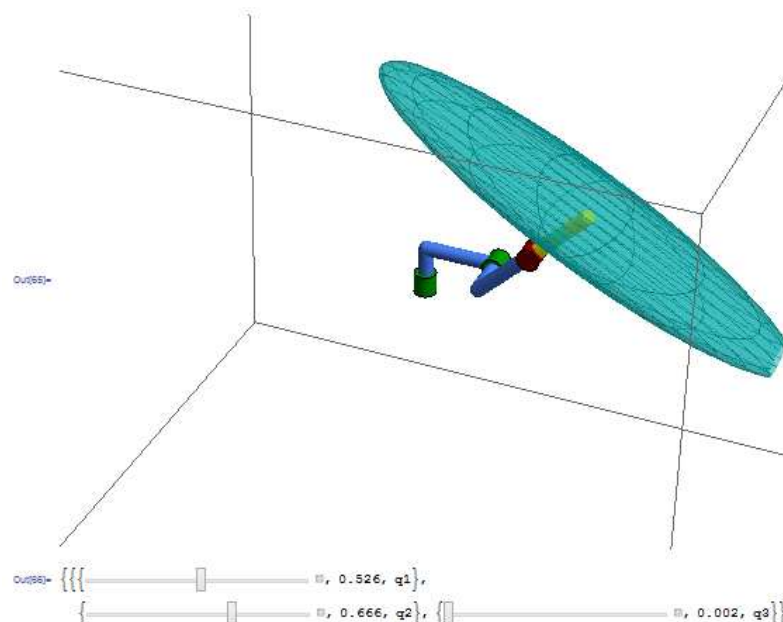
```
(*Algorithme de creation de robot seul*)
(*idem que pour la generation de robot seul*)
Clear["q*"]
matriceDH = {{q1, q2, 0}*20, {0, 1/8, q3}*2*Pi, {20, 20, 20}, {90°, 0, 0}};
MatrixForm[{{d}, {θ}, {r}, {α}}] MatrixForm[matriceDH]
MatrixForm[MatriceHomo[matriceDH, 0, Length[Transpose[matriceDH]]]] // Dynamic
matriceDHLiaisons = CreateLiaisons[matriceDH];
robot := CreateRobot[matriceDHLiaisons];

(*Algorithme de creation de l'ellipsoïde de manipulabilité*)
(*creation de la matrice homogène*)
mHomo = MatriceHomo[matriceDH, 0, Length[Transpose[matriceDH]]];
(*creation de la matrice géométrique*)
mGeom = MatriceGeom[mHomo];
(*creation de la jacobienne*)
mJacobienn = Jacobienne[mGeom, matriceDH];
(*suppression des termes non linéaires de la jacobienne*)
mJacobiennTranslations := Transpose[Transpose[mJacobienn][[1 ;; 3]]][[1 ;; 3]];
(*Ouverture du package MultivariateStatistics comprenant la creation d'ellipsoïdes*)
Needs["MultivariateStatistics`"]
(*creation de l'ellipsoïde de manipulabilité à partir de la base propre de la jacobienne*)
ellipsoide :=
Graphics3D[{RGBColor[0, .49, .49], Opacity[0.5],
Ellipsoid[VecteurFinal[matriceDH, 0, Length[Transpose[matriceDH]]][[1]],
Sequence@@Eigensystem[mJacobiennTranslations]]}],

(*affichage des volumes du robot et de son ellipsoïde de manipulabilité dans une seule fenêtre*)
Show[{robot, ellipsoide},
ImageSize -> {650, 475}, PlotRange -> {{-100, 100}, {-100, 100}, {-100, 100}}, ViewAngle -> Pi/15,
Lighting -> "Neutral"
] // Dynamic
(*creation des boutons de commande du robot*)
Boutons[matriceDH]
```

$$\text{Out[55]} = \begin{pmatrix} d \\ \theta \\ r \\ \alpha \end{pmatrix} = \begin{pmatrix} 20 q_1 & 20 q_2 & 0 \\ 0 & \frac{\pi}{4} & 2 \pi q_3 \\ 20 & 20 & 20 \\ 90^\circ & 0 & 0 \end{pmatrix}$$

$$\text{Out[55]} = \begin{pmatrix} 0.698165 & -0.715936 & 0. & 48.1054 \\ 0. & 0. & -1. & -19.92 \\ 0.715936 & 0.698165 & 0. & 38.9809 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$



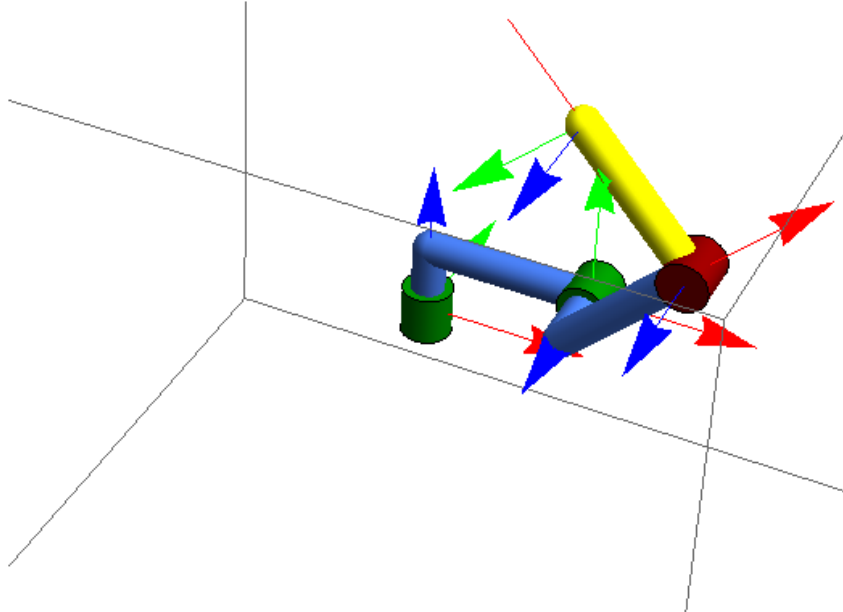
```
(*Algorithme de creation de robot seul*)
(*idem que pour la generation de robot seul*)
Clear["q*"]
matriceDH = {{q1, q2, 0} * 20, {0, 1/8, q3} * 2 * Pi, {20, 20, 20}, {90°, 0, 0}};
MatrixForm[{{d}, {θ}, {r}, {α}}] MatrixForm[matriceDH]
MatrixForm[MatriceHomo[matriceDH, 0, Length[Transpose[matriceDH]]] // Dynamic
matriceDHliaisons = CreateLiaisons[matriceDH];
robot := CreateRobot[matriceDHliaisons];




(*creation des repères liés à chacune des pièces du robot*)
reperes := CreateReperes[matriceDHliaisons, 20];

(*affichage des volumes du robot et des repères liés à chacune de ses pièces dans une seule fenêtre*)
Show[{robot, reperes},
  ImageSize -> {650, 475}, PlotRange -> {{-50, 50}, {-50, 50}, {-50, 50}}, ViewAngle -> Pi / 15, Lighting -> "Neutral"
] // Dynamic
(*creation des boutons de commande du robot*)
Boutons[matriceDH]

Out[69]= 
$$\begin{pmatrix} d \\ \theta \\ r \\ \alpha \end{pmatrix} = \begin{pmatrix} 20 q_1 & 20 q_2 & 0 \\ 0 & \frac{\pi}{4} & 2 \pi q_3 \\ 20 & 20 & 20 \\ 90^\circ & 0 & 0 \end{pmatrix}$$


Out[70]= 
$$\begin{pmatrix} -0.679953 & -0.733255 & 0. & 20.5431 \\ 0. & 0. & -1. & -9.04 \\ 0.733255 & -0.679953 & 0. & 37.4472 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$


Out[74]= 

Out[75]= {{{  , 0.432, q1},
  {  , 0.452, q2}, {  , 0.244, q3}}}
```

Pour les 3 exemples suivant, les résultats sont les même que ceux présentés précédemment mais le code pour les générer a été simplifié. Je n'ai donc pas recopié les résultats.

```
(*Generation simple de robot seul*)
Clear["q*"]
matriceDH = {{q1, q2, 0} * 20, {0, 1/8, q3} * 2 * Pi, {20, 20, 20}, {90°, 0, 0}};
CreateRobotTotal[matriceDH, 50, -50, 50, -50, 50, -50]
```



```
(*Generation simple de robot avec les repères liés à chacune de ses liaisons*)
Clear["q*"]
Mtest = {{q1, q2, 0} * 20, {0, 1/8, q3} * 2 * Pi, {20, 20, 20}, {90°, 0, 0}};
CreateRobotReperes[Mtest, 100, -100, 100, -100, 100, -100, 20]
```

L'exemple suivant montre comment générer un robot avec 2 arborescences série.

```
(*Exemple de génération de robot à 2 arborescences*)
Clear["q*"]

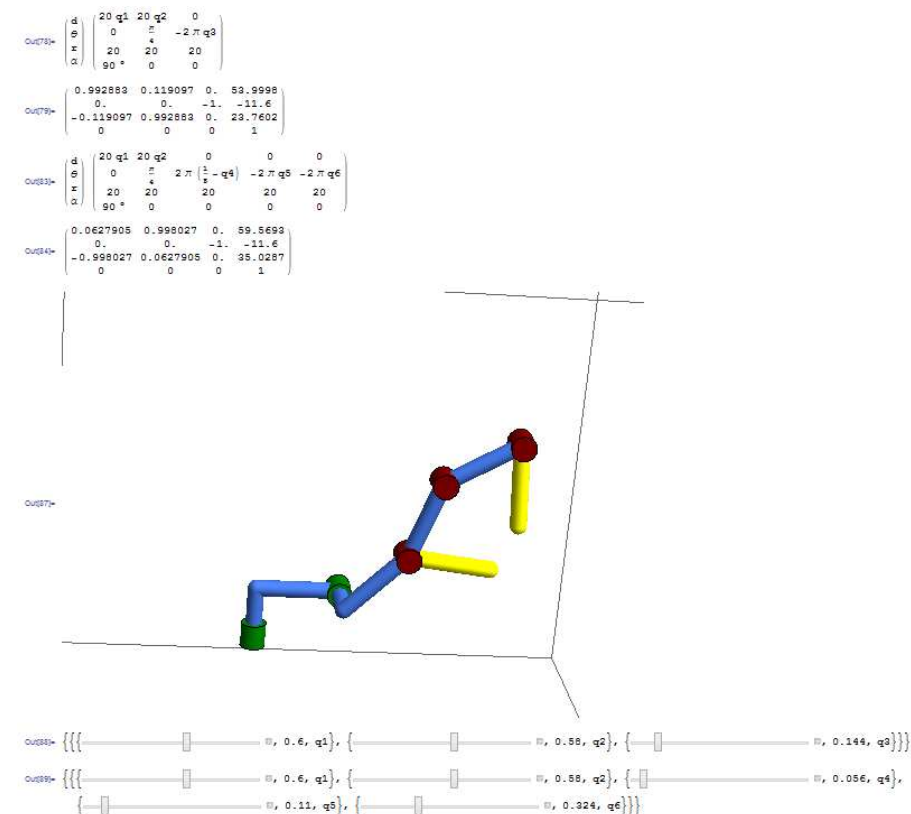
(*generation de la première arborescence*)
matriceDH1 = {{q1, q2, 0} * 20, {0, 1/8, -q3} * 2 * Pi, {20, 20, 20}, {90°, 0, 0}};
MatrixForm[{{d}, {θ}, {v}, {α}}] MatrixForm[matriceDH1]
MatrixForm[MatriceHomo[matriceDH1, 0, Length[Transpose[matriceDH1]]]] // Dynamic
matriceDHLiaisons1 = CreateLiaisons[matriceDH1];
robot1 := CreateRobot[matriceDHLiaisons1];

(*generation de la deuxième arborescence*)
matriceDH2 = {{q1, q2, 0, 0, 0} * 20, {0, 1/8, -q4 + 1/8, -q5, -q6} * 2 * Pi, {20, 20, 20, 20, 20}, {90°, 0, 0, 0, 0}};
MatrixForm[{{d}, {θ}, {v}, {α}}] MatrixForm[matriceDH2]
MatrixForm[MatriceHomo[matriceDH2, 0, Length[Transpose[matriceDH2]]]] // Dynamic
matriceDHLiaisons2 = CreateLiaisons[matriceDH2];
robot2 := CreateRobot[matriceDHLiaisons2];

(*affichage du robot complet*)
Show[{robot1, robot2},
  ImageSize → {650, 475}, PlotRange → {{-50, 100}, {-50, 100}, {-50, 100}}, ViewAngle → Pi/15, Lighting → "Neutral"
] // Dynamic

(*creation des boutons de commande de la première arborescence*)
Boutons[matriceDH1]

(*creation des boutons de commande de la deuxième arborescence*)
Boutons[matriceDH2]
```



Résumé

Ce projet a consisté en la création d'un programme sous Mathematica qui permet de générer automatiquement le squelette d'un robot et le commander à partir de son tableau de Denavit-Hartenberg.

Abstract

This project consisted in creating a program with Mathematica to automatically generate the structure of a robot and to command it from its Denavit-Hartenberg table.



Wolfram *Mathematica* 9
Seamlessly Flow Ideas to Results:
Compute, Develop, Deploy the *Mathematica* Way

