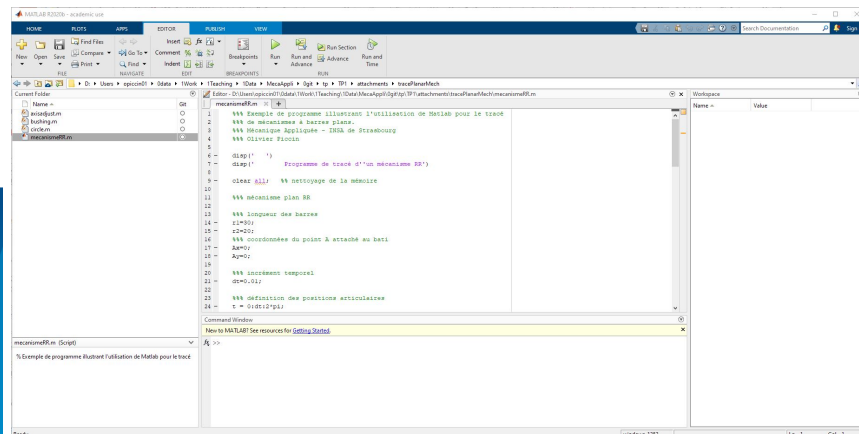


# TP1 – Mécanique Appliquée

## Étude d'un mécanisme plan avec MATLAB

L'objectif de ce TP est de proposer une introduction aux fonctionnalités de base du logiciel MATLAB ainsi qu'à son environnement de programmation. Ces connaissances seront nécessaires pour la suite des travaux de ce cours.

Ce TP comporte quatre parties. La première exploite uniquement le mode «ligne de commande» de MATLAB. La seconde et la suivante font appel à la manipulation de fichiers scripts, l'édition et la mise au point de fonctions MATLAB. La dernière partie consiste à programmer dans MATLAB la résolution d'un mécanisme plan à 6-barres.



## 1 Utilisation de MATLAB

### 1.1 Interpréteur de commande

Après avoir lancé MATLAB, se placer dans un répertoire de travail sur `C:\temp` et examiner le résultat de la saisie des commandes suivantes :

```
help help
help more
more
help inv
help arith
help format
format compact
help helpdesk
help diary
```


Il est souvent utile de conserver une trace des différentes commandes émises depuis ligne de commande. La commande `diary` permet d'enregistrer l'ensemble des actions qui sont réalisées sur la ligne de commande. Par exemple, la commande `diary sortie0.txt` va placer toutes les commandes et leur résultat dans le fichier `sortie0.txt`. L'enregistrement est arrêté par la commande `diary off`.

Le fichier `sortie0.txt` peut ensuite être édité puis commenté pour expliquer ce que **MATLAB** a effectué avec chaque commande.

---

```
A = [3 1 0 7; 4 -2 1 8; -3 1 6 -1; 2 -5 1 4]
b = [3 1 4 -3];
x = A\b
b = b'
x = A\b
r = b - A*x
Ainv = inv(A)
temp1 = Ainv*A;
temp2 = A*Ainv;
temp1 == temp2
AL1 = A(1,:)
AC1 = A(:,1)
AC2 = A(:,2)
AA2 = A(1:3, 1:3)
AA3 = A(2:4, 2:4)
A2 = 2.0*A
A3 = 2.*A
A4 = A^2
A5 = A.^2
Acar = A*A
A6 = A.*A
sum(A)
max(A)
max(max(A))
H = hilb(4)
Ident = eye(5)
Null = zeros(4)
c1 = 1:2:5
c2 = 2:6
c = [c1 c2]
x2 = A\c
x3 = A(:,1) + A(:,2) + A(:,3) + ... % Placer ceci sur une ligne
A(:,4) % Placer ceci sur la ligne suivante
what
whos
cd
which hilb
diary off
```

---

Sur le fichier journal obtenu, vous pourrez ajouter des commentaires pour mieux comprendre les réponses de **MATLAB** aux diverses commandes. **MATLAB** est également un outil permettant de tracer des graphes ou de visualiser des données comme illustré avec le code ci-dessous : Rechercher les moyens de placer sur ces tracés des étiquettes, une légende, un titre (voir les aides sur la fonction `plot` et la fenêtre de tracé). **MATLAB** dispose d'un système d'aide efficace permettant de trouver/retrouver des commandes ou des exemples d'utilisation de commandes. Il suffit de cliquer sur  pour lancer l'interface graphique (figure 1) et formuler une recherche.

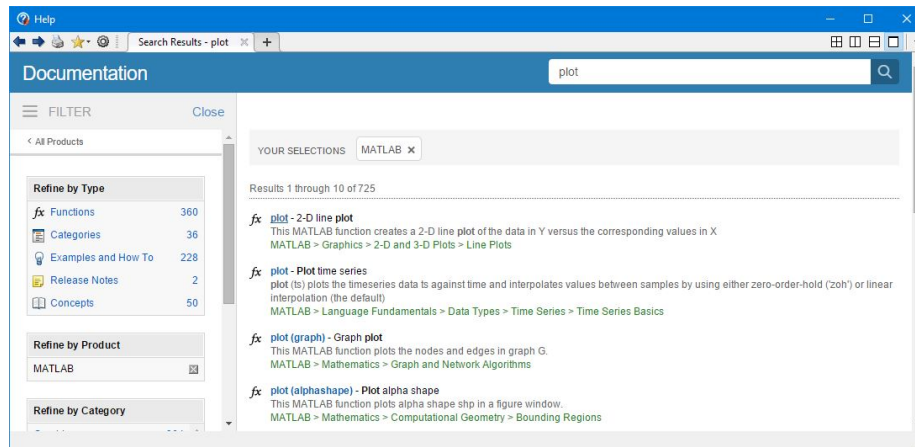


FIGURE 1 – Aide dans MATLAB.

## 1.2 Fichiers \*.m dans MATLAB

Il existe deux variétés de fichiers contenant des instructions **MATLAB** ; leur nom comporte une extension **.m**. Le premier type de fichier, appelé **script**, correspond à un ensemble de commandes **MATLAB** qui seront exécutées en séquence. Ce type de fichier peut servir à automatiser des calculs. On peut, à l'aide d'un éditeur de texte, modifier des paramètres avant chaque nouvelle exécution des calculs. Une autre possibilité est de faire demander dans le script les nouveaux paramètres à chaque nouvelle exécution des calculs.

Le second type de fichier, appelé **fonction**, contient la définition d'une fonction qui prend en entrée des arguments fournis par l'utilisateur (ou une autre fonction) et retourne en sortie des valeurs calculées. Les deux sortes de fichiers sont exécutés en saisissant leur nom (sans l'extension **.m**) à l'invite de commande ou bien en utilisant le nom du fichier dans un autre script ou fonction. Construire la fonction suivante et la tester pour  $n = 100$  et  $n = 1000$ .

---

```
function fractal(n)
x=[1 3 2 1];
y=[1 1 1+sqrt(3) 1];
xp=1; yp=1;
hold on;
axis('equal');
axis('off');
plot(x,y);
hold on;
tic;
for i=1:n
    de=fix(round(rand*6+0.5));
    if (de == 1 | de == 2)
        xp=(xp+x(1))/2; yp=(yp+y(1))/2;
    elseif (de == 3 | de == 4)
        xp=(xp+x(2))/2; yp=(yp+y(2))/2;
    elseif (de == 5 | de == 6)
        xp=(xp+x(3))/2; yp=(yp+y(3))/2;
    end
    plot(xp,yp,'r. ');
end
toc;
hold off;
```

---

Ajouter des commentaires pour expliquer ce que réalise la fonction. Il faut remarquer que les commentaires insérés juste après la 1ère ligne apparaîtront lorsque la commande `help fractal` est lancée.

## 2 Création de fonctions – Débogage dans MATLAB

Écrire une fonction MATLAB nommée `rot2quat` qui transforme une description de rotation donnée sous la forme  $\text{rot} = (\vec{a}, \theta)$  en un quaternion  $\text{quat} = (\lambda_0, \lambda_1, \lambda_2, \lambda_3)$ . Le vecteur  $\vec{a}$  ne sera pas nécessairement unitaire et l'angle  $\theta$  sera donné en degrés. Écrire ensuite une fonction `quat2mr` qui transforme un quaternion en matrice de rotation. Tester enfin à l'aide d'un script les deux fonctions précédentes sur les rotations suivantes :

$$\begin{aligned}\text{rot}_1 &= \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}^T, 90 \\ \text{rot}_2 &= \begin{bmatrix} 0 & 1 & 0 \end{bmatrix}^T, 60 \\ \text{rot}_3 &= \begin{bmatrix} 1 & 0 & 1 \end{bmatrix}^T, 20 \\ \text{rot}_4 &= \begin{bmatrix} 1 & 1 & 1 \end{bmatrix}^T, -30 \\ \text{rot}_5 &= \begin{bmatrix} 3 & 1 & 0 \end{bmatrix}^T, 20\end{aligned}$$

Les propriétés des matrices obtenues seront vérifiées (orthogonalité et valeur du déterminant). Durant la mise au point des fonctions précédentes, utiliser si nécessaire les outils de débogage via l'interface graphique (figure 2) : mise en place de points d'arrêt, visualisation des valeurs de variables en cours d'exécution, etc.

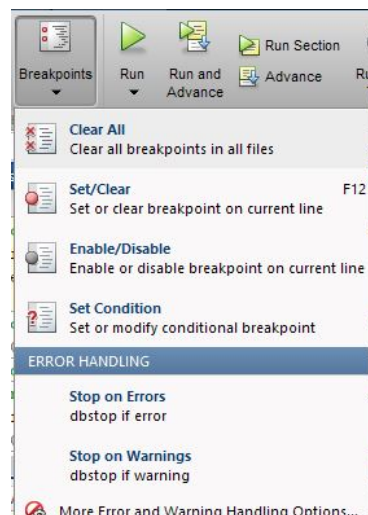


FIGURE 2 – Outils de débogage.

### 3 Programmation dans MATLAB d'un mécanisme plan

On considère le mécanisme représenté sur la figure 3. Ce mécanisme peut servir de base à la réalisation pratique d'une patte de robot marcheur. Il ne comporte que des liaisons de révolution. La pièce d'entrée **Crank** est mise en rotation à une vitesse constante de  $\omega = 1$  tr/s ce qui entraîne la pièce **Link 5** dans un mouvement particulier. Le but de l'étude est de construire le modèle permettant d'évaluer **la loi de mouvement** de ce mécanisme de jambe en observant particulièrement l'extrémité  $E$  de la pièce 5. La modélisation sera menée dans le but de programmer la fonction de résolution numérique de ce mécanisme par la méthode de Newton-Raphson.

#### 3.1 Paramétrage et description de la géométrie du mécanisme

$\overrightarrow{AB} = l_{0a} \vec{x}_{0a}$	$\overrightarrow{AC} = l_{0b} \vec{x}_{0b}$	$\overrightarrow{AD} = l_1 \vec{x}_1$	$\overrightarrow{GD} = l_{2a} \vec{x}_{2a}$	$\overrightarrow{GF} = l_{2b} \vec{x}_{2b}$
$\overrightarrow{CG} = l_3 \vec{x}_3$	$\overrightarrow{BH} = l_4 \vec{x}_4$	$\overrightarrow{HF} = l_{5a} \vec{x}_5$	$\overrightarrow{FE} = l_{5b} \vec{x}_5$	$\overrightarrow{HE} = l_5 \vec{x}_5$
$\theta_1 = (\vec{x}_0, \vec{x}_1)$	$\theta_2 = (\vec{x}_0, \vec{x}_{2a})$	$\theta_3 = (\vec{x}_0, \vec{x}_3)$	$\theta_4 = (\vec{x}_0, \vec{x}_4)$	$\theta_5 = (\vec{x}_0, \vec{x}_5)$
$x_E$	$y_E$	$\alpha_{0a} = (\vec{x}_0, \vec{x}_{0a})$	$\alpha_{0b} = (\vec{x}_0, \vec{x}_{0b})$	$\alpha_0 = \alpha_{0a} + \alpha_{0b}$
$\alpha_2 = (\vec{x}_{2a}, \vec{x}_{2b})$				

**NB** : les valeurs numériques de longueurs de la figure 3 sont données en mm.

#### 3.2 Calculs formels préparatoires

La description géométrique du mécanisme est faite dans le repère fixe  $(A, \vec{x}_0, \vec{y}_0)$ .


1. Écrire les équations de contrainte du mécanisme sous la forme  $F(X) = 0$ . Préciser les composantes du vecteur inconnu  $X$ .
2. Calculer la jacobienne de  $F$  ainsi que les configurations de singularité.
3. Calculer le modèle cinématique direct du mécanisme. On exprimera les relations donnant les vitesses  $\dot{x}_E$  et  $\dot{y}_E$  en fonction de la vitesse d'entrée  $\dot{\theta}_1$ .

#### 3.3 Programmation dans MATLAB

Tout d'abord, il s'agit d'appliquer la méthode de Newton-Raphson vue en cours pour la résolution du mécanisme, c'est à dire résoudre numériquement l'équation  $F(X) = 0$ . Ensuite, le résultat précédent sera utilisé pour simuler et visualiser le mécanisme lorsque la barre d'entrée tourne à une vitesse constante. Les principales étapes du travail sont les suivantes :

1. Partant d'une configuration initiale dans laquelle le mécanisme est **démonté**, écrire le codage de l'algorithme précédent qui va calculer la position du mécanisme **assemblé**. La configuration initiale choisie sera estimée à partir de la figure 3.
2. Visualiser la solution calculée.
3. Observer et commenter l'influence des valeurs initiales données sur la convergence/non-convergence, la vitesse de convergence.
4. Inclure le programme de résolution précédent dans une boucle de simulation en faisant varier le temps  $t$  de 0 à  $T_f = 5$  s par incrément  $dt = 0.05$ .
5. Faire varier l'incrément de temps  $dt$  et observer l'effet produit.
6. Tracer la trajectoire de  $E$  dans le plan  $(\vec{x}_0, \vec{y}_0)$  ainsi que les courbes d'évolution de la vitesse de  $E$  en projection suivant  $\vec{x}_0$  et  $\vec{y}_0$  en fonction de l'angle d'entrée  $\theta_1$ .
7. Tracer l'évolution de l'angle  $\theta_5$  et de la vitesse angulaire  $\dot{\theta}_5$  de la barre de sortie  $HE$  en fonction de l'angle de la barre d'entrée (courbes à tracer en deg et deg/s).

### Fichiers de base pour la programmation

L'archive 7-zip fournie dans le fichier attaché ici  donne des éléments de base pour la programmation du mécanisme dans MATLAB. Le fichier `script.m` constitue le script principal appelant d'autres fonctions pour la résolution et le tracé du mécanisme. Certains des fichiers fournis restent bien sûr à compléter.

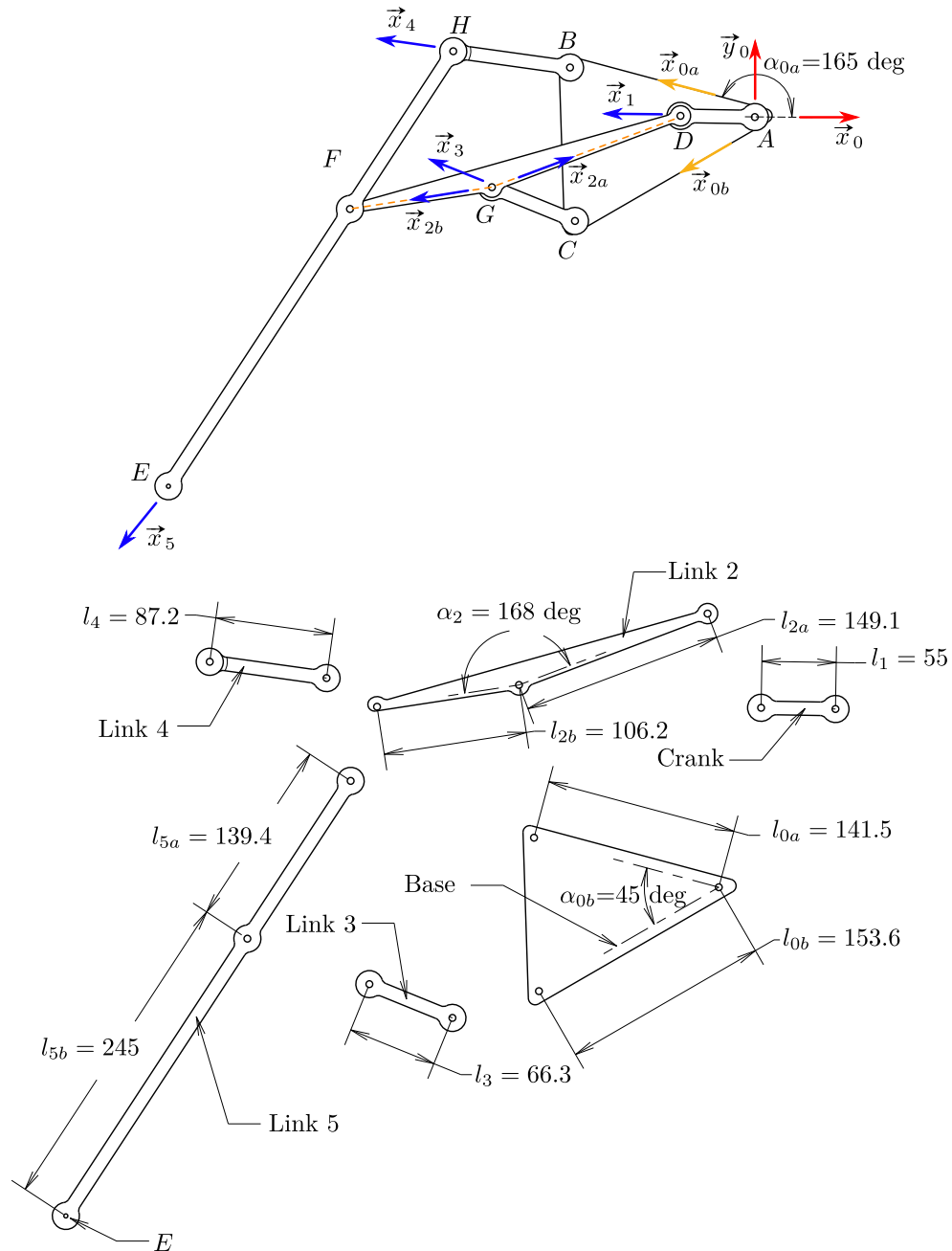


FIGURE 3 – Mécanisme pour la locomotion à pattes.