Initiation à Matlab

Ce document présente une découverte de Matlab *par l'exemple. Cette approche à été utilisée dans le cadre du cours d'Automatique des étudiants de formation continue à l'* INSA de Toulouse, *dont provient le document initial. Elle est aujourd'hui utilisée pour l'initiation à* Matlab *des étudiants de formation par alternance de l'*Ecole Nationale de Physique de Strasbourg.

1 Principes de Matlab

Matlab est l'outil de référence pour la simulation numérique, notamment en ce qui concerne l'Automatique[1]. Il offre des possibilités avancées que ce soit en matière d'identification ou de commande. Il permet, de manière plus générale, de résoudre une grande diversité de problèmes de simulation, dans des domaines aussi variés que le traitement du signal, les statistiques ou la vision, pour ne citer que quelques exemples. L'apprentissage de Matlab se fera en s'appuyant sur l'étude d'un moteur à courant continu.

1.1 Généralités

Avec Matlab les calculs sont numériques (une variable doit avoir une valeur) et basés sur la manipulation de scalaires, de vecteurs et de matrices.

Définir un scalaire Pour définir le réel $r = 2 * \pi$:

```
>>r=2*pi
```

Définir un vecteur Pour définir le vecteur $x = \begin{pmatrix} 1 \\ -1 \end{pmatrix} = (1 \quad -1)^T$:

`>>x=[1 ;-1]` ou `>>x=[1 -1]`

Définir une matrice Pour définir la matrice $A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix}$:

`>>A=[1 2 3; 4 5 6]` ou `>>A=[1,2,3;4,5,6]`

Opérations sur les matrices (ainsi que sur les scalaires et les vecteurs, le cas échéant)
- addition : `A+B`
- soustraction : `A-B`
- multiplication : `A*B` et `B*A`
- inversion : `inv(A)`
- transposition : `transpose(A)` ou `A'`

[1] On parle d'outil de CACSD (✍Computer Aided Control Systems Design).

Etude des matrices

- valeurs propres : `eig(A)`
- rang : `rank(A)`
- trace : `trace(A)`
- déterminant : `det(A)`

Quelques matrices utiles

- matrice identité de dimension n : `eye(n)`
- matrice de zéros de dimension $m \times n$: `zeros(m,n)`
- matrice de uns de dimension $m \times n$: `ones(m,n)`

De très nombreux algorithmes de calcul sont par ailleurs disponibles pour résoudre la plupart des problèmes courants (ou non ...) en mathématiques. Parmi ceux-ci, on peut citer les fonctions sur les polynômes (qui sont définis comme des vecteurs lignes composés des coefficients du polynôme entrés par ordre décroissant du degré). Par exemple, si on cherche les racines (complexes) de $x^2 - x + 1 = 0$:

```
>>roots([1 -1 1])
>>ans=
 0.5000 + 0.8660i
 0.5000 - 0.8660i
```

1.2 Aide en ligne

La bonne utilisation de l'aide en ligne est fondamentale pour travailler correctement avec Matlab.

Si l'on souhaite obtenir de l'aide sur certaines fonctions dont on connaît le nom, on utilise la fonction `help`. Par exemple :

```
>> help conv

 CONV Convolution and polynomial multiplication.
 C = CONV(A, B) convolves vectors A and B. The resulting
 vector is length LENGTH(A)+LENGTH(B)-1.
 If A and B are vectors of polynomial coefficients, convolving
 them is equivalent to multiplying the two polynomials.

 See also XCORR, DECONV, CONV2, FILTER, and CONVMTX in the Signal
 Processing Toolbox.
```

Si l'on cherche les noms des fonctions se rapportant à un sujet précis, on utilise la fonction `lookfor` (puis le mot en anglais). Par exemple :

```
>>lookfor polynom
>>POLYEIG Polynomial eigenvalue problem.
>>CONV Convolution and polynomial multiplication.
>>DECONV Deconvolution and polynomial division.
```

....

2 Utilisation de la Control Toolbox

2.1 Généralités

La boîte à outils dédiée à la commande (Control Toolbox) permet de disposer de nombreux outils d'analyse pour l'automatique.

Définition du système par sa fonction de transfert Soit le système décrit par :

$$G(s) = \frac{2s+1}{s^2+2s+1} = 2\frac{s+1/2}{(s+1)^2},$$

où s désigne la variable de Laplace. A l'aide de Matlab, on peut définir alternativement :

`>>F=tf([2 1],[1 2 1])` (numérateur et dénominateur de la fonction de transfert)
ou

`>>F=zpk([-1/2],[-1 -1],2)` (zéros, pôles et facteur de gain de la fonction de transfert)

Pour constituer un système à l'aide de différents sous-systèmes on peut effectuer différentes opérations. Soit `G1` et `G2` les représentations des deux systèmes. Les combinaisons de base sont :

`>>G1*G2`	ou	`>>series(G1,G2)`	`G1` en série avec `G2`
`>>G1+G2`	ou	`>>parallel(G1,G2)`	`G1` en parallèle avec `G2`
`>>feedback(G1,G2)`			`G1` bouclé par `G2`

On peut obtenir diverses informations sur le système défini par sa représentation G :

`>>pole(G)`	donne les pôles du système
`>>step(G)`	trace la réponse indicielle
`>>impulse(G)`	trace la réponse impulsionnelle
`>>bode(G)`	trace le diagramme de Bode
`>>nyquist(G)`	trace le diagramme de Nyquist
`>>nichols(G)`	trace le diagramme de Black-Nichols
`>>rlocus(G)`	trace le lieu d'Evans
`>>rlocfind(G)`	donne les valeurs des pôles et du gain correspondant sur le lieu d'Evans
`>>damp(G)`	donne les pôles, ainsi que la pulsation propre et l'amortissement associés à chaque pôle
`>>pzmap(G)`	place les pôles et les zéros dans le plan complexe

2.2 Prise en main de Matlab et de la Control Toolbox

On rappelle ici la modélisation du moteur à courant continu (MCC) dont l'étude va permettre d'illustrer les concepts fondamentaux de la Control Toolbox de Matlab. La fonction de transfert reliant la vitesse de rotation du rotor à la tension appliquée à l'induit s'écrit :

$$G(s) = \frac{\Omega(s)}{U(s)} = \frac{K}{(1+\tau_{el}s)(1+\tau_{em}s)},$$

avec :

$$\begin{aligned} K &= \frac{K_{em}}{Rf + K_{em}^2} \text{ gain statique du système,} \\ \tau_{em} &= \frac{RJ}{Rf + K_{em}^2} \text{ constante de temps électromécanique,} \\ \text{et } \tau_{el} &= \frac{L}{R} \text{ constante de temps électrique.} \end{aligned}$$

On rappelle que dans ce modèle R représente la résistance de l'induit du moteur, L son inductance ; f est le coefficient de frottement visqueux et J le moment d'inertie du rotor ; K est le rapport couple-courant (supposé égal au rapport force électromotrice-vitesse de rotation).

Pour la mise au point d'un programme ou des calculs très ponctuels, vous pouvez taper vos instructions sur la ligne de commande. Néanmoins :

> A RETENIR – **Dès que l'on a une séquence d'instructions a exécuter, on a tout intérêt à les regrouper sous forme d'un fichier script (fichier *.m).**

Si un fichier a l'extension `.m` (par exemple `nomFichier.m`), alors il sera exécuté en tapant son nom (`>>nomFichier`) sur la ligne de commande.

1. Eteindre sa calculatrice et l'enfouir dans son sac pour le reste du TP (et des autres TPs d'ailleurs). En cas de non respect de cette consigne s'attendre à de graves représailles…
2. Créer un script qui comporte les différentes opérations détaillées ci-dessous Pour cela on peut utiliser l'éditeur de Matlab (`>>edit`). Des commentaires peuvent être introduits à l'aide du symbole %.
3. Définir tout d'abord les diverses constantes du problème (dans un script nommé calcul_constantes_mmc.m par exemple). Les valeurs numériques choisies correspondent à un MCC un Maxon F 2260, numéro 885 étudié en cours :

$$\begin{aligned} R &= 1,44\,\Omega \\ L &= 5,6\ 10^{-4}\ H \\ J &= 1,29\ 10^{-4}\ kg.m^2 \\ f &= 7,2\ 10^{-5}\ m.N.s \\ K_{em} &= 0,10\ m.N.A^{-1} \end{aligned}$$

4. Calculer le gain K et les constantes de temps électrique τ_{el} et électromécanique τ_{em} du MCC. Définir alors sa fonction de transfert. Pour cela, on note `Num` et `Den` le numérateur et le dénominateur de la fonction de transfert.
5. Par la fonction appropriée, calculer les pôles de cette fonction de transfert. Vérifier que ces pôles valent $-\tau_{el}^{-1}$ et $-\tau_{em}^{-1}$.

6. Créer une figure (avec la fonction `figure`) et la diviser en deux sous-tracés (avec la fonction `subplot`). Dans le premier, tracer la réponse indicielle du MCC à un échelon unitaire de tension. A l'aide de la souris, observer les caractéristiques accessibles du tracé (clic droit puis relâcher pour les caractéristiques, pointer la courbe et clic gauche puis rester appuyé pour les valeurs). Dans la seconde sous-figure, tracer le diagramme de Bode du MCC. Analyser les différents tracés.
7. L'asservissement de vitesse du moteur à courant continu est défini à la figure 1. Il comporte un correcteur proportionnel de gain $K_p = 10$. La fonction de transfert du capteur de vitesse est assimilée à un gain pur noté K_ω. La sortie de ce capteur valant $10\ V$ pour une vitesse de rotation de $3000\ tours/min$ calculer K_ω en unités SI.

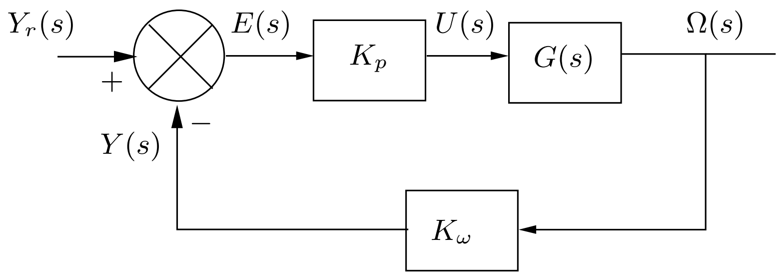


FIG. 1 – Schéma de l'asservissement de vitesse du MCC

8. Définir ensuite les fonctions de transfert du système en boucle ouverte et en boucle fermée en combinant les fonctions de transfert des différents blocs.
9. Sur une même figure, tracer les réponses indicielles du système en boucle ouverte et en boucle fermée.
10. Tracer les diagrammes de Bode, Black et Nyquist du système en boucle ouverte sur trois figures différentes (avec toujours $K_p = 10$). Identifier les marges de stabilité du système sur ces tracés.
11. En utilisant une boucle (`>>help for`), tracer sur un même graphe les réponses indicielles du système en boucle fermée pour les valeurs de K_p égales à $10,\ 100$ et 1000. Vérifier la cohérence de ces réponses avec les marges de stabilité relevées à la question précédente.

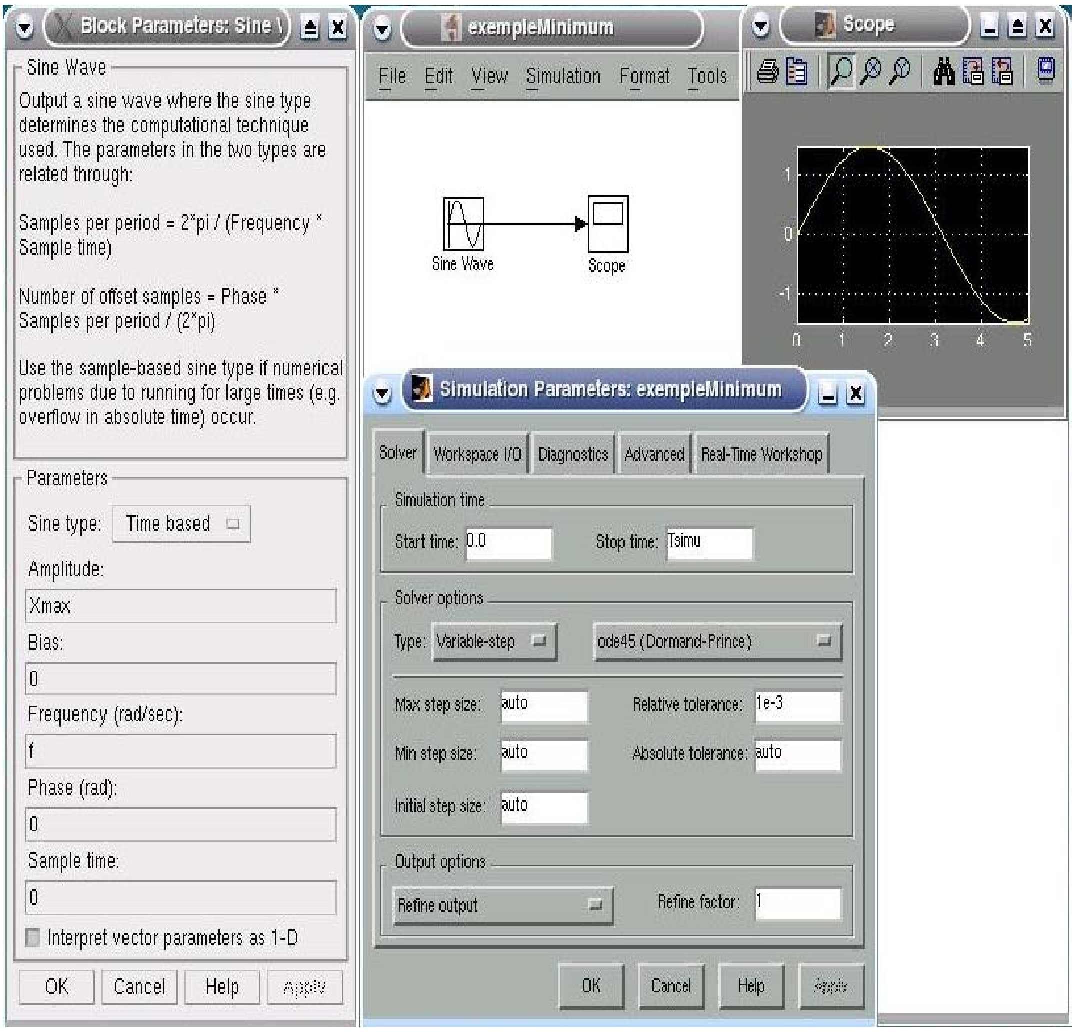
3 Utilisation de Simulink

3.1 Généralités

Simulink est une autre boîte à outils de Matlab qui permet de faire des simulations de systèmes définis à l'aide d'un outil graphique. On se propose ici d'utiliser Simulink pour définir l'asservissement en vitesse du moteur à courant continu. On pourra ainsi visualiser notamment les réponses du système à différents types d'entrées.

Pour lancer Simulink, on peut soit utiliser les menus disponibles, soit taper sur la ligne de commande `>>simulink`. Pour créer un nouveau modèle Simulink choisir `New` dans le menu `File`, puis `Model`. Une feuille de travail apparaît, sur laquelle on va pouvoir définir graphiquement notre système. Les différents outils disponibles seront trouvés dans les menus correspondants : sources, visualisation, automatique continue, automatique discrète, fonctions mathématiques, fonctions et tables, automatique non-linéaire, signaux et systèmes. De par sa nature graphique Simulink peut être aisément découvert intuitivement. Cet outil utilise la technique de *drag and drop* (sélectionner et faire glisser). Il est facile de positionner les éléments nécessaires dans la fenêtre du modèle. Ensuite, on relie ces éléments entre eux pour constituer le modèle. Chaque élément possède une description et éventuellement des paramètres qui peuvent être modifiés. Pour y accéder double-cliquer sur un élément.

Par exemple si on veut visualiser le signal d'un générateur sinusoïdal, on utilise la source correspondante (menu `Sources`) et un oscilloscope (menu `Sinks`). On connecte ensuite ces deux éléments en attrapant la sortie du générateur et amenant la souris enfoncée sur l'entrée de l'oscilloscope. La simulation est jouée en cliquant sur `Run`, dans le menu `Simulation`. La encore, on peut définir l'ensemble de la simulation à l'aide d'un script. En effet, Simulink partage les variables de l'espace de travail Matlab (variables globales). On peut ainsi définir le modèle Simulink à l'aide de variables dont les valeurs sont définies dans un script. On peut jouer la simulation depuis la ligne de commande (donc lancer cette simulation depuis un script). Ainsi sur l'exemple précédent, on obtient le modèle `exempleMinimum.mdl` et le script ci-après.



```
% Visualisation d'un signal sinusoïdal d'amplitude 1.5, de
% fréquence 1 Hz, sur un horizon de 5 s.
% le modèle simulé (voir ci-dessus) porte le nom exempleMinimum.mdl
Tsimu=5
Xmax=1.5
f=1
sim('exempleMinimum')
```

3.2 Prise en main de Matlab et de la Simulink

Selon la méthode décrite précédemment, on répondra aux questions suivantes, dont certaines reprennent largement l'étude effectuée pour la prise en main de la Control Toolbox. Néanmoins il est conseillé de créer un nouveau script.

1. A l'aide de Simulink créer le modèle de l'asservissement de vitesse vu dans la partie 2.2. On parcourra pour cela les menus de Simulink pour trouver les éléments nécessaires. En particulier le bloc fonction de transfert, nommé `Transfer Fcn` sera trouvé dans le menu `Continuous`. Les constantes, le numérateur et le dénominateur de la fonction de transfert seront définis dans le script qui pilotera les simulations, à l'image de l'exemple précédent.
2. Sur une même figure, tracer la consigne et la réponse indicielle du système en boucle ouverte. Le tracé sera fait sur un horizon de temps judicieusement choisi.

   Note : on peut envoyer à un oscilloscope autant de signaux que l'on veut. Par exemple, si l'on souhaite afficher deux signaux différents il faut utiliser un multiplexeur (`Mux` dans le menu `Signals and Systems`) pour les mettre sur une même ligne.

3. Définir le système en boucle fermée selon le schéma de la figure 1 avec $K_p = 100$. Sur une même figure, tracer la consigne et la réponse indicielle du système en boucle fermée.