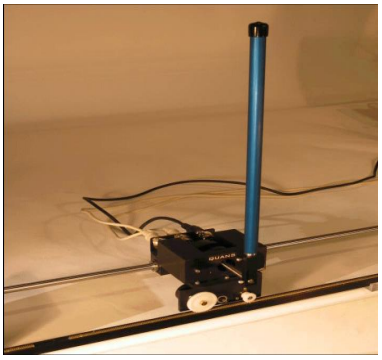
Université Larbi Ben M'hidi, Oum El Bouaghi

Faculté des Sciences et Sciences Appliquées

Département Génie Electrique





Polycopié De Travaux Pratiques

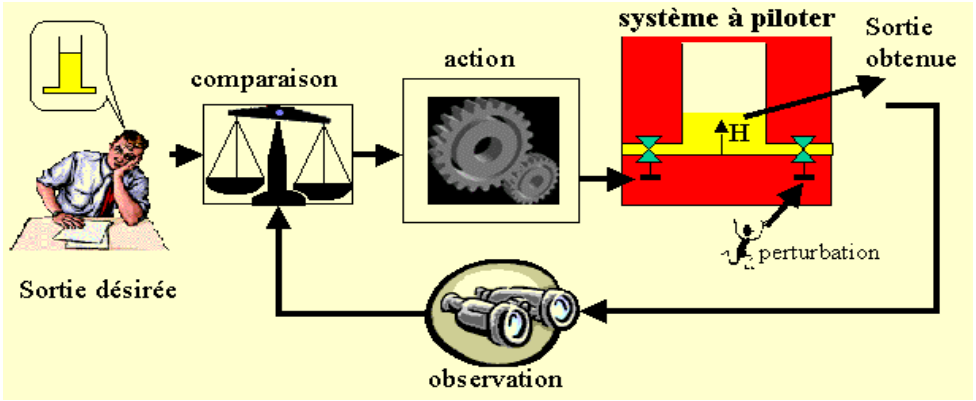
Commande dans l'espace d'état

Master -Informatique Industrielle-

1ère année

Réalisé par : MEGRI Abderrahim Fayçal

Maître de conférences B



Déclaration

Ce polycopié regroupe l'ensemble des travaux pratiques destinés aux étudiants en première année, master **Informatique Industrielle**.

De ce fait, ce polycopié ne sera utilisé qu'à des fins strictement pédagogiques et académiques.

Le travail a été réalisé, en se référant à plusieurs travaux dirigés et pratiques de différents auteurs, dont la majorité sont cités dans les références bibliographiques

Public Cible

Ce Travail est destiné aux étudiants en première année (S1) master **Informatique Industrielle**.

Présentation des Sujets

Ces travaux pratiques ont pour objectif de faire découvrir des méthodes de conception de commande de systèmes dans le domaine temporel, et cela au travers de représentations d'état des modèles de différents systèmes.

- Le TP 1 porte sur l'étude des systèmes électrique, mécanique et électromécanique dans l'espace d'état tout en tenant compte de la maîtrise des principales fonctionnalités du logiciel de calcul Matlab et sa Toolbox graphique Simulink.
- Le TP 2 porte sur la synthèse d'une loi de commande par retour d'état en utilisant la méthode de placement de pôles.
- Le TP 3 porte sur la synthèse d'une loi de commande par retour de sortie.
- Le TP 4 porte sur la synthèse d'une loi de commande par retour d'état avec observateur afin d'estimer les états du système.
- Le TP 5 porte sur la synthèse d'une loi de commande par retour d'état estimé dans le domaine discret.
- Le TP 6 porte sur la synthèse d'une loi de commande par retour d'état en utilisant la méthode dite LQR pour ''Linear Quadratic Regulator''.
- Le TP 7 porte sur la synthèse d'une loi de commande par retour d'état en présence de perturbations aléatoires en utilisant la méthode dite LQG pour ''Linear Quadratic Gaussien''.

Table des matières

TP N°1	Etude des systèmes dans l'espace d'état Sous Matlab et Simulink	Département de Génie Electrique

Objectifs Généraux :

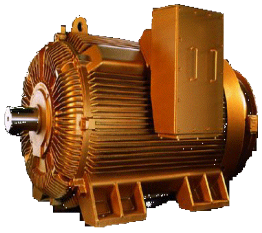
- Modélisation dans l'espace d'état
- Analyse d'un système d'état
- Utilisation du logiciel Matlab-Simulink

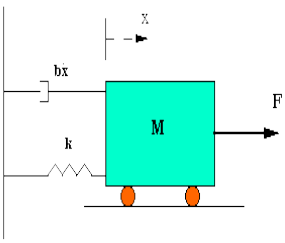
Pré-Requis :

- Cours d'asservissement
- Notions sur l'informatique

Equipement et Accessoires :

- Micro-ordinateur
- Logiciel Matlab
- Imprimante





TP N°1 : **Etude des systèmes dans**

L'espace d'état sous Matlab et Simulink

I.1. Introduction

Pendant longtemps, l'automaticien a utilisé exclusivement la notion de relation entrée-sortie. Or, pour les systèmes complexes, ce point de vue se révèle être une représentation incomplète des phénomènes.

La connaissance des variables internes d'un système peut présenter un intérêt certain en ce sens qu'elles permettent de déterminer l'état d'un système à un instant donné.

Ces variables d'état sont donc l'ensemble des variables nécessaires à la caractérisation de l'état d'un système. Elles sont regroupées dans un vecteur appelé vecteur d'état.

I.2. Rappels

Les équations d'état d'un système sont constituées de l'ensemble des équations matricielles suivantes :

$$\begin{cases} \dot{\underline{x}}(t) = A\underline{x}(t) + B\underline{u}(t) & \text{équation d'état} \\ \underline{y}(t) = C\underline{x}(t) + D\underline{u}(t) & \text{équation d'observation} \end{cases}$$

Avec :

$\underline{x}$: vecteur des variables d'état

$\underline{u}$: vecteur des commandes

$\underline{y}$: vecteur des sorties

A : matrice d'évolution de dimension $(n \times n)$(n :ordre du système)

B : matrice de commande de dimension $(n \times l)$(l :nombre d'entrées du système)

C : matrice d'observation de dimension $(m \times n)$ (m :nombre de sorties)

D : matrice de transmission directe de dimension $(m \times l)$

Ces équations sont établies pour les systèmes multivariables, dans le cas particulier des systèmes monovariable qui nous intéresse, on peut réécrire ces équations de la façon suivante :

$$\begin{cases} \dot{\underline{x}}(t) = A\underline{x}(t) + Bu(t) \\ y(t) = C\underline{x}(t) + Du(t) \end{cases}$$

Dans la plupart des systèmes rencontrés en automatique, il n'existe pas de lien direct entre l'entrée et la sortie du système. Par conséquent, dans la représentation d'état de tels systèmes, la matrice D de transmission directe est nulle.

Le choix des variables d'état conduit à des formes particulières pour les matrices, ces structures spécifiques sont appelées forme canonique, qu'on peut classer en trois formes [8] :

Forme canonique diagonale

La matrice d'état est diagonale et les éléments de la diagonale sont les valeurs propres.

Forme canonique commandable

Une ligne de la matrice d'état correspond aux coefficients du polynôme caractéristique de la fonction de transfert

Forme canonique observable

Une colonne de la matrice d'état correspond aux coefficients du polynôme caractéristique de la fonction de transfert

I.3. Présentation du logiciel

Matlab (MATrix LABoratory) est l'outil de référence pour la simulation numérique, notamment en ce qui concerne l'Automatique.

Matlab est une application Windows. En cliquant deux fois sur l'icône **Matlab**, s'ouvre alors la fenêtre principale. Cette fenêtre est divisée en plusieurs parties, comme le montre la Figure I.1 ci-dessous.

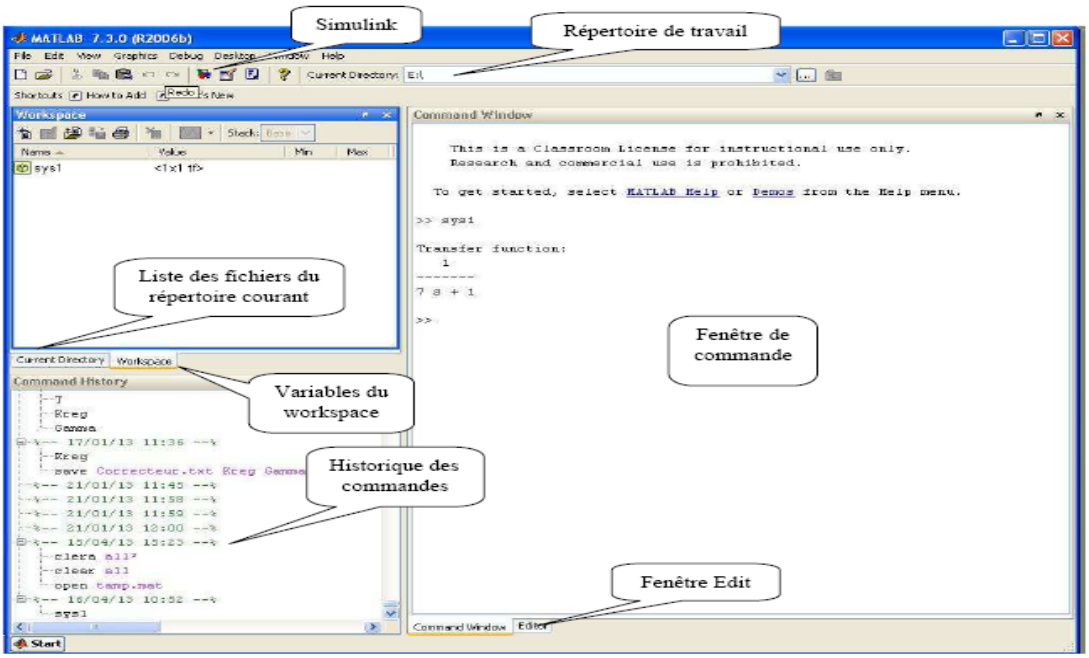


Figure I.1 : IHM (Interface Homme Machine) de **Matlab**

Programmation sous Matlab :

- Lancer Matlab sous windows, une fenêtre (command window) est alors affichée.
- Cliquer sur File et choisir "New" ensuite "Mfile", une nouvelle fenêtre est ouverte dans laquelle on peut écrire notre programme.
- Donner un nom au programme à enregistrer.
- Pour l'exécution du programme, taper son nom dans la fenêtre (Command window) ou cliquer sur l'icône **run**  dans la barre d'outils de Matlab.

Une boîte à outils de Matlab correspond à un ensemble de fonctions disponibles dans la fenêtre de travail que l'on peut appeler à l'invite ≫

De même, pour chaque commande particulière, vous avez accès à deux niveaux d'aide :

1. Un 1er niveau d'aide par : ≫ help 'nom de la commande'
2. Un 2eme niveau d'aide, avec navigation hypertexte par : ≫helpwin 'nom de la commande' .

   Exemple: ≫help **ss** ou ≫helpwin **ss**

Matlab possède une boîte à outils réservée à l'étude des systèmes de commande « control system Toolbox », ainsi qu'un environnement graphique de simulation des systèmes dynamiques **Simulink** sous forme de schémas en blocs.

Pour lancer Simulink, on clique sur l'icône  dans la barre d'outils de Matlab, ou bien par la commande ≫ simulink à l'invite, dans la fenêtre de travail de Matlab.

Ceci nous donnera accès à différents menus ou boîtes de Simulink ; ainsi on ouvrira une nouvelle fenêtre Simulink à partir de laquelle on pourra créer un nouveau fichier modèle et placer les éléments constitutifs du système à étudier.

La librairie de Simulink comprend de différentes sections permettant d'aborder de nombreux aspects de la commande des systèmes.

Dans notre polycopié on portera notre attention sur les éléments suivants comme le montre la Figure I.2 :

- **Step** : se trouve dans **Sources**, générateur de l'échelon.
- **Transfert fcn** : se trouve dans **Continuous**, permet de définir une fonction de transfert. Changez les paramètres pour qu'elle corresponde à la fonction de transfert demandée.
- **State-space :** se trouve dans **Continuous**, permet de définir une représentation d'état. changer les matrices A,B ,C,D pour qu'elle correspondre à la représentation d'état demandée.
- **Gain :** se trouve dans **Math Operations**, permet de multiplier l'entrée par un gain k(scalaire ou vecteur)
- **Mux** : se trouve dans **Signal routing**, permet de multiplexer plusieurs signaux dans un fil.
- **Sum** : se trouve dans **Math Operations**, permet de réaliser le comparateur. Il faut choisir les signes + et -.
- **Scope** : se trouve dans **Sinks**, c'est un scope rudimentaire pour avoir rapidement un tracé des courbes.
- **ToWorkspace**: il se trouve dans **Sinks**, permet de récupérer le résultat de la simulation dans une variable exploitable sur Matlab (ligne de commande).

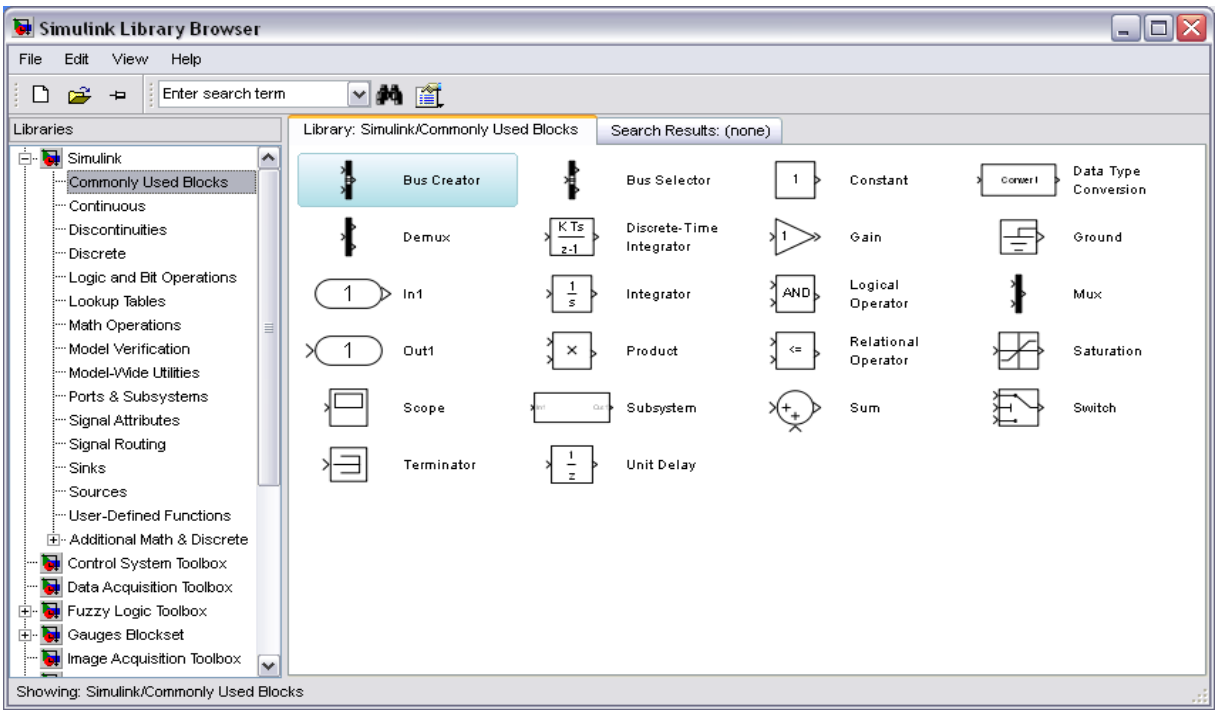


Figure I.2 : interface Simulink

I.4. Descriptions des systèmes

Dans toute la suite de ce TP, les systèmes considérés n'auront qu'une entrée et qu'une sortie.

I.4.1. Exemple 1 : système électrique

Considérons un circuit électrique alimenté par une tension $u(t)$, constitué des éléments suivants : résistance *R*, inductance *L* et capacité *C* en série (Figure I.3) ,en notant q la charge aux bornes du condensateur et *i* l'intensité du courant qui traverse le circuit.

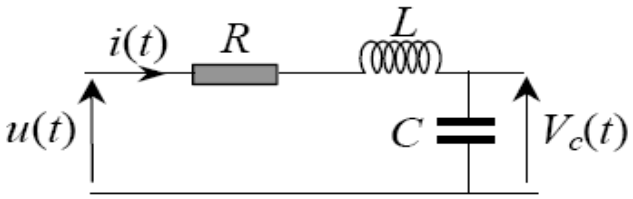


Figure I.3: Schéma d'un circuit RLC

La loi des mailles conduit à:

$$u(t) = Ri(t) + L\frac{di(t)}{dt} + V_c(t)$$

La formule suivante relie l'intensité du circuit à la charge et donc à la tension aux bornes du condensateur :

$$i(t) = C\frac{dV_c(t)}{dt} = \frac{dq(t)}{dt}$$

On arrive aisément à l'équation différentielle temporelle comportant une dérivée d'ordre 2

$$LC \cdot \frac{d^2V_c(t)}{dt^2} + RC \cdot \frac{dV_c(t)}{dt} + V_c(t) = u(t)$$

Données numériques du système :$R = 400\Omega, C = 100\mu F, L = 10H$

Pour la modélisation dans l'espace d'état du système, on précise que les variables $i(t)$ et $q(t)$ sont des variables d'états importantes et donc à prendre en compte.

D'où le choix suivant pour le vecteur d'état :

Le vecteur d'état :$\begin{cases} x_1(t) = i(t) \\ x_2(t) = q(t) \\ x_3(t) = V_c(t) \end{cases}$, l'entrée est $u(t)$, la sortie $y(t) = x_3(t)$

I.4.2. Exemple 2 : Système mécanique

Une masse m, retenue par un système ressort +amortisseur, se déplace sur un plan sous l'effet d'une force extérieure f (Figure I.4)

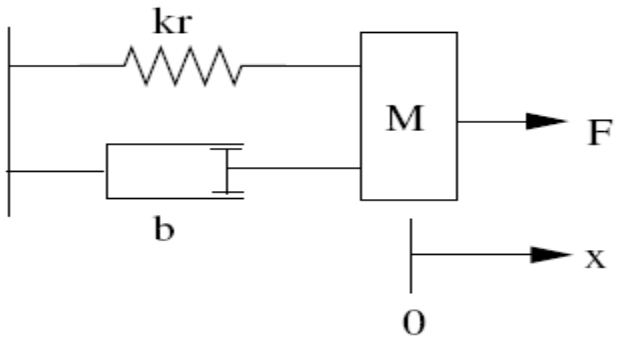


Figure I.4 : Schéma d'une masse en translation

La loi de Newton permet d'écrire l'équation du système :

$M \cdot \ddot{x}(t) + k_r \cdot x(t) + b \cdot \dot{x}(t) = F(t)$, équation du second ordre.

Données numériques du système :$k_r = 2, b = 8, M = 3$

On considérera comme état, le vecteur d'état suivant :$\underline{x}(t) = \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} = \begin{bmatrix} x(t) \\ \dot{x}(t) \end{bmatrix}$

L'entrée est la force $F(t)$ et la sortie est la position $x(t)$

I.4.3. Exemple 3 : Système électromécanique ''moteur à courant continu''

Un moteur à courant continu (MCC), (Figure I.5), est un dispositif électromécanique qui convertit une énergie électrique d'entrée en énergie mécanique.

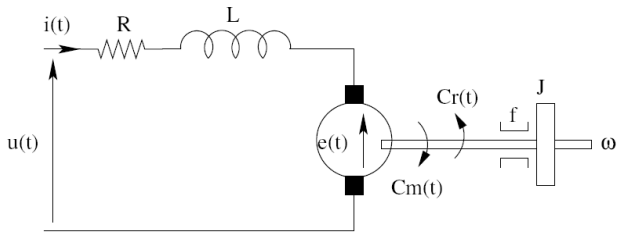


Figure I.5 : Schéma du moteur à courant continu

Pour la partie électrique, on calcule la tension aux bornes de l'induit.

L'équation électrique, liant la tension *u* aux bornes de l'induit et le courant d'induit *i* s'écrit :

$$u(t) = R \cdot i(t) + L\frac{di(t)}{dt} + e(t)$$

Où *R* est la résistance de l'induit du moteur, *L* son inductance et e la force électromotrice, qui est proportionnelle à la vitesse de rotation du rotor :

$$e(t) = k \cdot w(t)$$

Pour la partie mécanique, on applique le principe fondamental de la dynamique autour de l'axe de rotation. L'équation mécanique rendant compte des couples agissant sur le rotor s'écrit :

$$C_m(t) = J\frac{d\mathrm{w}}{dt} + f\mathrm{w}(t)$$

Où C_m est le couple moteur, f le coefficient de frottement visqueux et *J* le moment d'inertie du rotor. Par construction, le couple moteur est proportionnel au courant d'induit *i* :

$$C_m(t) = k \cdot i(t)$$

On arrive aisément à l'équation différentielle temporelle comportant une dérivée d'ordre deux

$$u(t) = \frac{J \cdot L}{k} \cdot \frac{d^2w(t)}{dt^2} + \frac{RJ + Lf}{k} \cdot \frac{dw(t)}{dt} + \frac{Rf}{k} \cdot w(t)$$

Données numériques du moteur à courant continu voir le tableau I.1

Coefficient de frottement visqueux	$f = 3 \cdot 10^{-4}$N/m/rad/s
Moment d'inertie	$J = 1.1 \cdot 10^{-3}$kg.m^2
Résistance de l'induit	$R = 4.8\Omega$
Inductance de l'induit	$L = 18.4mH$
Coefficient de fcém (c'est aussi constant du moteur en N.m/A)	$k = 0.45$ V/rad/s

Tableau I.1: Données numériques du moteur à Courant Continu

On considérera comme état :

- Courant d'induit $x_1(t) = i(t)$,
- Vitesse angulaire $x_2(t) = w(t)$,
- Entrée : tension d'alimentation $u(t)$,
- Sortie : position angulaire $y(t) = \theta(t)$.

I.5. Etude et Analyse de la représentation d'état des systèmes

❖ Préparation : (à remettre au début du TP)

- ✓ Donner la représentation d'état de chacun des trois systèmes présentés ci-dessus,
- ✓ Parmi les éléments $A, B, C, D, \underline{x}, \underline{u}, \underline{y}$, quels sont ceux qui varient en fonction de la représentation d'état choisie? Pourquoi les autres éléments restent-ils identiques quel que soit la représentation d'état?
- ✓ La représentation d'état d'un système est-elle unique ?
- ✓ Ecrire les équations générales permettant, de passer de la représentation d'état d'un système à la représentation externe (Fonction de transfert), en utilisant la transformation de Laplace,
- ✓ En prenant la transformée de Laplace de l'équation différentielle du système, donner sa fonction de transfert qui relie l'entrée avec la sortie. Quel est l'ordre du système ?

❖ Répondre à toutes les questions suivantes pour chaque système

- ✓ Créer un fichier de type M-file et en utilisant la fonction Matlab **ss**, construire la représentation d'état continue du système,
- ✓ Indiquer la dimension de chacune des matrices *A*, *B*, *C*, et *D* , en utilisant la fonction Matlab **size**,
- ✓ Calculer la fonction de transfert $H(p) = C.[pI - A]^{-1} \cdot B$ en utilisant les fonctions Matlab **syms, inv,eye**,
- ✓ Trouver la représentation d'état sous la forme canonique diagonale en utilisant la fonction Matlab **canon** avec l'option 'modal',
- ✓ Trouver la représentation d'état sous la forme canonique observable en utilisant la fonction Matlab **canon** avec l'option 'companion',
- ✓ Comment vérifier si deux modèles d'état sont équivalents du point de vue du comportement entrée-sortie ?
- ✓ En utilisant la fonction Matlab **ss2tf**, passer de la représentation d'état (représentation interne) du système à sa fonction de transfert (représentation externe),
- ✓ En utilisant la fonction Matlab **tf2ss**, passer de la fonction de transfert (représentation externe) du système à sa représentation d'état (représentation interne),
- ✓ Analyser la position des pôles et des zéros du système, et déterminer s'il est stable en boucle ouverte ou pas, tout en justifiant. Pour cela utiliser la fonction Matlab **pzmap**,
- ✓ Calculer les valeurs propres de la matrice d'état en utilisant la fonction Matlab **eig**,

Faire le point en répondant à cette question : Que peut-on dire sur les valeurs propres de A et les pôles du système ?

- ✓ Calculer le gain statique du système en utilisant la fonction Matlab **dcgain**,
- ✓ Analyser les performances temporelles du système (fonction de transfert et représentation d'état) en utilisant la fonction Matlab **step**, (En cliquant sur le tracé avec le bouton droit, il est possible d'obtenir des renseignements sur le temps de réponse, le temps de montée, le gain statique ainsi que le dépassement),
- ✓ Analyser les performances fréquentielles du système(fonction de transfert et représentation d'état) en utilisant la fonction Matlab **bode(** En cliquant sur le tracé avec le bouton droit, il est possible d'obtenir des renseignements sur le gain statique et les marges de gain et de phase)**,**
- ✓ Lancer Simulink, construire le modèle du système. Pour cela, on recherchera dans les menus Simulink les différents blocs nécessaires (fonction de transfert ou représentation d'état continue, gain, sommateur, oscilloscope pour visualiser…etc), que l'on fera glisser dans le modèle. Une fois les éléments essentiels présents, il suffit de les lier entre eux avec la souris c'est-à-dire tracer des connections entre la sortie du premier bloc et l'entrée du deuxième bloc. Ne pas oublier de sauvegarder le schéma dans votre répertoire de travail avant de l'exécuter,
- ✓ Simuler les réponses indicielle et impulsionnelle du système en boucle ouverte,
- ✓ Faire une étude comparative entre les deux résultats obtenus par le programme (M-file) et par le modèle Simulink,
- ✓ Interpréter les résultats,
- ✓ Tracez l'allure de la réponse indicielle unitaire en boucle fermée du système en utilisant les fonctions Matlab **step** et **feedback**,
- ✓ Simuler dans Simulink, la réponse indicielle du système en boucle fermée.

TP N°2	Commande par retour d'état Par placement de pôles	Département de Génie Electrique

Objectifs Généraux :

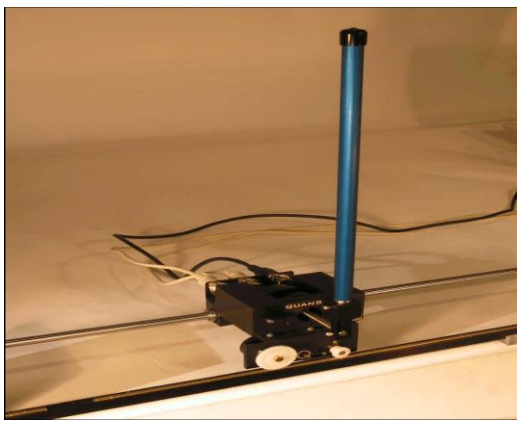
1. Modélisation et simulation d'un pendule inversé sous Matlab-Simulink.
1. Linéarisation au voisinage d'un point d'équilibre.
2. Synthèse d'une loi de commande par retour d'état

Pré-Requis :

- Modélisation, représentation d'état.
- Analyse d'un système d'état (stabilité, contrôlabilité)

Equipement et Accessoires :

- Micro-ordinateur
- Logiciel Matlab
- Imprimante



TP N°2 : **Commande par retour d'état**

Par placement de pôles

II.1. Introduction

Ce TP a pour objectif d'illustrer les concepts de commande par retour d'état appliqués aux systèmes linéaires monovariables. Les principes et propriétés d'une commande par retour d'état sont étudiés en simulation dans l'environnement logiciel Matlab/Simulink.

II.2 Rappels

II.2.1.La commande par retour d'état

La commande par retour d'état est une technique de contrôle qui consiste à calculer l'erreur entre la valeur de référence $r(t)$ et la valeur de l'état mesurée $x(t)$ multipliée par un gain de retour K et d'utiliser cette valeur en commande.

La figure II.1 montre le schéma bloc d'une commande par retour d'état.

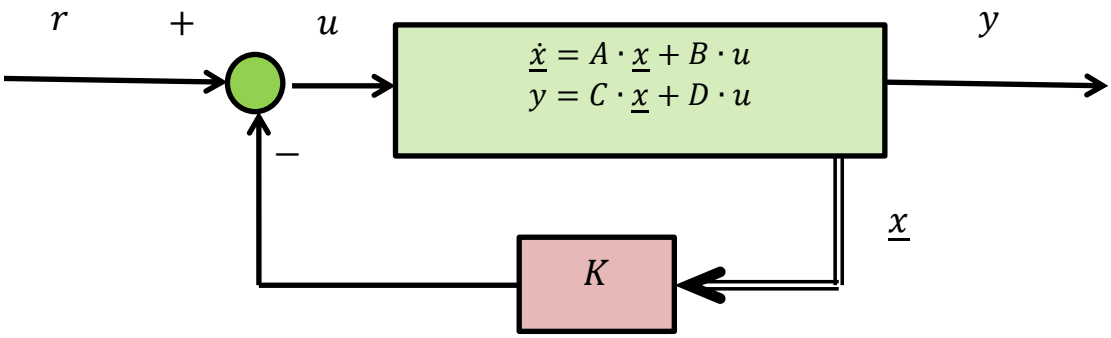


Figure II.1:schéma de la commande par retour d'état

La commande et l'équation d'état de ce système sont les suivantes :

$$u(t) = \left(r(t) - K \cdot \underline{x}(t)\right)$$

$$\dot{\underline{x}}(t) = (A - B \cdot K) \cdot \underline{x}(t) + B \cdot K \cdot r(t)$$

Il est possible de stabiliser le système en choisissant le gain de retour d'état K de manière à placer correctement les valeurs propres de $(A - B \cdot K)$

II.2.2 Obtention de la matrice de retour d'état K

Il y a deux méthodes principales pour obtenir le gain de retour d'état K.

- **Placement de pôles direct**

Cette méthode consiste à calculer le gain de retour d'état K de manière à placer les pôles (les valeurs propres) de la matrice $(A - B \cdot K)$ à des valeurs précise. Les valeurs choisies sont

évidemment stables, et sont souvent tirées des performances souhaitées sur le système corrigé. Il faut par contre veiller à ce que la valeur de *K* ne soit pas aberrante.

La fonction **acker** (ou **place)** de Matlab permet d'effectuer un placement de pôles direct.

- **La commande optimale linéaire quadratique LQR** sera étudiée dans le TP N°06.

II.3. Description du système

Le système auquel on va s'intéresser est le système mécanique représenté par la figure suivante (Figure II.1)[7] :

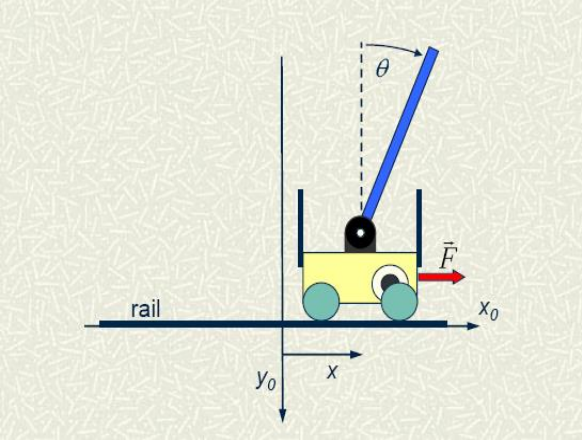


Figure II.2 : Schéma de l'ensemble pendule-chariot

Il s'agit d'un système composé d'un chariot et d'un pendule.

Le but de cette application consiste à asservir la position du chariot tout en maintenant le pendule inversé dans sa position verticale.

II.3.1. Modélisation du système

Montrer à l'aide des équations d'Euler – Lagrange (Annexe2) que l'ensemble chariot-pendule peut se mettre sous la forme de deux équations differentielles non linéaires suivantes :

$$\left(\frac{ml^2}{4}+J\right)\ddot{\theta}+\frac{ml}{2}(\ddot{x}\cos(\theta)-gsin(\theta))=0$$

$$(M+m)\ddot{x}+\frac{ml}{2}\left(\ddot{\theta}\cos(\theta)-\dot{\theta}^2\sin(\theta)\right)=F(t)$$

Si on se limite aux petites variations de θ autour du point de fonctionnement $\theta_0=0$, correspondant à la position verticale de la barre, on peut écrire :$\{\cos\theta\approx 1, sin\theta\approx\theta\}$ et on peut également simplifier l'expression suivante : $\ddot{\theta}\,cos\theta-\dot{\theta}^2 sin\theta\approx\ddot{\theta}$

- ✓ Montrer qu'au voisinage du point de fonctionnement (l'état d'équilibre) le système linéarisé est donné par l'ensemble d'équations suivant :

$$\begin{cases} (M+m)\ddot{x} + \dfrac{ml}{2}\ddot{\theta} = F(t) \\ \left(J + \dfrac{ml^2}{4}\right)\ddot{\theta} + \dfrac{ml}{2}(\ddot{x} - g\theta) = 0 \end{cases}$$

II.3.2. Représentation d'état du système

- ✓ Mettre le système linéarisé sous la forme d'état suivante :

$$\begin{cases} \dot{\underline{x}}(t) = A\underline{x}(t) + Bu(t) \\ y(t) = C\underline{x}(t) + Du(t) \end{cases}$$

En utilisant le vecteur d'état défini comme suit : $\underline{x} = [\theta \quad \dot{\theta} \quad x \quad \dot{x}]^T$,avec la sortie y= $x(t)$

Donnée numériques du système voir le tableau II.1

position angulaire	θ
longueur du pendule	l=0.5m
la force appliquée au chariot	F
la masse du pendule	m=0.1kg
la masse du chariot	M=2kg
accélération de la pesanteur	$g \approx 10m/s^2$
moment d'inertie du pendule	$J = \dfrac{ml^2}{12}$

Tableau II.1 : données numériques du système

- ✓ Créer le système d'état en boucle ouverte en utilisant la fonction Matlab **ss**,
- ✓ Trouver la fonction de transfert en boucle ouverte de l'ensemble chariot-pendule, en utilisant la fonction Matlab **ss2tf.**

II.4.Analyse du système en boucle ouverte

- ✓ Calculer les pôles du système, utiliser la fonction Matlab **roots**,
- ✓ Calculer les valeurs propres de la matrice *A*, utiliser la fonction Matlab **eig**,
- ✓ Que peut-on dire sur les valeurs propres de *A* et les pôles du système ?
- ✓ à partir de ces valeurs, que peut-on dire sur la stabilité du système ?
- ✓ Dans Matlab, simuler la réponse indicielle de consigne 0.5, la réponse impulsionnelle et le diagramme de Bode du système,

- ✓ Dans Simulink, simuler la réponse indicielle de consigne 0.5 et la réponse impulsionnelle du système.

II.5. Commande par retour d'état par placement de pôles

II.5.1. Commandabilite

Avant de synthétiser un système de commande, il est important de savoir si le système est commandable ou pas [9] .

- ✓ Après avoir construit la matrice de commandabilité à l'aide de la fonction Matlab **ctrb**, vérifier la commandabilité du système en utilisant les fonctions Matlab **det** et **rank**,
- ✓ A partir de la Figure II.1, donner le modèle d'état du Système en boucle fermée.

II.5.2. Calcul de la matrice de retour d'état *K*

- ✓ Déterminer l'utilité de la matrice de retour d'état *K*,
- ✓ Choisir des valeurs propres pour la nouvelle matrice d'état ($A - BK$). Pour cela, vous partirez des valeurs propres de *A*. Justifier ce choix ?
- ✓ Faire réaliser par votre programme le calcul du gain de retour d'état (Annexe 3),
- ✓ Comparer le résultat avec celui trouvé en utilisant les fonctions Matlab **place** ou **acker**.

II.6. Analyse du système en boucle fermée

- ✓ Créer un fichier de type M-file contenant la représentation d'état du système bouclé. Utiliser la fonction Matlab **ss**. Observer la position des pôles et des zéros du système en boucle fermée et les comparer à celle du système en boucle ouverte. Utiliser la fonction Matlab **pzmap**. Conclure sur l'effet d'une commande par retour d'état concernant la position des pôles et des zéros,
- ✓ En utilisant Matlab, Simuler le système bouclé en choisissant un échelon de consigne 0.5. Observer les signaux de sortie et de commande ainsi que les états du système, utiliser la fonction Matlab **step**,
- ✓ Comparer la réponse indicielle avec celle obtenue sous Matlab avec la fonction **lsim**,
- ✓ De même, en utilisant Simulink, simuler la réponse indicielle du système bouclé,
- ✓ Commenter les performances du système en boucle fermée par rapport au système en boucle ouverte.

Remarque : si les performances du régime transitoire ne vous conviennent pas, n'hésitez pas à modifier les valeurs propres de $(A - BK)$ puis recalculez K.

II.7. Commande par retour d'état avec action intégrale

- Lorsqu'on effectue une commande par retour d'état, l'erreur statique n'est pas forcément nulle. Ceci n'est généralement pas gênant dans la mesure où l'objectif poursuivi est une régulation, sauf dans le cas où l'objectif consiste à suivre une trajectoire.

  On peut alors procéder à l'insertion d'un intégrateur dans la chaine directe comme illustré sur la figure II.3 suivante :

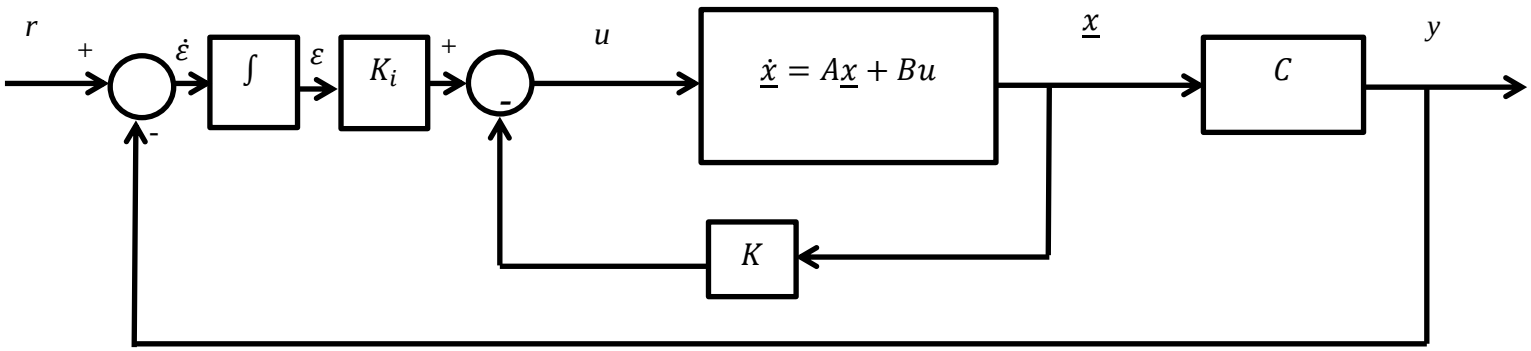


Figure II.3 : Commande par retour d'état avec action intégrale

- ✓ Définir le système augmenté et calculer une valeur de K_i permettant de réduire l'erreur statique par la méthode **place (acker)** de Matlab,
- ✓ En utilisant les équations non linéaires, construire le modèle du pendule inversé +chariot sous Simulink,
- ✓ Tester les performances du système en boucle fermée. On comparera pour cela les résultats obtenus avec le modèle complet et ceux obtenus avec le modèle linéaire.
- **Introduction d'une perturbation :**En présence de perturbations sur l'entrée, nous étudions le système par retour d'état avec l'intégrateur.
- ✓ Simuler pendant 20s pour étudier l'effet des perturbations additives suivantes au niveau de l'entrée :

- Impulsion de 15% de la consigne à t=8s,
- Echelon de 15% de la consigne entre t=8s et t=10s,

- ✓ Vérifier le rejet des perturbations additives.

TP N°3	**Commande par retour de sortie**	Département de Génie Electrique

Objectifs Généraux :

1. Modélisation et simulation d'un hélicoptère sous MATLAB/SIMULINK.
2. Synthèse d'une loi de commande par retour de sortie.

Pré-Requis :

- Modélisation, représentation d'état.
- Analyse d'un système d'état (stabilité, contrôlabilité)
- Synthèse d'une loi de commande par retour d'état

Equipement et Accessoires :

- Micro-ordinateur
- Logiciel Matlab
- Imprimante

TP N°3 Commande par retour de sortie

III.1. Introduction

La commande par retour de sortie est un cas particulier de la commande par retour d'état. En effet, il suffit d'annuler, a priori, toutes les composantes du gain du régulateur correspondant aux composantes non mesurées du vecteur d'état.

Le but de ce TP est la synthèse d'une loi de commande par retour de sortie pour la stabilisation verticale d'un hélicoptère.

III.2 Rappels

III.2.1. Commande par retour de sortie

Il est également possible de stabiliser un système par le retour de sortie conventionnel (figure) il vient alors que :

$$\dot{\underline{x}} = A \cdot \underline{x} + B \cdot (r - K \cdot y)$$
$$y = C \cdot \underline{x}$$

Soit : $\dot{\underline{x}} = A \cdot \underline{x} + B \cdot r - B \cdot K \cdot \left(C \cdot \underline{x}\right)$

Alors :

$$\dot{\underline{x}} = (A - B \cdot K \cdot C) \cdot \underline{x} + B \cdot r$$
$$y = C \cdot \underline{x}$$

III.2.2. Théorème [2]

La paire matricielle $(A - B \cdot K \cdot C, B)$ est commandable si et seulement si la paire matricielle (A, B) est commandable. La paire matricielle $(A - B \cdot K \cdot C, C)$ est observable si et seulement si la paire matricielle (A, C) est observable.

III.3. Description du système :

Soit un hélicoptère assimilé à un point de masse m, se déplaçant sur un axe vertical, et soumis à son poids, à une force de frottement de coefficient α et à la poussée de son rotor *f*.

Figure III.1 : Hélicoptère

z et $\dot{z}$ représentent respectivement l'altitude et la vitesse verticale du véhicule [3].
D'après la seconde loi de Newton, l'accélération subie par un corps est proportionnelle à la résultante des forces qu'il subit, et inversement proportionnelle à sa masse m : $m\vec{a} = \sum \vec{F}$

d'où $$m\ddot{z} = -\alpha\dot{z} - mg + f$$

avec $m = 1kg$ et $\alpha = 0.7$

III.3.1. Représentation d'état du système

Nous posons le vecteur d'état suivant : $\underline{x} = \begin{bmatrix} z \\ \dot{z} \end{bmatrix}$, la variable d'entrée $u = f - mg$ et la sortie $y = z$

- ✓ Trouver la représentation d'état du système de la forme :

$$\begin{cases} \dot{\underline{x}} = A\underline{x} + Bu \\ y = C\underline{x} + Du \end{cases}$$

- ✓ Définir le système linéaire sous Matlab à l'aide de la commande **ss**,
- ✓ Déterminer la fonction de transfert du modèle linéaire. utiliser la fonction Matlab **ss2tf.**

III.4. Analyse du système en boucle ouverte

- ✓ Analyser la stabilité du système. Utiliser la fonction Matlab **eig**,
- ✓ Simuler la réponse à des conditions initiales, en utilisant la fonction Matlab **ode45** et une S_function, cette fonction sera décrite sous Matlab et enregistrée sous le même nom .

III.5. Commande par retour d'état par placement de pôles

III.5.1. Commandabilité et observabilité

- ✓ Calculer la matrice de commandabilité. Donner son rang. utiliser les fonctions de Matlab **ctrb** et **rank**,

- ✓ Conclure sur la commandabilite du système,
- ✓ Vérifier l'observabilité du système en utilisant la matrice d'observabilité. Utiliser les fonctions Matlab **obsv**, **det** et **rank**.

III.5.2. Calcul de la matrice de retour d'état *K*

Mettre en place une commande par retour d'état : $u(t) = -K\underline{x} + r$ (Figure III.2) amenant le système en boucle fermée à une pulsation caractéristique $w_0 = 1rad/s$ et à un amortissement $\xi = 0.75$

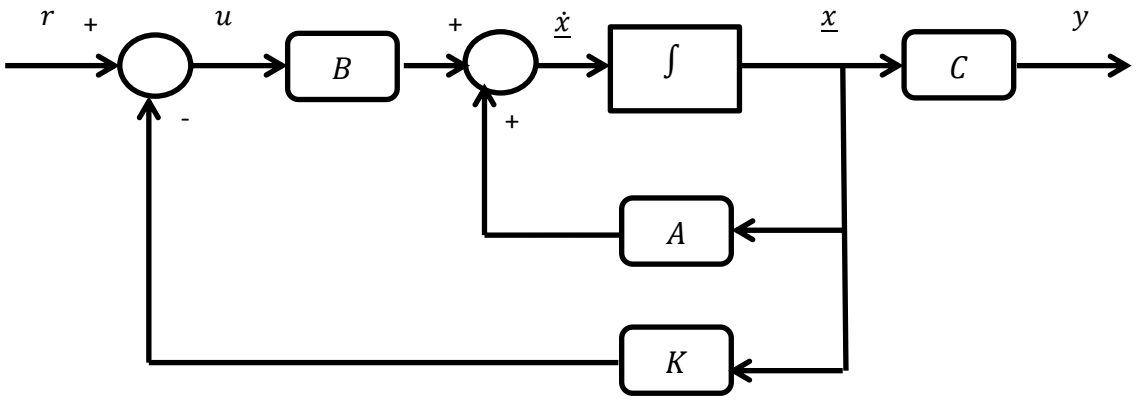


Figure III.2 : Schéma bloc de la commande par retour d'état

En introduisant $u = r - K \cdot \underline{x}$ et l'équation d'observation dans l'équation dynamique, la représentation d'état suivante apparaît en boucle fermée :

$$\underline{\dot{x}} = (A - B \cdot K) \cdot \underline{x} + B \cdot r \quad \text{où} \quad K = \begin{bmatrix} k_1 \\ k_2 \end{bmatrix}$$

La matrice d'évolution du système en boucle fermée est donnée par:

$$\tilde{A} = A - B \cdot K = \begin{bmatrix} 0 & 1 \\ \dfrac{-k_1}{m} & \dfrac{-(k_2 + \alpha)}{m} \end{bmatrix}$$

À partir de ce système :

- ✓ Trouver le polynôme caractéristique du système en boucle fermée, en utilisant les fonctions Matlab **syms** et **poly**,

  Par identification directe avec la dynamique voulue pour le système régulé :

$$p^2 + 2\xi w_0 p + {w_0}^2 = 0$$

- ✓ Donner l'expression littérale des pôles continus $p_1$et p_2 en fonction de ξ et de w_0 utiliser les fonctions Matlab **syms** et **roots**,
- ✓ Refaire la même question précédente en utilisant la fonction Matlab **ord2**,
- ✓ Discuter la stabilité du système en fonction de $\alpha\ et\ m$,
- ✓ Trouver la représentation d'état sous la forme canonique observable en utilisant la fonction Matlab **canon** avec l'option 'companion', puis transposer le résultat pour Obtenir la forme canonique commandable,
- ✓ déterminer l'expression littérale des termes k_1 et k_2 en fonction de $\xi, w_0, m\ et\ \alpha$.
  - ❖ Pour $m = 1kg$ et $\alpha = 0.7$
- ✓ Calculer le gain de retour d'état $K = [k_1 \quad k_2]$
- ✓ Comparer le résultat avec celui fourni en utilisant les fonctions Matlab **place** ou **acker**,
- ✓ Observer la position des pôles et des zéros du système en boucle fermée et les comparer à celle du système en boucle ouverte, utiliser la fonction Matlab **pzmap.**

III.5.3. Simulation du système avec retour d'état

- ✓ Simuler le système linéarisé en boucle fermée (avec retour d'état), utiliser la fonction matlab **lsim,**
- ✓ Refaire le même travail dans **Simulink.**

III.6. Commande par retour de sortie

Puisqu'il est parfois difficile ou coûteux d'exploiter la mesure de tout le vecteur d'état *x*, un autre choix peut consister à se contenter de l'information présentée au niveau du vecteur de sortie *y*. Ce sont alors ces seules sorties qui sont utilisées pour la contre-réaction comme le montre la figure III.3 ci-dessous :

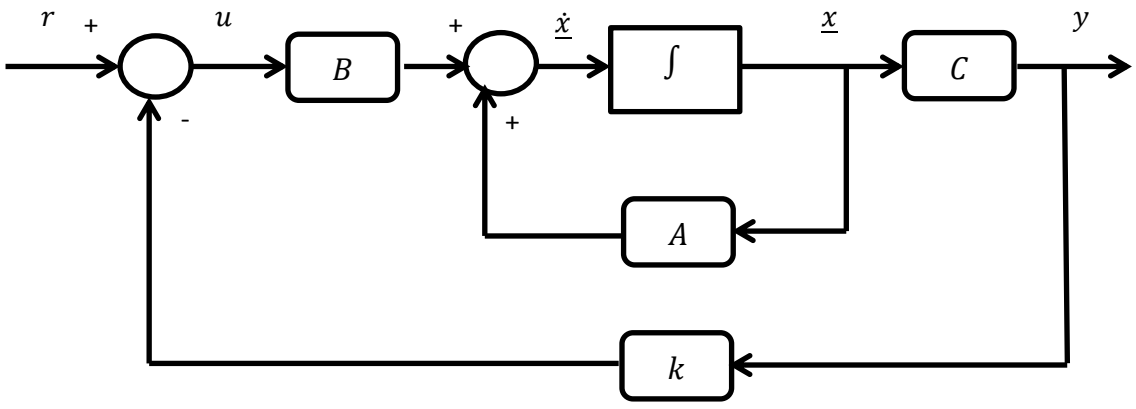


Figure III.3 : Schéma bloc de la commande par retour de sortie

En introduisant $u = r - ky$ et l'équation d'observation dans l'équation dynamique, la représentation d'état suivante apparaît en boucle fermée :

$$\dot{\underline{x}} = (A - BkC)\underline{x} + Br$$

La matrice d'évolution du système en boucle fermée est donnée par :

$$\tilde{A} = A - BkC = \begin{bmatrix} 0 & 1 \\ \frac{-k}{m} & \frac{-\alpha}{m} \end{bmatrix}$$

III.6.1. Calcul du gain de retour de sortie *k*

- ✓ Vérifier la commandabilité et l'observabilité du système bouclé par retour de sortie. Utiliser les fonctions Matlab **ctrb**,**obsv**, et **rank**,
- ✓ Trouver le polynôme caractéristique du système en boucle fermée, en utilisant les fonctions de Matlab **syms** et **poly**,

Par identification directe avec la dynamique voulue pour le système régulé :

$$p^2 + 2\xi w_0 p + {w_0}^2 = 0$$

- ✓ Trouver le gain de retour k en fonction de w_0 et ξ,
- ✓ Discuter le choix de *k*,
- ✓ Calculer k pour une pulsation naturelle $w_0 = 1rd/s$ et un amortissement $\xi = 0.35$,
- ✓ Observer la position des pôles et des zéros du système en boucle fermée et les comparer à celle du système en boucle ouverte, utiliser la fonction de Matlab **pzmap**.

III.6.2. Simulation du système avec retour de sortie

- ✓ Simuler le système linéarisé en boucle fermée (avec retour de sortie), utiliser la fonction Matlab **lsim** ,
- ✓ Refaire le même travail dans **Simulink,**
- ✓ Faire une comparaison entre la commande par retour d'état et la commande par retour de sortie.

TP N°4	**Synthèse d'une loi de commande Par retour d'état avec observateur**	Département de Génie Electrique

Objectifs Généraux :

2. Modélisation et simulation d'une bille en suspension magnétique sous Matlab-Simulink.
3. Synthèse d'une loi de commande par retour d'état estimé.

Pré-Requis :

- Modélisation, représentation d'état.
- Analyse d'un système d'état (stabilité, contrôlabilité)
- Synthèse d'une loi de commande par retour d'état

Equipement et Accessoires :

- Micro-ordinateur
- Logiciel Matlab
- Imprimante

TP N°4 :

Synthèse d'une loi de commande

Par retour d'état avec observateur

IV.1. Introduction

Ce TP a pour objectif d'illustrer le concept d'observation sur des systèmes linéaires monovariables. Les principes et propriétés d'un observateur en boucle ouverte et en boucle fermée (réglé par placement de pôles) seront étudiés en simulation dans l'environnement logiciel MATLAB/SIMULINK.

La visée du TP précédent était de démontrer que la synthèse d'une loi de commande par retour d'état nécessite la connaissance de l'état complet du système. Or, celui-ci n'est généralement pas mesurable, soit par manque de capteur, soit parce que l'état choisi ne correspond pas à une variable physique du système. Dans ce cas, il est nécessaire d'obtenir une estimation de cet état à partir des mesures disponibles qui sont généralement l'entrée et la sortie du système à commander. Une telle estimation est possible à condition que le système considéré soit observable.

On met donc en place, en parallèle avec le système considéré, un reconstructeur d'état ou observateur ayant comme entrées, les entrées et les sorties du système et dont la sortie est une estimation de l'état du système considéré. Sous l'hypothèse de linéarité du modèle du processus, la structure de base de l'observateur est toujours la même, mais sa réalisation dépendra du contexte choisi : continu ou discret, déterministe ou stochastique.

Dans le cas où ce modèle est un modèle déterministe, le reconstructeur d'état sera appelé observateur [8].

IV.2 Rappels

IV.2.1. Observateur complet

La structure de l'observateur est analogue à celle du système à observer à laquelle s'ajoute un terme de correction qui prend en compte la différence entre la sortie réelle du système et la sortie estimée. Cela peut s'exprimer par l'équation d'évolution suivante [8]:

$$\dot{\underline{\hat{x}}}(t) = A \cdot \underline{\hat{x}}(t) + B \cdot u(t) + L \cdot \big(y(t) - \hat{y}(t)\big), \text{ et } \quad \hat{y}(t) = C \cdot \underline{\hat{x}}(t)$$

Le vecteur L représente le gain de l'observateur

Cette expression s'écrit :

$$\dot{\underline{\hat{x}}}(t) = (A - L \cdot C) \cdot \underline{\hat{x}}(t) + B \cdot u(t) + L \cdot y(t)$$

En pratique, la méthode consiste à fixer les valeurs de $(A - L \cdot C)$ de façon a obtenir des valeurs propres plus rapides que celles du système. Ceci revient à choisir les valeurs propres de $(A - L \cdot C)$ ayant des parties réelles négatives plus grandes en valeur absolue que celles des valeurs propres de A.

IV.2.2. Observateur d'ordre réduit

Supposons que sur les n variables d'état, *m* variables soient accessibles à la mesure. Dans ce cas on peut poser :

$$\underline{x}(t) = \begin{bmatrix} \underline{x}_1(t) \\ \underline{x}_2(t) \end{bmatrix}$$

$\underline{x}_1(t)$: vecteur comportant les m variables mesurables

$\underline{x}_2(t)$: vecteur comportant les n-m variables non-mesurables

On peut choisir une représentation d'état de la façon suivante :

$$\dot{\underline{x}}(t) = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \cdot \underline{x}(t) + \begin{bmatrix} B_1 \\ B_2 \end{bmatrix} \cdot u(t) \text{ et } y(t) = C \cdot \underline{x}(t) = [I_m \quad 0_{n-m}] \cdot \begin{bmatrix} \underline{x}_1(t) \\ \underline{x}_2(t) \end{bmatrix}$$

On cherche à réaliser un observateur pour les (n-m) variables d'état, c'est-à-dire pour les variables d'état du vecteur $\underline{x}_2$.

On va construire un observateur qui va s'appuyer à la fois sur la connaissance de l'entrée et sur les variables mesurables, c'est-à-dire les sorties.

Pour cela on définit le vecteur auxiliaire :$\underline{z}(t) = \hat{\underline{x}}_2(t) - L' \cdot y(t)$

Après calcul, on obtient alors :

$$\dot{\underline{z}}(t) = S\underline{z}(t) + Py(t) + Nu(t)$$

Avec

$$\begin{cases} S = (A_{22} - L' \cdot A_{12}) \\ N = (B_2 - L' \cdot B_1) \\ P = (A_{21} + A_{22} \cdot L' - L' \cdot A_{11} - L' \cdot A_{12} \cdot L') \end{cases}$$

Les variables $\underline{x}_2(t)$ sont simplement reconstruites par la relation : $\hat{\underline{x}}_2(t) = \underline{z}(t) + L' \cdot y(t)$

Par conséquent, le gain L' de l'observateur réduit doit être choisi tel que sa dynamique $(A_{22} - L' \cdot A_{12})$soit plus rapide que celle du système.

IV.3. Description du système

Un système de lévitation magnétique est schématisé sur la Figure IV.1.

Il est composé d'un électro-aimant de résistance *R* et d'inductance *L*, alimenté par une tension $u(t)$ et un courant $i(t)$. La deuxième partie du système est formée par une pièce métallique mobile sur laquelle s'exerce une force d'attraction ou de répulsion *F* telle que [5] :

$$F = k\frac{i^2}{h}$$

La position e de la pièce mobile est mesurée par un capteur optique de gain unitaire

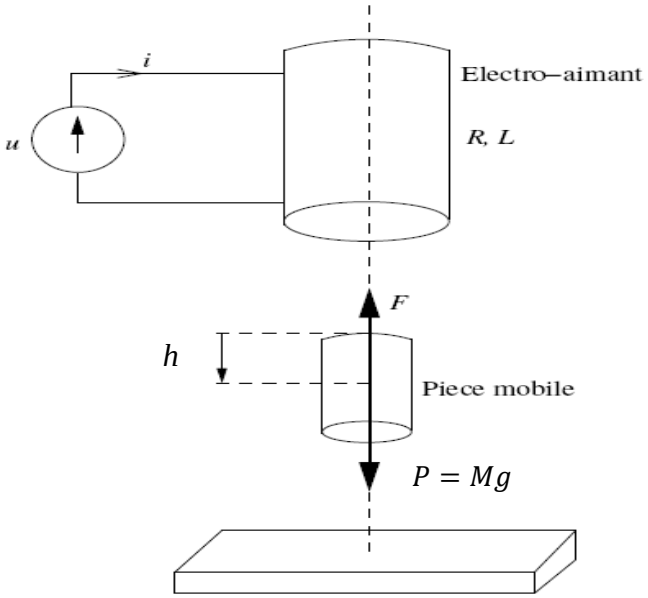


Figure IV.1 : Schéma simplifié d'un système de lévitation magnétique.

L'équation électrique de l'électro-aimant et l'équation dynamique de la pièce métallique mobile sont données par :

$$M\frac{d^2h}{dt^2} = Mg - \frac{Ki^2}{h}$$ (Équation mécanique)

$$u = L\frac{di}{dt} + Ri$$(Équation électrique)

Données numériques de la bille en suspension magnétique voir le tableau

Masse de la bille	M= 0.05 Kg
Résistance	$R = 1\ \Omega$
Inductance de la bobine	$L = 0.01\ H$
k : détermine la force magnétique appliquée sur la bille	$k = 0.0001$
g : Constante de gravité	g=9.81m.s^{-2}

Tableau IV.1: données numériques du système

- ✓ Donner le modèle d'état non linéaire caractérisant le comportement dynamique du système. On considèrera comme état, le vecteur suivant :

$$\underline{x}(t) = \begin{bmatrix} x_1(t) \\ x_2(t) \\ x_3(t) \end{bmatrix} = \begin{bmatrix} h(t) \\ \dot{h}(t) \\ i(t) \end{bmatrix}$$

L'entrée est la tension d'alimentation $u(t)$ et la sortie est la position $h(t)$

Toutefois, autour d'un point de fonctionnent ($\overline{x} = [0.01 \quad 0 \quad 7]^T, \overline{u} = 0$), il peut être modélisé par un système linéaire sous la forme d'état suivante :

$$\underline{\dot{x}} = A\underline{x} + Bu$$
$$y = C\underline{x} + Du$$

- ✓ Utiliser la fonction Matlab **jacobian** pour trouver *A*(*x*, *u*) et *B*(*x*, *u*),
- ✓ Evaluer numériquement les matrices *A* et *B* au point de fonctionnement ($\bar{x}$, $\bar{u}$), Pour ce faire, utiliser la fonction **eval** de Matlab,
- ✓ Définir la fonction de transfert entre $u(t)$ et $h(t)$ en utilisant la fonction **ss2tf** de matlab.

IV.4. Analyse du système en boucle ouverte

IV.4.1. Etude de la stabilité du système

- ✓ Déterminer les pôles du système à partir de la fonction de transfert trouvée et les comparer aux valeurs propres de la matrice d'état, en utilisant les deux fonctions **eig** et **roots** de matlab,
- ✓ A partir de ces valeurs, que peut-on dire sur la stabilité du système ?

IV.4.2. Commandabilité

Avant de synthétiser un système de commande, il est important de savoir si le système est commandable ou pas.

- ✓ Après avoir construit la matrice de commandabilité à l'aide de la fonction Matlab **ctrb**, vérifier la commandabilité du système en utilisant les fonctions Matlab **det** et **rank**.

IV.4.3. Observabilité

Afin de synthétiser un observateur pour un système, il est important de savoir si le système est observable ou pas.

- ✓ Vérifier l'observabilité du système en utilisant la matrice d'observabilité. Utiliser les fonctions Matlab **obsv**, **det** et **rank**.

IV.5. Commande par retour d'état par placement de pôles

IV.5 .1. Calcul de la matrice de retour d'état *K*

- ✓ Déterminer les coefficients k_1, k_2, k_3 à l'aide de la fonction **place** ou la formule d'Ackerman **acker** de Matlab, permettant de placer les pôles en boucles fermées à $-50, -10+10j, -10-10j$,
- ✓ Simuler à l'aide de Matlab et Simulink le comportement attendu en boucle fermée,
- ✓ Visualiser notamment les variables de commande et de sortie.

IV.6. Commande par retour d'état avec observateur

Toutes les variables d'état n'étant pas mesurées, il est nécessaire de faire la synthèse d'un observateur complet afin de pouvoir mettre en œuvre la commande calculée précédemment. La méthode employée est une méthode de placement de pôles, c'est-à-dire que l'on doit calculer le gain *L* correspondant à des pôles spécifiés pour l'observateur.

Il s'agit de l'observateur de Luenberger Figure IV.2 :

$$\underline{\hat{x}}(t) = A\underline{\hat{x}}(t) + Bu(t) + L\big(y - C\underline{\hat{x}}(t)\big)$$

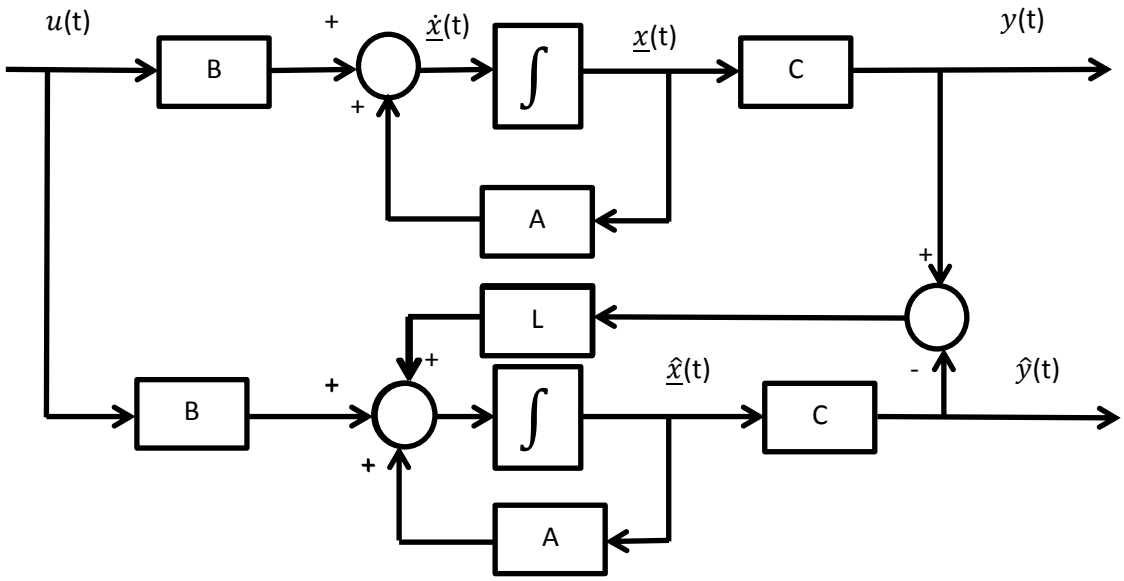


Figure IV.2 : Schéma bloc de l'observateur du système

IV.6.1. Calcul du gain de l'observateur *L*

- ✓ Calculer la matrice de gain de l'observateur, dont la dynamique est régie par les pôles -100, $-102, -103$, Il faut utiliser judicieusement la fonction **place** ou **acker** afin de permettre le calcul de l'observateur,

- ✓ Montrer que l'ensemble (système +observateur) peut se mettre sous la forme suivante :

$$\begin{bmatrix} \dot{\underline{x}} \\ \dot{\hat{\underline{x}}} \end{bmatrix} = \begin{bmatrix} A & 0 \\ L \cdot C & A - L \cdot C \end{bmatrix} \cdot \begin{bmatrix} \underline{x} \\ \hat{\underline{x}} \end{bmatrix} + \begin{bmatrix} B \\ B \end{bmatrix} \cdot u(t)$$

- ✓ Vérifier la stabilité de l'ensemble (système +observateur) en utilisant la fonction Matlab **eig,**
- ✓ vérifier la commandabilité de l'ensemble (système +observateur) en utilisant les fonctions Matlab **det** et **rank**,
- ✓ Etudier le comportement de l'observateur en régime libre si on initialise celui-ci avec des conditions initiales différentes de celles du système.

IV.7. Analyse du système en boucle fermée

- ✓ Simuler, sous matlab-simulink, le comportement de la boucle fermée complète (commande par retour d'état + observateur complet), ensuite visualiser la réponse à un échelon de 0.001 en utilisant la commande **lsim** de Matlab. On accordera notamment une grande importance aux évolutions des variables d'état et de commande. Puis initialiser l'observateur par des conditions initiales différentes de celles du système,
- ✓ Analyser les conséquences sur le comportement global de la boucle fermée, des modifications des dynamiques du retour d'état et de l'observateur,

- ✓ Compléter la synthèse par le calcul du gain N permettant d'obtenir une erreur statique nulle en régime permanent entre la consigne et la hauteur de la bille (Figure IV.3), et cela en utilisant la fonction **rscale (Annexe 04)**,

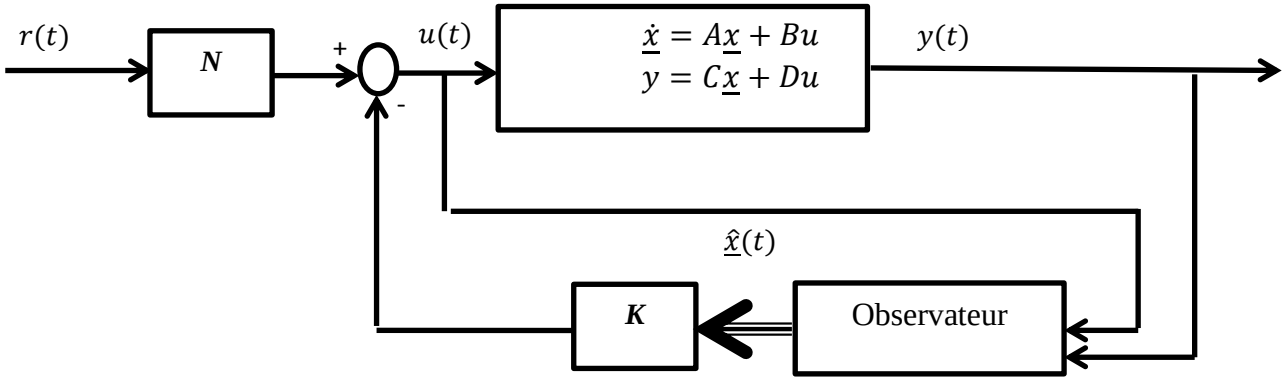


Figure IV.3 : schéma bloc d'un retour d'état estimé

- ✓ Analyser les conséquences sur le comportement global de la boucle fermée de la modification des dynamiques du retour d'état et de l'observateur.

IV.8.Commande par retour d'état avec observateur réduit

- Puisque la hauteur de la bille est mesurée, on peut se limiter à estimer simplement sa dérivée et le courant (Figure IV.4)
  - ✓ Calculer directement le gain L' de l'observateur réduit, pour avoir les valeurs propres $-105, -107$ pour la matrice $A_{22} - L' \cdot A_{12}$,
  - ✓ Simuler sous Matlab-Simulink le comportement de la boucle fermée complète (commande par retour d'état + observateur réduit). On accordera une grande importance aux évolutions des variables d'état et de commande.

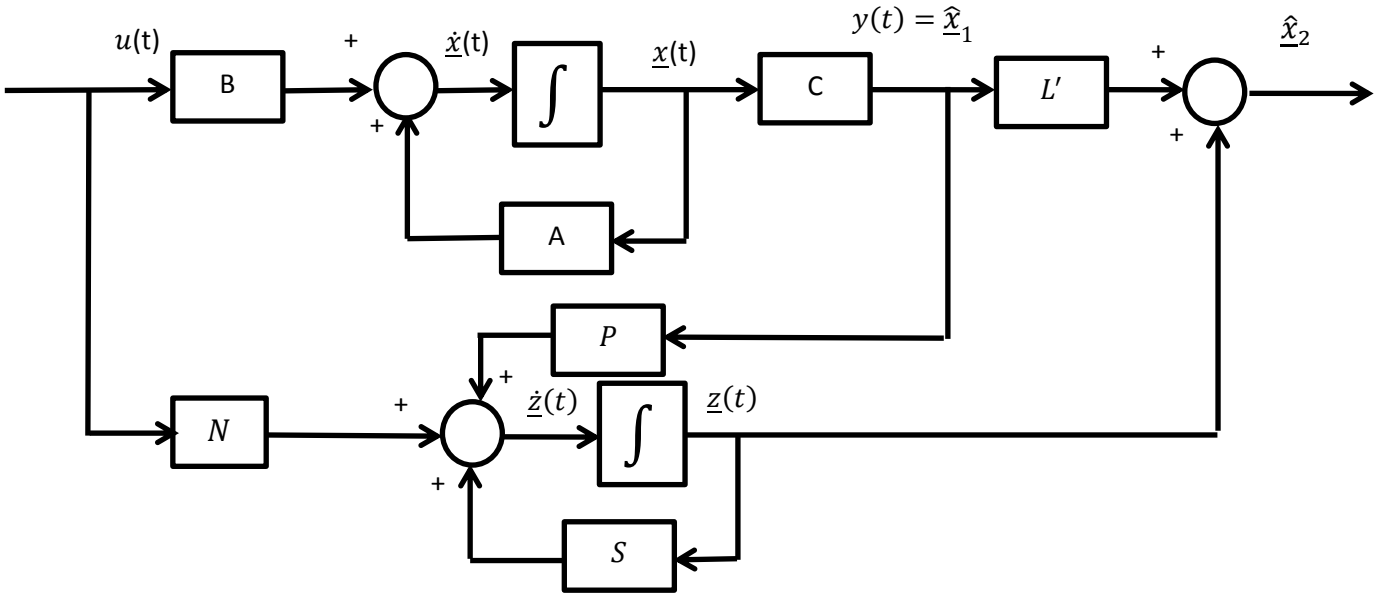


Figure IV.4: schéma bloc d'un observateur réduit

TP N°5	Commande par retour d'état avec observateur à temps discret	Département de Génie Electrique

Objectifs Généraux :

1. Modélisation et simulation d'un moteur à courant continu sous MATLAB/SIMULINK.
2. Synthèse d'une loi de commande par retour d'état estimé dans le domaine discret.

Pré-Requis :

- Modélisation, représentation d'état.
- Analyse (stabilité, contrôlabilité, observabilité)
- Synthèse d'une loi de commande par retour d'état
- Principe d'observateur
- Principe d'échantillonnage

Equipement et Accessoires :

- Micro-ordinateur
- Logiciel Matlab
- Imprimante

TP 5 : Commande par retour d'état

avec observateur à temps discret

V.1. Introduction

La commande des systèmes était dans le passé le privilège des techniques de commande analogiques. De nos jours la diffusion de l'électronique numérique fait que la quasi-totalité des commandes sont faites à l'aide micro contrôleurs, calculateurs, DSP…

Ce saut technologique implique l'utilisation d'outils tels que la transformation en z ou les variables d'état discrètes.

Le but de cette manipulation est de chercher la représentation d'état discrète du système réel, de choisir convenablement la période d'échantillonnage, de tracer les réponses temporelles et d'étudier la stabilité du système en présence d'un retour d'état estimé.

V.2. Rappels

V.2.1. Représentation d'état discrète

Un système discret est représenté par des équations de récurrence d'état et de sortie selon les notations conventionnelles suivantes :

$$\underline{x}(k+1) = F \cdot \underline{x}(k) + G \cdot u(k)$$
$$y(k) = C \cdot \underline{x}(k) + D \cdot u(k)$$

F, G dépendent de la période d'échantillonnage T_e, et C et D demeurent inchangées

V.2.2. Observateur d'état plein

Un observateur d'ordre plein (ou complet) permet de reconstituer toutes les variables d'état, sans les mesurer, à partir des seules informations sur la commande appliquée et la mesure des sorties[4].

On définira alors la structure de l'observateur à partir de la représentation échantillonnée du système :

$$\underline{\hat{x}}(k+1) = [F - LC] \cdot \underline{\hat{x}}(k) + Gu(k) + Ly(k)$$

Le calcul de *L* revient à imposer les valeurs propres de la matrice d'état de l'observateur :

$F - L \cdot C$, et par application de la formule d'ackerman (fonction Matlab **acker**) avec les matrices transposées F^T et C^T

V.2.2. Observateur d'ordre réduit

Il n'est pas toujours utile de reconstruire tous les états si on peut accéder à la mesure d'un certain nombre de variables d'état. On se contente de reconstruire les états non mesurés ce qui nécessite une partition de l'espace entre ce qui est mesuré, $\underline{x_1}$ et ce qui doit être estimé, $\underline{x_2}$

On cherche donc à élaborer une structure d'observateur réduit donnant une estimation de $\underline{x_2}$ soit $\underline{\hat{x}_2}$.

La représentation d'état s'écrit comme suit

$$\begin{bmatrix} \underline{x_1}(k+1) \\ \underline{x_2}(k+1) \end{bmatrix} = \begin{bmatrix} F_{11} & F_{12} \\ F_{21} & F_{22} \end{bmatrix} \cdot \begin{bmatrix} \underline{x_1}(k) \\ \underline{x_2}(k) \end{bmatrix} + \begin{bmatrix} G_1 \\ G_2 \end{bmatrix} \cdot u(K)$$

Après calcul on obtient l'observateur réduit sous la forme suivante

$$\underline{\hat{x}_2}(k) = F_e \underline{\hat{x}_2}(k-1) + G_e u(k-1) + L' \underline{x_1}(k) + L_e \underline{x_1}(k-1)$$

Avec : $\begin{cases} F_e = F_{22} - L' \cdot F_{12} \\ L_e = F_{21} - L' \cdot F_{11} \\ G_e = G_2 - L' \cdot G_1 \end{cases}$

Il s'agit de calculer L' par la formule d'Ackerman (fonction Matlab **acker**) en raisonnant sur le couple F_{22}, F_{12}

Alors, on place à présent les pôles de la matrice d'état de l'observateur réduit :$F_{22} - L \cdot F_{12}$

V.3. Description du système

Reprenons le cas du moteur à courant continu dont nous avons établi les équations d'état (**TP N 01**).

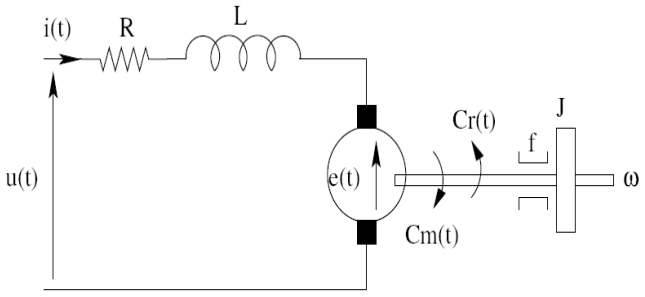


Figure V.1: Schéma du moteur à courant continu

- L'équation du mouvement de rotation de l'arbre du moteur et la loi d'Ohm dans l'induit sont les suivantes :

$$\begin{cases} u(t) = R \cdot i(t) + L\dfrac{di(t)}{dt} + k \cdot w(t) \\ J\dfrac{dw}{dt} + fw(t) = k \cdot i(t) \end{cases}$$

- Les paramètres du moteur et de sa charge sont les suivants :

Coefficient de moteur $k = 0.45\ \mathrm{N \cdot m/A}$ (c'est aussi le coefficient de f.c.e.m en V/rad/s),

Coefficient de frottement visqueux $f = 3 \cdot 10^{-4}\mathrm{N \cdot m/rad/\ s}$,

Moment d'inertie $J = 1{,}1 \cdot 10^{-3}\mathrm{kg}.m^2$,

Résistance de l'induit $R = 4{,}8\Omega$,

Inductance de l'induit $L = 18.4\ \mathrm{mH}$.

- L'induit est alimenté par un amplificateur de puissance de gain en tension $\gamma = 30$,
- La vitesse est mesurée par la dynamo tachymétrique. elle délivre 10V/10 rad/s(coefficient $\beta = 0{,}1$),
- Le courant d'induit est mesuré par capteur à effet Halll qui délivre 1V/A (coefficient de conversion α).

Avec :

- Vecteur d'état : $x_1(t) = \beta \cdot w(t)$ la vitesse et $x_2(t) = \alpha \cdot i(t)$ le courant
- Commande : entrée de l'amplificateur de puissance
- Vecteur de sortie : $y(t) = x_1(t)$ la vitesse

Les équations d'état sont les suivantes :

$$\dot{x}_1(t) = -\frac{f}{J} \cdot x_1(t) + \frac{k \cdot \alpha}{J \cdot \beta} \cdot x_2(t)$$

$$\dot{x}_2(t) = -\frac{k \cdot \beta}{L \cdot \alpha} \cdot x_1(t) - \frac{R}{L} \cdot x_2(t) + \frac{\gamma \cdot \beta}{L} u(t)$$

Et

$$y(t) = x_1(t)$$

✓ Mettre le système linéarisé sous la forme d'état suivante :

$$\begin{cases} \dot{\underline{x}}(t) = A \cdot \underline{x}(t) + B \cdot u(t) \\ y(t) = C \cdot \underline{x}(t) + D \cdot u(t) \end{cases}$$

V.4. Discrétisation du système

❖ Choisir une période d'échantillonnage $T_e = 2\ ms$

✓ Discrétiser (avec bloqueur d'ordre 0) la représentation d'état continue en utilisant la fonction Matlab **c2dm** (option 'zoh'), pour obtenir un nouveau modèle,

- ✓ En utilisant la fonction Matlab **ss2tf**, passer de la représentation d'état (représentation interne) du système à sa fonction de transfert $H(z^{-1})$ (représentation externe),
- ✓ Trouver la représentation d'état sous la forme canonique diagonale en utilisant la fonction Matlab **canon** avec l'option '**modal**',
- ✓ Trouver la représentation d'état sous la forme canonique observable en utilisant la fonction Matlab **canon** avec l'option '**companion**',puis transposer le résultat pour
- ✓ Obtenir la forme canonique commandable,

V.5. Analyse du système en boucle ouverte

- ✓ Analyser la position des pôles et des zéros de la fonction de transfert du système échantillonné bloqué $H(z^{-1})$. Est-il stable en boucle ouverte ? Justifier. Utiliser les fonctions Matlab **pzmap** et **axis**,
- ✓ Calculer les valeurs propres de la matrice d'état du système discrétisé en utilisant la fonction Matlab **eig**,
- ✓ Faire le point en répondant à cette question : Que peut-on dire sur les valeurs propres de F et les pôles du système ?
- ✓ Tracer la réponse impulsionnelle du système discrétisé, utiliser la fonction matlab **dimpulse**, que peut-on dire sur la stabilité ?
- ✓ Calculer le gain statique du système en utilisant la fonction Matlab **dcgain**,
- ✓ Analyser les performances temporelles en superposant sur une même figure la réponse indicielle du système continu et celle du système discrétisé, utiliser les fonctions Matlab **step** et **dstep,** à partir de la réponse indicielle, déterminer le dépassement et le temps de montée,
- ✓ Refaire la même question précédente en utilisant les fonctions Matlab **lsim** , **dlsim et stairs**,
- ✓ Analyser les performances fréquentielles en superposant sur une même figure la représentation fréquentielle de Bode du système continu et celle du système discrétisé. Utiliser la fonction Matlab **bode**. (Déterminer les marges de stabilité à partir du tracé de Bode),
- ✓ Discuter les différences entre les deux modèles pour valider la discrétisation ,
- ✓ Dans Simulink, simuler les réponses indicielle et impulsionnelle du système en boucle ouverte.

V.6 Commande par retour d'état à temps discret

On désire construire un contrôleur de type commande par retour d'état permettant d'atteindre des objectifs dynamique et statique exigés pour la boucle fermée (Figure VI.4) :

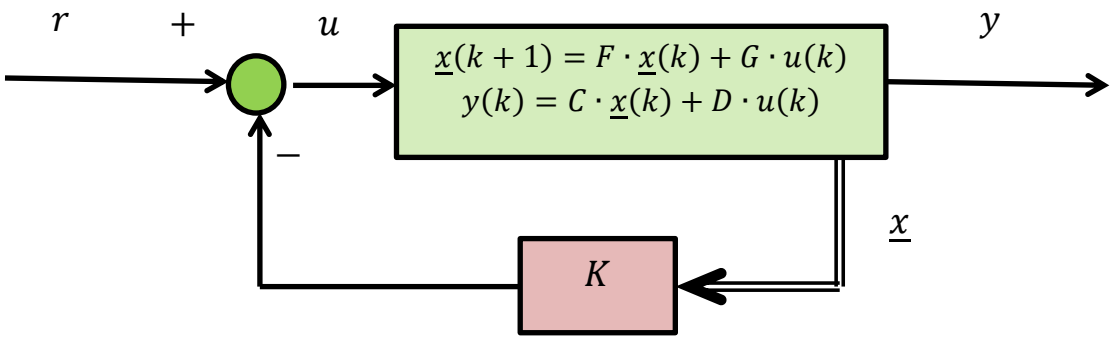


Figure V.2 : Schéma bloc du retour d'état à temps discret

- ✓ Après avoir construit la matrice de commandabilité du système discret à l'aide de la fonction Matlab **ctrb**, vérifier la commandabilité du système en utilisant les fonctions Matlab **det** ou **rank**. Dans quelle situation se trouve-t-on ?

- ✓ Calculer la matrice *K* du contrôleur à l'aide de la fonction **place** de Matlab, afin de placer les deux pôles aux valeurs suivantes :

$$z_1 = 0.8, \quad z_2 = 0.1$$

- ✓ Simuler, à l'aide de Matlab et Simulink, le comportement attendu en boucle fermée. Visualiser notamment les variables de commande et de sortie,
- ✓ Compléter la synthèse par le calcul du gain $\overline{N}$ permettant d'obtenir une erreur statique nulle en régime permanent, et cela après plusieurs essais (Figure V.3),

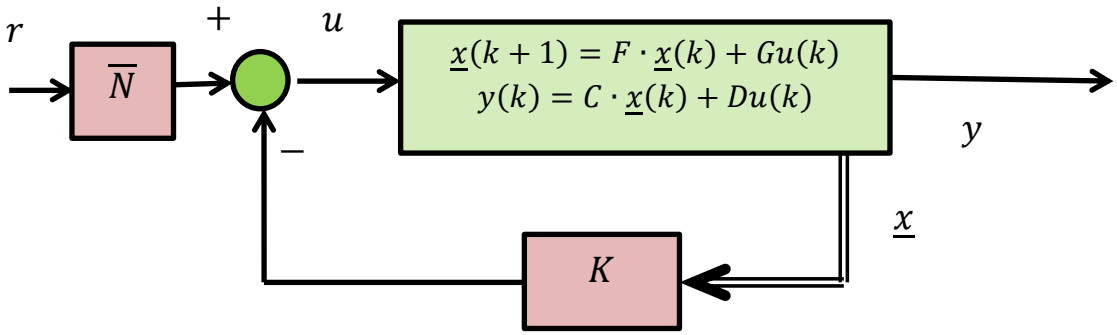


Figure V.3 : Schéma bloc du retour d'état à temps discret avec gain

V.7. Commande par retour d'état avec observateur complet à temps discret

- ❖ Toutes les variables d'état n'étant pas mesurées, faire la synthèse d'un observateur complet afin de pouvoir mettre en œuvre la commande calculée précédemment.
- ✓ Vérifier l'observabilité du système en utilisant la matrice d'observabilité. Utiliser les fonctions Matlab **obsv**, **det** et **rank**,
- ✓ Calculer la matrice de gain de l'observateur L, dont la dynamique est régie par les deux pôles à l'origine(réponse en temps minimal), Utiliser judicieusement la fonction **place** ou **acker** pour permettre de calculer l'observateur.

V.7.1. Analyse du système en boucle fermée

On s'intéresse à la synthèse d'une commande estimée où la loi de commande a pour expression :

$$u(k) = -K\underline{\hat{x}}(k) + r(k)$$

- ✓ Montrer que le système en boucle fermée (observateur + retour d'état) peut se s'écrire sous la forme suivante :

$$\begin{bmatrix}\underline{x}(k+1)\\ e(k+1)\end{bmatrix} = \begin{bmatrix}F-GK & GK\\ 0 & F-LC\end{bmatrix}\begin{bmatrix}\underline{x}(k)\\ e(k)\end{bmatrix} + \begin{bmatrix}G\cdot\overline{N}\\ 0\end{bmatrix}\cdot r(k)$$

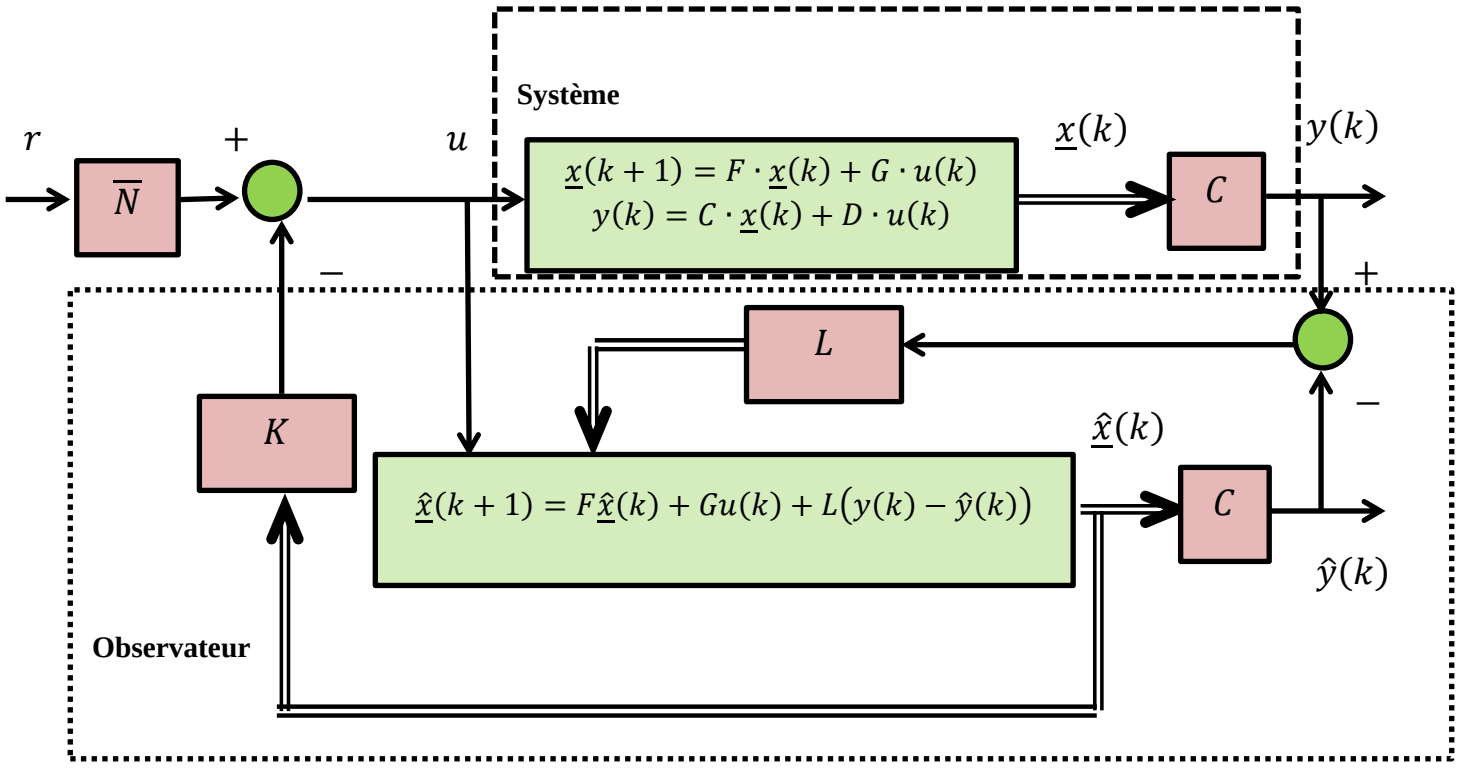


Figure V.4 : Schéma bloc du retour d'état estimé à temps discret

- ✓ Simuler sous Matlab-Simulink le comportement de la boucle fermée complète (commande par retour d'état + observateur complet). On accordera une grande importance aux évolutions des variables d'état et de commande,
- ✓ Analyser les conséquences sur le comportement global de la boucle fermée de la modification des dynamiques du retour d'état, de l'observateur, et de la période d'échantillonnage.

V.8. Commande par retour d'état avec observateur réduit à temps discret

- Puisque la vitesse angulaire est mesurée, on peut se limiter à estimer simplement le courant
- ✓ Calculer directement le gain L de l'observateur réduit, pour avoir une valeur propre nulle pour la matrice $F_{22} - L \cdot F_{12}$,
- ✓ Simuler sous Matlab-Simulink le comportement de la boucle fermée complète (commande par retour d'état + observateur réduit). On accordera une grande importance aux évolutions des variables d'état et de commande.

TP N°6	Commande linéaire Quadratique Etude et mise en œuvre	Département de Génie Electrique

Objectifs Généraux :

1. Modélisation et simulation d'un bras commandé par moteur sous MATLAB/SIMULINK.
2. Synthèse d'une loi de commande par retour d'état de type Linéaire Quadratique.

Pré-Requis :

- Modélisation, représentation d'état.
- Analyse (stabilité, contrôlabilité, observabilité)
- Synthèse d'une loi de commande par retour d'état
- Principe d'observateur

Equipement et Accessoires :

- Micro-ordinateur
- Logiciel Matlab
- Imprimante

TP N°6 : Commande linéaire Quadratique

Etude et mise en œuvre

VI.1. Introduction

La commande linéaire quadratique LQ a l'avantage et l'inconvénient de ne pas imposer le choix des valeurs propres de la matrice de commande corrigée. Ceci est un avantage car le concepteur n'a pas à réfléchir sur le choix des valeurs propres, mais c'est également un inconvénient car on ne choisit donc pas la dynamique du système [1].

VI.2. Rappels

Le principe de la commande Linéaire Quadratique (LQ), souvent appelée "commande optimale", est de synthétiser le vecteur K qui contient les gains associés à chaque variable d'état, il sera calculé en utilisant l'algorithme de la commande linéaire quadratique LQR qui minimise un critère quadratique

Le tableau VI.1 suivant résume le principe de la commande linéaire quadratique [10] :

Cas continu		Cas discret	
Système	$\begin{cases}\dot{\underline{x}}(t) = A \cdot \underline{x}(t) + B \cdot u(t) \\ y(t) = C \cdot \underline{x}(t) + D \cdot u(t)\end{cases}$ (1)	Système	$\begin{cases}\underline{x}(k+1) = F \cdot \underline{x}(k) + G \cdot u(k) \\ y(k) = C \cdot \underline{x}(k) + D \cdot u(k)\end{cases}$ (6)
Critère	$J = \frac{1}{2}\int_0^{\infty}(\underline{x}^T Q x + u^T R u)dt$ (2)	Critère	$J(u) = \frac{1}{2}\sum_{k=1}^{\infty}\begin{bmatrix}x(k)^T \cdot Q \cdot x(k) + \\ u(k)^T \cdot R \cdot u(k)\end{bmatrix}$ (7)
Solution	$PA + A^T P - PBR^{-1}B^T P + Q = 0$ (3) $K = R^{-1} * B^T * P$ (4)	Solution	$P = F^T(P - PG$ $(G^T PG + R)^{-1}G^T P)F + Q$ (8) $K = R^{-1} * G^T * P$ (9)
Commande	$u(t) = -K\underline{x}(t)$ (5)	Commande	$u(k) = -K\underline{x}(k)$ (10)

Tableau VI.1 : Principe de la commande linéaire quadratique

VI.3. Description du système

Considérons le modèle d'un bras manipulateur entraîné par un moteur à courant continu par l'intermédiaire d'un engrenage, nous supposons que le moment d'inertie du moteur est négligeable par rapport à celui du bras du robot [10].

Le bras étant considéré comme un point de masse *m* attaché à l'extrémité d'une tige de longueur *l* et d'une masse négligeable.

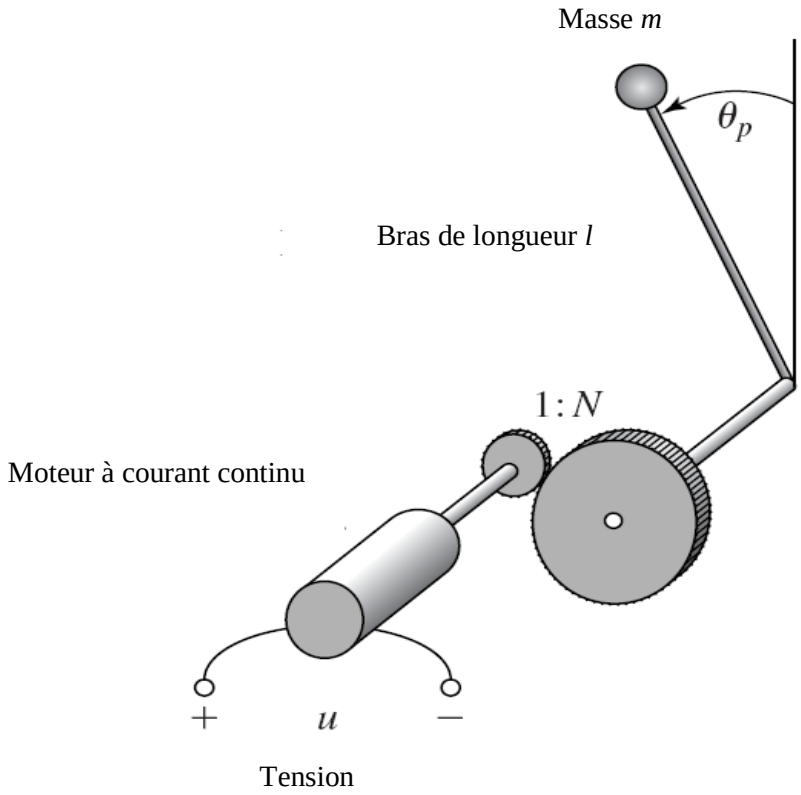


Figure VI.1 : Description du système étudié

VI.3.1. Modélisation du système

Un modèle du système peut être élaboré en considérant d'une part les équations de base qui décrivent le comportement d'un moteur à courant continu, et d'autre part les équations de la dynamique du bras selon les lois de Newton.

- Nous utilisons la deuxième loi de Newton pour décrire l'équation modélisant la dynamique du bras,

$$I_a \frac{d^2\theta_p}{dt^2} = mglsin(\theta_p) + T_p \qquad (11)$$

Avec T_p le couple appliqué sur le bras manipulateur est donné par :

$$T_p = NT_m = NK_m i_a$$

Où : T_m est le couple délivré par le moteur, et est également proportionnel au courant i_a qui le traverse. K_m étant la constante du couple-moteur, et i_a le courant d'induit.

L'angle de rotation de la plate-forme θ_p est directement proportionnel à celui du moteur θ_m

$$\theta_m = N\theta_p$$

N : le rapport de transmission

En remplaçant l'inertie de la masse *m* , $I_a = ml^2$ et l'expression du couple T_p, on trouve :

$$ml^2 \frac{d^2\theta_p}{dt^2} = mglsin(\theta_p) + NK_m i_a \qquad (12)$$

- Le stator du moteur est composé d'aimants permanents qui fournissent un champ magnétique supposé constant, son affaiblissement en fonction du courant rotorique restant faible.

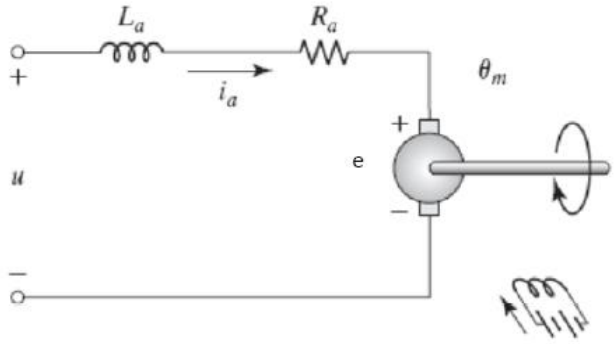


Figure V.2 : Moteur à courant continu

Selon cette hypothèse, et en négligeant l'inductance de l'enroulement, $L_a \approx 0$, la loi des mailles de Kirchhoff donne :

$$R_a i_a + e = u(t) \qquad (13)$$

Où e est la force contre électromotrice égale à :$e = K_e \frac{d\theta_m}{dt} = NK_e \frac{d\theta_p}{dt}$

et K_e est la constante de vitesse, on trouve :

$$R_a i_a + K_e N \frac{d\theta_p}{dt} = u(t) \quad (14)$$

En remplaçant l'expression du courant de i_a dans l'équation du bras on trouve :

$$ml^2 \frac{d^2\theta_p}{dt^2} = mglsin(\theta_p) + NK_m \left(\frac{u}{R_a} - \frac{K_e N \frac{d\theta_p}{dt}}{R_a} \right) \quad (15)$$

Avec :

Longueur du bras manipulateur	l=1m
masse	m=1Kg
Rendement	N =10
Constante du couple moteur	K_m =0.1
Constance de vitesse	K_e =0.1
Résistance aux bornes	R_a =1Ω

Tableau VI.2 : Données numériques du moteur à courant continu

On cherche à déterminer le modèle non linéaire représenté par un modèle d'état. Pour cela, on définit le vecteur d'état comme suit: $\underline{x} = \left[\theta_p \quad \frac{d\theta_p}{dt}\right]^T$ et la sortie $y = x_1$

Donc, en utilisant l'équation (15), on obtient le modèle d'état du bras manipulateur suivant :

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} x_2 \\ \frac{g}{l} sin(x_1) - \frac{K_e K_m N^2}{ml^2 R_a} x_2 + \frac{NK_m}{ml^2 R_a} u \end{bmatrix}$$

Ce système est non linéaire. Toutefois, autour d'un point de fonctionnent ($\bar{x} = 0, \bar{u} = 0$), il peut être modélisé par un système linéaire sous la forme d'état suivante :

$$\underline{\dot{x}} = A\underline{x} + Bu$$
$$y = C\underline{x} + Du$$

- ✓ Simuler le comportement du modèle non linéaire pour une entrée nulle dans un modèle Simulink.

VI.3.2. Linéarisation du système non linéaire

- ✓ Utiliser la fonction Matlab **jacobian** pour trouver $A(x, u)$ et $B(x, u)$,
- ✓ Evaluer numériquement les matrices A et B au point de fonctionnement $(\bar{x}, \bar{u})$,
  Pour ce faire, utiliser la fonction **eval** de Matlab,
- ✓ Déterminer les matrices C et D ayant les bonnes dimensions dans le cas où seulement l'angle θ_p est mesuré.

VI.4. Analyse du système en boucle ouverte

- ✓ Créer une représentation d'état continue (SysC) avec la fonction **ss** de Matlab , puis dans Simulink,
- ✓ Calculer les pôles du système. Les pôles correspondent aux valeurs propres de la matrice A. Utiliser la fonction Matlab **eig**. Conclure,
- ✓ Vérifier la commandabilité du système en exécutant les fonctions Matlab **ctrb** et **rank.**

VI.5. Commande linéaire quadratique LQ

VI.5.1. Cas Continu

Pour obtenir le correcteur dans le domaine continu, la méthode LQR cherche le vecteur K qui minimise le critère de l'équation (2) du tableau VI.1,

Les matrices Q, et R étant symétriques avec $Q \geq 0$ et $R > 0$

On choisira

$$Q = C^T * C = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \text{ et } R = 1$$

- ✓ Calculer la matrice P la solution de l'équation de Riccati (3) du tableau VI.1, par la fonction Matlab **care**,
- ✓ Calculer le gain de retour d'état en utilisant la relation (4) du tableau VI.1,
- ✓ Comparer les matrices P et K obtenues, avec celles qui sont obtenues avec la fonction **lqr** de Matlab,
- ✓ Simuler le comportement du système pour différentes valeurs de Q et R en fonction des réponses obtenues. Conclure,
- ✓ Montrer quelles composantes de Q et R ont de l'importance dans la régulation?
- ✓ Quelles valeurs de Q et R conservez-vous ? Justifier votre réponse.

VI.5.2. Cas discret

- ❖ Prendre une période d'échantillonnage de **1ms** ,

- ✓ Discrétiser (avec bloqueur d'ordre 0) la représentation d'état continue en utilisant la fonction Matlab **c2dm** (option 'zoh'), pour obtenir un nouveau modèle (SysD) et comparer les deux résultats,
- ✓ Analyser la position des pôles et des zéros du système discrétisé. Est-il stable en boucle ouverte ? Justifier. Utiliser les fonctions Matlab **pzmap** et **axis**,
- ✓ Simuler le comportement des modèles continu et discret pour une entrée nulle dans un fichier Simulink,
- ✓ Discuter les différences entre les deux modèles pour valider la discrétisation,
- ✓ Vérifier la commandabilité du système discret en exécutant les fonctions Matlab **ctrb** et **rank,** conclure,
- ✓ On cherche à trouver une commande optimale (LQR) dans le domaine discret avec les matrices de pondérations Q et R suivantes :

$$Q = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \text{ et } R = 1$$

- ✓ Implémenter l'équation de Riccati (8) du tableau VI.1 sous forme récurrente. Définir un nombre d'itérations suffisamment grand pour assurer une "bonne" convergence de l'algorithme,
- ✓ Après avoir obtenu une approximation de P, calculer la matrice de gain K par la relation (9) du tableau VI.1,
- ✓ Calculer la matrice P la solution de l'équation de Riccati (8), par la fonction Matlab **dare**,
- ✓ Calculer le gain de retour d'état par la relation (9),
- ✓ Comparer les matrices P et K obtenues avec celles qui sont obtenues avec la fonction **dlqr** de Matlab,
- ✓ Analyser les performances temporelles et fréquentielles du système bouclé et les comparer à celles du système en boucle ouverte. Utiliser les fonctions Matlab **lsim** et **bode**,
- ✓ Réaliser sous SIMULINK le système en boucle fermée complet (Système +Contrôleur).

TP N°7	Commande linéaire Quadratique Gaussienne (LQG)	Département de Génie Electrique

Objectifs Généraux :

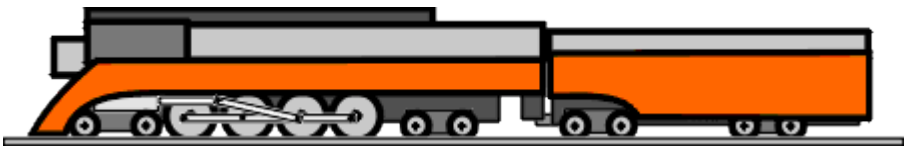
1. Modélisation et simulation d'un train électrique sous MATLAB/SIMULINK.
2. Conception d'un estimateur de Kalman
3. Synthèse d'une loi de commande de type linéaire quadratique gaussienne LQG .

Pré-Requis :

- Modélisation, représentation d'état.
- Analyse (stabilité, contrôlabilité, observabilité)
- Principe d'estimateur
- Synthèse d'une loi de commande par retour d'état de type Linéaire Quadratique.

Equipement et Accessoires :

- Micro-ordinateur
- Logiciel Matlab
- Imprimante



TP N°7 :

Commande Linéaire Quadratique

Gaussienne « LQG »

VII.1. Introduction

La commande linéaire quadratique gaussienne, dite commande LQG, est similaire à la commande LQ, à la différence près qu'elle intègre un estimateur dans la boucle de commande [1].En effet, la commande LQG, réunit un correcteur LQ et un estimateur de Kalman dans un souci particulier de réduire les bruits de mesure (Figure VII.1) .

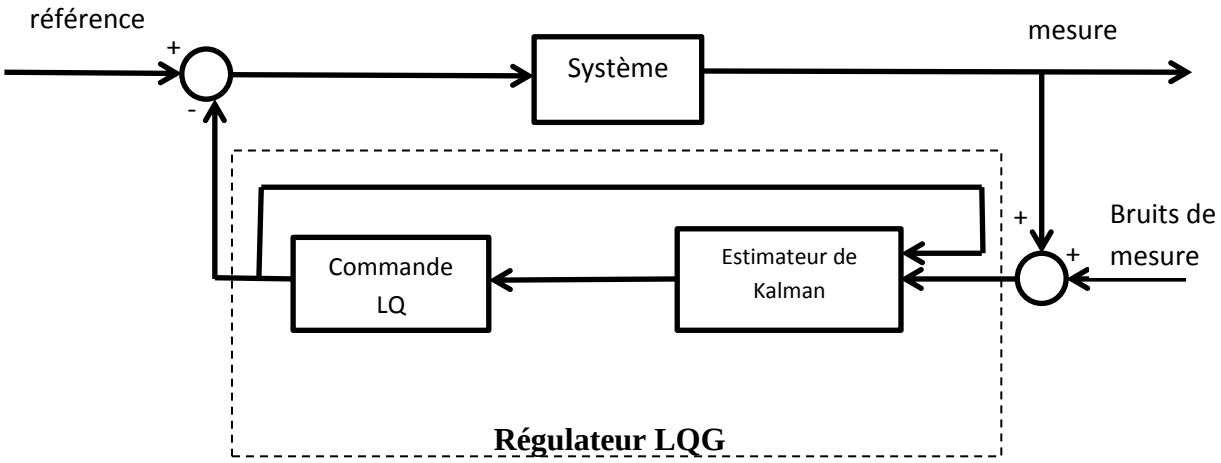


Figure VII.1 : Principe de la commande linéaire quadratique gaussienne

VII.2.Rappels

VII.2.1. Commande optimale linéaire quadratique gaussienne LQG

- **Cas continu**

Dans le cas de systèmes bruités, où interviennent des phénomènes aléatoires, le système sera donné par :

$$\begin{cases} \dot{\underline{x}}(t) = A \cdot \underline{x}(t) + B \cdot u(t) + T \cdot w(t) \\ y(t) = C \cdot \underline{x}(t) + D \cdot u(t) + v(t) \\ \underline{z} = N \cdot \underline{x} \end{cases}$$

Où w et v représentent des bruits blancs, de moyenne nulle, indépendants, avec respectivement pour matrice de covariance W et V.

$$E[w(t)w^T(t)] = W \text{ et } E[v(t)v^T(t)] = V$$

avec $W \geq 0$ et $V > 0$

Nous recherchons une loi de commande qui minimise le critère :

$$J = E\left\{\int_0^{\infty} \left[\underline{z}^T(t) Q \underline{z}(t) + u^T(t) R u(t)\right] dt\right\}$$

Où Q et R deux matrices de pondération avec : $Q = Q^T \geq 0$ *et* $R = R^T > 0$

Il faut noter que *R* est un scalaire si le système à commander est un système mono-entrée

Dans le but d'améliorer les performances et de rejeter les effets des perturbations éventuelles, on utilise le schéma-bloc de la Figure VII.3.

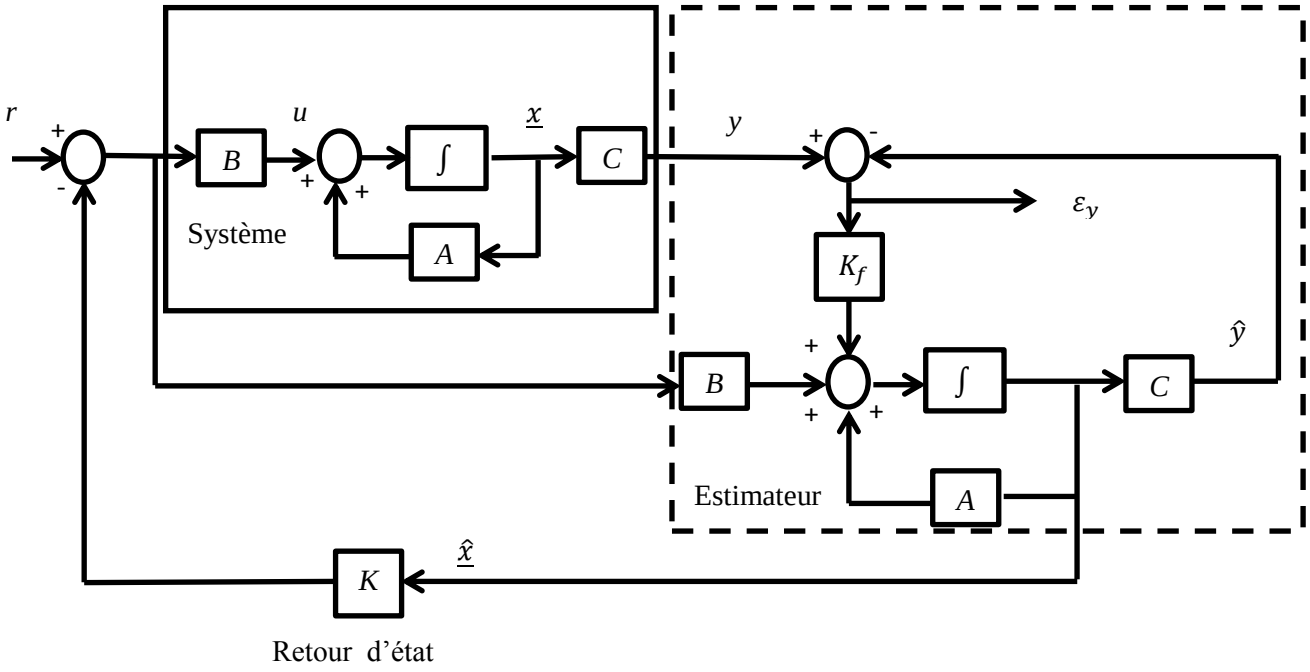


Figure VII.1 : schéma bloc de l'ensemble retour d'état+estimateur

Pour résoudre ce problème, on va utiliser le principe du théorème de séparation, qui énonce que la solution est divisée en deux parties :

1. Calcul de l'estimateur de Kalman de gain $K_f = P_f C^T V^{-1}$:

$$\dot{\hat{\underline{x}}} = A\underline{\hat{x}} + Bu + K_f\left(y - C\underline{\hat{x}} - Du\right)$$

Avec P_f est la solution de l'équation de Riccati suivante :

$$P_f A^T + A P_f - P_f C^T V^{-1} C P_f + T W T^T = 0$$

Avec $P_f = P_f^T > 0$

2. Calcul du régulateur optimal LQ : $u = -K\underline{\hat{x}}$ avec $K = R^{-1}B^T P$ et P solution de

$$PA + A^T P - PBR^{-1}B^T P + N^T QN = 0$$

Donc le réglage du correcteur LQG nécessite la donnée de quatre matrices de pondération : Q et R pour le retour d'état, V et W pour l'estimateur.

La méthode de réglage la plus simple repose sur un réglage séparé : régler V et W de sorte que l'état soit bien reconstruit et régler Q et R pour avoir un bon retour d'état.

- **Cas discret**

A l'image de la commande LQG à temps continu, la version à temps discret consiste en la combinaison d'un filtre de Kalman à temps discret et d'un retour d'état.

VII.3. Description du système

Nous allons utiliser l'exemple d'un train électrique composé d'une locomotive et d'un wagon. En supposant que le train ne se déplace que dans une seule direction, nous souhaitons élaborer un contrôleur qui permette un démarrage et un freinage progressifs ainsi qu'un déplacement à vitesse constante [3].

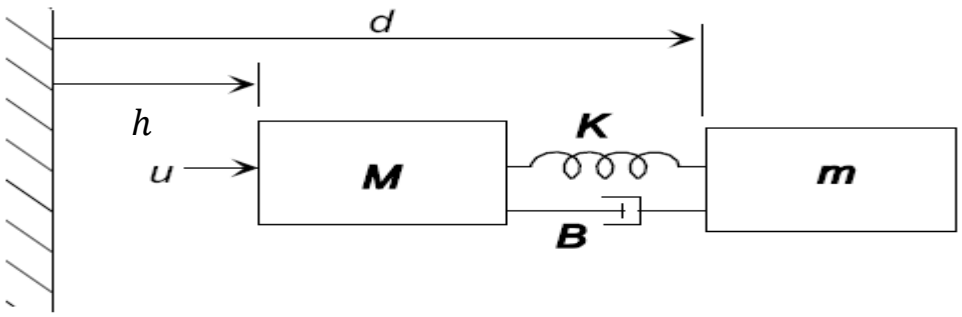


Figure VII.2 : schéma du train électrique

- Les masses de la locomotive et du wagon sont représentées par M et m,
- Ils sont reliés par un ressort de coefficient de dureté K,
- La force appliquée par le moteur de la locomotive est notée u,
- Le coefficient d'amortissement est noté B.

Donnée numériques du système voir le tableau VII.1 :

masse de la locomotive	M=1kg
masse du wagon	m=0.1
coefficient de dureté K	K=0.091
coefficient d'amortissement B	B =0.0036

Tableau VII.1 : données numériques du système

VII.3.1. Modélisation du système mécanique

D'après la seconde loi de Newton, l'accélération subie par un corps est proportionnelle à la résultante des forces qu'il subit, et inversement proportionnelle à sa masse m.

$$m\vec{a} = \sum \vec{F}$$

- ✓ Montrer à l'aide de la seconde loi de Newton que l'ensemble du train électrique peut se mettre sous la forme de deux équations differentielles linéaires suivantes :

$$M\ddot{h} + B(\dot{h} - \dot{d}) + K(h - d) = u$$
$$m\ddot{d} + B(\dot{d} - \dot{h}) + K(d - h) = 0$$

VII.3.2. Représentation d'état du système continu

- ❖ On définit le vecteur d'état de la manière suivante $\underline{x} = [d \quad \dot{d} \quad h \quad \dot{h}]^T$, et on considère que la sortie du système est la vitesse de la locomotive, l'équation de la sortie est : $y = \dot{h}$.

- ✓ Définissez les équations d'état et d'observation du train électrique,
- ✓ Donnez les matrices *A, B, C* et *D* en mettant les équations sous la forme :

$$\dot{\underline{x}} = A \cdot \underline{x} + B \cdot u$$
$$y = C \cdot \underline{x} + D \cdot u$$

VII.4. Analyse du système en boucle ouverte

- ✓ Calculer les valeurs propres de la matrice *A*. Utiliser la fonction Matlab **eig**,
- ✓ Que peut-on dire sur les valeurs propres de *A* et les pôles du système ?
- ✓ A partir de ces valeurs, que peut-on dire sur la stabilité du système ?
- ✓ Dans Matlab, simuler la réponse indicielle, et la réponse impulsionnelle,
- ✓ Refaire le même travail en utilisant Simulink.

VII.4.1.1. Commandabilité et observabilité

Avant de synthétiser un système de commande avec un estimateur, il est important de savoir si la paire (A, B) est commandable, et si la paire (A, C) est observable.

En utilisant Matlab

- ✓ Calculer la matrice de commandabilité. Utiliser la fonction Matlab **ctrb**,
- ✓ Calculer la matrice d'observabilité. Utiliser la fonction Matlab **obsv,**
- ✓ Déterminer le rang des deux matrices. Utiliser la fonction Matlab **rank**,
- ✓ En déduire si le système est commandable ou pas, et observable ou pas.

V.5. Commande linéaire quadratique gaussienne à temps continu

VII.5.1. Calcul du gain de l'estimateur de Kalman K_f

- ✓ Calculer la matrice de gain d'adaptation du filtre de Kalman K_f afin de minimiser la variance de l'erreur d'estimation $\underline{\varepsilon_x} = \underline{x} - \underline{\hat{x}}$ de l'état du système en régime permanent. Utiliser la fonction Matlab **kalman**
  Avec $R_w = 10^{-4}$ **et** $R_v = 10^{-7}$

VII.5.2. Calcul de la matrice de retour d'état K

- ✓ Calculer le vecteur de gain K, en utilisant la fonction Matlab **lqr**.
  En posant $Q = C^T C$ et $R = 1$,

VII.6. Analyse du système en boucle fermée

- ✓ Montrer que la représentation d'état du contrôleur LQG peut s'écrire sous la forme suivante :

$$\begin{bmatrix} \dot{\underline{\hat{x}}} \\ \underline{u} \end{bmatrix} = \begin{bmatrix} A - BK - K_f C + K_f DK & K_f \\ -K & 0 \end{bmatrix} \begin{bmatrix} \underline{\hat{x}} \\ \underline{u} \end{bmatrix}$$

- ✓ Montrer que la représentation d'état en boucle fermée du système complet (système +estimateur) peut s'écrire sous la forme suivante :

$$\begin{bmatrix} \dot{\underline{x}} \\ \dot{\underline{\varepsilon}}_x \\ \varepsilon_y \end{bmatrix} = \begin{bmatrix} A - BK & BK & B \\ 0 & A - K_f C & 0 \\ 0 & C & 0 \end{bmatrix} \begin{bmatrix} \underline{x} \\ \underline{\varepsilon}_x \\ e \end{bmatrix}$$

Où $\underline{\varepsilon_x} = \underline{x} - \underline{\hat{x}}$ désigne l'erreur d'estimation de l'état.

- ✓ En utilisant Matlab, simuler la réponse indicielle du système en boucle fermée,
- ✓ De même, en utilisant Simulink, simuler la réponse indicielle du système en boucle fermée,
- ✓ Commenter les performances du système en boucle fermée corrigé par rapport au système en boucle fermée non corrigé.

  Remarque : si les performances du régime transitoire ne conviennent pas, ne pas hésiter à modifier la valeur de pondération R puis à recalculer K.

VII.7. Calcul direct du bloc contrôleur LQG avec Matlab

- ✓ Calculer le contrôleur LQG '$G_c(p)$ (FigureVII.3), utiliser la fonction Matlab **lqg**,
- ✓ Calculer le système complet en boucle fermée, utiliser la fonction Matlab **feedback**,
- ✓ Tester les performances du système en boucle fermée. On comparera pour cela les résultats obtenus avec le modèle complet et ceux obtenus avec le modèle linéaire, utiliser la fonction Matlab **step**.

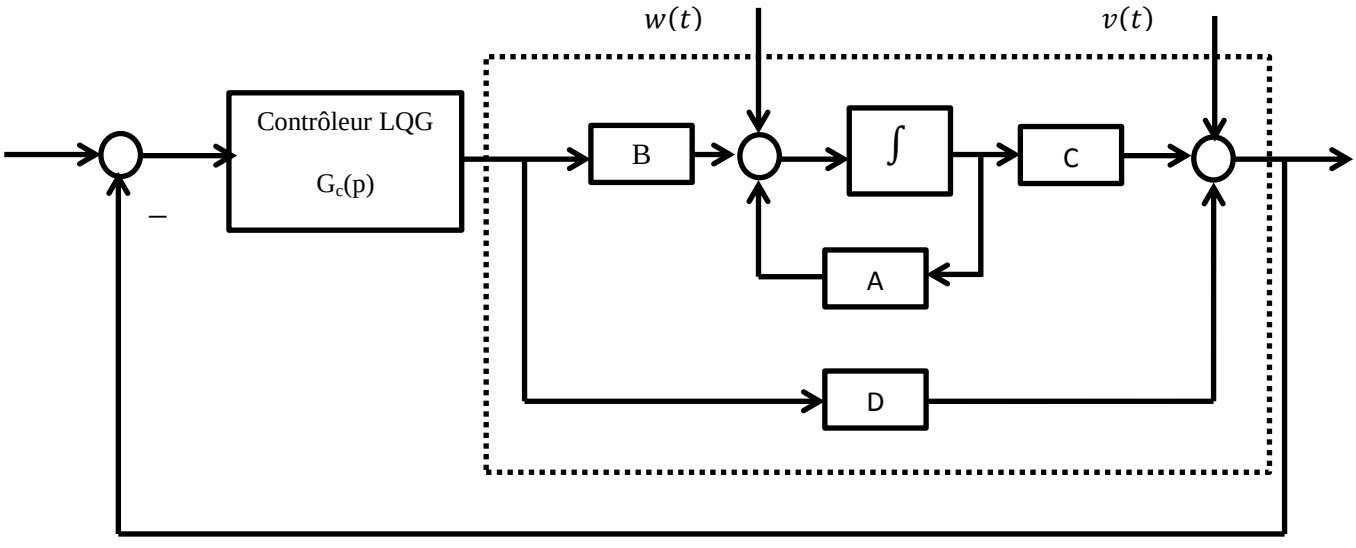


Figure VII.3 : schéma bloc du contrôleur LQG

VII.8. Commande linéaire quadratique gaussienne à temps discret

VII.8.1 Représentation d'état discrète du système

- ✓ Discrétisez le modèle d'état continu avec la fonction Matlab **c2dm** (option 'ZOH'), pour obtenir un nouveau modèle d'état discret sous la forme suivante :

$$\underline{x}(k+1) = \mathrm{F} \cdot \underline{x}(k) + G \cdot u(k)$$
$$y(k) = C \cdot \underline{x}(k) + D \cdot u(k)$$

Avec une période d'échantillonnage $T = 0.4s$

Dans le cas où la boucle de commande développée fonctionne en temps réel sur le système, on peut s'intéresser à la synthèse d'une commande par retour d'état estimé. Si on répond à un objectif de régulation, la loi de commande utilisant l'estimateur de Kalman pour estimer l'état $x(t)$ s'écrit sous la forme :

$$u(k) = -K\underline{\hat{x}}(k)$$

La figure VII.4 présente le schéma-bloc du système continu du train électrique et du retour d'état estimé

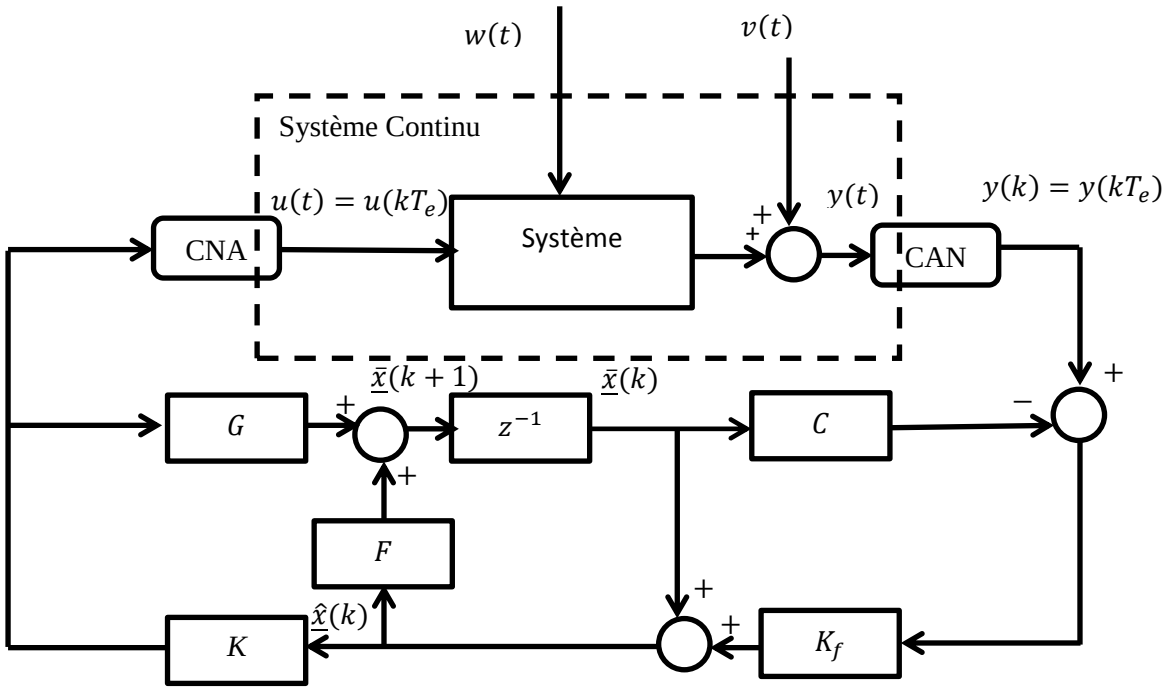


Figure VII.4 Commande par retour d'état estimé à temps discret du système continu

VII.8.1. Calcul de la matrice de retour d'état K

- ✓ Créer le système d'état échantillonné (F, G, C, D), utiliser la fonction Matlab **ss** ,
- ✓ Choisir les matrices $Q = \begin{bmatrix} Q_{11} & 0 \\ 0 & Q_{22} \end{bmatrix}$ et $R = \lambda$,
- ✓ Calculer le gain de retour d'état *K*, utiliser la fonction Matlab **dlqr.**

VII.8.2. Calcul du gain de l'estimateur discret de Kalman K_f

L'état estimé $\hat{x}(k)$ est reconstruit à partir des éléments disponibles $y(k)$ et $u(k)$, cet estimateur s'appelle le filtre de Kalman qui est défini par :

$$\underline{\hat{x}}(k) = \underline{\bar{x}}(k) + K_e \cdot \big(y(k) - C \cdot \underline{\bar{x}}(k)\big)$$
$$\underline{\bar{x}}(k) = \mathrm{F} \cdot \underline{\hat{x}}(k-1) + \mathrm{G} \cdot \mathrm{u}(\mathrm{k}-1)$$

- ✓ Trouver l'estimateur et le gain de l'estimateur K_f , utiliser la fonction Matlab **kalman (**Figure VII.5),

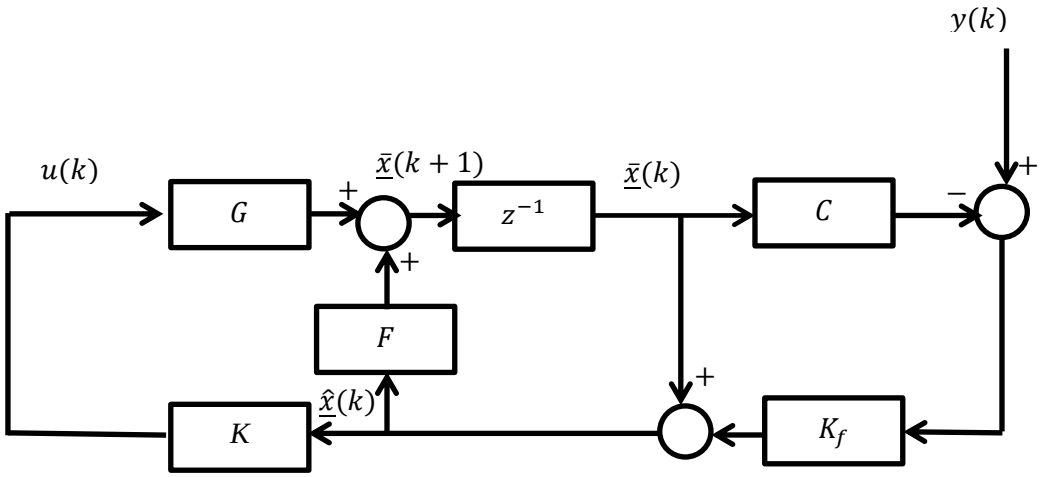


Figure VII.5 : Schéma bloc de l'estimateur de Kalman

- ✓ Simuler sous Matlab le système en boucle fermée complet, utiliser les fonctions Matlab **lqgreg** et **feedback, lsim**,
- ✓ Vérifier le rejet des perturbations additives sur l'entrée et la sortie (Impulsion de 10% de la consigne à t=10s).

Références Bibliographiques

[1]Daniel ALAZARD, ‘‘Régulation LQ/LQG ,notes de cours’’, Toulouse, campus supaero,

[2]David BENSOUSSEN, ‘’commande moderne approche par modeles continus et discrets, Ecole Polytechnique De Montreal**,**2008

[3]Antoine DROUIN « Systèmes Linéaires, Analyse moderne des systèmes dynamiques Temps discret, exemples d'application, Ecole Nationale de l'Aviation Civile - ENAC, France , Septembre 2012 .

[4] Raymond KONN ‘’commande analogique et numérique des systèmes’’ ellipses édition, 2010.

[5] Michel KREUTNER, « Travaux pratiques, Identification, commande et stabilité des systèmes »,université de Metz, 2002/2003

[6] Edouard LAROCHE, ’’cours de Commande Optimale ‘’, Ecole Nationale Supérieure de Physique de Strabourg, France2010

[7] Michel MARIE, Mohand MOKHTARI, ‘’ Application de Matlab 5 et Simulink 2, Springer, 1998.

[8] Jean François Massieu , Philipe DORLEANS ‘’Modélisation et analyse des systèmes linéaires, ellipses edition,1998.

[9]Micky RAKOTONDRABE,’’Polycopié de travaux pratiques, Commande des systèmes multivariables, CSM’’, Université de Franche-Comté à Besançon, France

[10] H. Zak STANISLAW, “Systems and Control”, School of Electrical and Computer Engineering Purdue University, New York Oxford, OXFORD UNIVERSITY PRESS, 2003

Annexe N°01 :
Quelques fonctions Matlab utiles

Lookfor	Pour trouver les fonctions relatives à un terme particulier
help	La fonction d'aide (taper :help LeNomDeLaFonction)
tf	Crée ou convertit un système sous une représentation fonction de transfert rationnelle
zpk	Crée ou convertit un système sous une représentation Zéros,Pôles, Gain.
ss	Crée ou convertit un système sous une représentation d'état
feedback	Connecte des systèmes en boucle fermée
Canon	donne la forme modale ou bien une forme canonique particulière
ss2ss	applique un changement de base donné
series	Connecte des systèmes en série (identique simple, multiplication des systèmes)
damp	Liste les fréquences naturelles ainsi que les coefficients d'amortissement associés
dcgain	Précise le gain statique
pole,eig	Liste les pôles
tzero	Liste les zéros du système
syms	Pour déclarer des variables symboliques
simplify	Pour obtenir une expression simplifiée
roots	Racines d'un polynôme
plot	représente graphiquement
grid	ajoute une grille sur la graphique courant
figure	génère une fenêtre graphique
hold on/off	permet/interdit la superposition des représentations graphiques
abs	le module
angle	la phase (d'une variable complexe)
real	la partie réelle
ctrb	Calcule la matrice de commandabilité d'un système d'état
acker	Implante la formule d'Ackermann

Jacobian	Pour dériver symboliquement des expressions
impulse(sys) ou y=impulse(sys, t)	calcul de la réponse impulsionnelle pour les instants définis par le vecteur t
step(sys) ou y=step(sys, t)	calcul de la réponse indicielle pour les instants définis par le vecteur t
y=lsim(sys, u, t)	calcul de la réponse à l'entrée u pour les instants définis par le vecteur t
bode (sys) ou bode(sys,w)	lieu de Bode (pour les pulsations du vecteur w)
nyquist (sys) ou nyquist(sys,w)	lieu de Nyquist (pour les pulsations du vecteur w)
nichols (sys) ou nichols(sys,w)	lieu de Black (pour les pulsations du vecteur w)
margin(sys)	calcul des marges de gain et de phase, ainsi que de leur pulsations associées
evalfr	Evalue la réponse à une fréquence donnée
ode23	pour intégrer numériquement des équations dynamiques continues
poly	pour obtenir les coefficients d'un polynôme à partir de ses racines
obsv	Calcule la matrice d'observabilité d'un système d'état
c2d	Transforme un objet continu en objet discret

Annexe N°02 : Modélisation du système

à n degrés de liberté

Introduction :

La modélisation mathématique est l'art de mettre en équation le comportement du système, souvent cela nécessite la connaissance des lois qui régissent le comportement des systèmes : loi de la physique, de la chimie, de l'économie.

Modélisation d'un système à n degrés de liberté par la méthode de lagrange :

Dans le cas général d'un système à plusieurs degrés de liberté, il y a autant d'équations de Lagrange que de degrés de liberté. Ainsi, si le système possède *n* degrés de liberté, il est nécessaire d'avoir *n* coordonnées généralisées $q_i \ \ (i = 1, \dots, n)$; nous aurons ainsi *n* équations de Lagrange :

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}_i}\right) - \frac{\partial L}{\partial q_i} + \frac{\partial D}{\partial \dot{q}_i} = F_i$$

$$L = E_c - E_p$$

Avec q_i : degré de liberté, $i = 1, \dots, n$

D: énergie dissipée par frottement,

F_i: force généralisée dans la direction du degré de liberté q_i,

E_c: Énergie cinétique,

E_p: Énergie potentielle.

Annexe N°03 : Commande par retour d'état

Soit un système continu décrit par l'équation d'état:

$$\begin{cases} \dot{x} = Ax + Bu \\ y = Cx + Du \end{cases}$$

Où u(t), y(t) et x(t) sont des vecteurs de dimension *m*, *l* et *n* et représentent respectivement la commande, la sortie (mesurée) et l'état du système. Les matrices *A*, *B*, *C* et *D* sont des matrices constantes de dimensions convenables.

Nous pouvons modifier tous les pôles du système si et seulement si le système est commandable, c'est-à-dire si et seulement si la paire (*A*,*B*) vérifie le critère de commandabilite

$$rang(\Phi_c) = n$$

Avec $\Phi_c = [B \quad AB \quad A^2B \quad \cdots \quad A^{n-1}B]$

Formule d'ackerman :

Elle donne directement l'expression de la matrice de retour

$$K = [0 \quad \cdots \quad 0 \quad 1]\left(\Phi_c(A,B)\right)^{-1} P_{BF}(A)$$

Avec

$$P_{BF}(A) = A^n + \alpha_{n-1}A^{n-1} + \cdots + \alpha_1 A^1 + \alpha_0 I$$

Application de la formule d'Ackerman

1. Vérifier la commandabilité de la paire (*A*, *B*)
2. Choisir les pôles correspondants au comportement désiré en Boucle Fermée BF
3. Calculer le polynôme caractéristique en BF à partir de ces pôles :

$$P_{BF}(p) = \prod_{i=1}^{n}\left(p - \lambda_{i,BF}\right) = p^n + \alpha_{n-1}p^{n-1} + \cdots + \alpha_1 p + \alpha_0$$

4. En déduire *PBF*(*A*). Appliquer la formule d'Ackerman

Annexe N°04

```
function[Nbar]=rscale(A,B,C,D,K)

s = size(A,1);
Z = [zeros([1,s]) 1];
N = inv([A, B; C, D]) * Z';
Nx = N(1:s);
Nu = N(1+s);
Nbar = Nu + K * Nx;
```