Aix*Marseille université

Faculté des Sciences

Master I&S

Régulation industrielle

Travaux Pratiques

Régulation par PI discret
Roue tournante et perturbation par freinage

Julien Marot, Rachid Outbib

2016-2017

1 Généralités

1.1 Présentation de la maquette

Le moteur faisant tourner la roue (fig. 1) est alimenté par une commande créée par Real-Time Windows Target. C'est une toolbox de Matlab qui permet de communiquer avec une machine par le biais d'une interface et qui s'utilise comme Simulink.

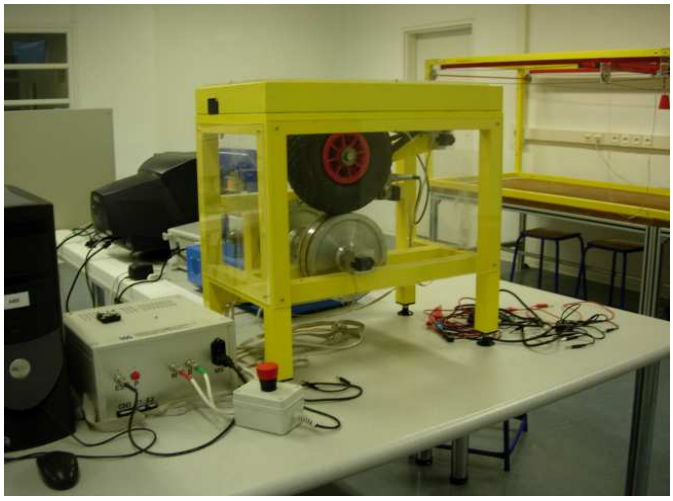

Figure 1: Système ABS

La figure 2 présente le schéma fonctionnel du dispositif expérimental. Remarquez la partie relation qui sert également d'interface de transfert et de convertisseur analogique numérique, la partie opérative qui se divise en effecteurs, perturbateur, et système physique.

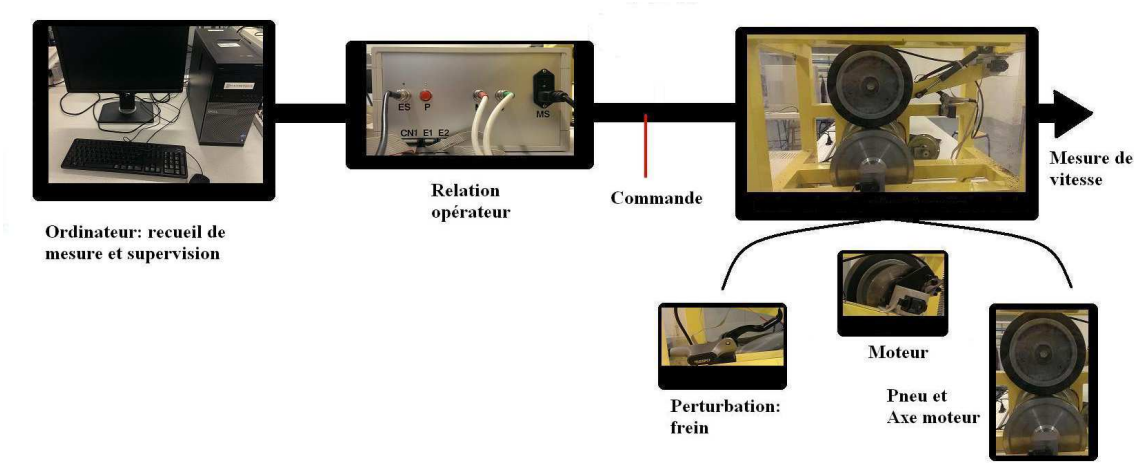

Figure 2: Schéma fonctionnel du système ABS

1.2 Objectif du TP

L'objectif est de réguler le système constitué par le moteur qui fait tourner la roue, en boucle fermée par un correcteur PI discrétisé.
On souhaite rattraper une vitesse de *consigne* appropriée. C'est un mode de fonctionnement proche de celui de l'ABS: la vitesse d'une roue ne doit pas être inférieure à la vitesse des autres roues lors du freinage.

1.3 Aspects techniques

Simulink permet de commander le moteur par le biais d'une interface. Matlab permet de traiter les données recueillies. Des programmes préremplis sont disponibles sur le site web suivant:
`https://sites.google.com/site/regulindusm1sisu3/`

- Real-Time Windows Target, Simulink : `Identification_ABS.mdl` permet de visualiser la sortie en boucle ouverte du système ABS, et de la comparer à la sortie du modèle qui sera calculé.
  `Correcteur_Discret_Pratique.mdl` permet de faire fonctionner l'ABS. Après chaque changement, il doit être sauvegardé et compilé.
  `Correcteur_Discret_Modele.mdl` contient un modèle de la fonction de transfert du système étudié.
- Matlab : `Identification_Correction_ABS.m` permet de construire et visualiser au prompt les fonctions de transfert discrètes du correcteur et du système.

2 Expression de la fonction de transfert du PI numérique

Le fichier Simulink contient un PI qui prend en entrée un signal discret. L'influence du PI sur ce signal discret est traduit par sa fonction de transfert C dont on souhaite calculer la version discrète

$C(z)$. Le PI implanté est de type mixte. La fonction de transfert $C(p)$ du PI continu mixte est:

$$C(p) = K(1 + \frac{1}{T_i p}) \tag{1}$$

où K et T_i caractérisent le PI.
Pour trouver la fonction de transfert numérique $C(z)$ du PI numérique, on applique la transformée en Z à l'association (BOZ;correcteur). C'est une méthode de discrétisation parmi d'autres:

$$C(z) = (1 - z^{-1}) * Z\{\frac{C(p)}{p}\} \tag{2}$$

Question 1: A partir des équations (1) et (2), montrez que l'on peut écrire la transmittance $C(z)$ sous la forme d'une fraction rationnelle:

$$C(z) = K_c \frac{1 - z0z^{-1}}{1 - z^{-1}} \tag{3}$$

$$BF(z) = K_m K_c \frac{1 - z0z^{-1}}{K_m K_c(1 - z^{-1}) + (1 - az^{-1})(1 - z^{-1})} \tag{4}$$

$$BF(z) = K_m K_c \frac{1 - z0z^{-1}}{\alpha + \beta z^{-1} + \gamma z^{-2}} \tag{5}$$

Question 2: Exprimez K_c et $z0$ en fonction de K et T_i.

3 Correcteur: choix des paramètres

Le choix du type de correcteur et les valeurs de ses paramètres dépendent des caractéristiques du système corrigé, qui sont déterminées dans ce TP par la méthode de Broïda (voir annexe).

En boucle ouverte, par le fichier Simulink `Identification_ABS.mdl`, donnez une consigne au système et visualisez la sortie: lancez l'expérience (triangle noir). Un ou plusieurs fichiers `data` et `data1` doivent être créés dans le workspace de Matlab.
Question 3: Appliquez la méthode de Broïda.
Question 4: Dans le programme Matlab `Identification_Correction_ABS.m`, créez la fonction de transfert continue et la fonction de transfert discrète du moteur. Vous n'avez pas à calculer "à la main" la fonction de transfert discrète du moteur ! Trouvez l'option correspondante pour la fonction `c2d.m` de Matlab.
La méthode de Broïda conduit à des paramètres préconisés pour la régulation PID (voir tableau ci-dessous). Le rapport entre les paramètres de constante de temps et de retard conduisent au choix du régulateur: soit P, soit PI, soit PID.

θ/τ	$\cdot > 20$	$10 < \cdot < 20$	$5 < \cdot < 10$	$2 < \cdot < 5$	$\cdot < 2$
Type de régulation	TOR	P	PI	PID	Autre

Question 5: Le choix du PI est-il justifié ?
Question 6: On souhaite par le PI compenser le pôle lent du système: déduisez-en la valeur de T_i.
On souhaite obtenir un système en boucle fermée qui soit critique. On peut obtenir la valeur du

gain du correcteur en utilisant le principe suivant: la fonction de transfert d'un système critique présente un pôle double. Cela conduit à la valeur de gain $K_c = \frac{\theta}{4*\tau*K_m}$ (vous n'avez pas à la calculer).
Question 7: Définissez les valeurs de Ti et K_c dans le programme Matlab `Identification_Correction_ABS.m`, et créez la fonction de transfert du correcteur (équation (4)). Pour cela, écrivez directement la fonction de transfert en fonction de z^{-1}.
Question 8: Avec la fonction Matlab `zpk.m` mettez cette fonction de transfert sous la forme zéros, pôle, gain.
Question 9: Comparez le résultat que vous obtenez ainsi au résultat donné par `c2d.m`.
Pour effectuer une régulation PI, un système doit être rebouclé. Le rebouclage est fait par le fichier Simulink `Correcteur_Discret_Pratique.mdl`, qui contient donc un soustracteur en plus d'un régulateur PI.
Question 10: Reportez les pôles, les zéros, et le gain de la fonction de transfert dans un bloc `Discrete Zero-Pole` de simulink. Calculez la fonction de transfert du système corrigé bouclé sur le programme Matlab. Implantez-la sur le fichier `Correcteur_Discret_Modele.mdl`.

Remarque: pour que le moteur fonctionne de façon linéaire, la commande appliquée ne doit pas saturer !
Question 11: D'après les spécifications du moteur, les valeurs de la commande doivent rester dans l'intervalle `[0:2.5]`. Quel élément de Simulink permet de s'assurer de cela ?
Question 12: Créez sous Simulink le régulateur discret que vous avez calculé avec Matlab. Visualisez par un scope le signal avant et après l'élément que vous venez de placer. Vérifiez qu'il n'y a pas de non-linéarité dans la commande envoyée au moteur.

Question 13: Comparez les signaux obtenus avec les fichiers `Correcteur_Discret_Pratique.mdl` et `Correcteur_Discret_Modele.mdl`. Obtenez-vous le résultat souhaité, *i.e.* une erreur statique nulle ? D'où peuvent provenir les différences observées ?

3.1 Influence d'une perturbation

Sur le fichier `Correcteur_Discret_Modele.mdl`, introduisez une perturbation: un échelon qui passe de 0 à -200 après 10 secondes (élément step de simulink).
Question 14: Où faut-il placer cet échelon pour simuler un freinage ? Vérifiez que le PI permet au système bouclé de s'affranchir de cette perturbation.
Par une expérience réalisée avec le fichier `Correcteur_Discret_Pratique.mdl`, appliquez un freinage mécanique à la roue, observez le comportement du système bouclé.
Dans les deux cas, visualisez la commande avec un scope.
Question 15: Comment évolue la commande lorsqu'on freine ?
Question 16: Quel est le paramètre dominant dans le comportement d'un correcteur PI ?
Question 17: Comment réagit à une perturbation un système corrigé par un PI ?
Que faut-il contrôler pour rester dans un domaine de fonctionnement linéaire du système régulé ?

3.2 Comparaison de deux méthodes de discrétisation

On souhaite comparer le résultat de discrétisation obtenu par l'association (BOZ;correcteur) d'une part et la transformée bilinéaire d'autre part. La transformée bilinéaire consiste à remplacer dans la fonction de transfert $C(p)$ la variable p de la façon suivante:

$$p \rightarrow \frac{2}{T_e}\frac{z-1}{z+1} \tag{6}$$

Remarquez que $C(z)$ s'exprime, comme lorsqu'un BOZ est utilisé, de la façon suivante:

$$C(z) = K_c\frac{1-z0z^{-1}}{1-z^{-1}} \tag{7}$$

Question 18: Dans quelles conditions les deux méthodes de discrétisation tendent-elles à être équivalentes ?

Annexe: Identification par la méthode de Broïda

On suppose que l'entrée échelon commence au temps t=0.
Soit `umax` l'amplitude de l'échelon.
Soit `ymax` la valeur asymptotique de la sortie.
Soit `t1` tel que `y(t1)=0.28*ymax`, et `t2` tel que `y(t2)=0.40*ymax`.
Le système est modélisé comme un système du premier ordre retardé, de gain K, de temps caractéristique θ, et de retard τ : $G(p) = e^{-\tau p}\frac{K}{1+\theta p}$
avec $K = ymax/umax$, $\theta = 5.5 * (t2 - t1)$, $\tau = 2.8 * t1 - 1.8 * t2$.

Aix*Marseille université

Faculté des Sciences

Master I&S

Régulation industrielle

Travaux Pratiques N°1

Régulation discrète PID de la température d'une pièce
par automate programmable Millenium 3

Julien Marot, Thibaut Raharijaona, Habib Nasser 2013-2014

1 Généralités

1.1 Présentation de la maquette

La maquette que vous allez utiliser sert à une application de domotique. Elle contient deux boîtes, concernant l'éclairage et le chauffage. On s'intéresse, dans ce TP, au chauffage. La boîte (blanche) à laquelle nous nous intéressons contient une petite ampoule et un capteur de température. La Figure 1 représente le dispositif expérimental.

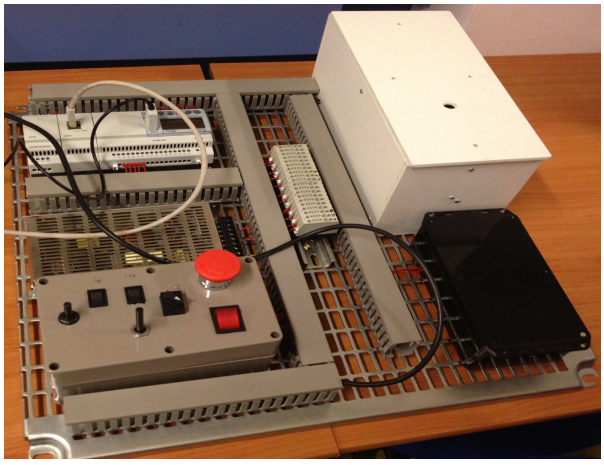

Figure 1: Maquette de domotique: automate programmable

1.2 Objectif du TP

L'objectif principal de ce TP est de réguler la température de la pièce en contrôlant la tension appliquée aux bornes d'une lampe chauffante.

Pour cela vous utiliserez trois logiciels.

1.3 Aspects techniques

Les trois logiciels utilisés sont les suivants:

- Crouzet Logic Software Millenium 3 : crée des diagrammes sous la forme de fichiers `.pm3` incluant des icônes graphiques qui sont reliées aux entrées et sorties de l'automate;
- SmartConfig : permet d'enregistrer des données transférées par l'automate via son port Ethernet sous la forme d'un fichier `.csv`;
- Matlab : permet d'exploiter les données écrites par l'automate dans le fichier `.csv`.

L'environnement de programmation principal est le logiciel Crouzet Logic Software M3 :

Lancez le logiciel et créez un programme que vous nommerez `Regulation_Temperature_Piece.pm3`. L'automate est équipé d'une extension "XA04" qui permet la commande analogique, et d'une extension "XN05" qui permet le transfert de données via un port ethernet. Lors de la création du programme vous devez donc les inclure. Choisir:

- l'automate XD 26 de référence 88970161;
- l'extension XA04 24VDC;
- l'extension XN05 24VDC;
- le langage FDB (Functional Block Diagram) comme type de programmation (pas le langage à contact (LADDER)).

Le fichier `Regulation_Temperature_Piece.pm3` qui conditionne le fonctionnement de la maquette est maintenant créé.

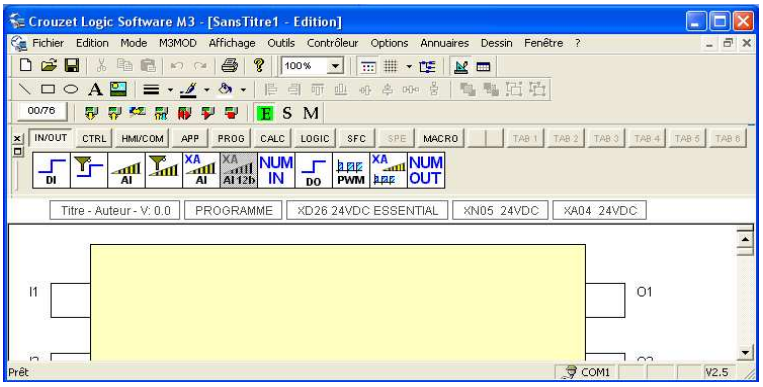

Figure 2: Fichier pm3 vierge

Vous utilisez l'extension **XN05** pour enregistrer et visualiser la mesure de température délivrée par la sonde. Sur le logiciel Logic Software M3, paramétrez l'extension XN05. Pour cela, cliquez sur le rectangle associé en lui attribuant l'adresse IP (statique) de la maquette que vous utilisez. plusieurs adresses IP sont disponibles:
Adresse IP 1 : 172.17.106.210
Adresse IP 2 : 172.17.106.211
Adresse IP 3 : 172.17.106.212
etc.

Vérifiez que celle que vous voulez utiliser est disponible: démarrer>exécuter>cmd OK.
tapez ping 172.17.106.212 (par exemple). Si le délai d'attente est dépassé, c'est que l'adresse est libre et que vous pouvez la choisir (voir Fig. 3). Précisément vous devez rentrer: 172.17.106.212 pour l'adresse IP et de passerelle, et 255.255.255.0 pour le masque de sous réseaux.

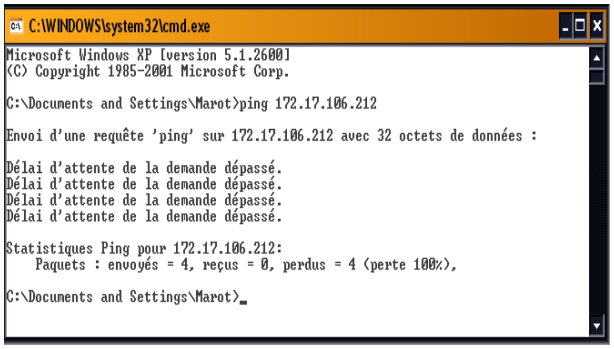

Figure 3: Test de disponibilité d'une adresse IP

Vous utiliserez l'extension **XA04** pour alimenter l'ampoule. Sur le logiciel Logic Software M3, paramétrez l'extension XA04 pour qu'elle produise un signal PWM. Pour cela, cliquez sur le rectangle XA04 24VDC, puis choisissez PWM dans l'onglet "extension analogique".
Voici quelques éléments de base de la programmation avec le logiciel Logic Software M3:
Les entrées: les entrées I1 à IA sont dites "numériques". En fait elles prennent deux valeurs: 0 ou 1, codées sur un bit. Les entrées IB à IG sont dites "analogiques". En fait elles sont aussi numériques mais codées sur 10 bits et non sur un seul bit.
Les blocs fonctions: on insère une fonction en la faisant glisser. Par exemple le "ou" logique (LOGIC).
Les sorties: Les sorties 01 à OA sont numériques (allumé ou éteint), les sorties OFXA et OGXA simulent une sortie "analogique" dans le sens où elles peuvent accepter une valeur entre 0 et 1023 (elles correspondent à l'extension XA04). Les sorties "ETH" correspondent au port ethernet.
Question 1: Qu'est-ce qui distingue les connexions vertes des connexions noires ?

2 Programme de régulation

Complétez le fichier `Regulation_Temperature_Piece.pm3` avec le logiciel Crouzet:

2.1 Eléments pour la mise en route

Placez en 03 une sortie numérique " digital output " (IN/OUT) à laquelle vous pouvez donner l'aspect d'une ampoule. Elle est reliée au chauffage. Placez en O4 une sortie numérique " digital output " (IN/OUT) à laquelle vous pouvez donner l'aspect d'un ventilateur.

Sur I2 et I3, placez l'entrée numérique "digital input" (IN/OUT). Reliez I2 à O3 (lampe chauffante) et I3 à O4 (ventilateur).

Il existe trois modes d'utilisation du logiciel: écriture E, simulation S, monitoring M. Faites un premier essai de simulation S en allumant la lampe de la sortie O3.

2.2 Mesure de température

Aidez-vous du schéma de la Fig. 2.3.
Afin de mesurer la température ambiante dans la pièce, on y a placé un capteur qui fonctionne dans une gamme de $T = -10$ à 40 degrés Celsius, et qui donne une sortie V entre 0 et 10 Volts. Le capteur est relié à la broche IC de l'automate. Cette mesure est donnée sous la forme numérique d'un nombre NUM codé sur 10 bits.

Question 2: trouvez la relation qui donne la température T en fonction de cette valeur numérique NUM. On affichera la température multipliée par 10. Complétez en fonction de cela l'élément Gain (CALC). Réalisez le branchement de la figure 2.2.

Remarquez l'élément Display. Il vous permettra de visualiser la température à l'intérieur de la boîte en temps réel. Les éléments NUM OUT permettront par la suite de transférer des données sur l'ordinateur.

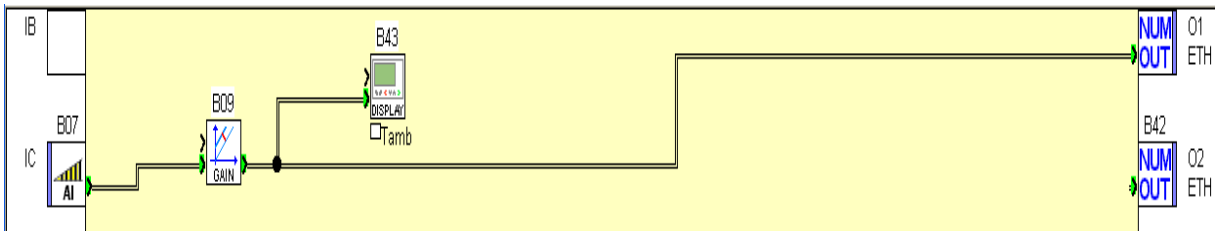

Figure 4: Programme pour la mesure de température

Faites une mesure de température par les actions suivantes:

- brancher le câble au port USB COM 1 et à l'automate;
- écrire (flèche jaune vers le haut);
- monitoring (M);
- lecture (rouge).

Question 3: la température s'affiche-t-elle sur l'écran de l'automate ? Le facteur de multiplication correspond-il au gain que vous avez choisi ?

2.3 Régulation PID

Poursuivez la construction de votre programme: modifiez le pour mettre en oeuvre une régulation PID.

Pour cela, placez les éléments PID (APP), Gain (CALC), Display (HMI/COM), réalisez le schéma de branchement pour effectuer la régulation.

L'élément ADD/SUB (CALC) permet d'ajouter une constante à la valeur maximum permise par le sélecteur.

L'utilité de la constante à ajouter à la consigne de température est la suivante: L'entrée ID mesure la tension donnée par le sélecteur. La lampe chauffante est alimenté par une tension à 0 à 10 Volts.
Question 4: Quelle est la valeur numérique TEMP donnée par l'entrée thermomètre ?

Question 5: Déduisez-en la tension délivrée par le sélecteur de votre maquette.

Question 6: Quelle valeur numérique NUM faut-il ajouter pour atteindre la consigne de 35°?

Un élément Gain après la sortie du PID multiplie par un facteur cette sortie. Celle-ci atteint une valeur max de 255.

Question 7: Quel facteur multiplicatif choisit-on, sachant que la valeur de commande donnée par OF XA est codée sur 10 bits ?

Sur la figure 2.3 sont présentés les éléments nécessaires à la mise en oeuvre du PID. A vous de faire les câblages adéquats.

On assimile le système constitué par la pièce à chauffer à un système d'ordre 1 retardé, de fonction de transfert:

$$G(p) = e^{-\tau p}\frac{K}{(1+Tp)} \tag{1}$$

avec $T \simeq 150$ sec. et $\tau \simeq 5$ sec. La valeur du gain est: $K = 0.02$. Ce informations sont importantes pour effectuer une bonne régulation: on ne régule pas de la même façon un système lent et un système rapide.

Le PID implanté dans le logiciel Millenium 3 est de type mixte. Sa fonction de transfert est la suivante:

$$C(p) = K_p(1 + \frac{1}{T_i p} + T_d p) \tag{2}$$

où K_p, T_i, T_d caractérisent le PID.
Lorsque K_p augmente, le système est plus rapide mais plus instable (plus les variations de sortie du système sont brutales). Lorsque $1/T_i$ augmente, le temps de montée est plus court mais il y a un dépassement plus important. Lorsque T_d augmente, le temps de montée change peu mais le dépassement diminue (le système est stabilisé).

Pour que le système soit stable, on choisit K_p proche de sa valeur minimale. Pour qu'il soit rapide, on choisit T_i du même ordre de grandeur que T, mais inférieur pour éviter les dépassement. On souhaite avoir un dépassement très faible ou pas de dépassement, donc on choisit T_d d'un ordre de grandeur au-dessus de τ mais d'un ordre de grandeur en dessous de T.

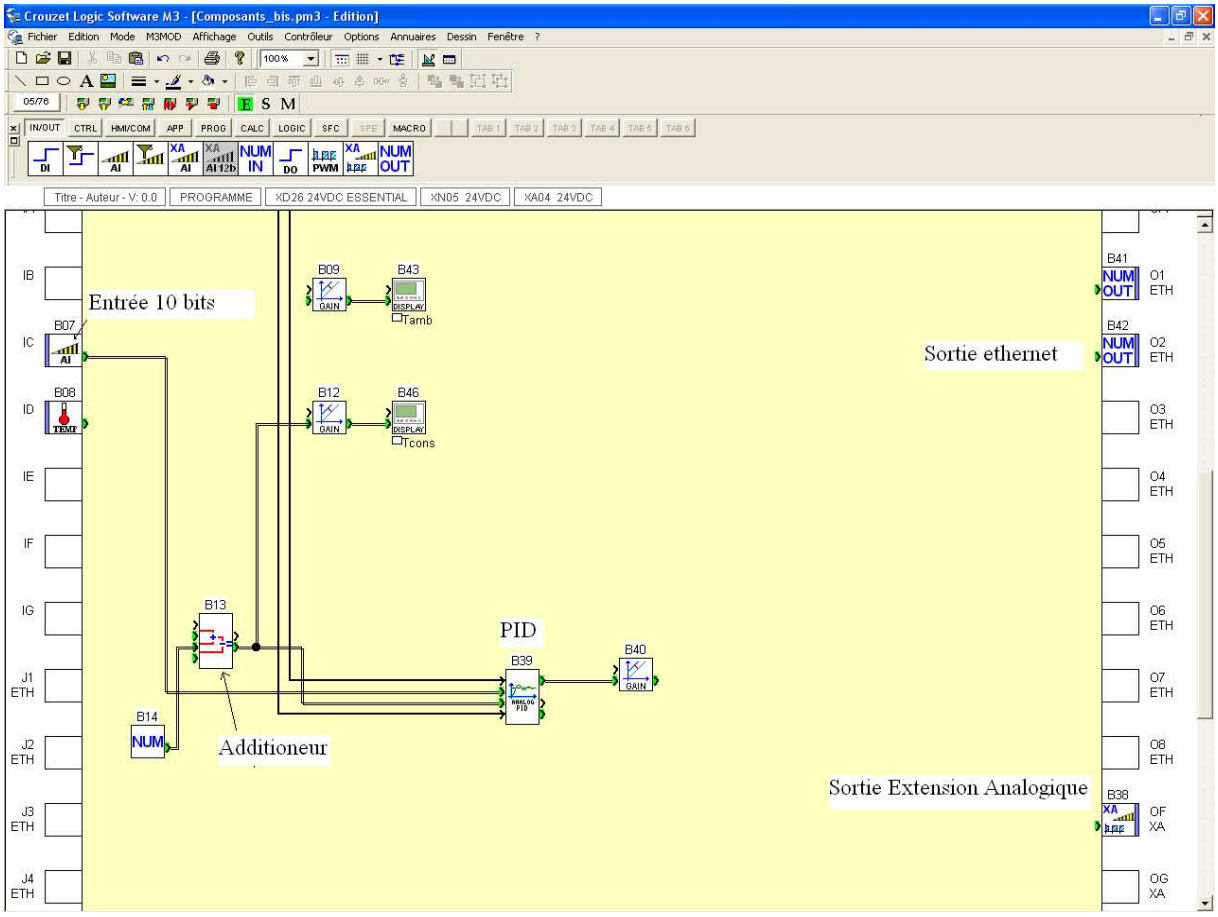

Figure 5: Eléments nécessaires à la mesure et à la régulation

Question 8: Quelles valeurs de K_p, T_i et T_d préconisez-vous ?
La régulation est réalisée à l'aide de la sortie analogique OFXA. La commande envoyée à la sortie OFXA est proportionnelle à l'écart $\epsilon(t) = T_{consigne} - T_{mesure}$ avec $T_{consigne} = 35°$.
Lors de l'expérience, si les valeurs de consigne varient trop, choisissez $T_d = 0$, et approchez la valeur du gain de la valeur minimale autorisée.

Question 9: Quel élément manque-t-il en apparence pour réaliser un rebouclage ? Où se trouve implicitement cet élément ?
Question 10: Quel est le temps de montée à 5% du système ? Comment varie-t-il si on augmente le gain du correcteur ? Vous est-il possible d'augmenter $1/T_i$ sans risquer un dépassement ? Si oui, comment évolue le temps de montée du système ? Vérifiez que la température atteint la consigne. Etudiez la robustesse du système régulé à des perturbations: augmentez pendant un court instant la consigne. Ramenez-la au niveau précédent puis faites tourner à faible vitesse le ventilateur.
Question 11: La consigne est-elle rattrapée ?

Annexes

Transfert de données

Mettez en oeuvre un protocole de transfert de données avec Smartconfig. L'enseignant doit vous fournir un document annexe vous expliquant ce protocole.

1. Lancez par double clic le fichier `.saf` que vous avez créé;
2. Paramétrez l'adresse IP adéquate sur SmartConfig > configure connection > en face de ETHERNET 172.17.106.210:502. L'adresse doit correspondre à celle que vous aviez donnée en paramétrant le XN05.
3. Vérifier que les registres de sortie qui recueillent les mesures de température sont bien pris en compte: sur SmartConfig > Donnée `ETH_OUT1_l1` et `ETH_OUT2_l1` sont bien présents dans la liste des registres ?
4. Paramétrez l'adresse IP adéquate sur SmartCommand: sur SmartConfig > Outils > Aller à SmartCommand (voir Fig; 2.3), puis sur SmartCommand > configure connection > en face de ETHERNET 172.17.106.210:502. L'adresse doit correspondre à celle que vous aviez donnée en paramétrant le XN05.

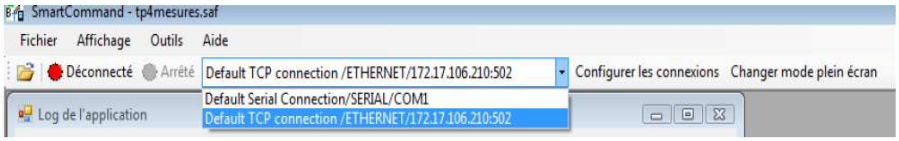

Figure 6:

Quittez SmartCommand (Fichier>Quitter).

Enregistrement d'une expérience

Lisez cette sous-section en entier. Vous répéterez deux fois la démarche qui y est décrite, une fois en boucle ouverte, une fois en boucle fermée pour la régulation.

Démarrage d'une expérience:

1. Faire tourner en monitoring la maquette. Appuyer sur lecture. Ne pas allumer la lampe pour l'instant.
2. Sur SmartConfig > Outils > Aller à SmartCommand.
3. Lancez l'enregistrement sur SmartCommand: Cliquez sur déconnecté et arrêté; les cases passent au vert ! On est prêt à visualiser la montée en température.

4. Ensuite sur Crouzet Logic Software: allumez la lampe et donnez une consigne de température (entrée thermomètre).

Remarque 2: l'étape 1 doit toujours être réalisée avant l'étape 2

Arrêt d'une expérience:

- Sur SmartCommand: appuyez sur en marche et sur connecté pour passer en mode "Déconnecté".
- Sur Crouzet Logic Software: appuyez sur stop.

Le fichier de mesures `mesures.csv` dans le répertoire `C:\TEMP` a dû être créé.
Sinon, en cas de problème: Si rien ne s'enregistre par SmartCommand: redéfinir l'adresse IP sur l'extension XN05 et sur SmartCommand. Vérifiez sur SmartConfig que les registres `ETH_OUT_I1` et `ETH_OUT2_I1` sont bien référencés. Si vous n'arrivez pas à établir la connexion, changer l'adresse IP dans le logiciel Crouzet millenium 3 et dans SmartCommand et SmartConfig (... 219 par exemple).

Exploitation des données avec Matlab

Les données de consigne et de température mesurée sont contenues dans le fichier `mesures.csv`. Prenez soin d'en supprimer la première ligne (en ouvrant le fichier avec un éditeur de texte). Créez le fichier `RegulationAutomate.m`, complétez-le avec les lignes suivantes et avec les lignes nécessaires pour l'affichage:

```
load mesures.csv
temperature=mesures(:,4)/10;
consigne=mesures(:,5);
```

Aix*Marseille université

Faculté des Sciences

Master I&S

Régulation industrielle

Travaux Pratiques

Régulation par PID discret sous Real-Time Windows Target
Moteur à courant continu Tergane 25

Julien Marot, Rachid Outbib

2016-2017

1 Généralités

1.1 Présentation de la maquette

La maquette est constituée du moteur à courant continu, d'un gain le précédant, et d'un rebouclage qui peut être ouvert. Elle est alimentée par une source externe, que vous commanderez lors de ce TP par Real-time windows target. *Attention : la maquette ne contient pas de régulateur PID intégré, il faut donc utiliser un régulateur créé sur l'ordinateur.*

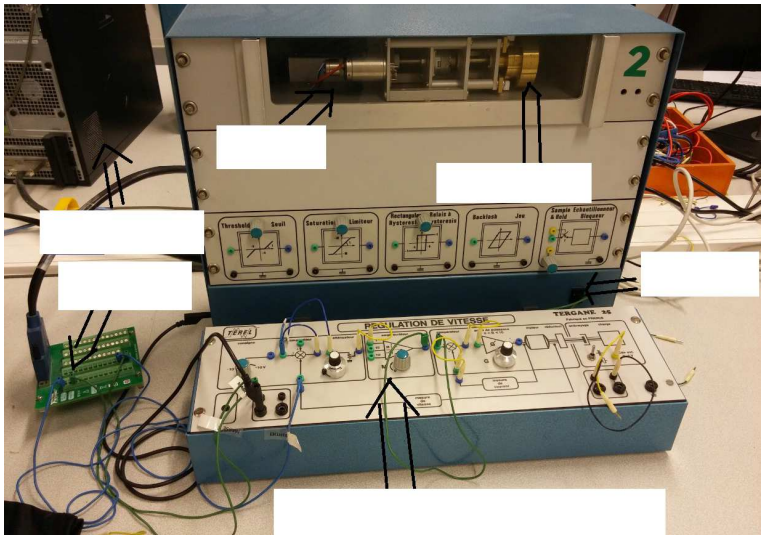

Figure 1: Tergane 25: éléments à compléter

Question préliminaire:

Complétez les blancs de la figure 1, et inspirez-vous de cette figure pour réaliser un schéma fonctionnel du dispositif expérimental (sous forme de dessin Paint-Brush). On doit y voir apparaître notamment la partie opérative avec les effecteurs et leur dénomination, le système physique, et l'interface de transfert (c'est-à-dire le convertisseur analogique numérique).

1.2 Objectif du TP

L'objectif est de réguler le système en boucle fermée par un correcteur PID discrétisé. Les paramètres de ce PID seront réglés par la méthode de Broïda, mais aussi en prenant en compte les contraintes de saturation de la commande du moteur.

1.3 Aspects techniques

L'acquisition des données est réalisé par Real-Time Windows Target. La visualisation des données est réalisée par Matlab.

Les fichiers que vous utiliserez et que vous aurez à compléter sont les suivants:

- Real-Time Windows Target : `Boucle_Ouverte_Tergane_25.mdl` permet d'étudier le système en boucle ouverte en vue de le caractériser; `Regulation_Tergane_25_PID.mdl` permet d'effectuer une régulation PID du Tergane 25;
- Matlab : `Identification_Regulation_Tergane_25.m` permet de calculer les paramètres du système à étudier et de calculer les paramètres du PID. Il permet aussi de confronter la réponse obtenue par le moteur en boucle fermée, et sa réponse théorique en fonction de la fonction de transfert discrète du système bouclé.

Demandez à l'enseignant de vous fournir ces fichiers.

Par ailleurs, les fichiers cités ci-dessus sont disponibles sur le site web suivant:

`https://sites.google.com/site/regulindusm1sisu3/` (Tergane 25 RTWT)
Il peut être nécessaire de les adapter selon l'interface de transfert (National Instruments, etc.) utilisée.

2 Fonction de transfert du PID numérique

Le fichier `Regulation_Tergane_25_PID.mdl` contiennent un PID qui prend en entrée un signal discret. L'influence du PID sur ce signal discret est traduite par sa fonction de transfert $C(z)$ que l'on souhaite calculer. Le fichier qui vous est fourni contient un PID mixte. La fonction de transfert $C(p)$ du PID continu mixte est:

$$C(p) = K(1 + \frac{1}{T_i p} + T_d p) \tag{1}$$

où K, T_i, T_d caractérisent le PID.
Pour trouver la fonction de transfert numérique $C(z)$ du PID numérique, on applique une transformée bilinéaire à la fonction de transfert continue. C'est une méthode de discrétisation parmi d'autres:

$$p \rightarrow \frac{2}{T_e}\frac{z-1}{z+1} \tag{2}$$

Question 1: A partir des équations (1) et (2), montrez que l'on peut écrire la transmittance $C(z)$ sous la forme d'une fraction rationnelle:

$$C(z) = \frac{c_0 + c_1 z^{-1} + c_2 z^{-2}}{(1 - z^{-1})(1 + z^{-1})} \tag{3}$$

Question 2: Exprimez c_0, c_1, c_2, en fonction de K, T_i, T_d.

On souhaite maintenant créer ce correcteur PID discret, sous real-time windows target (en procédant de la même façon qu'avec simulink).

3 Correcteur: choix des paramètres

Le choix du type de correcteur et les valeurs de paramètres du correcteur dépendent des caractéristiques du système à corriger. Ces caractéristiques sont déterminées dans ce TP par la méthode de Broïda (voir annexe). La maquette Tergane 25 contient un moteur à courant continu qui peut être modélisé par un système d'ordre 1 retardé, ou par un système d'ordre 2 car le retard du système est très faible devant sa constante de temps.

Mise en oeuvre:
En boucle ouverte, donnez une consigne au système et enregistrez la sortie par le fichier RTWT `Boucle_Ouverte_Tergane_25.mdl`:

Décrivez brièvement dans votre compte-rendu les éléments du programme. Les branchements adéquats doivent normalement être déjà faits. L'entrée de la carte est connectée à la sortie du système, la sortie à l'entrée du système non bouclé. Demandez à l'enseignant responsable une vérification.

Le fichier `Boucle_Ouverte_Tergane_25.mdl` a besoin d'une valeur de paramètre, la période d'échantillonnage T_e pour pouvoir être compilé et pour fonctionner.

Dans le fichier `Identification_Regulation_Tergane_25.m`, donnez une valeur faible de la période d'échantillonnage:
`Te=1E-4`
Lancez le fichier le fichier `Identification_Regulation_Tergane_25.m`.
Compilez le fichier `Boucle_Ouverte_Tergane_25.mdl`. Réalisez la connection à la cible puis lancez l'expérience (flèche noire).
Visualisez l'entrée et la sortie sur le scope.
Appliquez la méthode de Broïda.

Question 3: La valeur de période d'échantillonnage choisie est-elle suffisamment faible pour que la condition de Shannon soit respectée ?

La méthode de Broïda préconise certaines valeurs de paramètres pour la régulation PID (voir tableau ci-dessous). Le rapport entre les paramètres de constante de temps et de retard conduisent au choix du régulateur: soit P, soit PI, soit PID. C'est une méthode très générale qui peut être adaptée au cas par cas.

θ/τ	$\cdot > 20$	$10 < \cdot < 20$	$5 < \cdot < 10$	$2 < \cdot < 5$	$\cdot < 2$
Type de régulation	TOR	P	PI	PID	Autre
K_p		$\frac{0.8\theta}{K_m\tau}$	$\frac{0.8\theta}{K_m\tau}$	$\frac{\theta/\tau+0.4}{1.2K_m}$	
T_i			$\frac{K_m\tau}{0.8}$	$\theta+0.4\tau$	
T_d				$\frac{\theta\tau}{2.5\theta+\tau}$	

Pour effectuer une régulation PID, un système doit être rebouclé. Cela est fait par le fichier `Regulation_Tergane_25_PID.mdl`.
Question 4: Complétez le fichier `Regulation_Tergane_25_PID.mdl` avec un PID.

Question 5: Les branchements sur la maquette sont-ils toujours adéquats ?
Sur le fichier `Regulation_Tergane_25_PID.mdl` placez des scopes pour visualiser la commande.

Lancement de l'expérience et précautions de réglage du PID:

donnez une valeur plus élévée à la période d'échantillonnage (sinon Matlab ne supporte pas la charge de calcul):
`Te=1E-3`

Question 6: Pourquoi ce choix pour la période d'échantillonnage ?

Essayez tout d'abord une régulation PI en supprimant la partie dérivateur ($T_d = 0$). Prenez ensuite la valeur calculée précédemment pour T_d.

Remarque 1: La commande du moteur ne doit pas saturer !
Vous devez donc visualiser les valeurs de commande, et vérifier qu'elles ne sortent pas de l'intervalle -10:10.

Vous choisirez une valeur faible de gain sur la maquette Tergane 25, que vous augmenterez progressivement, jusqu'à atteindre la limite de saturation.

4 Simulation

Sur le fichier Matlab `Identification_Regulation_Tergane_25.m`:
Vous créerez la fonction de transfert discrétisée de l'ensemble correcteur-moteur. Vous utiliserez la fonction Matlab `c2d.m`, avec comme troisième option `'tustin'`, transformée bilinéaire.
Question 7: Quelle démarche faut-il adopter théoriquement ? On discrétise le correcteur et le moteur successivement, puis on multiplie les deux résultats ? Ou on multiplie les deux fonctions de transfert continues, et on discrétise le résultat de la multiplication ?
Question 8: Comparez les signaux obtenus sous Matlab avec lsim et ceux obtenus avec RTWT. Sont ils ressemblants ?
Question 9: Effectuez la démarche suivante: discrétisez séparément le correcteur et le moteur. La forme obtenue pour la fonction de transfert du correcteur PID est-elle la même que la forme proposée par l'équation (3) ? Pourquoi ?

Créez la fonction de transfert discrète égale au produit des deux fonctions de transfert discrètes. Utilisez à nouveau la fonction lsim pour étudier le comportement de cette fonction de transfert.
Question 10: Le signal de sortie simulé ressemble-t-il au signal obtenu avec RTWT ? Faites quelques commentaires.

Question 11: Le résultat obtenu sans dérivateur est-il satisfaisant ? Quel est normalement le rôle du dérivateur ? Si le résultat obtenu est satisfaisant, pourquoi n'a-t-on pas besoin du dérivateur ?

Question 12: Refaites l'expérience de régulation en changeant la fréquence d'échantillonnage. La période d'échantillonnage a-t-elle une influence sur la régulation ?

5 Conclusion, commentaires

Question 13: Quel est le paramètre dominant dans le comportement d'un correcteur PID ?
Question 14: Que faut-il contrôler pour rester dans un domaine de fonctionnement linéaire ?

Annexe: Identification par la méthode de Broïda

On suppose que l'entrée échelon commence au temps t=0.
Soit `umax` l'amplitude de l'échelon.
Soit `ymax` la valeur asymptotique de la sortie.
Soit `t1` tel que `y(t1)=0.28*ymax`.
Soit `t2` tel que `y(t2)=0.40*ymax`.

Le système est modélisé comme un système du premier ordre retardé, de gain K, de temps caractéristique θ, et de retard τ: $G(p) = e^{-\tau p}\frac{K}{1+\theta p}$ Tels que: $K = ymax/umax$, $\theta = 5.5 * (t2 - t1)$, $\tau = 2.8 * t1 - 1.8 * t2$.

Aix*Marseille université

Faculté des Sciences

Master I&S

Régulation industrielle

Travaux Pratiques

Moteur triphasé
Identification par la méthode de Streicj
Régulation par PI discret

Julien Marot, Rachid Outbib 2016-2017

Le support d'étude de ce TP est un moteur triphasé asynchrone. Le moteur triphasé présente l'intérêt suivant: il permet la création d'un champ magnétique tournant, utile pour les moteurs de forte puissance. Le moteur tourne grâce à des bobinages disposés à 120 degrés les uns des autres. Les machines asynchrones sont utilisées aujourd'hui dans de nombreuses applications, notamment dans le transport (métro, trains, propulsion des navires), dans l'industrie (machines-outils), dans l'électroménager. Pour le pilotage de ces moteurs, il est impératif de séparer la tension de commande (basse) de la tension de puissance (élevée). La tension de commande doit être en basse tension car opérateur humain est amené à intervenir. C'est pour cela qu'il est nécessaire de différencier physiquement sur un circuit ces deux tensions. Ainsi sur le moteur étudié pendant le TP, *la tension de puissance est d'environ 400 V, mais la tension de commande est obligatoirement inférieure ou égale à 10 Volts en valeur absolue.*

1 Introduction, présentation de la maquette

Le moteur est alimenté par une commande, que l'on peut envoyer à partir de Real-Time Windows Target et d'une carte National Instruments. *Attention : la maquette ne contient pas de régulateur PI intégré, il faut donc utiliser un régulateur créé sous Real-Time Windows Target.*

La Figure 1 représente le dispositif expérimental.

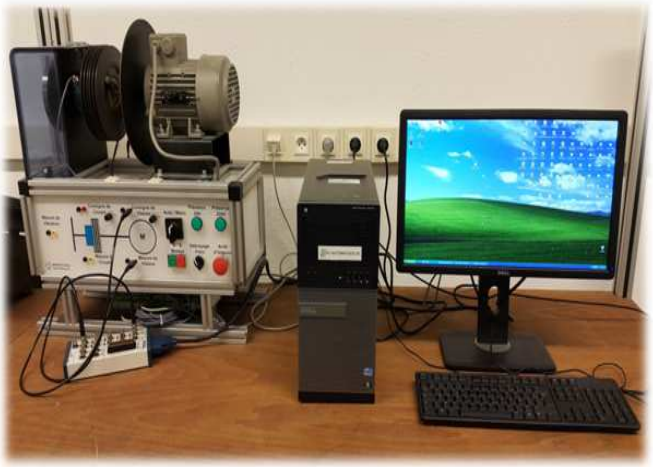

Figure 1: Moteur asynchrone

La Figure 2 représente le schéma fonctionnel du dispositif expérimental. Remarquez les bobinages, le convertisseur analogique numérique/convertisseur numérique analogique.

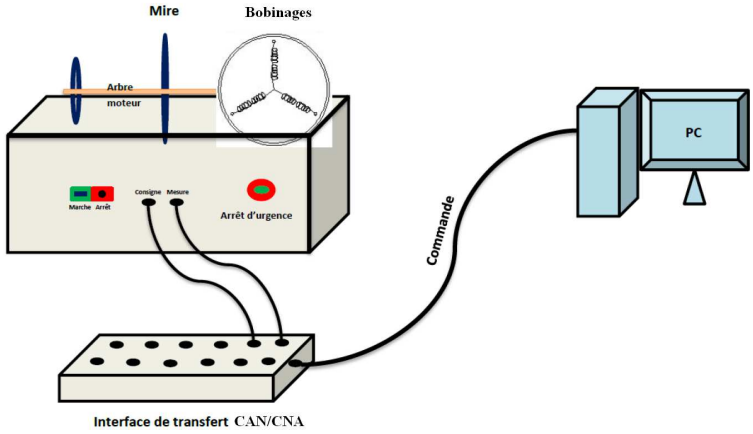

Figure 2: Moteur asynchrone: schéma fonctionnel

1.1 Objectif du TP

L'objectif est de réguler le système en boucle fermée par un correcteur PI discrétisé. Les paramètres de ce PI implanté sous Real-Time Windows Target seront réglés d'après la méthode de Ziegler et Nichols selon une identification en boucle ouverte de type Streicj, mais aussi en prenant en compte les contraintes de saturation de la commande du moteur.

1.2 Le protocole d'acquisition et de visualisation des données

L'acquisition des données est réalisée par Real-Time Windows Target. La visualisation des données est réalisée par Matlab. Les fichiers cités ci-dessous vous seront fournis par l'enseignant:

- Real-Time Windows Target: `Boucle_Ouverte_Moteur_Triphase.mdl` permet d'étudier le système en boucle ouverte en vue de le caractériser. `Regulation_PI_Moteur_Triphase.mdl` permet de réaliser la régulation PI du moteur et de confronter la réponse obtenue par le moteur en boucle fermée à sa réponse théorique en fonction des modèles discrets et des paramètres de PI discrétisé obtenus.
- Script Matlab: `Identification_Correction_Moteur_Triphase.m` permet de visualiser les signaux et de calculer les paramètres du PI.

2 Expression de la fonction de transfert du PI numérique

Le fichier Real-Time Windows Target contient un PID qui prend en entrée un signal discret. Pour obtenir un PI à la place d'un PID, il suffit d'annuler un paramètre ... Lequel ?
L'influence du PI sur ce signal discret est traduit par sa fonction de transfert C dont on souhaite calculer la version discrète $C(z)$.

Le fichier Real-Time Windows Target doit donc implanter un PI mixte. La fonction de transfert $C(p)$ du PI continu mixte est:

$$C(p) = K(1 + \frac{1}{T_i p}) \tag{1}$$

où K et T_i sont le gain et la constante de temps.
Pour trouver la fonction de transfert numérique $C(z)$ du PI numérique, on applique une transformée bilinéaire à la fonction de transfert continue. C'est une méthode de discrétisation parmi d'autres:

$$p \rightarrow \frac{2}{T_e}\frac{z-1}{z+1} \tag{2}$$

A partir des équations (1) et (2), montrez que l'on peut écrire la transmittance $C(z)$ sous la forme d'une fraction rationnelle:

$$C(z) = K_c\frac{1 - z0z^{-1}}{1 - z^{-1}} \tag{3}$$

Exprimez K_c et $z0$ en fonction de K et T_i.

Ce correcteur PI discret sera inséré dans le fichier Real-Time Windows Target `Regulation_PI_Moteur_Triphase.` pour réguler le moteur.

3 Détermination pratique des fonctions de transfert du moteur et du correcteur

Le choix du type de correcteur et les valeurs de paramètres du correcteur dépendent des caractéristiques du système corrigé, qui sont déterminées dans ce TP par la méthode de Streicj.

3.1 Identification par la méthode de Streicj

On suppose que l'entrée échelon commence au temps $t = 0$.
Soit Δx l'amplitude de l'échelon (valeur asymptotique - valeur initiale).
Soit Δy l'amplitude de la sortie.

Le système est modélisé par la fonction de transfert suivante:

$$G(p) = e^{-\tau p}\frac{K_m}{(1 + \theta p)^n} \tag{4}$$

Les paramètres à identifier sont donc:

Le gain statique $K_m = \frac{\Delta y}{\Delta x}$,
le retard τ,
la constante de temps θ,
l'ordre n.

Méthode:

- On trace la tangente au point d'inflexion I pour déterminer deux valeurs : T1 et T2. Voir la figure 3.1 pour la mesure de ces deux temps.
- Relever T1 et T2, en déduire l'ordre n en utilisant le tableau ci-dessous. Entre deux lignes du tableau, on choisit la valeur de n la plus petite.
- Déterminer la constante de temps θ à partir de la valeur $\frac{T2}{\theta}$ du tableau.
- Déterminer le retard τ quand il existe à partir de la différence entre la valeur de T1 mesurée et celle donnée par la colonne $\frac{T1}{T2}$ du tableau.

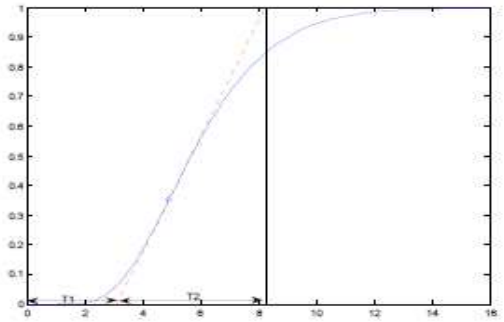

Figure 3: Méthode pour obtenir T1 et T2

n	$\frac{T1}{\theta}$	$\frac{T2}{\theta}$	$\frac{T1}{T2}$
1	0	1	0
2	0.28	2.72	0.1
3	0.8	3.7	0.22
4	1.42	4.46	0.32
5	2.10	5.12	0.41
6	2.81	5.70	0.49

Mise en oeuvre:
En boucle ouverte, donnez une consigne au système et enregistrez la sortie par le fichier Real-Time Windows Target `Boucle_Ouverte_Moteur_Triphase.mdl`:

La commande du moteur doit recevoir une entrée fournie par une sortie ao, par exemple ao0 ("output") de la carte National Instruments. La sortie du moteur doit être mesurée par une entrée ai, par exemple ai0 ("input") de la carte. Vérifiez la cohérence entre les branchements sur la maquette et le fichier RTWT `.mdl`. Vérifiez également le nom des dossiers spécifiés pour l'enregistrement (éléments de sauvegarde des signaux).

Compilez, connectez-vous à la cible, et lancez l'expérience (triangle noir). Un ou plusieurs fichiers de données doivent être créés dans votre dossier de travail ou dans le workspace.

Visualisez l'entrée et la sortie par le fichier Matlab `Identification_Correction_Moteur_Triphase.m`. Vérifiez la cohérence entre le nom des fichiers de données créés par Real-Time Windows Target et le nom des fichiers chargés sous Matlab par la fonction `load`.

Appliquez la méthode de Streicj. Vérifiez la validité du résultat d'identification par la fonction `tf` de Matlab.

3.2 Régulation: méthode et paramètres

Le correcteur adéquat est choisi d'après le rapport des constantes de temps du système. Ensuite (une fois le choix du PI justifié), on détermine les paramètres K et Ti du correcteur, par le tableau ci-dessous construit par Ziegler et Nichols. Il s'agit des paramètres du correcteur en version continue, avant qu'il ne soit discrétisé.

θ/τ	$\cdot > 20$	$10 < \cdot < 20$	$5 < \cdot < 10$	$2 < \cdot < 5$	$\cdot < 2$
Type de régulation	TOR	P	PI	PID	Autre
K		$\frac{\theta}{K_m * \tau}$	$0.9\frac{\theta}{K_m * \tau}$	$1.2\frac{\theta}{K_m * \tau}$	
T_i			3.3τ	2τ	
T_d				0.5τ	

Effectuez la régulation PI grâce au fichier Real-Time Windows Target `Regulation_PI_Moteur_Triphase.mdl`, qui contient donc un soustracteur en plus d'un régulateur PI.

Faites les branchements adéquats sur la maquette: *attention*, on injecte la consigne du système en boucle fermée. Les branchements doivent permettre de visualiser et d'enregistrer la commande, pour vérifier qu'elle ne sort pas des valeurs limite -10 et 10.

Réglage pratique du PI, précautions:

La commande du moteur ne doit pas saturer !
Vous devez donc visualiser et enregistrer les valeurs de commande, en ajoutant un scope et un élément To File dans le fichier `Regulation_PI_Moteur_Triphase.mdl`).

Vous choisirez une valeur faible de gain, que vous augmenterez progressivement, jusqu'à atteindre la limite de saturation. Vérifiez que les valeurs de la commande restent dans l'intervalle `[-10:10]`.

Lancez l'expérience. Commentaires?

3.3 Comparaison des réponses théorique et discrète

Sur le fichier `Identification_Correction_Moteur_Triphase.m`: par la fonction Matlab `c2d.m` avec comme troisième option `'tustin'` discrétisez le moteur. Vous obtiendrez la fonction de transfert discrète théorique du système 'Moteur'. Pourquoi l'option 'tustin' ?

De même, discrétisez le régulateur PI: d'abord avec la même fonction `c2d.m`, ensuite en créant la fonction que vous avez vous-même calculée (voir Equation (3)). Pour cela, créez au préalable l'élément `z=tf('z',Te)`.

Ensuite sur le fichier RTWT `Regulation_PI_Moteur_Triphase.mdl`, vous créerez la fonction de transfert discrétisée du moteur. Vous pourrez ainsi comparer la sortie théorique du système avec sa sortie obtenue en pratique.

Note: La fonction `zpk` permet de représenter la fonction de transfert sous la forme zéros, pôles, gain, et de faciliter la représentation sous simulink avec l'élément "Discrete zero pole".

Vous visualiserez la consigne, la commande, et la sortie du système bouclé avec des "scopes". Comparez les signaux obtenus en sortie du système théorique et du système pratique.

3.4 Expériences complémentaires

Refaites l'expérience de régulation en changeant la fréquence d'échantillonnage. La période d'échantillonnage a-t-elle une influence sur la régulation ?

4 Conclusion et commentaires

Quel est le paramètre dominant dans le comportement d'un correcteur PI ? Que faut-il contrôler pour rester dans un domaine de fonctionnement linéaire du système régulé ?

Aix*Marseille université

Faculté des Sciences

Master I&S

Régulation industrielle

Travaux Pratiques

Régulation d'une boule en sustentation magnétique par PID numérique

Julien Marot, Rachid Outbib

2016-2017

1 Généralités

1.1 Présentation de la maquette

La maquette est constituée de la boule en sustentation, d'un électro-aimant, d'un régulateur PID. Elle peut être alimentée de l'extérieur (GBF, Labview, real-time windows target, etc.) ou de l'intérieur.
Remarque 1: Attention ! Les coefficients du PID sont préréglés, il ne faut pas les modifier en début de TP.
Pour que la boule soit stable en sustentation, il faut se placer dans les conditions suivantes:
KPID: 8
Coefficient I: 2
Coefficient D: 9,9 (max)
Filtrage: maximum (molette tournée à fond vers la droite)
Il faut placer la boule de façon à ce que l'ombre de la main ne couvre pas la photodiode.

Avec, par exemple, une consigne signal carré variant de -3.25 à -4, de période 5 sec.

La figure 1 schématise le dispositif expérimental.

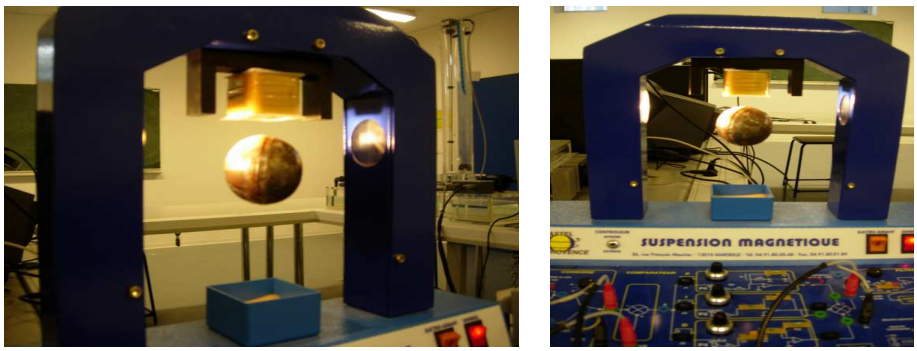

Figure 1: Boule en sustentation magnétique

1.2 Objectif du TP

Pour maintenir la boule en équilibre la maquette est dotée d'un PID qui assure la stabilité du système en boucle fermée. L'objectif du présent TP est de caractériser ce correcteur. Le montage étudié dans ce TP permet de toucher du doigt le phénomène de la sustentation électromagnétique, dont le principe de base est le même que celui des prototypes de trains allemands et japonais.

1.3 Aspects techniques

La boule contient un aimant qui lui permet de rester en sustentation dans l'air lorsqu'elle atteint un état d'équilibre. Sa position d'équilibre *Yfinal*, à valeurs négatives, est donnée par sa distance à l'électroaimant.

Pour mesurer la position de la boule, une source de lumière est placée dans le même plan que la boule, juste au-dessus. Un capteur photoélectrique lui est associé à l'opposé de la boule. Ainsi, plus la boule se déplace vers le haut, moins le capteur reçoit de lumière, et inversement plus la boule se déplace vers le bas plus le capteur reçoit de lumière. On en déduit la position de la boule. Voici les outils de contrôle:

- La consigne peut être fournie par le montage lui-même ou par un élément extérieur. On choisit d'alimenter la maquette de l'extérieur, par Real-Time Windows Target. C'est une toolbox de Matlab qui permet d'envoyer des signaux vers une machine et d'en recevoir, par le bias d'une plaquette du type National Instruments. On choisit de générer un signal sinusoïdal de valeurs négatives comme *Yfinal*.

- la boucle de rétroaction : cette boucle permet de mettre en oeuvre le processus de régulation. La chaîne directe comprend un PID pour lequel les coefficients caractéristiques (gains et constantes de temps) ont été ajustés manuellement;

- l'outil de mesure numérique : un élément "analog input" permet de faire l'acquisition d'un signal de tension, comme la sortie du système, ou la sortie du contrôleur PID. On peut créer deux éléments "analog input" au maximum. Chacun correspond à une entrée ai (ai0 et ai1) de la carte.

Question 1: Réalisez un schéma fonctionnel du dispositif expérimental. On doit y voir apparaître notamment sous forme de dessin la partie opérative avec les effecteurs et leur dénomination, le système physique, l'interface de transfert (c'est-à-dire le convertisseur analogique numérique).

2 Fonction de transfert du PID numérique

On s'intéresse à l'identification du correcteur. Ainsi, le signal d'entrée du système à caractériser sera ici le signal d'erreur ϵ, et le signal de sortie du système à caractériser sera ici la commande, notée w. On note N la longueur de ces deux signaux. Il sont reliés par la fonction de transfert C du correcteur.
La maquette contient un PID monté en parallèle. La fonction de transfert $C(p)$ du PID continu monté en parallèle est: $C(p) = K + \frac{1}{T_i p} + T_d p$ où K, T_i, T_d sont des constantes caractéristiques.
Pour trouver la fonction de transfert $C(z)$ du PID numérique, on utilise les approximations numériques de la dérivée et de l'intégrale.
La relation entre l'entrée $\epsilon(t)$ du PID et sa sortie $w(t)$ est donnée par l'équation (1).

$$w(t) = K\epsilon(t) + \frac{1}{T_i}\int_0^t \epsilon(\tau)d\tau + T_d\frac{d\epsilon}{dt} \tag{1}$$

Pour obtenir l'équivalent numérique temporel, on utilise une approximation de l'intégrale et de la dérivée de ϵ :
-Intégrale: $u(k) = u(k-1) + T_e\epsilon(k)$
-Dérivée: $v(k) = \frac{\epsilon(k)-\epsilon(k-1)}{T_e}$.
L'approximation numérique de (1) est: $w(k) = K\epsilon(k) + \frac{T_e}{T_i}\sum_{j=0}^{j=k}\epsilon(j) + T_d\frac{(\epsilon(k)-\epsilon(k-1))}{T_e}$

Question 2: donnez l'expression de

$$w(k) - w(k-1) \tag{2}$$

Question 3: en considérant la transformée en z de l'équation (2) montrez que $C(z) = \frac{W(z)}{\epsilon(z)} = K + \frac{T_e}{T_i}\frac{1}{1-z^{-1}} + \frac{T_d}{T_e}\frac{1-2z^{-1}+z^{-2}}{1-z^{-1}}$

Question 4: Montrez que la transmittance échantillonnée du correcteur s'écrit $C(z) = \frac{c_0+c_1z^{-1}+c_2z^{-2}}{1-z^{-1}}$

Question 5: Exprimez c_0, c_1, c_2, en fonction de K, T_i, T_d.

On souhaite maintenant identifier le correcteur par la méthode des moindres carrés, c'est-à-dire trouver les valeurs de c_0, c_1, c_2, puis de K, T_i, T_d. Exprimez K, T_i, T_d en fonction de c_0, c_1, c_2.

3 Identification des paramètres du PID numérique

Le but dans cette partie est d'utiliser Real-Time Windows Target et Matlab pour trouver la valeur des paramètres du PID numérique.

3.1 Première acquisition et affichage des signaux

Créez un dossier de travail dans lequel vous enregistrerez tous vos fichiers. Certains fichiers préremplis vous seront fournis par l'enseignant.

Remarque 2: Il faut éteindre le système boule en sustentation après chaque expérience, pour éviter une surchauffe !

Les programmes RTWT et Matlab sont à utiliser et compléter.

Sur le fichier RTWT `Input_Output_Real_Time_WT.mdl`, vérifiez l'intervalle des valeurs du signal de consigne : `[-6 ;-5]` (environ);

Faire les branchements adéquats : connecter les entrées ai0 et ai1 ("inputs") à la sortie du système bouclé, et à la sortie du correcteur PID. Si nécessaire, rajoutez un élément "Analog Input" sur le fichier RTWT, qui s'utilise comme Simulink. Connecter la sortie ao0 ("output") à la consigne du système bouclé (vérifiez que l'interrupteur est sur "ext"). La sortie permet d'envoyer un signal de tension sinusoïdal. C'est un signal bien approprié à une caractérisation par la méthode des moindres carrés : le signal ne doit pas être trop uniforme.
Vérifiez la cohérence entre les branchements sur la maquette et le fichier RTWT

`Input_Output_Real_Time_WT.mdl`. Compilez, connectez à la cible, et lancez l'expérience (flèche noire). Les données du `Input_PID`, `Output_PID` doivent être créées dans le workspace.

Le fichier Matlab principal est `Main_Characterisation_PID_Boule.m`.
Il fait tourner deux sous-fichiers: `Acquisition_Entree_Sortie_Boule.m` qui charge et affiche les signaux, `Apply_Least_Squares_Characterization.m` qui applique les moindres carrés.

Analysez les signaux grâce au fichier Matlab `Acquisition_Entree_Sortie_Boule.m`: ajoutez si nécessaire les lignes de commande qui sauvegardent les signaux générés (`save Output_PID Output_PID` par exemple). Visualisez l'entrée et la sortie du correcteur.
Le fichier `Apply_Least_Squares_Characterization.m` sera utilisé et commenté par la suite.

3.2 Méthode des moindres carrés

Pour appliquer la méthode d'identification des moindres carrés au correcteur, on utilise comme données les signaux d'entrée et de sortie du contrôleur ϵ et w.

La méthode des moindres carrés est mise en oeuvre de la façon suivante. On considère la fonction de transfert du correcteur à caractériser: $C(z) = \frac{c_0+c_1z^{-1}+c_2z^{-2}}{1-z^{-1}}$

Question 6: Ecrivez l'équation aux différences reliant l'entrée ϵ à la sortie w du correcteur.

Question 7: Ecrivez sous forme matricielle cette équation aux différences pour l'ensemble des échantillons $w(3)$, $w(4)$, ... $w(N)$.
Un vecteur θ sera utilisé tel que:

$$\theta = [c_0 c_1 c_2 1]^T \tag{3}$$

Question 8: Montrez que cette forme matricielle s'exprime de la façon synthétique suivante

$$\mathbf{w} = \mathbf{\Psi}\theta \tag{4}$$

avec

$$\mathbf{\Psi} = [\Delta|\Lambda] \tag{5}$$

Δ est une matrice à $N-2$ lignes et 3 colonnes; Λ est une matrice à $N-2$ lignes et 1 colonne. Δ est construite à partir à partir du signal d'entrée du correcteur ϵ, et Λ est construite à partir du signal de sortie du correcteur w.

Question 9: Construisez la matrice $\mathbf{\Psi}$ sur votre programme Matlab `Apply_Least_Squares_Characterization.m`.

Question 10: Comment calcule-t-on l'estimée des paramètres $\hat{\theta}$? Vérifiez et complétez votre programme Matlab en conséquence. La fonction Matlab `pinv.m` sera utile. Connaissant $\hat{\theta}$, on a ainsi caractérisé entièrement le système.

Question 11: Calculez K, T_i, T_d à partir de c_0, c_1, c_2, et T_e la période d'échantillonnage. Mettez en oeuvre ce calcul sur le programme Matlab `Main_Characterisation_PID_Boule.m`.

3.3 Evaluation de la qualité de l'estimation obtenue

On souhaite évaluer la qualité de l'estimation obtenue. Pour cela, on compare le signal enregistré par real-time windows target au signal de sortie modélisé.

- Vérifiez et/ou complétez le programme `Apply_Least_Squares_Characterization.m`, pour afficher l'erreur entre ces deux signaux.
- La quatrième composante du vecteur de paramètres θ à estimer, que l'on note c_3, doit être égale à 1. Vérifiez que cette valeur est proche de 1 en affichant cette valeur calculée par Matlab.
- Vérifiez la valeur du gain du PID en mesurant la sortie du correcteur proportionnel au lieu de la sortie du correcteur PID. Attention, il y a multiplication par -1 sur la maquette entre les deux sorties ! Cela ne doit pas fausser votre jugement. Mesurez aussi l'entrée ϵ du PID directement. Pour cela vous pouvez renommer le nom des variables "To Workspace".
- Complétez les lignes de code écrites à la fin du programme `Main_Characterisation_PID_Boule.m` qui permettent de faire tourner les éléments "From Workspace" du fichier simulink `modele_PID.mdl`. Comparez les sorties du système corrigé simulé et du système corrigé réel. Avez-vous bien caractérisé le PID ?

3.4 Estimation raffinée des paramètres

Plusieurs méthodes permettent d'améliorer les résultats d'estimation obtenus. Vérifiez l'influence de ces méthodes sur la valeur de c_3 et sur le critère d'erreur entre le signal de sortie enregistré et le signal modèle.

- Le débruitage des signaux d'entrée et de sortie: sur le programme `Main_Characterisation_PID_Boule.m`, appliquez un filtre médian aux signaux `Sortie_Correcteur` et `Entree_Correcteur`. Utilisez la fonction `medfilt1` de Matlab, avec une taille de filtre 3, que vous pourrez faire varier.
- La durée d'acquisition: faites un enregistrement plus long.

Annexe A: La méthode d'identification des moindres carrés dans le cas d'un système discret

On présente ici la méthode des moindres carrés dans le cas d'un système discret. Le système suit un modèle de type ARX (auto régressive external).

A.1. Système discret d'ordre quelconque

Le système est représenté par l'équation aux différences suivante:

$$a_n y(k-n)+a_{n-1}y(k-(n-1))+...+a_1 y(k-1)+y(k) = b_m u(k-m)+b_{m-1}u(k-(m-1))+...+b_0 u(k) \tag{6}$$

qui peut s'écrire:

$$y(k) = -a_n y(k-n)-a_{n-1}y(k-(n-1))-...-a_1 y(k-1)+b_m u(k-m)+b_{m-1}u(k-(m-1))+...+b_0 u(k) \tag{7}$$

En prenant la transformée en Z de l'équation (6):

$$Y(z) = \frac{b_m z^{-m} + b_{m-1} z^{-(m-1)} + ... + b_0}{a_n z^{-n} + a_{n-1} z^{-(n-1)} + ... + a_1 z^{-1} + 1} U(z) \tag{8}$$

L'équation (10) peut s'écrire:

$$Y(z) = \frac{B(z^{-1})}{A(z^{-1})} U(z) \tag{9}$$

ou:

$$A(z^{-1})Y(z) = B(z^{-1})U(z) \tag{10}$$

Lorsque les mesures sont entachées d'une erreur e, qui peut provenir d'un bruit ou d'erreurs de mesures:

$$A(z^{-1})Y(z) = B(z^{-1})U(z) + e(z) \tag{11}$$

On peut noter l'équation (11) de la façon suivante:

$$Y(z) = \frac{B(z^{-1})}{A(z^{-1})} U(z) + \frac{1}{A(z^{-1})} e(z) = \frac{B(z^{-1})}{A(z^{-1})} U(z) + \nu(z) \tag{12}$$

L'équation (12) est celle d'un modèle arx.
L'équation (7) peut s'écrire:

$$y(k) = \phi_{\mathbf{discret}}(k)\theta \tag{13}$$

avec:

$$\phi_{\mathbf{discret}}(k) = [-y(k-n),\ -y(k-(n-1)),...,\ -y(k-1),\ u(k-m),\ u(k-(m-1)),...,\ u(k)] \tag{14}$$

soit:
$\phi_{\mathbf{discret}}(k) = [\phi_1, \phi_2, ..., \phi_{n+m+1}]$.

Le vecteur θ est construit à partir des paramètres du système à étudier:

$\theta = [a_n,\ a_{n-1}, ...,\ a_1,\ b_m,\ b_{m-1}, ...,\ b_0]^T$
soit:
$\theta = [\theta_1, \theta_2, ..., \theta_{n+m+1}]^T$.

Le vecteur θ contient $n+m+1$ paramètres qui caractérisent entièrement le système.
Ainsi on a besoin de $n+m+1$ équations au moins.
Ces equations sont obtenues par la définition de plusieurs échantillons de y.

Un échantillon $y(k)$ dépend des n échantillons qui le précèdent dans l'équation aux différences (7). Ainsi les $n+m+1$ équations sont obtenues via $2*n+m+1$ échantillons de y. Soit N le nombre d'échantillons, avec pour condition sur N: $N > 2n+m$. Il est conseillé de prendre $N >>> 2n+m$.
Soit $\mathbf{y}$ le vecteur d'échantillons.
$\mathbf{y} = [y((k-n)T_e), y((k-n+1)T_e), ..., y((k+n+m)T_e)]^T$
En considérant l'équation (14) et en remplaçant k par les valeurs $k-n$ à $k+n+m$, on peut exprimer $\mathbf{y}$ sous forme matricielle:
$\mathbf{y} = \mathbf{\Psi}\theta$.

La matrice $\mathbf{\Psi}$ peut être exprimée grâce à l'équation (14).

A.2 Système discret d'ordre 2

Par exemple dans le cas d'un système d'ordre 2:
On considère un système discret d'ordre 2 $C(z)$, à 4 paramètres inconnus: b_0, b_1, a_1, a_2.

$$C(z) = \frac{b_0 + b_1 z^{-1}}{1 + a_1 z^{-1} + a_2 z^{-2}} \tag{15}$$

Le système est de degré 1 au numérateur et un degré 2 au dénominateur. Le degré du dénominateur est nommé "ordre" du système.

On a besoin de 4 équations, et donc de 6 échantillons au minimum.
Cependant, on choisit de prendre un nombre maximum d'équation pour obtenir le meilleur résultat possible en évitant les erreurs de calcul numérique.

On remarque que k doit être supérieur ou égal à 2. Avec $k = 3$, on obtient:

$$\begin{bmatrix} y(3) \\ y(4) \\ ... \\ y(N) \end{bmatrix} = \begin{bmatrix} u(3) & u(2) & -y(2) & -y(1) \\ u(4) & u(3) & -y(3) & -y(2) \\ ... & ... & ... & ... \\ u(N) & u(N-1) & -y(N-1) & -y(N-2) \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ a_1 \\ a_2 \end{bmatrix} + \begin{bmatrix} \nu(3) \\ \nu(4) \\ ... \\ \nu(N) \end{bmatrix} \tag{16}$$

Sous forme matricielle:

$$\mathbf{y} = \mathbf{\Psi}\theta + \nu \tag{17}$$

où

$$\mathbf{\Psi} = [\Delta| - \Lambda] \tag{18}$$

Δ et Λ sont des matrices à $N-2$ lignes et 2 colonnes; $\mathbf{\Psi}$ est une matrice de taille $N-2$ par 4.

Par la solution des moindres carrés on obtient l'expression de l'estimée des paramètres $\hat{\theta}$:

$$\hat{\theta} = (\mathbf{\Psi}^H \mathbf{\Psi})^{-1} \mathbf{\Psi}^H \mathbf{y} \tag{19}$$

Note concernant le choix du signal d'entrée:

Pour que la matrice $\mathbf{\Psi}$ admette une pseudo-inverse sans problème de conditionnement de matrice, *i.e.* pour qu'elle soit facilement inversible, il faut que toutes ses lignes et colonnes soient différentes. Pour cela il faut que le signal d'entrée ne soit pas trop uniforme (un signal créneau ou sinusoïdal est conseillé).
On construit l'estimée de la réponse obtenue de la façon suivante:

$$\hat{\mathbf{y}} = \mathbf{\Psi}\hat{\theta} \tag{20}$$