UE21 - Régulation industrielle

TD N°6 : Commande linéaire-quadratique (commande LQ)

Dans ce TD charnière, on va apprendre à réaliser la structure complète d'une régulation multivariable (même si le premier exemple est en fait monovariable). On supposera que l'on veut contrôler les p sorties d'un systèmes de dimension n, à l'aide de p entrées. On ne supposera pas que l'on mesure les n variables d'état, donc la structure de contrôle comportera un observateur. Dans un premier temps, on supposera connaitre les équations du système (pas besoin d'identification) et on supposera que la consigne est 0 (cette remarque prend tout son sens quand le système à contrôler est non linéaire et donc que le modèle est son linéarisé).

Exercice 1– Un cas relativement simple .

Le système à réguler est supposé décrit par les équations $\mathbf{x}(k+1) = \mathbf{A}\mathbf{x}(k) + \mathbf{B}u(k)$ avec

$$\mathbf{A} = \begin{pmatrix} 1 & 0.1 & 0 \\ 0 & 1 & 0.1 \\ -0.1 & -0.2 & 0.97 \end{pmatrix} \text{ et } \mathbf{B} = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$$

et on supposera que $\mathbf{x_1}(0) = 10$ et $\mathbf{x_2}(0) = \mathbf{x_3}(0) = 0$. On prendra pour temps d'échantillonnage $T_e = 0{,}1\,\text{s}$.

La sortie que nous voulons réguler est $\mathbf{x}_1(k) = \mathbf{C}\mathbf{x}(k)$ donc $C = \begin{pmatrix} 1 & 0 & 0 \end{pmatrix}$.

Pour commencer, on va étudier ce modèle *en boucle ouverte*, ce qui est toujours une bonne idée.

1. Calculez le spectre de la matrice $\mathbf{A}$ (**eig**(A)) et commenter la stabilité du système sans contrôleur (utiliser aussi **abs**( ));
2. Simuler le système en boucle ouverte (u=0), cela devrait confirmer la réponse à la question précédente.

On aura intérêt à créer le script suivant (et à l'exécuter).

```
A = [1, 0.1, 0; 0, 1, 0.1; -0.1, -0.2, 0.97];
B = [0; 0; 1];
C = [1, 0, 0];
D = 0;
x0 = [10; 0; 0];
Te = 0.1;
```

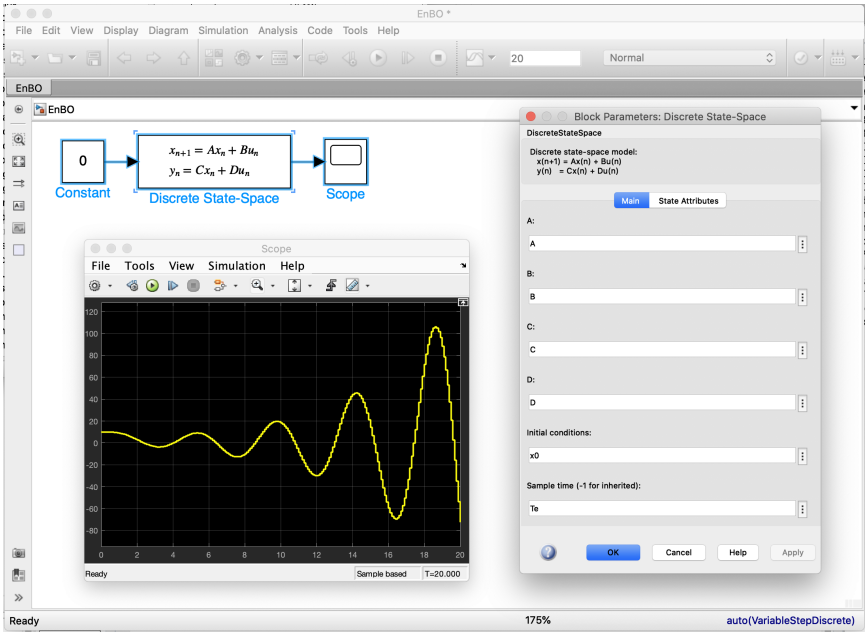


FIGURE 1 – Système en boucle ouverte

On se propose donc de stabiliser ce système autour de 0, pour commencer.

3. Faire une régulation PID de ce système.
4. Placer une saturation sur le contrôle et tester.

Exercice 2– Commande LQ .

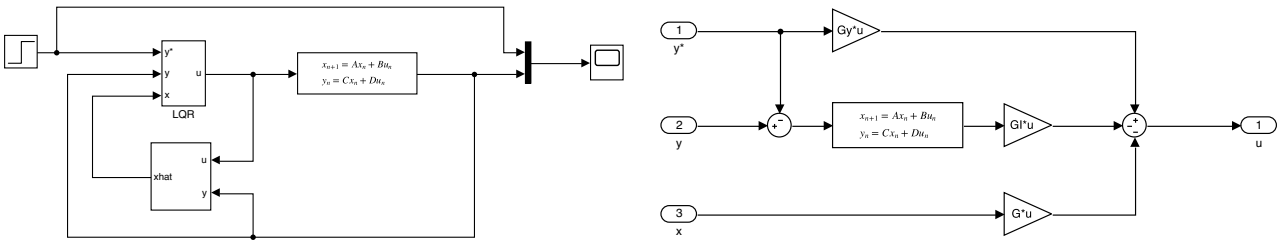
La commande LQ est une commande linéaire multivariable. Elle ressemble à la commande plus basique dite "à placement de pôle" qui consiste à choisir des gains de façon à assurer la stabilité du système. Elle en diffère par la façon de spécifier ces gains, à l'aide de fonctions de coûts. La théorie qui conduit à la construction des matrices de gain est du niveau des (bonnes) écoles d'ingénieur. On se contentera - comme d'habitude - d'utiliser les fonctions matlab qui, rappellons-le, existeront dans tout logiciel permettant la commande avancée.

La façon de spécifier les performances désirées en fonction de **Q** et **R** sera vue lors de la séance de TP.

1. Tester la commande LQ sur ce système. Il faudra compléter le script précédent par :

   A1=A;
   B1=B;
   C1=C;
   D1=D;

2. Comparer avec le PID
3. Essayez d'autres systèmes, y compris multivariables (vous pourrez télécharger l'ensemble des scripts et modèles Simulink sur le site `http://s2i-geiitd/SARII/`)



(a) Commande LQG (b) Contrôleur

FIGURE 2 – Un contrôleur LQ en matlab

Annexe mathématique

Le but d'un régulateur est de réguler la sortie autour d'une consigne que l'on peut modifier. Autour de 0, le contrôle s'écrit

$$\mathbf{u} = -\mathbf{G}\mathbf{x}$$

Si on veut stabiliser le système autour d'une consigne autre que $y = 0$, disons $y = y^*$, alors le contrôle s'écrira

$$\mathbf{u} = \mathbf{G}_y y^* - \mathbf{G}\mathbf{x} \tag{1}$$

où $\mathbf{G}_y$ est définie en écrivant le vecteur d'état $x = \begin{pmatrix} y \\ z \end{pmatrix}$ où la variable à commander y contitue les q premières composantes de x et dans ce cas, la matrice $\mathbf{Gy}$ correspond aux q premières colonnes de $\mathbf{G}$. Autrement dit :

$$\mathbf{G}x = \begin{pmatrix} \mathbf{G}_y & \mathbf{G}_z \end{pmatrix} \begin{pmatrix} \mathbf{y} \\ \mathbf{z} \end{pmatrix}$$

Dans le cas de l'exercice, le vecteur d'état est de dimension 3 et la variable à commander est la première composante de $\mathbf{x}$ donc $q = 1$ et $\mathbf{G}_y$ est la première composante de $\mathbf{G}$ (donc, ici, un scalaire).

La formule (1) s'écrit aussi

$$\mathbf{u} = \mathbf{G}_y(y^* - y) - \mathbf{G}_z z$$

et on reconnait la structure d'un contrôleur proportionnel-dérivé.

Le régulateur que l'on a construit est en fait un régulateur proportionnel-dérivé. Par nature, ce type de régulateur peut présenter des erreurs statiques. Il faut donc ajouter un intégrateur de façon à obtenir l'analogue d'un contrôleur PID multivariable. On pourrait se contenter d'ajouter un terme proportionnel à l'erreur intégrée mais on risque de déstabiliser le système.

Il faut donc être plus malin.

On va agrandir l'espace d'état $\mathbf{x}$ en lui adjoignant q variables $\mathbf{y}_I$ qui résultent de l'intégration de l'écart entre les valeurs désirées et les valeures réelles, donc

$$\mathbf{y}_I(k+1) = \mathbf{y}_I(k) + T_e(\mathbf{y}(k) - \mathbf{y}^*)$$

On pose donc

$$\mathbf{A}_e = \begin{pmatrix} A & {}^n\mathbf{0}_q \\ {}^q\mathbf{Id}_q & {}^q\mathbf{0}_n \end{pmatrix} \text{ et } \mathbf{B}_e = \begin{pmatrix} B \\ {}^p\mathbf{0}_q \end{pmatrix}$$

Cela revient, dans le système, à écrire :

$$\begin{aligned} \mathbf{x}(k+1) &= \mathbf{A}\mathbf{x}(k) + \mathbf{B}\mathbf{u}(k) \\ \mathbf{y}_I(k+1) &= \mathbf{y}_I(k) + T_e(\mathbf{y}(k) - y^*) \end{aligned}$$

Pour résumer, les matrices admettent les structures suivantes :

$$\mathbf{A_e} = \begin{array}{|c|c|c|} \multicolumn{2}{c}{\multirow{2}{*}{\mathbf{A}}} & \multirow{2}{*}{\mathbf{0}} \\ \multicolumn{2}{c}{} & \\ \hline \mathbf{Id} & \mathbf{0} & \mathbf{Id} \\ \hline \end{array} \qquad \mathbf{B_e} = \begin{array}{|c|} \hline \mathbf{B} \\ \hline \mathbf{0} \\ \hline \end{array}$$

et

$$\mathbf{G} = \begin{array}{|c|c|c|} \hline \mathbf{G}_y & \mathbf{G}_z & \mathbf{G}_I \\ \hline \end{array}$$

Le contrôleur obtenu a de bonnes qualités mais il nécessite beaucoup de connaissances sur le procédé à contrôler. En effet, puisque $\mathbf{u}$ est calculé à partir de $\mathbf{x}$, il faut mesurer toutes les composantes de x. Pour un

système mécanique, cela revient à mesurer positions *et* vitesses. Pour un système complexe, il est en général impossible de mesurer toutes les variables d'état. On va donc apporter une derniere amélioration à notre contrôleur : l'adjonction d'un observateur.

Un observateur est un modèle du système à commander, qui réagit exactement de la même façons aux entrées que le système, et que l'on vient caler sur les sorties mesurées. Ainsi, l'état interne – connu – de l'observateur ressemble à l'état interne inconnu du système. La construction d'un observateur ressemble beaucoup à la construction d'un contrôleur, et c'est donc un problème à part entière.

Considérons le système $\mathbf{x}(k+1) = \mathbf{A}\mathbf{x}(k) + \mathbf{B}\mathbf{u}(k)$, on veut construire un système dont l'état interne sera noté $\mathbf{v}$ tel que $\mathbf{x}(k) - \mathbf{v}(k)$ tende rapidement vers 0.

Posons

$$\mathbf{v}(k+1) = \mathbf{A}\mathbf{v}(k) + \mathbf{B}\mathbf{u}(k) + \mathbf{K}(\mathbf{y}(k) - \mathbf{C}\mathbf{v}(k))$$

en notant $\mathbf{y}(k)$ la sortie mesurée du système (qui n'est pas forcément la sortie à contrôler mais on le suppose pour simplifier les notations et parce que c'est souvent le cas).

On pourrait utiliser un observateur par placement de pôles ! En effet, si $\mathbf{K}$ est une matrice telle que $\mathbf{A} - \mathbf{KC}$ est g.a.s. alors

$$\begin{aligned}\mathbf{v}(k+1) - \mathbf{x}(k+1) &= \mathbf{A}\mathbf{v}(k) + \mathbf{B}\mathbf{u}(k) + \mathbf{K}(\mathbf{y}(k) - \mathbf{C}\mathbf{v}(k)) - \mathbf{A}\mathbf{x}(k) - \mathbf{B}\mathbf{u}(k) \\ &= \mathbf{A}(\mathbf{v}(k) - \mathbf{x}(k)) + \mathbf{K}(C\mathbf{x}(k) - \mathbf{C}\mathbf{v}(k)) \\ &= (\mathbf{A} - \mathbf{KC})(\mathbf{v}(k) - \mathbf{x}(k))\end{aligned}$$

et donc $\mathbf{x}(k) - \mathbf{v}(k)$ tend vers 0 par stabilité de $\mathbf{A} - \mathbf{KC}$ et la vitesse est spécifiée par ses pôles.

En pratique, on va plutôt utiliser un bloc existant dans Matlab qui s'appelle "Kalman filter" et qui est bien plus efficace.

La Figure 3 montre l'implémentation de la commande LQ précédente en Scilab (à comparer avec la Figure 2 qui montre la même commande en Matlab.

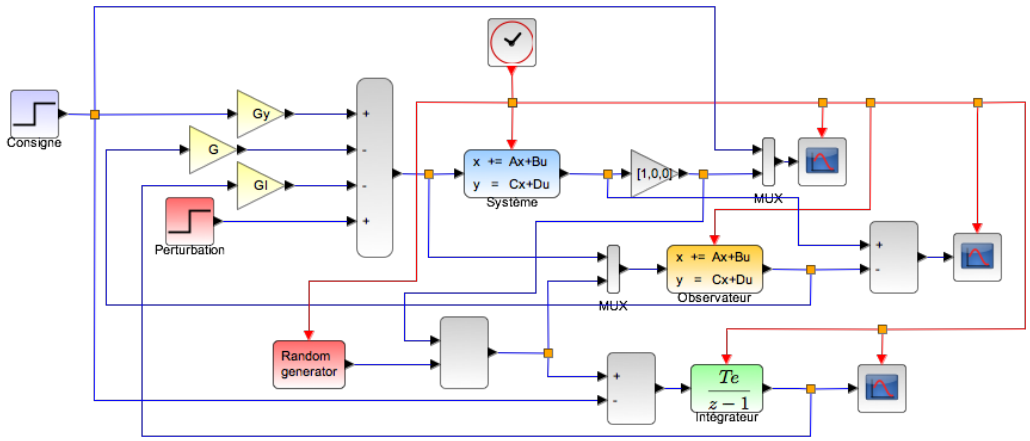


FIGURE 3 – Un contrôleur LQ en scilab