RÉCAPITULATIF DE LA DÉMARCHE DU COURS D'AUTOMATIQUE

IS2120 "SYSTÈMES AUTOMATIQUES"

Cette fiche récapitule la démarche suivie dans le cours, illustrée sur l'exemple du moteur électrique traité en deuxième partie d'amphi en Matlab/Simulink. La démarche est volontairement très formalisée, et à adapter selon les cas. L'exemple est également volontairement très simple, de manière à illustrer la démarche globale (il n'y a aucune notion théorique supplémentaire par rapport au cours IS2110 "Systèmes Automatiques embarqués" de première année).

La démarche générale :

(1) réécrire les équations modélisant le système physique sous forme d'état en faisant bien apparaître les entrées (commande, perturbation, bruit) et les sorties (mesurée et à commander) : c'est le *modèle de simulation*. Programmer ce modèle en Simulink

(2) calculer les points d'équilibre. Choisir quelques points d'équilibre "représentatifs" dans le domaine utile de fonctionnement. Utiliser ces points comme conditions initiales du modèle Simulink, et évaluer grossièrement les performances (stable/instable ? Si stable, performances statiques et dynamiques en poursuite et rejet ?)

(3) linéariser le modèle de simulation autour des points d'équilibre de l'étape précédente et examiner à chaque fois les valeurs propres. S'il y a un groupe de valeurs propres à partie réelle "très négative" et un groupe de valeurs propres "proches" de l'origine, c'est la marque que le système peut sans doute être simplifié par perturbations singulières. Vérifier la cohérence entre les valeurs propres et le comportement de la simulation

(4) (éventuellement) chercher un "petit" paramètre et simplifier le système par perturbations (dans les cas compliqués, il peut être nécessaire de faire un changement de variables pour réécrire le système en forme standard). Le modèle simplifié est valable "globalement" et est très utile pour comprendre le système et concevoir des contrôleurs non-linéaires (ce dernier point sort du cadre de ce cours). Programmer ce modèle simplifié en Simulink et vérifier qu'il approxime bien le système non-simplifié

(5) linéariser le système simplifié obtenu à l'étape précédente autour des points d'équilibre de l'étape 2 et programmer dans la Control Toolbox la "collection" des systèmes ainsi obtenus. Choisir un de ces systèmes linéarisés comme *modèle de commande*, qui servira à concevoir un contrôleur linéaire (choisir par exemple le système qui a un comportement "moyen", ou bien le plus "défavorable", selon les cas). Programmer ce modèle de commande en Simulink et vérifier qu'il approxime bien le système non-simplifié autour du point d'équilibre "moyen".

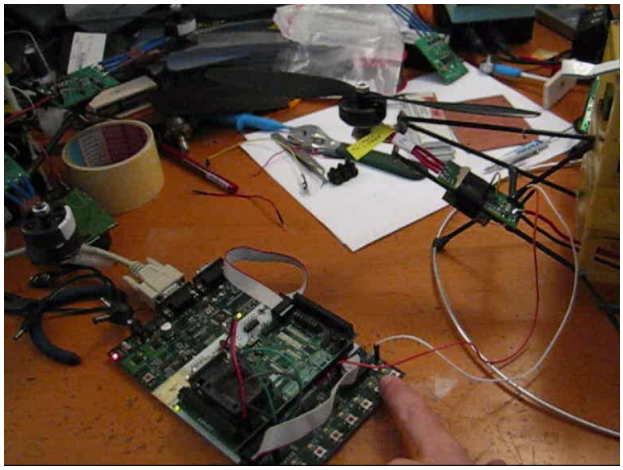
(6) à l'aide des méthodes du cours, concevoir sur la base du modèle de commande un contrôleur linéaire qui assure les performances désirées. Programmer le contrôleur dans la Control Toolbox, et boucler sur la "collection" de systèmes de l'étape précédente. Ajuster les paramètres du contrôleur pour avoir un comportement satisfaisant

(7) implémenter le contrôleur en Simulink (penser au "décalage" par rapport au point d'équilibre et/ou aux conditions initiales du contrôleur), et vérifier qu'il fonctionne de manière satisfaisante sur le modèle de simulation de l'étape 1. Sinon, réfléchir... et recommencer la démarche

---

Date: 14 novembre 2011.

Pour illustrer cette démarche, on considère l'exemple très simplifié d'un moteur électrique à courant continu entraînant une hélice (photo ci-dessous). Il s'agit de contrôler la vitesse de rotation, qui est supposée directement mesurée par un capteur (bruité). Les sections ci-dessous reprennent la démarche point par point. Le programme Matlab/Simulink de simulation se compose du fichier de commandes `script.m` et de plusieurs planches Simulink listées dans la suite. On rappelle que toute variable définie dans le “workspace” Matlab est connue de Simulink, et peut donc être utilisée dans n'importe quel bloc Simulink.



1. Le modèle de simulation

En notant ω la vitesse de rotation de l'hélice, I l'intensité du courant dans le moteur et U la tension appliquée au moteur, on peut écrire

$$\begin{aligned} J\dot{\omega} &= \tau_{em} - \tau_l && \text{(équation mécanique)} \\ L\dot{I} + RI &= U - K\omega + d_2 && \text{(équation électrique)} \\ \tau_{em} &= KI && \text{(couple électro-magnétique)} \\ \tau_l &= \operatorname{sign}(\omega) K I_{fr} + b\,|\omega|\,\omega + K d_1 && \text{(couple de charge).} \end{aligned}$$

Le couple de charge comporte la traînée de l'hélice dans l'air $b\,|w|\,w$, proportionnelle au carré de la vitesse, et le frottement sec sur les paliers du moteur, modélisé par $\operatorname{sign}(\omega) K I_{fr}$; d_1 représente une perturbation de couple, d_2 une perturbation de la tension d'alimentation.

Les paramètres R, L, I_{fr}, K, J, b sont considérés constants, avec comme valeur nominale (en unités SI)

$$R = 0.56 \qquad L = \text{4E-3} \qquad I_{fr} = 0.4 \qquad K = \text{1.19E-2} \qquad J = \text{3.45E-5} \qquad b = \text{2.4E-7}.$$

La sortie mesurée y_m est la vitesse de rotation ω entachée d'un bruit de mesure additif ν. La sortie à commander est $y_c := \omega$; on souhaiterait de plus que le courant I reste dans des limites acceptables, soit environ $10A$.

Le système se réécrit sous forme d'état

$$\begin{aligned} \dot{\omega} &= \frac{1}{J}\big(KI - \operatorname{sign}(\omega) K I_{fr} - b\,|\omega|\,\omega - K d_1\big) \\ \dot{I} &= \frac{1}{L}\big(U - RI - K\omega + d_2\big) \\ y_m &= \omega + \nu \\ y_c &= \omega. \end{aligned}$$

C'est ce qui est programmé dans la planche Simulink `plant.mdl`. Le système étant de dimension 2, il faut un vecteur de conditions initiales de dimension 2 dans le bloc `Integrator`.

On notera qu'il peut y avoir des problèmes numériques de simulation en particulier avec les solvers “stiff” quand on passe par $\omega = 0$, le système n'étant pas continu en ce point (à cause de la fonction `sign`).

2. Points d'équilibre

Les points d'équilibre sont par définition les $(\bar{\omega}, \bar{I}, \bar{U}, \bar{d}_1, \bar{d}_2)$ tels que

$$0 = K\bar{I} - \text{sign}(\bar{\omega})KI_{fr} - b\,|\bar{\omega}|\,\bar{\omega} - K\bar{d}_1$$
$$0 = \bar{U} - R\bar{I} - K\bar{\omega} + \bar{d}_2.$$

En paramétrant par $(\bar{\omega}, \bar{d}_1, \bar{d}_2)$, c.-à-d. par la sortie à commander et les entrées de perturbation, on a

$$\bar{I} = \text{sign}(\bar{\omega})I_{fr} + \frac{b}{K}\,|\bar{\omega}|\,\bar{\omega} + \bar{d}_1$$
$$\bar{U} = R\bar{I} + \bar{\omega} - \bar{d}_2.$$

On vérifie expérimentalement dans la planche Simulink `plant.mdl` qu'on est bien sur un point d'équilibre (le système reste dans l'état initial). Le morceau correspondant de `script.m` est

```
clear all, clc

%% Nominal plant parameters
R = 0.56;
L = 4e-3;
Ifr = 0.4;
K = 1.19e-2;
J = 3.45e-5;
b = 2.4e-7;

%% Compute equilibrium point (initial condition for plant.mdl)
d1bar = 0;
d2bar = 0;
wbar = 300;

Ibar = sign(wbar)*Ifr + b*abs(wbar)*wbar/K + d1bar;
Ubar = R*Ibar+K*wbar-d2bar;

[wbar, Ibar ,Ubar, d1bar, d2bar]
```

3. Linéarisation du modèle de simulation

Le système linéarisé autour d'un point d'équilibre $(\bar{\omega}, \bar{I}, \bar{U}, \bar{d}_1, \bar{d}_2)$ s'écrit

$$J\delta\dot{\omega} = K\delta I - 2b\,|\bar{\omega}|\,\delta\omega - K\delta d_1$$
$$L\delta\dot{I} = \delta U - R\delta I - K\delta\omega + \delta d_2.$$

On notera qu'il n'est pas défini au point de dicontinuité $\bar{\omega} = 0$; autour de $\bar{\omega} \neq 0$, le terme $\text{sign}(\omega)I_{fr}$ disparaît à la linéarisation (c'est une constante). Sous forme matricielle, le linéarisé se réécrit

$$\begin{pmatrix} \delta\dot{\omega} \\ \delta\dot{I} \end{pmatrix} = \begin{pmatrix} -\frac{2b}{J}\,|\bar{\omega}| & \frac{K}{J} \\ -\frac{K}{L} & -\frac{R}{L} \end{pmatrix} \begin{pmatrix} \delta\omega \\ \delta I \end{pmatrix} + \begin{pmatrix} 0 & -\frac{K}{J} & 0 \\ \frac{1}{L} & 0 & \frac{1}{L} \end{pmatrix} \begin{pmatrix} \delta U \\ \delta d_1 \\ \delta d_2 \end{pmatrix}$$

$$\begin{pmatrix} \delta y_m \\ \delta y_{c_1} \\ \delta y_{c_2} \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} \delta\omega \\ \delta I \end{pmatrix} + \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} \delta U \\ \delta d_1 \\ \delta d_2 \end{pmatrix}$$

Le morceau correspondant de `script.m` est

```
%% Linearization around 3 "representative" equilibrium points
% create array of Control Toolbox LTI systems
wtab = [100 300 500];
for i = 1:3;
```

```
    A = [-2*abs(wtab(i))*b/J K/J; -K/L -R/L];
    B = [0; 1/L];
    Bd = [-K/J 0; 0 1/L];
    C = [1 0];
    D = 0;
    Dd = zeros(1,2);
    sys(:,:,i) = augstate(ss(A,[B Bd],C,[D Dd]));
end
set(sys,'InputName',{'U','d1','d2'},'OutputName',{'wm','w','I'})
eig(sys) % 1 slow, 1 fast eigenvalue => use singular perturbations
```

En examinant les valeurs propres de la matrice A (commande `eig(sys)`), on peut voir qu'il y a 2 valeurs propres négatives, d'ordre de grandeur très différent. C'est un signe qu'il est sans doute possible de simplifier le système par perturbations singulières (il faudrait en toute rigueur vérifier que les valeurs propres restent bien séparés sur tout un domaine de points d'équilibre).

4. Simplification du modèle de simulation par perturbations singulières

On "devine" que L est un petit paramètre et qu'en conséquence I est une variable "rapide" et ω une variable "lente" (en toute rigueur, il faudrait normaliser le système et faire apparaître un petit paramètre ε sans dimension). En utilisant la méthode des perturbations singulières, on fait $L = 0$, d'où $0 = U - RI - K\omega + d_2$ (la deuxième équation devient non-différentielle). On en tire $I = \frac{U - K\omega + d_2}{R}$, qu'on injecte dans la première équation différentielle, soit

$$J\dot{\omega} = \frac{K}{R}(U - K\omega + d_2) - \operatorname{sign}(\omega) K I_{fr} - b\,|\omega|\,\omega - K d_1.$$

On obtient ainsi l'*approximation lente*

$$\begin{aligned} J\dot{\omega} &= \frac{K}{R}U - \left(\frac{K^2}{R} + b\,|\omega|\right)\omega - \operatorname{sign}(\omega) K I_{fr} + K\left(\frac{d_2}{R} - d_1\right) \\ I &= \frac{U - K\omega + d_2}{R}. \end{aligned}$$

Ce système ne contient plus qu'une seule équation différentielle, avec en conséquence une condition initiale de dimension 1. Sous forme d'état et avec les sorties,

$$\begin{aligned} \dot{\omega} &= \frac{K}{JR}U - \frac{1}{J}\left(\frac{K^2}{R} + b\,|\omega|\right)\omega - \operatorname{sign}(\omega)\frac{K I_{fr}}{J} + \frac{K}{J}\left(\frac{d_2}{R} - d_1\right) \\ I &= \frac{U - K\omega + d_2}{R} \\ y_m &= \omega + \nu \\ y_c &= \omega. \end{aligned}$$

Si besoin (par exemple pour faire une boucle d'asservissement sur le courant, ce qu'on ne fera pas ici), on peut également considérer l'*approximation lent-rapide*

$$\begin{aligned} J\dot{\omega} &= \frac{K}{R}U - \left(\frac{K^2}{R} + b\,|\omega|\right)\omega - \operatorname{sign}(\omega) K I_{fr} + K\left(\frac{d_2}{R} - d_1\right) \\ L\dot{I} &= U - RI - K\omega + d_2. \end{aligned}$$

Ce système est de dimension 2, avec en conséquence 2 conditions initiales ω_0 et I_0.

Le modèle de simulation et ses approximations lente et lent-rapide sont simulés dans la planche Simulink `plant_model.mdl` ; on constate expérimentalement que les approximations sont excellentes et valables sur tout le domaine de fonctionnement.

5. Le modèle de commande

Conformément à la démarche du cours, le *modèle de commande*, qui sera utilisé pour concevoir le contrôleur est le modèle obtenu après simplification par perturbations singulières et linéarisation, soit

$$J\delta\dot{\omega} = -\left(\frac{K^2}{R} + 2b\,|\bar{\omega}|\right)\delta\omega + \frac{K}{R}\delta U - K\delta d_1 + \frac{K}{R}\delta d_2$$
$$\delta I = \frac{\delta U - K\delta\omega + \delta d_2}{R}.$$

Sans surprise, on note que le système obtenu en simplifiant d'abord par perturbations puis par linéarisation est le même que celui obtenu en simplifiant d'abord par linéarisation puis par perturbations.

Sous forme matricielle, et avec les sorties désirées,

$$\delta\dot{\omega} = -\frac{1}{J}\left(\frac{K^2}{R} + 2b\,|\bar{\omega}|\right)\delta\omega + \begin{pmatrix}\frac{K}{JR} & -\frac{K}{J} & \frac{K}{JR}\end{pmatrix}\begin{pmatrix}\delta U\\ \delta d_1\\ \delta d_2\end{pmatrix}$$
$$\begin{pmatrix}\delta y_m\\ \delta y_c\\ \delta I\end{pmatrix} = \begin{pmatrix}1\\ 1\\ -\frac{K}{R}\end{pmatrix}\delta\omega + \begin{pmatrix}0 & 0 & 0\\ 0 & 0 & 0\\ \frac{1}{R} & 0 & \frac{1}{R}\end{pmatrix}\begin{pmatrix}\delta U\\ \delta d_1\\ \delta d_2\end{pmatrix}.$$

Le modèle de simulation et le modèle de commande sont simulés dans la planche Simulink `plant_model.mdl` ; on constate expérimentalement que le modèle de commande approxime bien le modèle de simulation tant qu'on reste suffisamment proche du point de d'équilibre autour duquel on a linéarisé. On notera que pour pouvoir comparer le système non-linéaire et son linéarisé, il faut "décaler" les entrées-sorties du linéarisé à partir des valeurs d'équilibre, c.-à-d. faire $(\delta U, \delta d_1, \delta d_2) = (U, d_1, d_2) - (\bar{U}, \bar{d}_1, \bar{d}_2)$ et $(y_m, y_{c_1}, y_{c_2}) = (\delta y_m, \delta y_c, \delta I) + (\bar{y}_m, \bar{y}_c, \bar{I})$.

Le morceau correspondant de `script.m` est

```
%% Linearized slow model
% create array of Control Toolbox LTI systems
% rem: linearized slow system = slow linearized system
for i = 1:3;
    As = -(K^2/R+abs(wtab(i))*2*b)/J;
    Bs = K/J/R;
    Bds = [-K/J K/J/R];
    Cs = 1;
    Ds = 0;
    Dds = zeros(1,2);
    Fs = [1; -K/R];
    Gs = [0; 1/R];
    Gds = [0 0; 0 1/R];
    syss(:,:,i) = ss(As,[Bs Bds],[Cs; Fs],[Ds Dds; Gs Gds]);
end
set(syss,'InputName',{'U','d1','d2'},'OutputName',{'wm','w','I'})

figure(1),step(sys,syss) % compare full and slow systems

%% Design model = linearized slow model for "middle" point defined by w=300
[As,Bs,Cs,Ds] = ssdata(syss(1,1,2)) % re-extract data (wm = output 1, U = output 1, i = 2)
```

6. Conception du contrôleur sur la base du modèle de commande

Le système de commande étant de dimension 1, on peut facilement le commander avec un contrôleur Proportionnel-Intégral. Le terme intégral assure une erreur statique nulle malgré des perturbations

(constantes) inconnues et des erreurs de modèle. Le contrôleur s'écrit

$$\begin{aligned}\delta\dot\eta &= k_I(\delta y_m - \delta\omega_r)\\ \delta U &= -k\delta y_m - \delta\eta,\end{aligned}$$

où et $\delta\omega_r$ est la vitesse désirée et k, k_I sont des paramètres à choisir.

En appliquant ce contrôleur au modèle de simulation, et en négligeant le bruit de mesure ν, on trouve

$$\begin{pmatrix}\delta\dot\omega\\ \delta\dot\eta\end{pmatrix} = \begin{pmatrix}-\alpha-k\beta & -\beta\\ k_I & 0\end{pmatrix}\begin{pmatrix}\delta\omega\\ \delta\eta\end{pmatrix} + \begin{pmatrix}0\\ -k_I\end{pmatrix}\delta\omega_r + \begin{pmatrix}-\frac{K}{J} & \frac{K}{JR}\\ 0 & 0\end{pmatrix}\begin{pmatrix}\delta d_1\\ \delta d_2\end{pmatrix},$$

où on a posé $\alpha := \frac{1}{J}\left(\frac{K^2}{R} + 2b\,|\bar\omega|\right)$ et $\beta := \frac{K}{JR}$.

Le polynôme caractéristique du système bouclé est donc

$$\begin{vmatrix}s+\alpha+k\beta & \beta\\ -k_I & s\end{vmatrix} = s^2 + s(\alpha+k\beta) + k_I\beta.$$

Pour qu'il soit égal au polynôme désiré $s^2 + 2\xi f_0 + f_0^2$, il suffit de choisir $k_I := f_0^2/\beta$ et $k := (2\xi f_0 - \alpha)/\beta$. Le morceau correspondant de `script.m` est

```
%% Tune PI controller
Tr = .1; % desired settling time
p0 = 2*pi/Tr;
kI = p0^2/Bs
k = (2/sqrt(2)*p0+As)/Bs
M = 0;
N = [-kI kI];
P = -1;
Q = [0 -k];
cont = ss(M,N,P,Q);
set(cont,'InputName',{'wr','wm'},'OutputName',{'U'})
etabar = -k*wbar-Ubar; % initial condition for no initial transient in Simulink

sysscl = lft(cont,syss,1,1); % closed-loop system, slow model
syscl = lft(cont,sys,1,1); % closed-loop system, full linear model
figure(2),step(sysscl,syscl) % compare closed-loop systems

% Finally, export controller in plant_cl.mdl to test it against the simulation model
```

Pour une véritable implémentation, il faut convertir le contrôleur en discret (on choisit ici un temps d'échantillonage T_s très petit devant la constante de temps T_r du système bouclé). Le morceau correspondant de `script.m` est

```
%% Discrete controller
Ts = Tr/20; % sampling time
syssd = c2d(syss,Ts); % exact zoh discretization for simulation in CTB
contd = c2d(cont,Ts,'foh'); % foh is better for controller discretization
sysscld = lft(contd,syssd,1,1); % closed-loop system
figure(3),step(sysscl,sysscld)

[Md,Nd,Pd,Qd] = ssdata(contd);
% Finally, export controller in plant_cld.mdl to test it against the simulation model
```

On vérifie expérimentalement sur les courbes que le fait de discrétiser le contrôleur ne dégrade pas ses performances tant qu'on respecte $T_s \ll T_r$.

7. Implémentation du contrôleur et vérification expérimentale sur le modèle de simulation

Le contrôleur étant conçu pour un modèle linéarisé, il faut "décaler" ses entrées-sorties à partir des valeurs d'équilibre, c.-à-d. faire $(\delta\omega_r, \delta\ y_m) = (w_r, y_m) - (\bar{\omega}, \bar{\omega})$ et $U = \delta U + \bar{U}$. C'est ce qui est implémenté dans la planche `plant_cl1.mdl`.

On vérifie expérimentalement que le contrôleur fonctionne de manière satisfaisante sur le modèle de simulation.

On peut simplifier l'implémentation du contrôleur en remarquant que grâce au terme intégral, il n'est en fait pas nécessaire de "décaler" les entrées-sorties du contrôleur à partir des valeurs d'équilibre. Il suffit de prendre comme condition initiale du contrôleur

$$\bar{\eta} = -k\bar{\omega} - \bar{U}.$$

C'est une remarque qui a une certaine importance pratique puisque cela dispense de calculer les valeurs d'équilibre. On vérifie expérimentalement que ça marche sur la planche `plant_cl2.mdl`.

8. Ce qui reste ensuite à faire

Il faut maintenant réaliser matériellement le contrôleur de l'étape précédente (par exemple programme embarqué sur un micro-contrôleur) et tester sur le système réel. C'est bien sûr un gros travail... Si ça ne marche pas (après correction des inévitables bugs matériels et logiciels), c'est probablement que le modèle de simulation n'était pas pertinent. Il faut alors s'interroger sur les effets physiques visiblement importants qui ont été négligés et recommencer la démarche.