Mini Projet - Calcul et Simulation sous Simulink (ou Xcos)

Projet – Simulation et commande d'un micro-drone quadrirotor
(Le travail eut être fait sous simulink ou sous Xcos)

Préambule :

Ce TP est basé sur un article de conférence sur la commande d'un micro drone quadrirotor :

Auteurs: Bouabdallah, S. ; Noth, A. ; Siegwart, R.
Titre: PID vs LQ Control Techniques Applied to an Indoor Micro Quadrotor
Conférence : IROS'2004, Sendai, Japan, 2004

L'article est téléchargeable depuis le site de L'Ecole Polytechnique Fédérale de Lausanne EPFL :
http://infoscience.epfl.ch/record/97531

Les paramètres nécessaires à la simulation sont donnés en Annexe (faire attention aux unités).
Il est demandé aux étudiants de :

- Lire l'article jusqu'au paragraphe IV (étudier principalement le paragraphe II).
- Vérifier la cohérence entre les équations des couples (7) et la figure 1.
- Vérifier la cohérence des équations (9) et l'exactitude des équations (1).
- Traduire en français les termes des tableaux de l'article (page 3)
- Vérifier le passage des équations du moteur à courant continu de (11) à (12) et de (12) à (13).

Partie 1 : Simulation du drone

On s'intéresse à la simulation des dynamiques de roulis, de tangage et de lacet du drone, et non pas à celle des mouvements de translation (selon les axes x, y et z).
Pour cette partie, les équations (09) et (10) sont suffisantes (Celui qui s'intéresse au modèle complet (rotations/translations) peut se référer au rapport de *thèse* du premier auteur).

a) En déduire de ces deux équations, la représentation d'état du système (modèle interne) $\dot{X} = f(X, \mathbf{Omega})$, en prenant comme vecteur d'état $X = \left[\phi\ ,\ \theta\ ,\ \psi\ ,\ \dot{\phi}\ ,\ \dot{\theta}\ ,\ \dot{\psi}\right]^T$. Les éléments du vecteur d'entrée $\mathrm{Omega} = [\Omega_1, \Omega_2, \Omega_3, \Omega_4]^{\mathrm{T}}$ représentent les vitesses de rotation des quatre hélices.

b) Créer un modèle Simulink (ou Xcos), réaliser quatre versions de cette partie mécanique (figure 1):

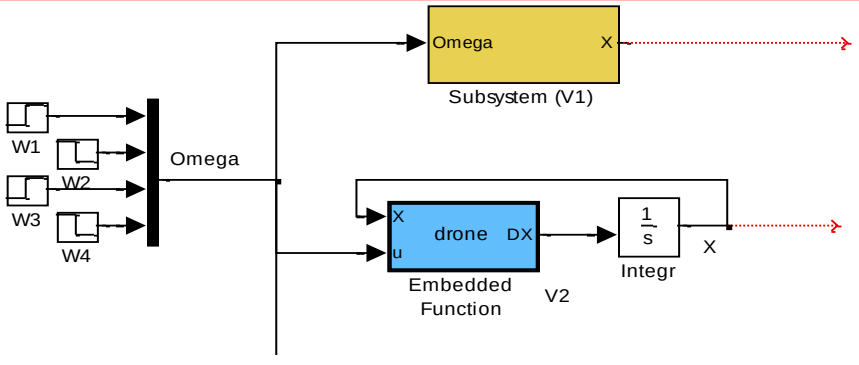


Figure 1 – Exemple de modèle Simulink (ou Xcos)

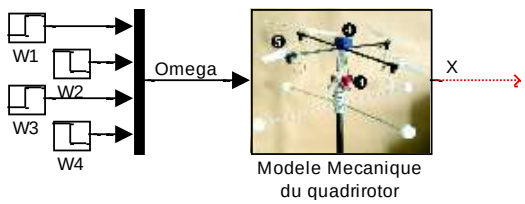


Figure 2 – Partie mécanique du drone dans un subsystem (ou superblock)

1. une version en utilisant six blocs *Integrator* et des blocs *Gain*, *Sum*, *Product*, *Integrator*, … (ne pas oublier les conditions initiales)
2. une version en utilisant six blocs *Integrator* et un bloc *Matlab Function*
3. une version en utilisant deux blocs *Integrator* ($q = [\,\phi\,,\ \theta\,,\ \psi\,]^T$) et un bloc *Matlab Function* (ne pas oublier de mettre à jour les conditions initiales – vecteur de trois éléments).
4. une version représentée par un bloc *Matlab Function* et un seul bloc *intégrateur* (ne pas oublier de mettre à jour les conditions initiales – vecteur d'état X de six éléments).
5. **(En plus)** une version représentée par un seul bloc *S-function*. S-function peut seul simuler un modèle décrit par à la fois des équations différentielles (temps continu) et des équations de récurrence (avec une ou plusieurs fréquences d'échantillonnage).

- Choisir des conditions initiales non nulles pour comparer les différentes versions. N'hésiter pas d'utiliser les *comparateurs* (bloc *sum*) pour tracer l'erreur entre les sorties (qui devrait être nulle).
- Faites en sorte que la *Matlab Function* ne contienne pas les définitions des paramètres du drone. Les paramètres ne doivent être définis que dans le fichier script d'initialisation :
  La déclaration des variables globales (avec *global*) n'est malheureusement pas admise dans une *Matlab Function*. Pour ce faire, une des solutions est d'ajouter des ports d'entrées au bloc *Matlab Function* et faire passer les paramètres à travers ces ports. Une seconde solution, plus intéressante, est d'utiliser l'outil *Editeur de Matlab Function* → *Tools* → *Edit Data/Ports*. Puis ajouter des données (*Add Data*) en le définissant comme *Parameters* et non pas comme des ports d'entrée ou de sortie. Une fois la simulation lancée, les paramètres seront pris du Workspace.

♦ Rajouter au modèle Simulink (ou Xcos) ce qu'il faut pour récupérer les signaux (attitude du drone) dans Workspace. Dans ce cas, vous pouvez lancer la simulation et tracer les courbes directement dans le fichier script d'initialisation.
♦ Choisir désormais **une de ces versions** pour simuler la partie mécanique du quadrirotor. Utiliser des blocs *Manual Switch* ou *Multiport Switch* dans *Signal Routines* pour sélectionner un des modèles.
♦ Mettre le tout dans un *Subsystem* (Entrée Omega, Sortie X) et y associer une image d'un quadriritor (Figure 2). Clic droit -> Mask subsystem

c) Première simulation : Les hélices ont toutes la même vitesse (150 rad/s). Le drone peut-il voler avec ces paramètres ? Justifier en calculant sa portance, sachant que la portance de chaque hélice est donnée par la relation : $F_i = b \cdot \Omega_i^{\,2}$. (Voir l'annexe et la figure Fig.3 de l'article)

- Simuler dans ce cas le système avec des conditions initiales nulles. Utiliser le bloc *Step* pour simuler les entrées Ω_i. Tracer les courbes des angles en fonction du temps. **Remarques !!!**

- Simuler le système avec des conditions initiales non nulles (exemple donner un roulis initial).

Deuxième simulation : Modifier les vitesses des hélices afin de simuler les trois cas suivants :

- 1er cas : uniquement le mouvement de roulis.

- 2ème cas : uniquement le mouvement de tangage.

- 3ème cas : uniquement le mouvement de lacet.

Pour chacun des cas, garder une portance totale constante : $\sum_1^4 F_i = cste$. Les résultats sont-ils cohérents ?

Partie 2 : Ajout du modèle des moteurs à courant continu

Dan cette partie, nous considérons comme entrées les tensions appliquées aux moteurs à courant continu (MCC) au lieu des vitesses des hélices. Pour cela, on ajoutera le modèle de chaque moteur au modèle précédent.
Nous considérons le modèle des MCC donné par l'équation (13), sachant que :

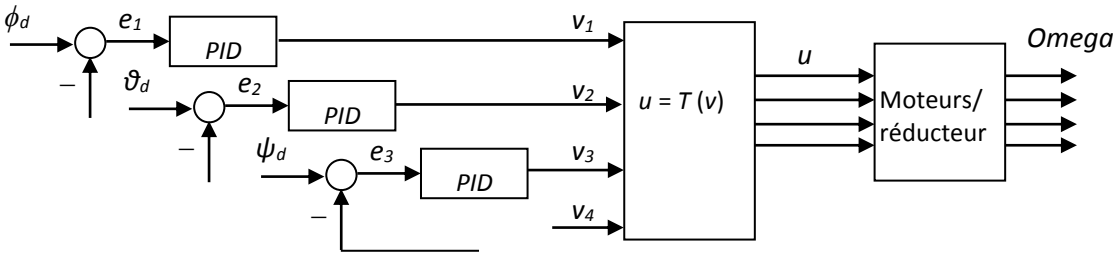
$$\omega_{m,i} = r \cdot \Omega_i$$

$\omega_{m,i}$ la vitesse du $i^{ème}$ moteur, Ω_i la vitesse du $i^{ème}$ hélice, r le facteur de réduction de chaque moteur

a) Simuler dans ce cas le système avec comme entrée les tensions u_i = 4.2 v (constantes).
Indication :
Utiliser le bloc *Manual Switch* dans *Signal Routines* pour choisir entre les sorties Ω_i des MCC et les valeurs précédentes de Ω_i.

Partie 3 : Ajout d'un régulateur PID

Pour contrôler l'attitude du drone, on implémente trois régulateurs PID (un PID pour chaque angle). Cependant, le système a quatre entrées (les tensions u_1, u_2, u_3, u_4). En se basant sur les équations (16), la commande du drone peut être réalisée selon le schéma suivant :



Les trois contrôleurs PID interviennent pour ajuster l'attitude $[\phi, \theta, \psi]^T$ autour de la configuration d'équilibre.
Ainsi, v(1) agit sur le couple de roulis, v(2) sur le couple de tangage et v(3) sur le couple de lacet.

b) Etudier l'équation (16) de l'article et vérifier s'il n'y a pas d'erreur.

Après rectification, la transformation T peut être définie par :

```
u(1)=0.5*sqrt(-2*v(2) + v(3) + v(4) );
u(2)=0.5*sqrt(-2*v(1) - v(3) + v(4) );
u(3)=0.5*sqrt( 2*v(2) + v(3) + v(4) );
u(4)=0.5*sqrt( 2*v(1) - v(3) + v(4) );
```

avec $\mathbf{v(4)} = cZ$ supposée être constante, elle correspond à la portance $F = mg$ (pour une attitude nulle).
En pratique, $v(4)$ sert à contrôler la position verticale du quadrirotor (elle influe sur la portance).

c) Comment a-t-on trouvé l'expression de la transformation T ?

Simplification:

- Pour simplifier, il est possible de choisir une transformation T de la forme:

```
u(1)=0.25* (-2*v(2) + v(3) + v(4) );
u(2)=0.25* (-2*v(1) - v(3) + v(4) );
u(3)=0.25* ( 2*v(2) + v(3) + v(4) );
u(4)=0.25* ( 2*v(1) - v(3) + v(4) );
```

d) Comment a-t-on trouvé l'expression de la transformation T (rectifier son expression si nécessaire) ?

e) Simuler le système régulé en utilisant trois blocs PID avec K_p , K_i , K_d les mêmes que ceux utilisés pour reproduire les figures Fig. 5 et Fig. 6 (page 4 de l'article). Ici, les angles désirés (les consignes) sont nuls.

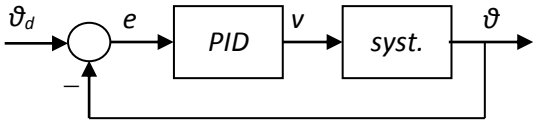
Indication :

Utiliser le bloc *Manual Switch* pour choisir entre les sorties u_i du régulateurs et des valeurs constantes de u_i.

Partie 4 : (Bonus) Réalisation d'un régulateur PID filtré discret

On veut contrôler l'attitude du drone avec trois régulateurs PID filtrés (dont la composante dérivée est filtrée) défini par :

$$PID(s) = K_p + K_i \frac{1}{s} + K_d \frac{s}{1+\frac{s}{N}}$$



On peut démontrer que le PID présenté ici peut être assimilé par un système continu (entrée e / sortie v), tel que :

$$\dot{x}_1 = e$$
$$\dot{x}_2 = N\left(K_d\, e - x_2\right)$$

Sortie : $v = K_p e + K_i x_1 + N(K_d e - x_2)$

a) Ajouter un bloc *Matlab Function* et un bloc *Intégrateur* pour simuler le PID filtré.

b) En pratique, le réglage ne se fait généralement pas en continu, la commande v est calculée une fois par période d'échantillonnage T_e. On utilise donc un régulateur discret.

A partir des équations du PID continu et par approximation des dérivées $\dot{x} \approx \frac{x(k+1)-x(k)}{Te}$, la représentation discrète du PID filtré s'écrit :

$$x_1(k+1) = x_1(k) + T_e\ e(k)$$
$$x_2(k+1) = x_2(k) + T_e\, N\left(K_d\, e(k) - x_2(k)\right)$$

Sortie : $v = K_p e + K_i x_1 + N(K_d e - x_2)$

Réaliser ce régulateur discret avec un bloc *Matlab Function* et un bloc *Retard (Unit Delay* du groupe *Descrete)*.

(En plus) d) Réaliser une autre version du PID discret avec un seul bloc *S-Function*.

Annexe

Table 3.1: OS4 propulsion group design variables

propeller		OS4	unit
mass	m_p	5.2	g
thrust coeff.	b	3.13e-5	$\mathrm{N\,s^2}$
drag coeff.	d	7.5e-7	$\mathrm{Nm\,s^2}$
inertia	J_r	6e-5	$\mathrm{kg.m^2}$
gearbox		**OS4**	**unit**
efficiency	η	90	%
mass	m_{gb}	7	g
max. torque		0.15	Nm
max. speed		1000	rad/s
red. ratio	r	4:1	
motor		**OS4**	**unit**
effi. at hover	η_m	64	%
mass	m_m	12	g
max. power	P_{el}	35	W
internal res.	R_{mot}	0.6	Ω
inertia	J_m	4e-7	$\mathrm{kg\,m^2}$
torque cst.	k	5.2	mNm/A

Table E.1: OS4 parameters.

name	parameter	value	unit [mksA]
mass	m	0.650	kg
inertia on x axis	I_{xx}	7.5e-3	$\mathrm{kg.m^2}$
inertia on y axis	I_{yy}	7.5e-3	$\mathrm{kg.m^2}$
inertia on z axis	I_{zz}	1.3e-2	$\mathrm{kg.m^2}$
thrust coefficient	b	3.13e-5	$\mathrm{N\,s^2}$
drag coefficient	d	7.5e-7	$\mathrm{Nm\,s^2}$
propeller radius	R_{rad}	0.15	m
propeller chord	c	0.04	m
pitch of incidence	θ_0	0.26	rad
twist pitch	θ_{tw}	0.045	rad
rotor inertia	J_r	6e-5	$\mathrm{kg.m^2}$
arm length	l	0.23	m