2025 – SEMESTRE 8

UE - EEAA

COMMANDE NUMÉRIQUE D'UNE MAQUETTE D'HÉLICOPTÈRE

AUTEUR :
M. KUMAR DEV

GROUPE : GIE2 – ED3 – TS21

ENCADRANTS :
M. GUILLARD HERVÉ
M. MECHBAL NAZIH
M. REBILLAT MARC

École Nationale Supérieur des Arts et Métiers
151 Boulevard de l'Hôpital, 75013 Paris, France

Conservatoire Nationale des Arts et Métiers
61 Rue du Landy, 93210 Saint-Denis, France

SOMMAIRE

INTRODUCTION

Ce travail s'inscrit dans le cadre de l'étude d'une **maquette d'hélicoptère didactique**, conçue pour illustrer la commande de systèmes dynamiques à plusieurs degrés de liberté. Cette maquette peut se mouvoir selon **trois axes** : l'élévation (angle ε), le tangage (θ) et le lacet (λ). Ces trois angles permettent de décrire intégralement la position de l'hélicoptère dans l'espace. Chacun de ces mouvements peut être activé indépendamment, et des **capteurs spécifiques** permettent d'en mesurer précisément les valeurs à tout instant.

L'étude expérimentale est focalisée ici sur **le mouvement d'élévation**. Il s'agit d'un basculement vertical de l'hélicoptère autour d'un axe horizontal, sous l'effet d'un couple généré par deux moteurs à hélice fixés à l'extrémité d'un bras tournant. Ces moteurs sont commandés électriquement par les tensions appliquées à droite et à gauche, et la somme de leurs effets produit une **force verticale** qui agit sur l'élévation du système.

Dans ce TP, on cherche à **maîtriser dynamiquement l'angle d'élévation**, en le maintenant à une valeur désirée malgré l'instabilité naturelle du système. En effet, le système est instable en boucle ouverte : une simple perturbation suffit à entraîner une déviation continue de l'élévation. L'enjeu est donc de concevoir une **commande stabilisante** capable de garantir le **suivi d'une consigne**, avec une bonne performance dynamique et une réponse robuste aux perturbations.

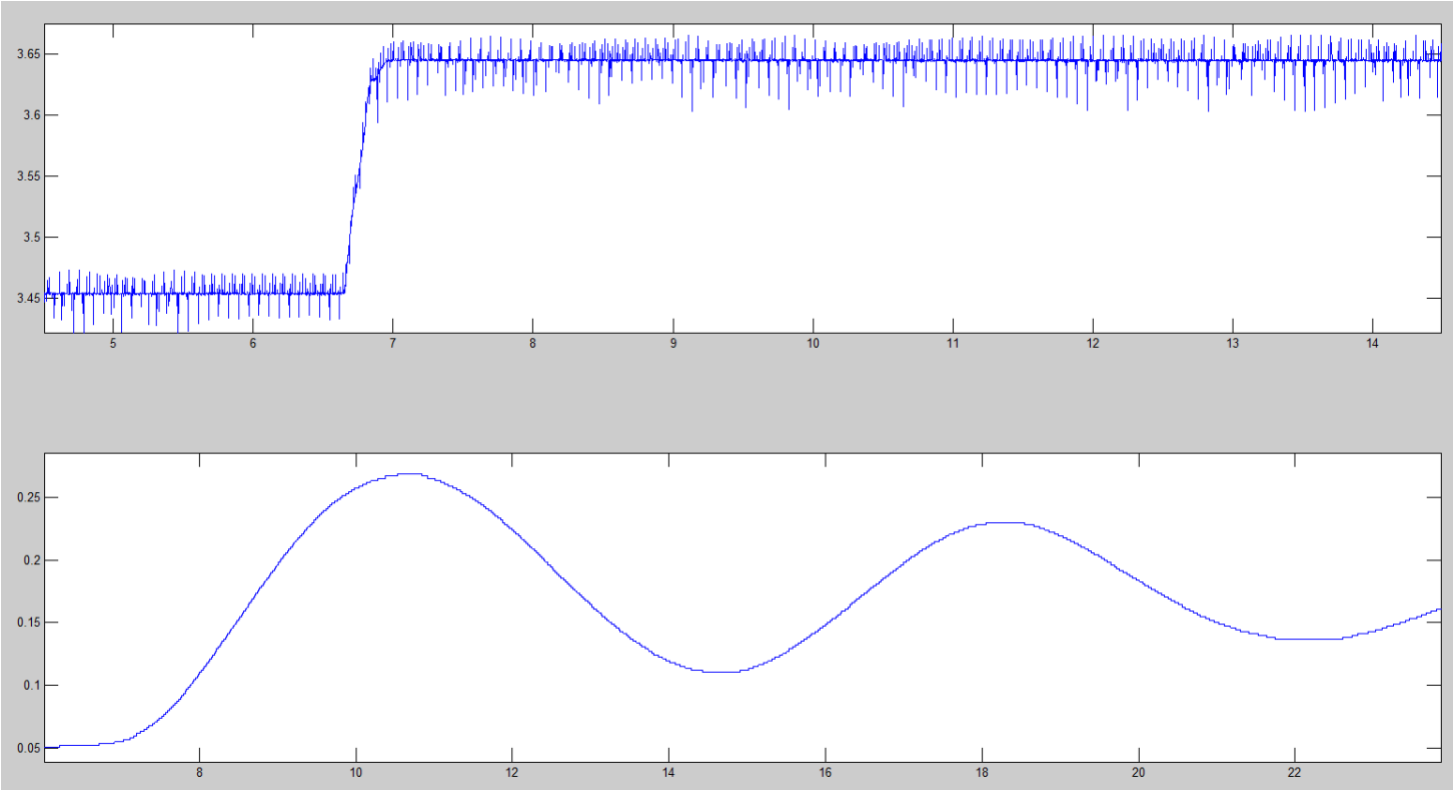
L'ensemble de l'asservissement repose sur une **chaîne numérique temps réel** : la maquette est pilotée par une **carte dSPACE DS1103/1104**, utilisée pour le prototypage rapide, et interfacée avec le logiciel Matlab/Simulink via un **bloqueur d'ordre zéro**. L'architecture informatique comprend une **liaison temps réel avec le calculateur PC**, dans lequel les lois de commande sont conçues puis compilées et injectées dans le DSP de la carte pour une exécution autonome.

Les outils logiciel utilisé dans cette étude sont dSPACE, MATLAB et SIMULINK.

I – ESSAI EN BOUCLE OUVERTE ET D'IDENTIFICATION DU MOUVEMENT D'ÉLÉVATION

Le mouvement d'élévation est défini par l'angle ε(t), mesuré par un capteur dédié situé au niveau du socle de la maquette. Ce mouvement dépend de la force résultant des deux hélices, elles-mêmes pilotées par les tensions appliquées aux moteurs gauche et droit, notées Ud(t) et Ug(t). Dans cette partie, seule la **somme** de ces deux tensions est considérée comme commande, soit U(t)=Ud(t)+Ug(t) et la sortie du système est l'angle d'élévation ε(t).
Pour identifier le comportement dynamique du système, une **méthode expérimentale d'automaticien** a été privilégiée. Elle consiste à bloquer les deux autres mouvements (tangage et lacet) afin d'isoler le comportement en élévation, puis à appliquer un **échelon de tension** via le potentiomètre d'élévation (commande manuelle), en se plaçant autour de l'état d'équilibre (hélicoptère à l'horizontale).

Une fois les données exportées, elles ont été traitées sous MATLAB et on obtient la courbe suivante :



L'échelon appliqué (courbe du haut) provoque une réponse typiquement **oscillante** de l'élévation (courbe du bas), avec un **dépassement**, des **oscillations visibles** et une stabilisation lente. Ce comportement est caractéristique d'un **système du second ordre sous-amorti**, ce qui justifie naturellement le **choix d'un modèle linéaire d'ordre 2**.

Nous avons donc le modèle suivant en fréquentielle avec S la variable de Laplace.

$$H_e(S) = \frac{K}{\frac{S^2}{\omega_0^2} + \frac{2\xi}{\omega_0} S + 1}$$

- *Facteur d'amortissement :*

$$\frac{D_2}{D_1} = \frac{D_3}{D_2} = e^{-\frac{\pi\xi}{\sqrt{1-\xi^2}}} \quad <-> \quad \frac{y_S - d_2}{d_1 - y_S} = \frac{d_3 - y_S}{y_S - d_2}$$

$$y_S = \frac{d_2^2 - d_1 d_3}{2d_2 - d_1 - d_3} = \frac{0.11^2 - 0.27 \times 0.23}{2 \times 0.11 - 0.27 - 2.23} \quad <-> \quad y_S = 0.17857$$

$$e^{-\frac{\pi\xi}{\sqrt{1-\xi^2}}} = \frac{y_S - d_2}{d_1 - y_S}$$

$$\xi = \frac{-\frac{1}{\pi}\ln^2\frac{y_S - d_2}{d_1 - y_S}}{1 + \frac{1}{\pi^2}\ln^2\frac{y_S - d_2}{d_1 - y_S}} = \frac{-\frac{1}{\pi}\ln^2\frac{0.17857 - 0.11}{0.27 - 0.17857}}{1 + \frac{1}{\pi^2}\ln^2\frac{0.17857 - 0.11}{0.27 - 0.17857}} \quad <-> \quad \boxed{\xi = 0.0912}$$

- *Pulsation de coupure :*

$$\omega_0 = \frac{\pi}{\sqrt{1-\xi^2}} = \frac{\pi}{\sqrt{1 - 0.0912^2}} \quad <-> \quad \boxed{\omega_0 = 0.8938 \text{ rad.s}^{-1}}$$

- *Gain :*

$$K = \frac{\Delta y}{2\,\Delta a} = \frac{0.17857 - 0.051}{2\,(3.649 - 3.46)} \quad <-> \quad \boxed{K = 0.3299 \text{ rad.v}^{-1}}$$

- *Fonction de transfert en Boucle Ouverte :*

$$H_e(S) = \frac{0.3299}{\frac{S^2}{0.8038^2} + \frac{2 \times 0.0912}{0.8038} S + 1} \quad <-> \quad \boxed{H_e(S) = \frac{0.2131}{S^2 + 0.1465\,S + 0.6461}}$$

On observe que notre système présente une erreur statique non nulle, révélée par un gain statique différent de l'unité lorsque l'on évalue la fonction de transfert en S=0. Pour assurer un suivi fidèle de la consigne à long terme, il devient alors nécessaire d'introduire un intégrateur dans la boucle de correction. Celui-ci jouera le rôle d'un gardien du régime permanent, contraignant le système à effacer toute erreur résiduelle au fil du temps. Dans cette perspective, nous nous apprêtons à embrasser une nouvelle représentation : celle de l'état. Par son formalisme rigoureux, elle ouvre les portes d'une analyse fine de la dynamique interne, et surtout, elle nous offre un cadre naturel pour intégrer l'erreur comme une variable à part entière. Ainsi, le placement judicieux des pôles nous permettra de façonner le comportement temporel du système selon nos exigences — entre stabilité, rapidité et précision. C'est sur cette voie que se dessine la suite de notre démarche.

II – CALCUL D'UN CORRECTEUR NUMÉRIQUE À RETOUR D'ÉTAT ET BOUCLE INTÉGRAL

A) Étude du système

1- Équation différentielle du système

$$\mathbf{H_e(S)} = \frac{\boldsymbol{\varepsilon(S)}}{\boldsymbol{U_d(S) + U_g(S)}} = \frac{\boldsymbol{K}}{\dfrac{\boldsymbol{S^2}}{\boldsymbol{\omega_0^2}} + \dfrac{\boldsymbol{2\xi}}{\boldsymbol{\omega_0}}\boldsymbol{S} + \mathbf{1}}$$

$$\boldsymbol{S^2\,\varepsilon(S) + 2\xi\omega_0\,S\,\varepsilon(S) + \omega_0^2\,\varepsilon(S) = \omega_0^2 K\,U(S)}$$

Laplace Inverse : $\boxed{\ddot{\varepsilon}(t) + 2\xi\omega_0\,\dot{\varepsilon}(t) + \omega_0^2\,\varepsilon(t) = \omega_0^2 K\,u(t)}$

2- Représentation d'état

$$\ddot{\varepsilon}(t) = -2\xi\omega_0\,\dot{\varepsilon}(t) - \omega_0^2\,\varepsilon(t) + \omega_0^2 K\,u(t)$$

Vecteur d'état, sortie et entrée du système :

$$x(t) = \begin{bmatrix}\varepsilon(t)\\ \dot{\varepsilon}(t)\end{bmatrix} \rightarrow \quad \dot{x}(t) = \begin{bmatrix}\dot{\varepsilon}(t)\\ \ddot{\varepsilon}(t)\end{bmatrix} \quad ; \quad y(t) = \varepsilon(t) \quad ; \quad u(t) = u_d(t) + u_g(t)$$

Système d'état :

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

$$\begin{cases}\begin{bmatrix}\dot{\varepsilon}(t)\\ \ddot{\varepsilon}(t)\end{bmatrix} = \begin{bmatrix}0 & 1\\ -\omega_0^2 & -2\xi\omega_0\end{bmatrix}\begin{bmatrix}\varepsilon(t)\\ \dot{\varepsilon}(t)\end{bmatrix} + \begin{bmatrix}0\\ \omega_0^2 K\end{bmatrix} u(t)\\ \varepsilon(t) = [1 \quad 0]\begin{bmatrix}\varepsilon(t)\\ \dot{\varepsilon}(t)\end{bmatrix} + [0]\,u(t)\end{cases}$$

Matrices (A,B,C,D) :

$$A = \begin{bmatrix}0 & 1\\ -\omega_0^2 & -2\xi\omega_0\end{bmatrix} \quad B = \begin{bmatrix}0\\ \omega_0^2 K\end{bmatrix} \quad C = [1 \quad 0] \quad D = 0$$

$$\boxed{A = \begin{bmatrix}0 & 1\\ -0.6461 & -0.146\end{bmatrix} \quad B = \begin{bmatrix}0\\ 0.2652\end{bmatrix} \quad C = [1 \quad 0] \quad D = 0}$$

3- *Discrétisation*

Le système étudié est initialement modélisé sous forme continue dans une représentation d'état classique avec les matrices AAA, BBB, CCC et DDD. Pour pouvoir implémenter le correcteur dans un système numérique, il est nécessaire de convertir ce modèle continu en un modèle discret. Cette conversion est réalisée ici par la méthode du **ZOH** (*Zero Order Hold*), qui est la plus répandue dans les systèmes d'asservissement numérique.

La méthode **ZOH** consiste à supposer que l'entrée du système reste constante pendant chaque intervalle d'échantillonnage. Cela correspond à un maintien en valeur nulle du dérivé de l'entrée entre deux instants kT et (k+1)T. Cette hypothèse est compatible avec les convertisseurs numériques type DAC (Digital-to-Analog Converter) utilisés en pratique, qui maintiennent la sortie constante entre deux mises à jour.

Via Matlab nous discrétisons notre système d'état afin d'obtenir F, G, C, D. Le code source est disponible en annexe.

```
a =
            x1        x2
   x1        0         1
   x2  -0.6461   -0.1466

b =
            u1
   x1        0
   x2   0.2652

c =
       x1  x2
   y1   1   0

d =
       u1
   y1   0

Continuous-time model.
```

$$Avec\ T_{ech} = 0.05s$$

→

```
F =

    0.9992    0.0498
   -0.0322    0.9919

G =

    0.0003
    0.0132

C =

     1     0

D =

     0
```

Lors de la discrétisation les matrices C et D reste inchangés, seule les matrices F et G sont modifier par les calculs suivants :

$$F = \exp(AT) = L^{-1}[(sI - A)^{-1}]$$

$$G = \int_0^T \exp(A\tau)d\tau\ B$$

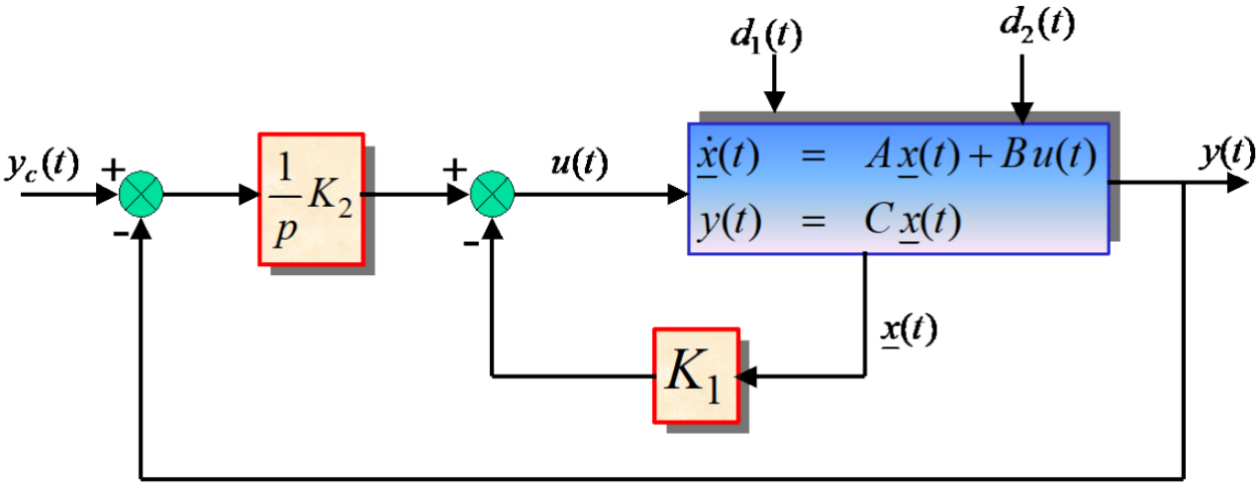
Nous obtenons ainsi notre représentation d'état en discret :

$$\boldsymbol{F = \begin{bmatrix} 0.9992 & 0.0498 \\ -0.032 & 0.9919 \end{bmatrix}} \quad \boldsymbol{G = \begin{bmatrix} 0.0003 \\ 0.0132 \end{bmatrix}} \quad \boldsymbol{C = [1 \quad 0]} \quad \boldsymbol{D = 0}$$

B) Dimensionnement du correcteur à retour d'état par effet intégrale

1- Commande par retour d'état à effet intégrale

Le calcul d'un correcteur numérique à retour d'état et bouclage intégral repose sur une idée élégante : observer le cœur du système — ses états internes — pour mieux le guider vers la trajectoire désirée. En enrichissant le modèle par l'intégration de l'erreur, on ajoute une mémoire de la faute, un rappel constant de la distance entre la consigne et la réalité.
C'est ce terme intégrateur qui, comme une conscience, veille à effacer toute déviation rapporter par des perturbations extérieures comme d1(t) et d2(t) ici ci-dessous.



Pour qu'un tel correcteur puisse pleinement exercer son pouvoir, deux conditions fondamentales doivent être satisfaite : **le critère de Kalman** et le critère de commandabilité augmenté.

En d'autres termes, il faut que la paire (F, G) du système soit gouvernable soit gouvernable (de rang plein n) et la matrice Fe du système augmenté soit de rang n+1.

Matrice de Gouvernabilité :

$$rang\ (G_v) = rang\ ([G \quad FG]) = n$$

$$\begin{cases} rang\ (G_v) = rang\ ([G \quad FG]) = n \\ rang\ (F_e) = n + 1 \end{cases}$$

Pour cela il suffit de vérifier que Gv est inversible :

$$\det G_v = \det \begin{bmatrix} 0.0003 & 0.0009 \\ 0.0132 & 0.0131 \end{bmatrix} = 0.0003 \times 0.0131 - 0.0009 \times 0.0132$$
$$= -7.9\ 10^{-6} \neq 0$$

Gv est donc de rang plein.

La matrice Fe est construite de la façon suivante :

$$F_e = \begin{bmatrix} F & 0 \\ C & 1 \end{bmatrix}$$

Le 1 en diagonale est justement l'ajout du pôle de l'intégrateur en discret $\frac{z}{z-1}$ puisque c'est une matrice triangulaire : $Valp(F_e) = \{Valp(F)\} + \{1\}$

Par la règle de Sarrus :

$$\det F_e = \det F = 0.9992 \times 0.9919 + 0.0322 \times 0.0428 = 0.9925 \neq 0$$

Ainsi **Fe est de rang plein**. Les matrices du système discrétisé répondent avec rigueur aux exigences. Ainsi, nous avons les clefs en main : les pôles du système fermé pourront être placés là où le concepteur le souhaite, ouvrant la voie à une dynamique sur mesure, à un asservissement robuste, et à une élévation sans faute.

2- *Système augmenté*

Nous obtenons ainsi le système augmenté suivant :

```
Fe =

    0.9992    0.0498         0
   -0.0322    0.9919         0
    1.0000         0    1.0000


Ge =

    0.0003
    0.0132
         0
```

➔ $F_e = \begin{bmatrix} 0.9992 & 0.0498 & 0 \\ -0.0322 & 0.9919 & 0 \\ 1 & 0 & 1 \end{bmatrix}$

➔ $G = \begin{bmatrix} 0.0003 \\ 0.0132 \\ 0 \end{bmatrix}$

3- *Horizon de commande*

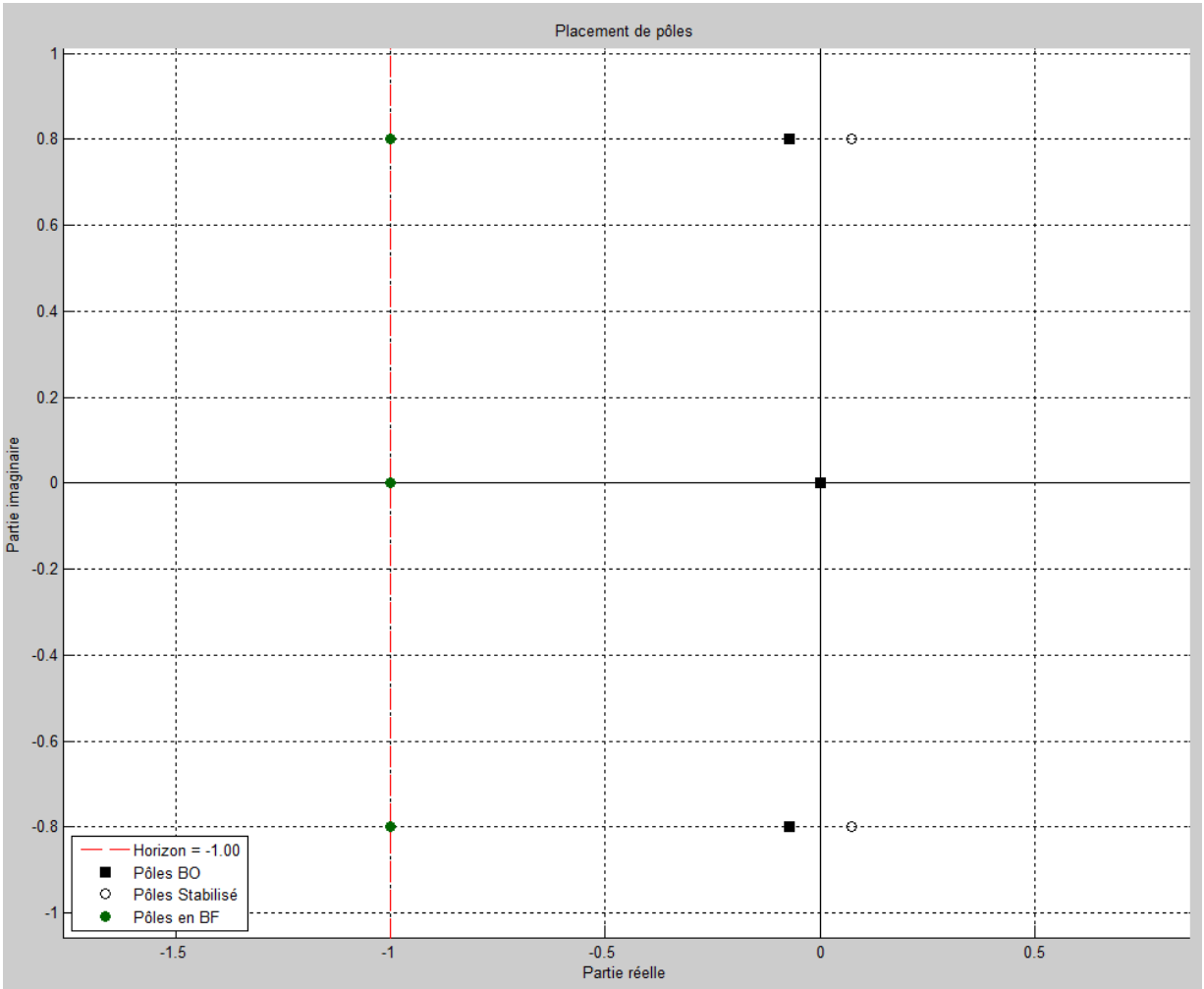
La période d'échantillonnage doit respecter le critère suivant :

$$\frac{\tau_{min}}{10} \leq T_{ech} \leq \frac{\tau_{min}}{5}$$

τ_{min} est la partie réelle du pôle le plus rapide en boucle fermé (donc après le placement de pôles).
Les pôles en continues sont les valeurs propres de A que nous obtenons avec Matlab :

```
>> eig(A)

ans =

  -0.0733 + 0.8005i
  -0.0733 - 0.8005i
```

Notre horizon de commande est placé à -1. Le placement de pôles va se faire de la façon suivante :



Les pôles sont donc tous sur l'horizon de commande -1. Sachant que Tc = 1s, nous avons donc $\tau_{min} = T_c = 1s$

```
Poles_C_BO =

         0
  -0.0733 + 0.8005i
  -0.0733 - 0.8005i


Poles_C_BF =

  -1.0000
  -1.0000 + 0.8005i
  -1.0000 - 0.8005i
```

$$\frac{1}{10} \leq T_{ech} \leq \frac{1}{5}$$

La période d'échantillonnage est bien inférieure à 0.1s ce qui ne pose pas de réel problème tant qu'elle est inférieure à au moins 5 fois τ_{min}.

4- Pôles en boucle fermé

Voici les transitions entre les pôles en continu en boucle ouverte à pôles en discret en boucle fermé via les formules d'Euler.

```
Poles_C_BO =

         0
  -0.0733 + 0.8005i
  -0.0733 - 0.8005i


Poles_C_BF =

  -1.0000
  -1.0000 + 0.8005i
  -1.0000 - 0.8005i


Poles_Z_BF =

   0.9512
   0.9505 + 0.0381i
   0.9505 - 0.0381i
```

Ainsi les pôles en discret en boucles fermé sont :

$$\mathbf{P_{BF-Z} = \{0.9512\ ;0.9505 \pm 0.0381\ i\}}$$

5- Gain de commande K_c

A partir des pôles le raisonnement se fait manuellement par identification des coefficients en égalisant 2 polynômes en boucle fermé obtenu et en résolvant un système d'équations.

$$\det(zI - (F - GK)) = P_{BF-Z}$$

Via matlab la commande acker nous donne ainsi la matrice de commande K suivante :

```
K =

   10.6434   10.2527    0.2882
```

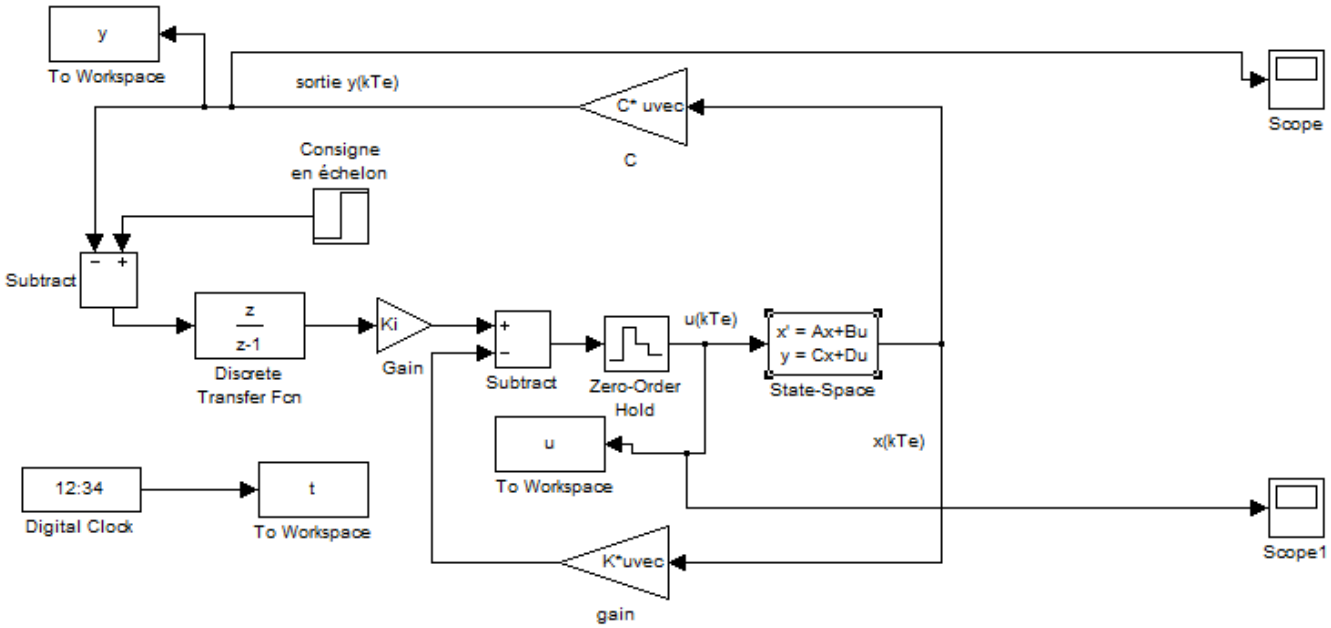
$$\mathbf{Kp = [10.6434 \quad 10.2527] \qquad K_I = [0.2882]}$$

III – ESSAI DE SIMULATION

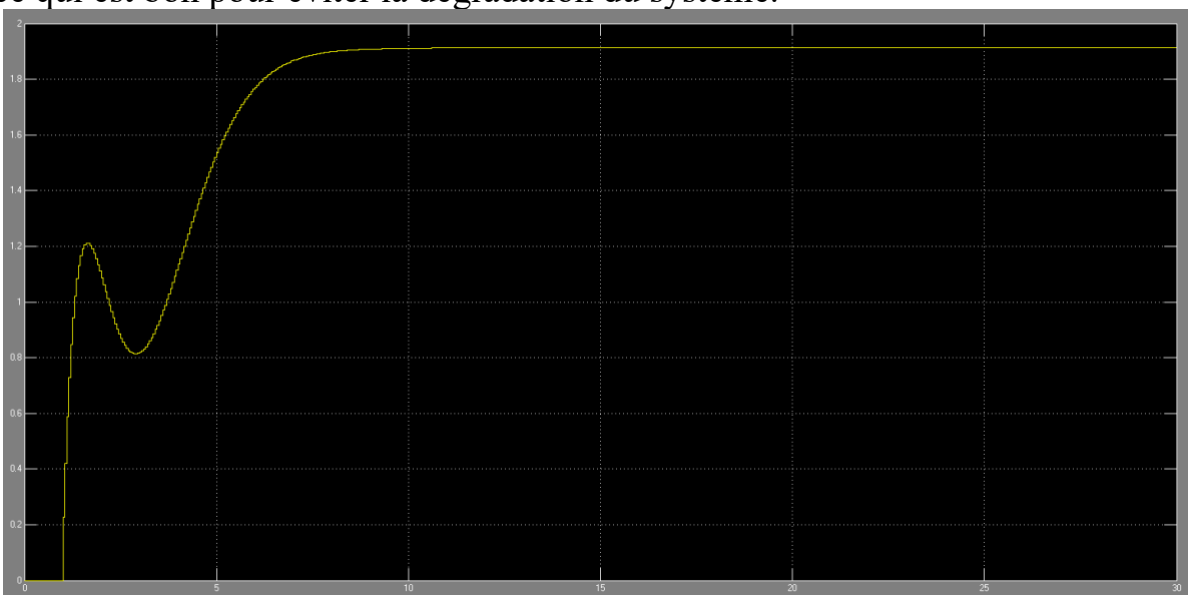
1- *Simulation avec correcteur*

À la suite du dimensionnement du correcteur par retour d'état intégral, une phase de simulation sous Simulink a été menée afin de valider les performances théoriques du contrôleur avant implémentation sur le système réel.

Le modèle utilisé reprend la représentation d'état discrétisée du système, intégrée dans une boucle fermée où l'action de commande est calculée à partir de l'état estimé et de l'erreur intégrée. Le schéma de commande inclut un intégrateur numérique, garantissant l'annulation de l'erreur statique en régime permanent.

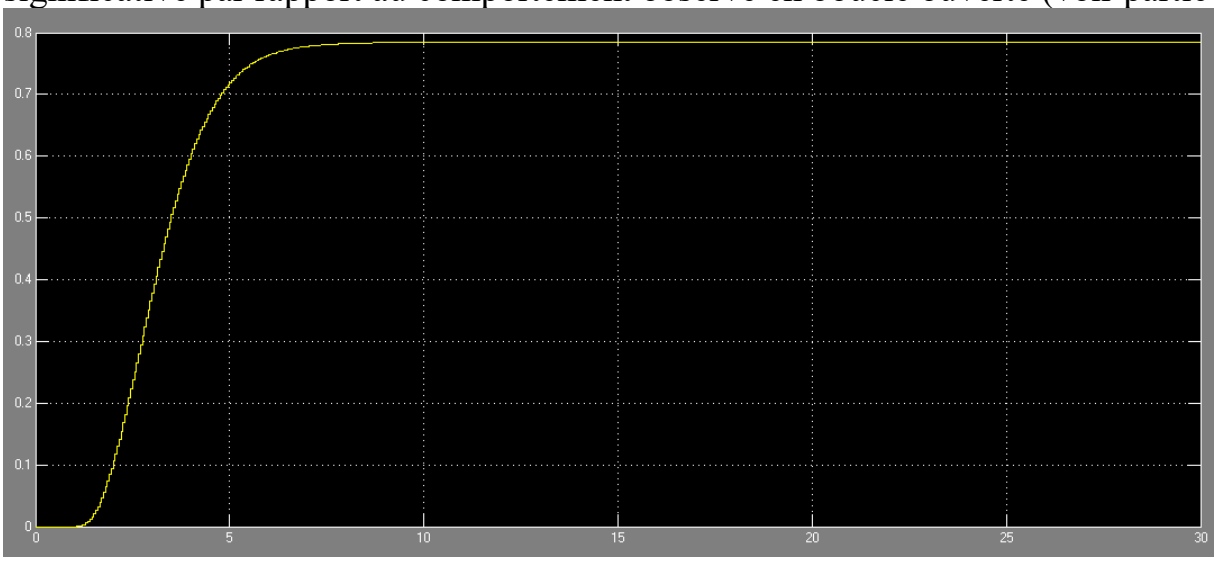


La **commande** délivrée atteint un **maximum de 1.9 V** sans oscillation brusque au début, ce qui est bon pour éviter la dégradation du système.



2- *Analyse de la réponse indicielle*

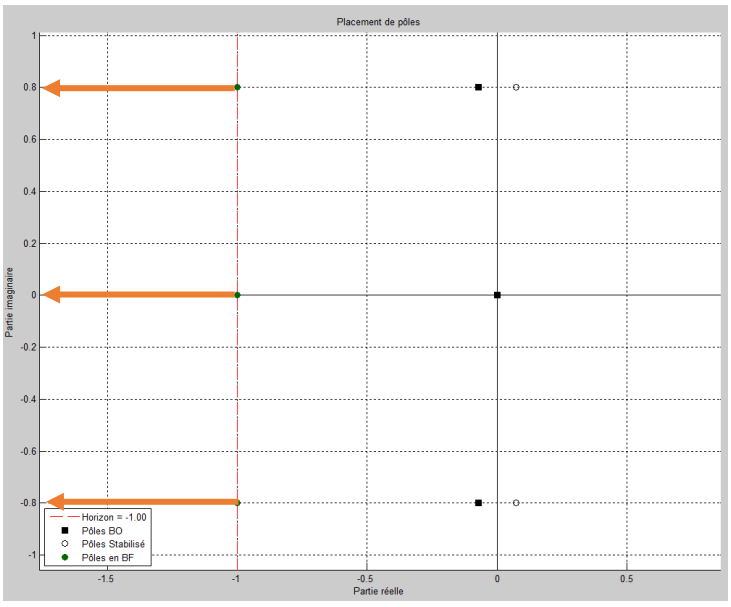
La **sortie en élévation** présente une dynamique **sans dépassement**, **sans oscillation**, et avec une **erreur statique nulle**, démontrant la bonne efficacité du correcteur. Le **temps de réponse** est estimé à **environ 6 secondes**, ce qui constitue une amélioration significative par rapport au comportement observé en boucle ouverte (voir partie I).



Cette simulation valide le bon fonctionnement du correcteur dans des conditions nominales et constitue une étape clé avant le déploiement sur le système réel. Le système présente désormais une réponse stable, assez rapide et conforme aux exigences spécifiées.

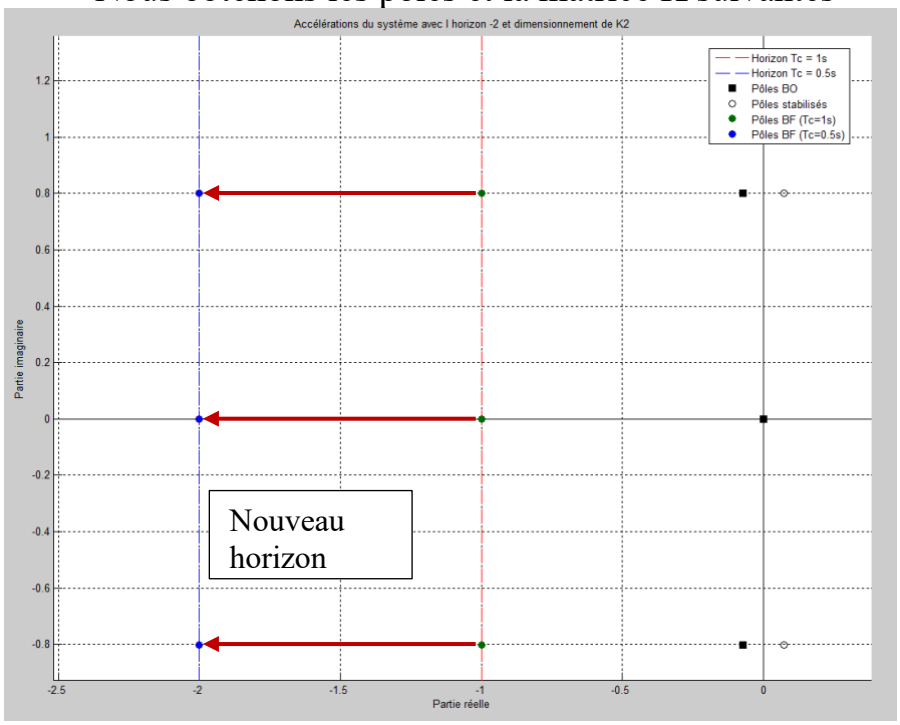
3- *Accélération du système*

Si l'on souhaite obtenir un temps de réponse en boucle fermée plus court, il est nécessaire de repositionner les pôles du système en boucle fermée plus à gauche dans le plan complexe, c'est-à-dire de **déplacer davantage l'horizon de placement**. Cela revient à **réduire le paramètre Tc**, qui fixe la partie réelle souhaitée des pôles du système fermé. En diminuant Tc on impose une dynamique plus rapide à la boucle fermée. Les pôles sont alors situés plus profondément dans la partie négative du plan réel, ce qui accélère la convergence de la réponse du système à une consigne. Cette stratégie permet donc de réduire significativement le **temps de réponse global**.



4- Calcul du nouveau correcteur et analyse de la réponse indicielle en boucle fermé

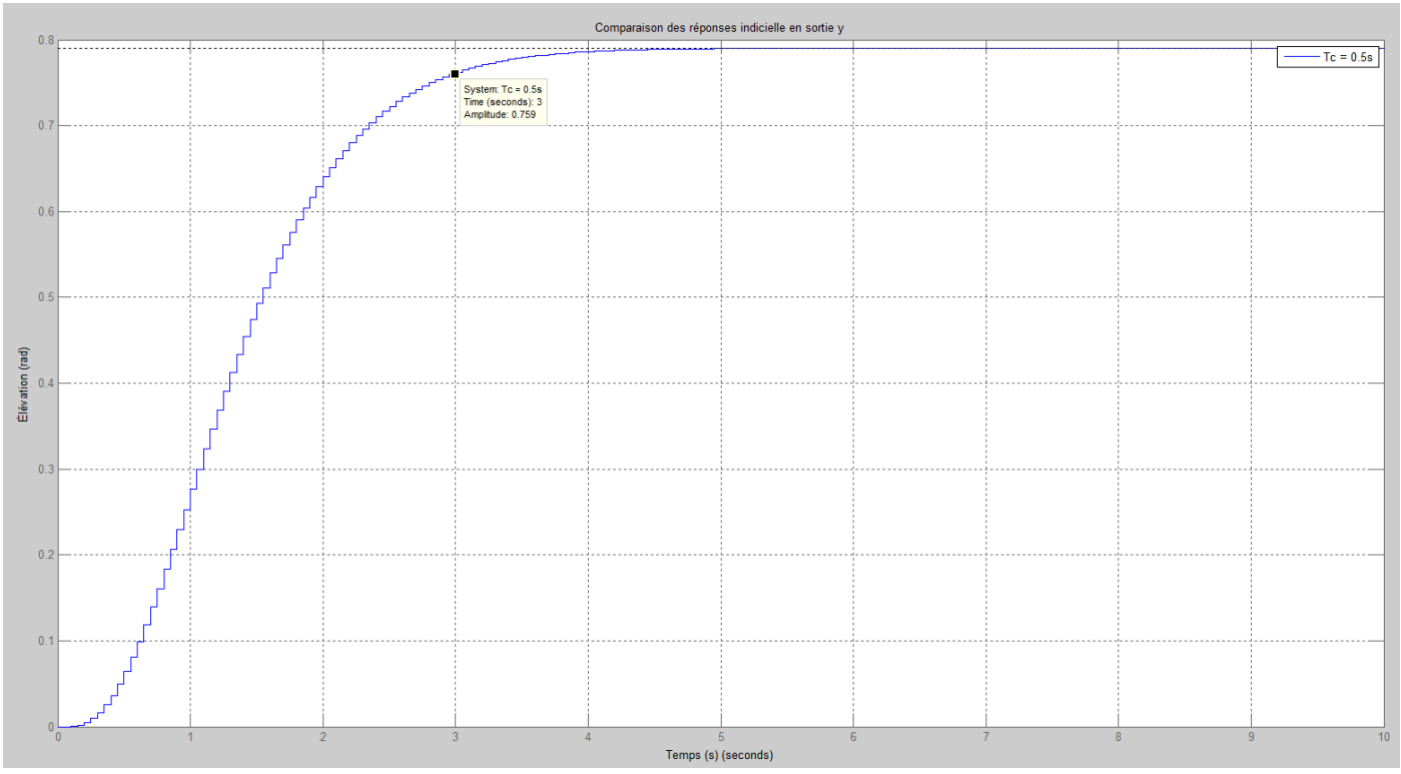
Nous choisissons de diviser notre temps Tc par 2 afin d'accélérer le système et le rendre 2 fois plus rapide et pourrais théoriquement réduire notre temps de réponse à 3s. Ainsi l'horizon de commande se trouve maintenant à -2. Nous obtenons les pôles et la matrice K suivantes




```
Poles_C_BO =

   0.0000 + 0.0000i
  -0.0733 + 0.8005i
  -0.0733 - 0.8005i


Poles_C_BF2 =

  -2.0000 + 0.0000i
  -2.0000 + 0.8005i
  -2.0000 - 0.8005i


Poles_Z_BF2 =

   0.9048 + 0.0000i
   0.9041 + 0.0362i
   0.9041 - 0.0362i


K2 =

   40.3459   20.0418    1.5137
```

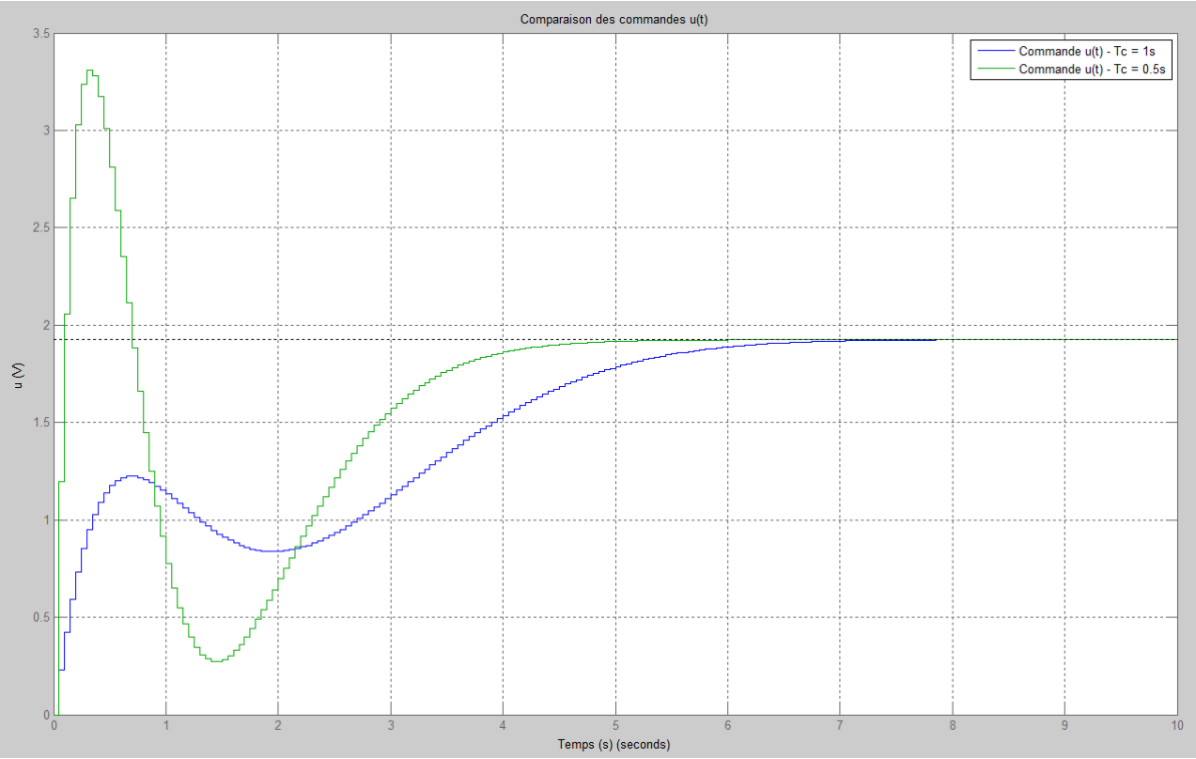
$$Kp = [40.3459 \quad 20.0418]$$
$$K_I = [1.5137]$$

Nous obtenons la réponse indicielle suivante comme attendu le temps de réponse s'est réduit à 3s notre système est bien plus rapide.



5- *Conclusion sur le second correcteur*

Notre correcteur K2 est plus rapide que le premier. Toutefois, cette accélération n'est pas sans conséquence.



L'analyse comparative des deux commandes associées aux correcteurs dimensionnés avec Tc=1 (courbe bleue) et Tc=0,5 (courbe verte) révèle des différences significatives tant en amplitude qu'en dynamique.

Tout d'abord, la commande associée à Tc=0,5s est nettement plus rapide mais aussi plus agressive. Elle atteint un pic d'environ **3.3 V** dès les premières itérations, ce qui reflète une forte sollicitation du système afin de garantir une montée plus rapide de la sortie. On note également une phase transitoire marquée par un dépassement de la consigne, suivie d'une oscillation d'amplitude modérée avant stabilisation. Ce comportement est typique d'un système rapide mais moins amorti.

En comparaison, la commande issue du correcteur avec Tc=1s est plus douce et plus progressive. Son pic est limité à **1.2 V**, ce qui traduit une sollicitation plus modérée de l'actionneur. La réponse est également plus amortie, avec une dynamique plus lisse et sans oscillation notable.

Ainsi, bien que le second correcteur permette une accélération significative du système, cela se fait au prix d'une **augmentation importante de l'énergie de commande**, ce qui peut s'avérer problématique dans un contexte de saturation des actionneurs ou de contrainte énergétique. Un compromis entre rapidité et stabilité reste donc essentiel selon les exigences du cahier des charges.

IV – IMPLANTATION DES CORRECTEURS OBTENUS

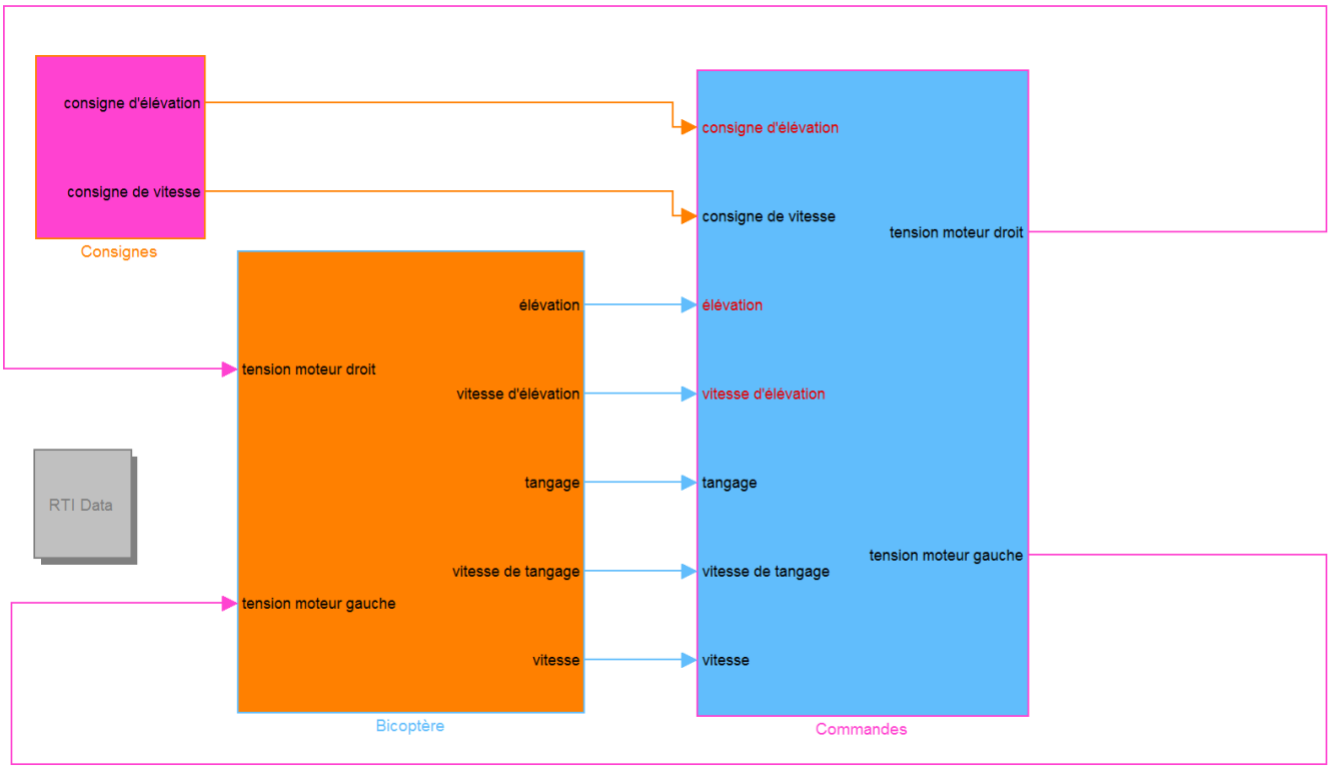
Après la phase de conception et de validation en simulation, l'étape suivante consiste à implanter le correcteur directement sur le système physique à l'aide de l'environnement **Simulink** couplé à la **carte dSPACE**. Cette interface temps réel permet de transférer automatiquement le schéma de commande développé en simulation vers la plateforme matérielle, en assurant l'acquisition et le traitement en continu des données. Le schéma ci-dessus représente l'architecture logicielle du système. On y distingue trois blocs fonctionnels :

Le bloc "Consignes" (en rose) permet d'imposer dynamiquement une consigne d'élévation ou de vitesse au système.

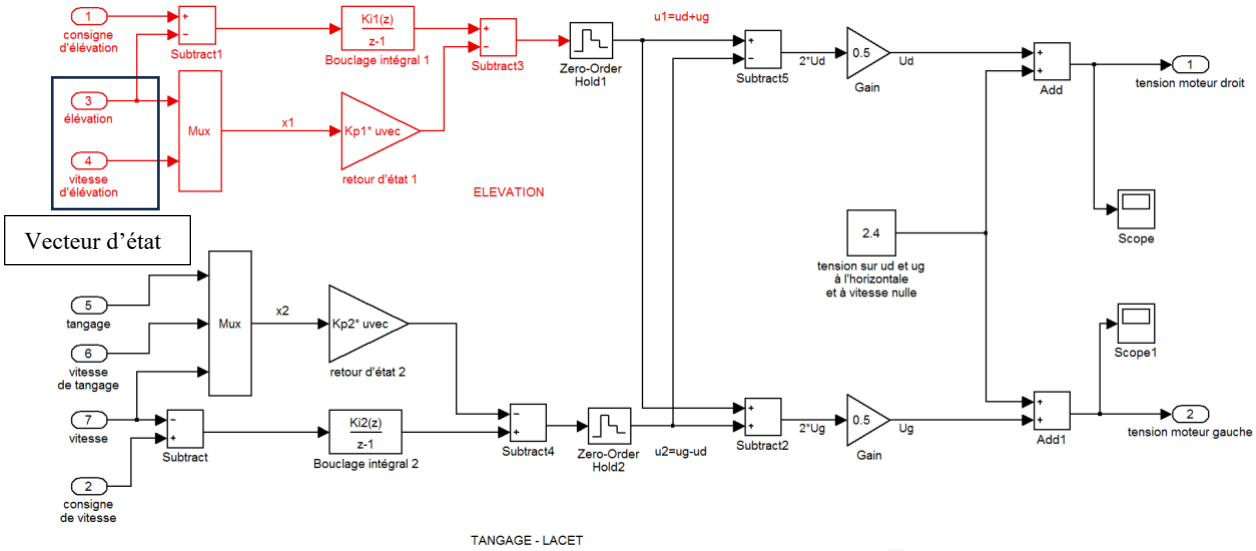
Le bloc "Commandes" (en bleu) contient les lois de commande élaborées, notamment le correcteur à retour d'état avec intégration d'erreur. Il reçoit en entrée les mesures physiques et calcule en sortie les tensions à appliquer aux moteurs.

Le bloc "Bicoptère" (en orange) symbolise le système réel. Il intègre les actionneurs (moteurs gauche et droit) ainsi que les capteurs permettant de mesurer l'élévation, le tangage, et leurs dérivées temporelles.

Les signaux échangés entre les blocs transitent via la **carte dSPACE** qui assure le lien entre le simulateur (hôte) et la machine cible, permettant une exécution en temps réel avec échantillonnage contrôlé. Cette infrastructure garantit une fidélité maximale entre les tests de simulation et le comportement réel du système.



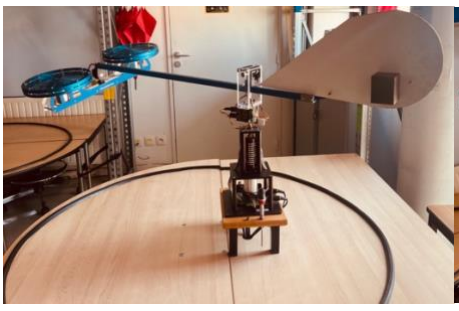
Le schéma présenté ci-dessous représente en détail la structure interne du **bloc "Commandes"** dédié à l'élévation et au tangage du système. C'est à ce niveau que le **correcteur numérique à retour d'état intégral** est **implanté dans Simulink** et exécuté sur la carte **dSPACE**. La partie en rouge concerne l'élévation, les étiquettes en entrée du bloc 3 et 4 encadrés sont justement notre vecteur d'état x(t).



Pour déployer ce correcteur sur le système, **il suffit de définir les gains Kp1, Ki1** dans l'espace de travail MATLAB avant d'exécuter le modèle Simulink. Les blocs associés récupèrent automatiquement les valeurs définies et les injectent dans le calcul en temps réel. Cela permet un ajustement rapide des paramètres sans avoir à modifier manuellement la structure du schéma.

```
Kp1 =

   10.6434   10.2527


Ki =

    0.2882
```

Après lancement de l'implémentation, le système se stabilise immédiatement en position horizontale, comme l'illustrent les photos ci-dessous. Le temps de réponse mesuré au chronomètre est de **5.78 s**, légèrement inférieur à celui obtenu en simulation (**6.2 s**), écart restant cohérent au vu des effets non modélisés (non-linéarités, saturation, délai carte dSPACE). Soumis à plusieurs perturbations manuelles, le système revient à l'équilibre en moyenne en **4.41 s**, confirmant la **robustesse et l'efficacité** du correcteur à retour d'état intégral. Ces résultats valident la performance du correcteur en condition réelle et ouvrent la voie à une extension du contrôle sur les autres axes du système.



CONCLUSION

Ce travail a permis de concevoir, simuler puis implémenter une **commande à retour d'état avec intégration** sur un système réel de type **hélicoptère**. En suivant une démarche rigoureuse, nous avons d'abord identifié les paramètres dynamiques du mouvement d'élévation à partir d'un essai en boucle ouverte, aboutissant à un modèle de second ordre fidèle. Ce modèle nous a ensuite permis de construire une représentation d'état, base indispensable à la synthèse d'un correcteur numérique.

La **discrétisation du système** via la méthode **ZOH**, suivie du **placement des pôles** dans le plan complexe discret, nous a permis d'atteindre les performances désirées en boucle fermée : **réduction de l'erreur statique**, **stabilité**, et **temps de réponse maîtrisé**. La commande intégrale s'est montrée robuste, tant en simulation qu'en situation réelle, avec une excellente capacité à rejeter les perturbations constantes.

OUVERTURES ET PERSPECTIVES

Malgré ces bons résultats, plusieurs **axes d'amélioration** peuvent être envisagés :

- L'ajout d'un **observateur d'état**, de type Luenberger ou Kalman, permettrait d'estimer les états internes non mesurés, réduisant ainsi la dépendance aux capteurs. Bien évidement il faut le système soit observable et que l'observateur soit plus rapide que la commande donc l'horizon d'observation doit être à gauche de l'horizon de commande.
- Le recours à des techniques **adaptatives** ou **robustes** (type H∞ ou sliding mode) ou les filtres de Kalman permettrait de mieux gérer les incertitudes paramétriques et non-linéarités du système réel.
- Enfin, la mise en œuvre de **commandes prédictives (MPC)** ou d'approches **à base d'intelligence artificielle** (réseaux neuronaux, apprentissage par renforcement) constitue une voie prometteuse pour les systèmes embarqués modernes à haut degré de liberté.

Ce TP a ainsi constitué une base solide pour aborder la commande de systèmes dynamiques complexes, en alliant modélisation, théorie du contrôle et expérimentation concrète.

ANNEXES – Code MATLAB

```
clc;
clear;

% === Données système continu ===
xhi = 0.0912;
w0 = 0.8038;
k = 0.3299;
T = 0.05;           % Période d'échantillonnage (s)
Tc = 1;        % Temps de réponse souhaité (s)

% === Modèle d'état continu ===
A = [0 1 ; -w0^2 -2*xhi*w0];
B = [0 ; k*w0];
C = [1 0];

% === Passage en discret ===
sys_disc = c2d(ss(A, B, C, 0), T, 'zoh');
[F, G, ~, ~] = ssdata(sys_disc);

% === systeme augmenté ===
Fe = [F, zeros(2,1); C, 1];
Ge = [G; 0];

% === placement de pôles) ===
Ae = [A, zeros(2,1); C, 0];
Poles_C_BO = eig(Ae);
Poles_C_BF = -1/Tc + 1i * imag(Poles_C_BO);
Poles_Z_BF = exp(Poles_C_BF * T);

% === Correcteur 1 ===
K1 = acker(Fe, Ge, Poles_Z_BF)
F_BF = Fe - Ge * K1;
B_echelon = [0; 0; -0.79];  % Consigne unitaire
C_y = [1 0 0];
C_u = -K1;
sys_y = ss(F_BF, B_echelon, C_y, 0, T);
sys_u = ss(F_BF, B_echelon, C_u, 0, T);

figure;
subplot(2,1,1);
step(sys_y, 10);
title('Réponse indicielle en sortie y en BF avec K1');
ylabel('y (rad)');
```

```
grid on;

subplot(2,1,2);
step(sys_u, 10);
title('Commande u(t)');
ylabel('u (V)');
xlabel('Temps (s)');
grid on;

figure;
hold on; grid on;
title('Placement des pôles pour Horizon à -1 et
dimensionnement de K1');
xlabel('Partie réelle'); ylabel('Partie imaginaire');
re_lim = [-3, 1];
im_lim = [-2, 2];
xlim(re_lim);
ylim(im_lim);
plot([-1/Tc -1/Tc], im_lim, 'r--', 'LineWidth', 1.2,
'DisplayName', sprintf('Horizon = %.2f', -1/Tc));
plot([0 0], im_lim, 'k', 'HandleVisibility', 'off');  % axe
imaginaire
plot(re_lim, [0 0], 'k', 'HandleVisibility', 'off');  % axe
réel

Poles_sym = -real(Poles_C_BO) + 1i * imag(Poles_C_BO);
Poles_C_BF = -1/Tc + 1i * imag(Poles_C_BO);  % recalcul propre

plot(real(Poles_C_BO), imag(Poles_C_BO), 'ks',
'MarkerFaceColor', 'k', 'DisplayName', 'Pôles BO');
plot(real(Poles_sym), imag(Poles_sym), 'ko',
'MarkerFaceColor', 'none', 'DisplayName', 'Pôles Stabilisé');
plot(real(Poles_C_BF), imag(Poles_C_BF), 'o', ...
    'MarkerEdgeColor', [0 0.4 0], 'MarkerFaceColor', [0 0.4
0], ...
    'LineWidth', 1.3, 'DisplayName', 'Pôles en BF');

legend('Location', 'southwest');
axis equal;
hold off;
```

```
% === Ajout d'un second correcteur avec Tc2 = 0.5 s ===
Tc2 = 0.5;
Poles_C_BF2 = -1/Tc2 + 1i * imag(Poles_C_BO);
Poles_Z_BF2 = exp(Poles_C_BF2 * T);
K2 = acker(Fe, Ge, Poles_Z_BF2)  % pas utilisé ici, juste pour
info

figure;
hold on; grid on;
title('Accélérations du système avec l horizon -2 et
dimensionnement de K2');
xlabel('Partie réelle'); ylabel('Partie imaginaire');

plot([0 0], [-2 2], 'k', 'HandleVisibility', 'off');
plot([-3 1], [0 0], 'k', 'HandleVisibility', 'off');
plot([-1 -1], [-2 2], 'r--', 'LineWidth', 1.2, 'DisplayName',
'Horizon Tc = 1s');
plot([-2 -2], [-2 2], 'b--', 'LineWidth', 1.2, 'DisplayName',
'Horizon Tc = 0.5s');

plot(real(Poles_C_BO), imag(Poles_C_BO), 'ks',
'MarkerFaceColor', 'k', 'DisplayName', 'Pôles BO');
plot(real(Poles_sym), imag(Poles_sym), 'ko',
'MarkerFaceColor', 'none', 'DisplayName', 'Pôles stabilisés');

plot(real(Poles_C_BF), imag(Poles_C_BF), 'o', ...
    'MarkerEdgeColor', [0 0.4 0], 'MarkerFaceColor', [0 0.4
0], ...
    'LineWidth', 1.3, 'DisplayName', 'Pôles BF (Tc=1s)');

plot(real(Poles_C_BF2), imag(Poles_C_BF2), 'o', ...
    'MarkerEdgeColor', [0 0 0.6], 'MarkerFaceColor', [0 0 1],
...
    'LineWidth', 1.3, 'DisplayName', 'Pôles BF (Tc=0.5s)');

xlim([-3 1]);
ylim([-2 2]);
axis equal;
legend('Location', 'southwest');
hold off;

F_BF2 = Fe - Ge * K2;
sys_y2 = ss(F_BF2, B_echelon, C_y, 0, T);  % Système avec K2
(Tc = 0.5s)
```

```
% === réponses indicielle y2(t) ===
figure;
step(sys_y2, 10);
legend( 'Tc = 0.5s');
title('Comparaison des réponses indicielle en sortie y');
xlabel('Temps (s)');
ylabel('Élévation (rad)');
grid on;
hold off;
% === Système boucle fermée pour la commande avec le second
correcteur (Tc = 0.5s) ===
C_u2 = -K2;
sys_u2 = ss(F_BF2, B_echelon, C_u2, 0, T);
figure;
step(sys_u, 10); hold on;
step(sys_u2, 10);
legend('Commande u(t) - Tc = 1s', 'Commande u(t) - Tc =
0.5s');
title('Comparaison des commandes u(t)');
xlabel('Temps (s)');
ylabel('u (V)');
grid on;
hold off;
```