Université de Namur

Faculté des sciences

Master en Sciences Mathématiques


# Systèmes et Contrôle : travail de groupe

*Auteurs*
A. Rida
A. Ryane
R. Nesrine
W. Lisa

*Professeur*
J. Winkin
*Assistants*
FG. Bierwart
J. Mohet
C. Sonveaux

2023-2024

# Table des matières

# 1 Introduction

Au cours de cette étude, notre principal objectif est d'explorer les problèmes de contrôle associés à un cas d'étude spécifique, à savoir le quadrirotor. Tout d'abord, nous avons procédé à la compréhension du modèle du système mécanique réel, suivi d'une analyse approfondie de ce système et de ses caractéristiques. De plus, nous avons réaliser des tests dynamiques pour évaluer le comportement du système. Notre drône sera le Parrot MiniDrone Rolling Spider.

La phase centrale de ce travail consistera à appliquer diverses stratégies de contrôle en vue d'atteindre des objectifs définis. Pour évaluer les performances des contrôles mis en œuvre, nous effectuerons des évaluations à deux niveaux : des analyses numériques, réalisées au moyen d'outils tels que Matlab et Simulink, ainsi que des expérimentations pratiques. Cette démarche nous permettra non seulement d'évaluer l'adéquation de nos méthodes à atteindre les objectifs, mais également d'identifier les discordances potentielles entre le modèle théorique et la réalité.

# 2 Modélisation

Dans cette partie, nous allons expliquer les étapes permettant d'obtenir la modélisation suivante pour la dynamique du quadrirotor.Notons que cette section s'est principalement basé sur les sources [1], [2] et [3].

$$
\begin{cases}
\dot{x} = v_x \\
\dot{y} = v_y \\
\dot{z} = v_z \\
\dot{v_x} = -\frac{c_f}{m} v_x + \frac{c_p}{m}(sin\psi sin\phi + cos\psi cos\phi sin\theta)(V_1^2 + V_2^2 + V_3^2 + V_4^2) \\
\dot{v_y} = -\frac{c_f}{m} v_y + \frac{c_p}{m}(cos\phi sin\psi sin\theta - cos\psi sin\phi)(V_1^2 + V_2^2 + V_3^2 + V_4^2) \\
\dot{v_z} = -\frac{c_f}{m} v_z - g + \frac{c_p}{m}(cos\theta cos\phi)(V_1^2 + V_2^2 + V_3^2 + V_4^2) \\
\dot{\phi} = \omega_\phi + \omega_\theta(sin\phi \tan\theta) + \omega_\psi(cos\phi tan\theta) \\
\dot{\theta} = \omega_\theta cos\phi - \omega_\psi sin\phi \\
\dot{\psi} = \frac{sin\phi}{cos\theta}\omega_\theta + \frac{cos\phi}{cos\theta}\omega_\psi \\
\dot{\omega_\phi} = \frac{rc_p}{I_{xx}}(V_1^2 - V_3^2) + \frac{I_{yy} - I_{zz}}{I_{xx}}\omega_\theta\omega_\psi \\
\dot{\omega_\theta} = \frac{rc_p}{I_{yy}}(V_2^2 - V_4^2) + \frac{I_{zz} - I_{xx}}{I_{yy}}\omega_\phi\omega_\psi \\
\dot{\omega_\psi} = \frac{c_t}{I_{zz}}(V_1^2 - V_2^2 + V_3^2 - V_4^2) + \frac{I_{xx} - I_{yy}}{I_{zz}}\omega_\phi\omega_\theta
\end{cases}
$$

Nous allons ensuite donner la description interne R = [A, B, C, D] correspondant à la modélisation de ce système en schématisant le système réel en premier lieu, ensuite en expliquant les variables et paramètres. Et puis nous allons développer les lois physiques permettant d'obtenir le modèle non linéaire ci-dessus. Nous allons finalement déterminer les équilibres, définir le vecteur d'état et de contrôle et déterminer le système différentiel linéaire temps invariant R = [A, B, C, D] où C est donné par:

$$C = \begin{pmatrix} 1&0&0&0&0&0&0&0&0&0&0&0 \\ 0&1&0&0&0&0&0&0&0&0&0&0 \\ 0&0&1&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&1&0&0 \\ 0&0&0&0&0&0&0&0&0&0&1&0 \\ 0&0&0&0&0&0&0&0&0&0&0&1 \end{pmatrix}$$

et la matrice D est nulle.

## 2.1 Schématisation du système

Dans cette section, nous allons présenter le fonctionnement du quadrirotor. D'abords, nous allons explorer les trois types de rotations possibles. Un aéronef en vol à la liberté de se mouvoir dans les trois dimensions, lui permettant de monter/descendre, de s'incliner à gauche/droite (Roll), de s'incliner en avant/arrière (Pitch), ou encore de pivoter sur lui-même (Yaw). Ces axes sont également connus respectivement sous les noms d'axes longitudinaux, latéraux et verticaux. Ils bougent en conjonction avec le véhicule et tournent par rapport à la terre en même temps que l'aéronef. Le quadrirotor a six degrés de liberté, comprenant trois mouvements de rotation et trois mouvements de translation.

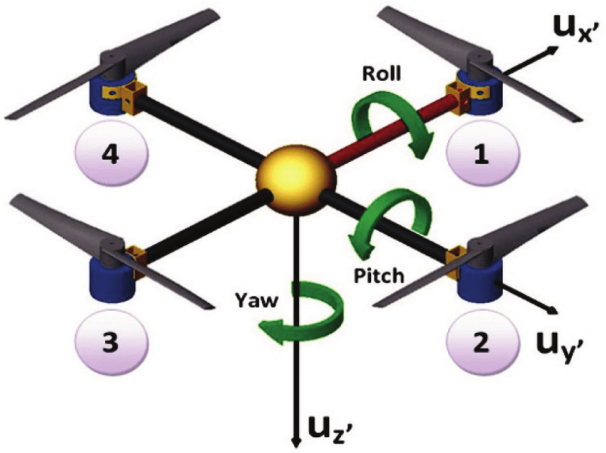


FIG. 2.1 – *Les 3 rotations d'un Quadrirotor: Lacet(Yaw), Roulis(Roll) et Tangage(Pitch)*

Le mouvement vertical est obtenu en ajustant la poussée générée par les moteurs. Augmenter la poussée fait monter le quadrirotor, tandis que la réduire le fait descendre. Et pour se déplacer horizontalement, le quadrirotor combine des mouvements de roulis et de tangage. Par exemple, incliner l'aéronef vers l'avant en combinant le roulis et le tangage permet de le faire avancer. Voici ci-dessous, une représentation des mouvements verticaux:

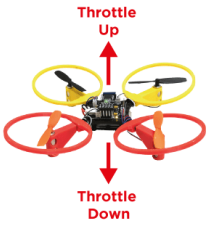


FIG. 2.2 – *Le mouvement de gaz(montée/descente)*

## 2.2 Variables et paramètres

Dans cette deuxième section, nous passerons en revue les variables et paramètres clés nécessaires pour comprendre en détail la dynamique d'un quadrirotor.

Nous avons en premier lieu les trois angles d'Euler, qui sont des paramètres cruciaux pour déterminer l'orientation du quadrirotor dans l'espace: Roll($\phi$), Pitch($\theta$) et Yaw($\psi$). Ensuite, nous avons les vitesses angulaires dans le repère mobile, ce qui signifie qu'elles sont mesurées par rapport à l'orientation actuelle du quadrirotor. Celles-ci déterminent les vitesses de rotations autour des axes: $\dot{\phi}$, $\dot{\theta}$ et $\dot{\psi}$. Enfin, nous considérons les vitesses angulaires dans le repère fixe, ce qui signifie qu'elles sont mesurées par rapport au monde extérieur et non à l'orientation du quadrirotor: $\omega_\phi$, $\omega_\theta$ et $\omega_\psi$. Ces variables et paramètres sont fondamentaux pour comprendre et contrôler la dynamique d'un quadrirotor. Ils permettent de mesurer et de contrôler l'orientation et les mouvements dans l'espace.

Ensuite, nous allons effectuer le calcul de la matrice de passage du repère fixe au repère mobile. Le repère mobile associé au quadrirotor effectue trois mouvements de rotation par rapport au repère fixe, ce qui nécessite donc la définition de trois matrices de rotation distinctes :

$$Rot_z(\psi)= \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad Rot_y(\theta)= \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix} \quad Rot_x(\phi)= \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi & \cos\phi \end{bmatrix}$$

Pour passer d'un vecteur e= $(x\ y\ z)^T$ dans le repère fixe à un autre vecteur e'=$(x'\ y'\ z')^T$ du repère mobile on utilise la matrice de passage R = $Rot_z(\psi)Rot_y(\theta)Rot_x(\phi)$. Ainsi on a e = R.e'. Voici la matrice R:

$$\mathrm{R} = \begin{bmatrix} \cos\psi\cos\theta & \sin\phi\sin\theta\cos\psi - \cos\phi\sin\psi & \cos\phi\sin\theta\cos\psi + \sin\phi\sin\psi \\ \cos\theta\sin\psi & \sin\phi\sin\theta\sin\psi + \cos\phi\cos\psi & \cos\phi\sin\theta\sin\psi - \sin\phi\cos\psi \\ -\sin\theta & \sin\phi\cos\theta & \cos\phi\cos\theta \end{bmatrix}$$

Voici également ci-dessous le calcul des vitesses angulaires dans le repère fixe qui nous sera utile. Notons que nous nous sommes basées sur la source [4] pour trouver cette relation.

$$\omega = \begin{pmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \end{pmatrix} = \begin{pmatrix} \dot{\phi} \\ 0 \\ 0 \end{pmatrix} + Rot_x(\phi)^{-1} \begin{pmatrix} 0 \\ \dot{\theta} \\ 0 \end{pmatrix} + (Rot_y(\theta)Rot_x(\phi))^{-1} \begin{pmatrix} 0 \\ 0 \\ \dot{\psi} \end{pmatrix}$$

Donc,

$$\omega = \begin{bmatrix} \dot{\phi} - \dot{\psi}\sin\theta \\ \dot{\theta}\cos\phi + \dot{\psi}\sin\phi\cos\theta \\ \dot{\psi}\cos\phi\cos\theta - \dot{\theta}\sin\phi \end{bmatrix}$$

Nous avons aussi nos 4 vitesses angulaires, c'est à dire nos moteurs, des 4 rotors qui seront nos 4 entrées: $V_1$, $V_2$, $V_3$ et $V_4$.

Maintenant, faisons une petite description des coefficients aérodynamiques. Nous avons le coefficient de Portance $c_p$ utilisé pour quantifier la portance qui est la composante perpendiculaire de la force subie par un corps placé dans un écoulement d'air. Ensuite, nous avons le coefficient de Traînée $c_t$ utilisé pour quantifier la résistance d'un objet dans un fluide. Et enfin, nous avons le coefficient de frottement de l'air $c_f$ utilisé pour quantifier le frottement fluide qui est la force que reçoit un objet se déplaçant dans un fluide.

Notons que, Drag (Trainée) et Lift (Portance) sont les deux composantes de la force aérodynamique. La force aérodynamique est la résistance que l'air exerce sur un véhicule en mouvement. La composante de Trainée (Drag) est la force qui s'oppose au mouvement du véhicule dans la direction du mouvement. Elle est responsable de la décélération du véhicule. La composante de Portance (Lift), en revanche, est perpendiculaire à la direction du mouvement. Elle est responsable de la sustentation du véhicule dans les airs, comme c'est le cas pour les ailes d'un avion. La portance permet au véhicule de s'élever contre la gravité. Ces deux composantes de la force aérodynamique sont essentielles pour comprendre comment un véhicule interagit avec l'air qui l'entoure.


FIG. 2.3 – *La dynamique d'un Quadrirotor*

## 2.3 Lois physiques

Nous allons dans cette section 2.3 définir les lois physiques qui nous permettront d'obtenir le système donné. Voici tout d'abord les spécifications des forces agissant sur le quadrirotor qui influencent son comportement en vol. Parmi ces forces, nous avons le poids (Weight). Il s'agit de la force gravitationnelle qui attire le quadrirotor vers le bas en raison de sa masse. Nous avons ensuite la force de poussée du rotor (Thrust) générée par les moteurs du quadrirotor. Elle est dirigée vers le haut et équilibre la force de poids, permettant ainsi au quadrirotor de s'élever dans les airs. Le contrôle de la poussée est essentiel pour le décollage, l'atterrissage et le maintien en altitude. De plus, nous avons la portance (Lift) qui est la force générée par l'aérodynamique des rotors ou des hélices. Elle s'exerce perpendiculairement à la direction du mouvement et permet au quadrirotor de maintenir son élévation. La portance est essentielle pour le vol en maintenant le quadrirotor en l'air. Il y a également la trainée (Drag) qui est la résistance de l'air qui s'oppose au mouvement du quadrirotor. Elle agit dans la direction opposée au mouvement du véhicule. La gestion de la traînée est importante pour contrôler la vitesse. Et enfin nous avons le mouvement (Motion) faisant référence aux forces internes générées par les moteurs et les contrôles du drône pour effectuer des manœuvres et des déplacements dans les 3 dimensions de l'espace. Ces forces sont essentielles pour comprendre la dynamique de vol d'un quadrirotor. Voici ci-dessous un dessin qui représente les forces que nous venons de citer:

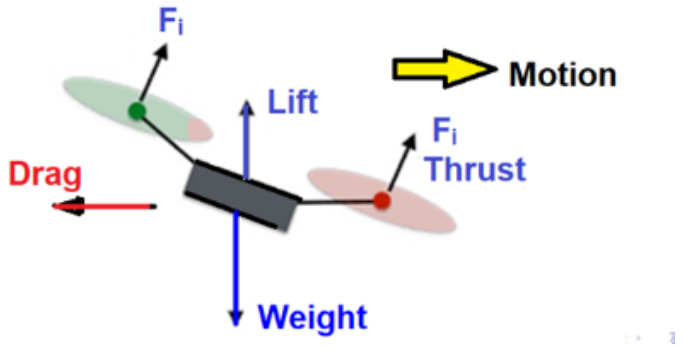


FIG. 2.4 – *Les forces*

Nous avons également d'autres forces comme la gravité $P = mg$ où m est la masse du quadrirotor et g la pesanteur. Il y a aussi les forces de poussées $F_i = c_p V_i^2$ avec $i$ = 1,2,3,4 où $V_i$ est vitesse de rotation du rotor i et $c_p$ le coefficient de portance des hélices. Nous avons aussi les forces de traînée des hélices $T_{hi} = c_t V_i^2$ avec $i$ = 1,2,3,4 où $V_i$ est la vitesse de rotation du rotor i et $c_t$ le coefficient de traînée des hélices.

## 2.4 Points d'équilibres et linéarisation

Nous allons déterminer nos points d'équilibres. Pour celà, on pose $X^* = f(X,U)$ notre système d'équations où $X^\intercal = \left[x,y,z,v_x,v_y,v_z,\phi,\theta,\psi,\omega_\phi,\omega_\theta,\omega_\psi\right]$, $U^\intercal = \left[V_1^2,V_2^2,V_3^2,V_4^2\right]$ et $f$: $\mathbb{R}^{12} \times \mathbb{R}^4 \to \mathbb{R}^{12}$. Nous avons que $X^* \in \mathbb{R}^{12}$ et $U^* \in \mathbb{R}^4$ est un point d'équilibre du système $\iff$ f(X*,$U^*$) = $[0]_{12\times1}$. Voici donc ci-dessous le système à résoudre:

$$\begin{cases}
\dot{x} = \mathrm{v}_x = 0 \Longrightarrow \mathrm{x} = \mathrm{x}_0 \text{ avec } x_0 \text{ une constante} & (1)\\
\dot{y} = \mathrm{v}_y = 0 \Longrightarrow \mathrm{y} = \mathrm{y}_0 \text{ avec } y_0 \text{ une constante} & (2)\\
\dot{z} = \mathrm{v}_z = 0 \Longrightarrow \mathrm{z} = \mathrm{z}_0 \text{ avec } z_0 \text{ une constante} & (3)\\
\dot{v_x} = 0 \Longrightarrow sin(\psi)sin(\phi) + cos(\psi)cos(\phi)sin(\theta) = 0 & (4)\\
\dot{v_y} = 0 \Longrightarrow cos(\phi)sin(\psi)sin(\theta) - cos(\psi)sin(\phi) = 0 & (5)\\
\dot{v_z} = 0 \Longrightarrow -g + \frac{c_p}{m}(cos(\theta)cos(\phi))(V_1^2 + V_2^2 + V_3^2 + V_4^2) = 0 & (6)\\
\dot{\phi} = 0 \Longrightarrow \omega_\phi + \omega_\theta(sin(\phi)tan(\theta)) + \omega_\psi(cos(\phi)tan(\theta)) = 0 & (7)\\
\dot{\theta} = 0 \Longrightarrow \omega_\theta cos(\phi) - \omega_\psi sin(\phi) = 0 & (8)\\
\dot{\psi} = 0 \Longrightarrow \frac{sins(\phi)}{cos(\theta)}\omega_\theta + \frac{cos(\phi)}{cos(\theta)}\omega_\psi = 0 & (9)\\
\dot{\omega_\phi} = 0 \Longrightarrow \frac{rc_p}{I_{xx}}(V_1^2 - V_3^2) + \frac{I_{yy}-I_{zz}}{I_{xx}}\omega_\theta\omega_\psi = 0 & (10)\\
\dot{\omega_\theta} = 0 \Longrightarrow \frac{rc_p}{I_{xx}}(V_2^2 - V_4^2) + \frac{I_{zz}-I_{xx}}{I_{yy}}\omega_\phi\omega_\psi = 0 & (11)\\
\dot{\omega_\psi} = 0 \Longrightarrow \frac{c_t}{I_{zz}}(V_1^2 - V_2^2 + V_3^2 - V_4^2) + \frac{I_{xx}-I_{yy}}{I_{zz}}\omega_\phi\omega_\theta = 0 & (12)
\end{cases}$$

Par l'équation (4) et (5), on a ce système:

$$\begin{cases}
sin(\psi)sin(\phi) + cos(\psi)cos(\phi)sin(\theta) = 0 & (a)\\
cos(\phi)s(\psi)sin(\theta) - cos(\psi)sin(\phi) = 0 & (b)
\end{cases} \tag{2.1}$$

On multiplie (a) par $cos(\psi)$ et on multiplie (b) par $sin(\psi)$, et en additionnant, on obtient:

$$2cos(\phi)sin(\theta) = 0$$

$$\Longrightarrow \phi = k\pi + \tfrac{\pi}{2} \text{ ou } \theta = k\pi$$

Prenons d'abord le premier cas où $\phi = k\pi + \frac{\pi}{2}$ et remplaçons dans (6):

$$-g + \frac{c_p}{m}(cos(\theta)cos(k\pi + \frac{\pi}{2}))(V_1^2 + V_2^2 + V_3^2 + V_4^2) = 0$$

On obtient donc que $g = 0$ mais ceci est impossible car la pesanteur ne peut pas être nulle. Donc notre premier cas n'a pas de solutions.

Prenons donc notre deuxième cas qui est que $\theta = k\pi$. Dans ce cas, le système d'équations 2.1 devient:

$$\begin{cases} \sin(\psi)sin(\phi) = 0 \Longrightarrow \psi = k\pi \text{ ou } \phi = k\pi \\ \text{-}\cos(\psi)sin(\phi) = 0 \Longrightarrow \psi = k\pi + \frac{\pi}{2} \text{ ou } \phi = k\pi \end{cases}$$

Et comme $\psi$ ne peut pas être égale en même temps à $k\pi$ et $k\pi + \frac{\pi}{2}$, la seule solution possible est $\phi = k\pi$. En remplaçant les valeurs de $\theta$ et $\psi$ dans l'équation (6):

$$g = \frac{c_p}{m}((-1)^{k+k'})(V_1^2 + V_2^2 + V_3^2 + V_4^2)$$

Or g ne peut pas être négatif donc $(-1)^{k+k'} = 1$ obligatoirement, c'est à dire que $cos(\theta)cos(\psi) = 1$. Et comme $\text{-}\frac{\pi}{2} \leqslant \phi \leqslant \frac{\pi}{2}$, $\text{-}\frac{\pi}{2} \leqslant \theta \leqslant \frac{\pi}{2}$ et $\text{-}\frac{\pi}{2} \leqslant \psi \leqslant \frac{\pi}{2}$, les seules valeurs de $\phi$ et $\theta$ qui vont vérifier $cos(\theta)cos(\psi) = 1$ sont $(\phi,\theta) = (0,0)$.

Nous avons ensuite remplacer nos angles dans nos équations (7), (8) et (9), et nous obtenons:

$$\begin{cases} \omega_\psi = 0 \\ \omega_\phi = 0 \\ \omega_\theta = 0 \end{cases}$$

Avec ceci et après avoir eu des simplification dans nos équations (10), (11) et (12) on obtient: $V_1^2 = V_2^2 = V_3^2 = V_4^2$. On a donc par (6):

$$g = \frac{c_p}{m}(4V_1^2) \Longrightarrow V_1^2 = \frac{mg}{4c_p}$$

Les points d'équilibres sont par conséquent:

$$X^{*\intercal} = \left[x_0,y_0,z_0,0,0,0,0,0,\psi_0,0,0,0\right]^\intercal$$

et

$$U^{*\intercal} = \left[\tfrac{mg}{4c_p},\tfrac{mg}{4c_p},\tfrac{mg}{4c_p},\tfrac{mg}{4c_p}\right]^\intercal$$

tel que $x_0$, $y_0$, $z_0$ sont des positions quelconques et $\psi_0$ est un angle quelconque.
Après avoir trouver les équilibres, on va imposer de petites déviations à notre système:

$$\begin{cases} \delta_X(t) = X(t) - X^* \text{ avec } X^* \text{ une position d'équilibre} \\ \delta_U(t) = U(t) - U^* \text{ avec } U^* \text{ l'entrée à l'équilibre} \end{cases}$$

On sait que X et U sont liées par l'équation différentielle suivante:

$$\dot{X} = f(X,U)$$

Donc en substituant ces deux relations, on obtient:

$$\dot{\delta}_X(t) = f(X^* + \delta_X(t),U^* + \delta_U(t))$$

Notre objectif est de linéariser le système de façon à trouver un système linéaire sous cette forme:

$$\dot{\delta}_X(t) = A\delta_X(t) + B\delta_U(t) \tag{2.2}$$

dont les matrices A et B sont les deux matrices qui constituent notre représentation d'état du système. Nous allons appliquer le développement en série de Taylor sur la fonction f avec le point d'équilibre $X^*$ avec $U^*$ qui est un vecteur d'entrée pour lequel l'équilibre existe, et nous allons négliger les termes d'ordre 2:

$$\dot{\delta}_X(t) \approx f(X^*,U^*)+\frac{\partial f}{\partial X}(X^*,U^*)\delta_X(t)+\frac{\partial f}{\partial U}(X^*,U^*)\delta_U(t) \tag{2.3}$$

$$\dot{\delta}_X(t) \approx \frac{\partial f}{\partial X}(X^*,U^*)\delta_X(t)+\frac{\partial f}{\partial U}(X^*,U^*)\delta_U(t) \tag{2.4}$$

Il s'agit donc bien d'un système linéaire temps invariant, car il y une relation linéaire dans le système 2.3 entre $\delta_X(t)$ et $\delta_U(t)$ et on a, de plus, en faisant une analogie avec le système 2.2:

$$A = \frac{\partial f}{\partial X}(X^*,U^*) \in \mathbb{R}^{12\times 12}$$

$$B = \frac{\partial f}{\partial U}(X^*,U^*) \in \mathbb{R}^{12\times 4}$$

Cette linéarisation autour des points d'équilibres nous permet d'étudier la relation linéaire qui existe entre les déviations $\delta_X$ et $\delta_U$ quand on oscille autour de l'équilibre ($X^*$, $U^*$). Voici les matrices A et B. On pose $U^{*\intercal} = [V^{*2}_1,V^{*2}_2,V^{*2}_3,V^{*2}_4]$

$$A = \begin{pmatrix}
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & -\frac{c_f}{m} & 0 & 0 & a & b & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & -\frac{c_f}{m} & 0 & -b & a & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & -\frac{c_f}{m} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{pmatrix} \quad B = \begin{pmatrix}
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
\frac{c_p}{m} & \frac{c_p}{m} & \frac{c_p}{m} & \frac{c_p}{m} \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
\frac{rc_p}{I_{xx}} & 0 & -\frac{rc_p}{I_{xx}} & 0 \\
0 & \frac{rc_p}{I_{yy}} & 0 & -\frac{rc_p}{I_{yy}} \\
\frac{c_t}{I_{zz}} & -\frac{c_t}{I_{zz}} & \frac{c_t}{I_{zz}} & -\frac{c_t}{I_{zz}}
\end{pmatrix}$$

avec $a = g\sin(\psi_0)$ et $b = g\cos(\psi_0)$

# 3 Analyse

Dans cette deuxième partie, nous allons entreprendre une analyse approfondie du modèle en utilisant Matlab. Cette analyse vise à explorer des propriétés cruciales du modèle que nous avons établi. De plus, nous allons illustrer la dynamique du système en examinant le comportement des trajectoires d'état et de sortie en réponse à divers types d'entrées et/ou d'états initiaux. Nous aborderons également la sélection de la matrice d'observabilité, en discutant du choix optimal et des alternatives réalisables pour enrichir notre analyse. L'analyse dynamique du modèle comprendra les éléments suivants : la stabilité interne et externe, la contrôlabilité et l'observabilité, la stabilisabilité et la détectabilité. De plus, nous allons réaliser des tests dynamiques en boucle ouverte, comprenant les trajectoires d'état libres ainsi que la réponse impulsionnelle et indicielle.

## 3.1 Stabilité interne

Par définition un système R = [A,B,C,D] est internement stable $\iff \forall\lambda \in \sigma(A), Re(\lambda) < 0$. On a obtenu comme vecteur des pôles ceci: (0, 0, 0, -3.6765, -3.6765, -3.6765, 0, 0, 0, 0, 0, 0). On voit donc qu'on a 9 valeurs propres égales à 0 donc le sytème n'est pas internement stable. Dans le contexte d'interprétation physique, cela suggère que le mouvement du drone n'est pas stable. Cela veut dire que le drone a tendance à diverger de la position d'équilibre.

### 3.1.1 Sous-espace stable

Le sous-espace stable $L^-(A) = \text{Span}\{v_1, v_2, v_3\}$, il contient 3 vecteurs linéairement indépendants, il est donc de dimension 3.

```
v1=[-0.2625,0,0,0.9649,0,0,0,0,0,0,0,0]

v2=[0,-0.2625,0,0,0.9649,0,0,0,0,0,0,0]

v3=[0,0,-0.2625,0,0,0.9649,0,0,0,0,0,0]
```

### 3.1.2 Sous-espace instable

Le sous-espace instable $L^{0+}(A) = \text{Span}\{e_1, e_2, e_3, e_9\}$, il contient 4 vecteurs de la base canonique linéairement indépendants, il est donc de dimension 4.

## 3.2 Stabilité externe

Soit $\hat{G}(s) = C(sI - A)^{-1}B + D$.
Le système R = [A,B,C,D] est externement stable $\iff P[\hat{G}] \subset \mathring{\mathbb{C}}_-$ .
On a dans la fonction de transfert au moins un pole égal à 0 donc le sytème n'est pas externement stable. Dans le cas de notre drône, cela signifie que le comportement du drone dans l'espace peut ne pas être stable sans une intervention active du système de contrôle pour maintenir une position stable dans l'environnement externe. En d'autres termes, le drone peut être sujet à des perturbations externes telles que des rafales de vent, des turbulences atmosphériques, etc. La stabilité externe est cruciale pour garantir que le

drone peut maintenir sa position souhaitée dans un environnement changeant, même en présence de perturbations.

## 3.3 Contrôlabilité

Définissons tout d'abord la matrice de contrôlabilité $\mathcal{C} = [B(AB)(A^2B)...(A^{n-1}B)]$

On sait que (A,B) c.c. $\Leftrightarrow rg(\mathcal{C}) = n$. On a obtenu à l'aide de Matlab que le rang valait 12 ce qui implique que le système est contrôlable. La contrôlabilité d'un système dynamique signifie la capacité à atteindre n'importe quel état souhaité à partir d'un état initial donné en utilisant des entrées de contrôle appropriées.

### 3.3.1 Sous-espace contrôlable

Pour ce qui est du sous-espace contrôlable, on a que C(A,B) = Im$\mathcal{C}$ contenant 12 vecteurs linéairement indépendants donc C(A,B) = $\mathbb{R}^{12}$

## 3.4 Observabilité

Définissons tout d'abord la matrice d'observabilité: $\mathcal{O} = [CCACA^2...CA^{n-1}]^T$.

Un théorème nous dit que (C,A) c.o. $\Leftrightarrow rg(\mathcal{O}) = n$ ce qui n'est pas le cas dans notre système, car le rang de la matrice d'observabilité est égale à 11 $\Rightarrow$ le rang n'est pas plein et donc le système n'est pas observable.
L'observabilité d'un système dynamique est liée à la capacité de déterminer l'état du système à partir des sorties mesurées. Un système est dit observable s'il est possible de reconstruire l'état du système à partir des sorties mesurées. Dans le contexte du drone, l'observabilité est importante pour garantir que nous pouvons estimer et reconstruire l'état actuel du drone.

### 3.4.1 Sous-espace inobservable

Ensuite, le sous-espace inobservable NO(C,A) = Span$\{e_9\}$ ce qui était prévu car le système n'est pas obsevable et que la dimension de ce sous espace est égale à 1.
Ici, $e_9$ est un vecteur d la base canonique de $\mathbb{R}^{12}$.

## 3.5 Stabilisabilité

Selon un théorème, (A,B) est stabilisable $\Leftrightarrow \forall s \in \mathbb{C}_+$,rg $\begin{bmatrix} sI - A & B \end{bmatrix} = n$. Le rang vaut 12 donc le sytème est stabilisable. La stabilisabilité est une propriété qui indique la possibilité de rendre le système stable en appliquant un contrôle approprié. Un système est dit stabilisable s'il existe un contrôle de rétroaction qui, lorsqu'il est appliqué au système, rend le système stable, c'est-à-dire que toutes les valeurs propres du système peuvent être placées dans la partie gauche du plan complexe.

### 3.5.1 Sous-espace stabilisable

Passons maintenant au sous-espace stabilisable défini comme :

$$\text{S(A,B)} = \text{C(A,B)} + \mathcal{L}^-(A)$$

Comme le sous espace controlable est de rang plein, alors le sous espace stabilisable est égale à $\mathbb{R}^{12}$

## 3.6 Détectabilité

Un théorème nous dit que (A,B) est détectable $\Leftrightarrow \forall s \in \mathbb{C}_+$,rg $\begin{bmatrix} sI - A \\ C \end{bmatrix} = n$

Le rang vaut 11 donc le système n'est pas détectable. La détectabilité est souvent liée à la matrice d'observabilité. Si le système est observable, alors il est également détectable. Si un système est non détectable, cela signifie que certaines composantes de l'état ne peuvent pas être déterminées à partir des sorties mesurées, ce qui peut rendre difficile ou impossible la conception d'un système de contrôle efficace. Mais donc, dans le cas de notre drone, la détectabilité concerne la capacité de déterminer l'état du drone à partir des sorties mesurées. Ces propriétés sont cruciales pour concevoir des systèmes de contrôle robustes et efficaces.

### 3.6.1 Sous-espace indétectable

Passons désormais au sous-espace indétectable qui est défini par:

$$ND(C,A) = NO(C,A) \cap \text{L}^{0+}(A)$$

Comme la base du sous espace inobservable est de dimension 1 et contient le Span$\{e_9\}$ et la base du sous espace instable est engendré par Span$\{e_1,e_2,e_3,e_9\}$ avec les $e_i$ vecteurs de la base canonique de $\mathbb{R}^{12}$ alors ND(C,A) = Span$\{e_9\}$

## 3.7 Trajectoires d'état libres

Pour tracer les trajectoires d'états libres de notre drone, on a résolu l'équation différentielle $\dot{X}(t) = AX(t)$ à l'aide de la fonction intégrée sur Matlab **ode45** et on a trouvé les graphes suivants:

### 3.7.1 Nos graphes

**Trajectoires d'état libres: Modes stables**

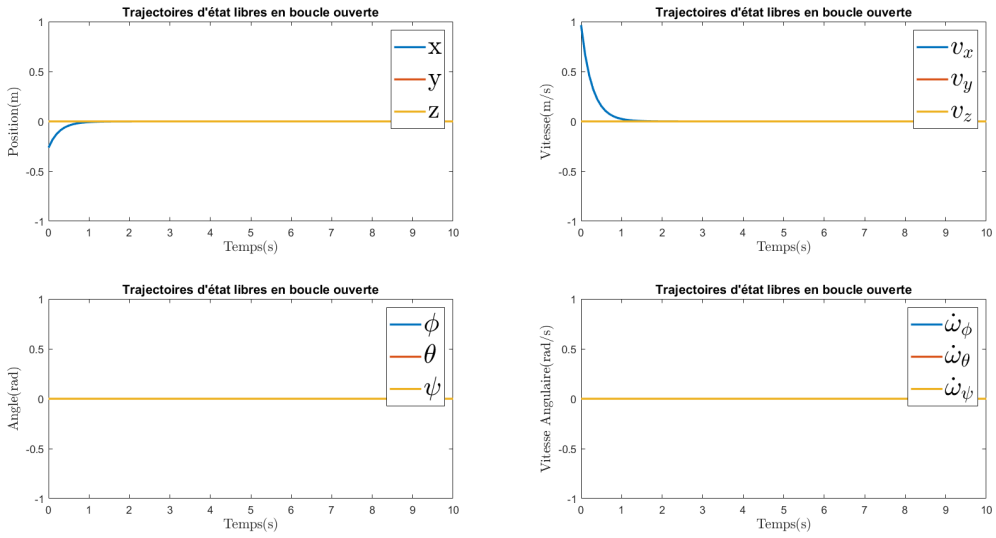


FIG. 3.1 – *Trajectoires d'états en boucle ouverte avec un vecteur propre associé à une valeur propre à partie réelle strictement négative (stable)*

**Trajectoires d'état libres: Modes instables**

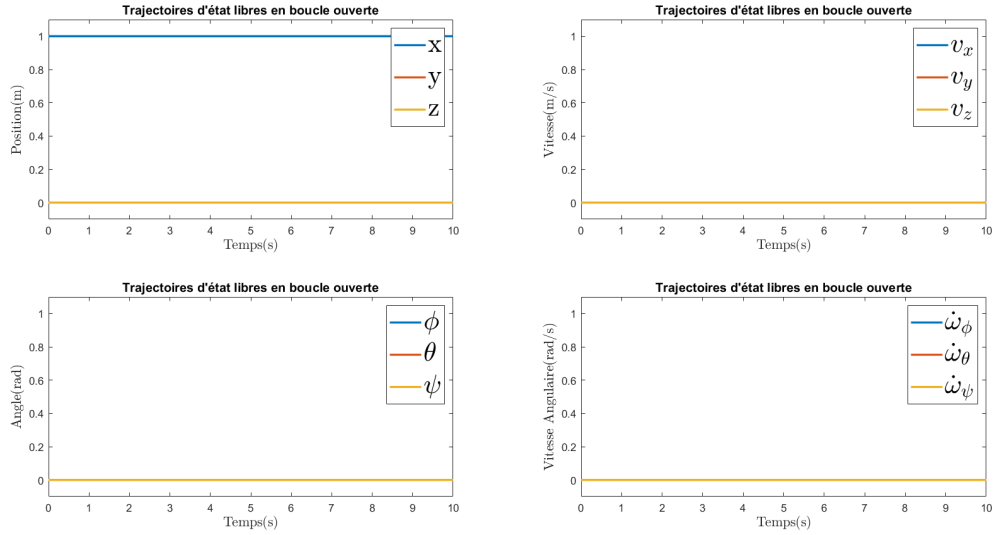


FIG. 3.2 – *Trajectoires d'états en boucle ouverte avec un vecteur propre associé à une valeur propre à partie réelle positive (instable)*

### 3.7.2 Interprétations

Nous remarquons ci-dessus à la figure 3.1 que nos trajectoires d'états tendent toutes vers 0 car nous avons pris le cas d'un vecteur propre associé à la valeur propre négative. On a donc que notre modèle est stable dans ce cas. Ensuite sur les graphiques de la figure 3.2, nos trajectoires d'états ne tendent pas toutes vers 0, x tend vers 1, car nous avons pris le cas d'un vecteur propre associé à la valeur propre positive. Nous avons donc que notre modèle est instable.

## 3.8 Réponse indicielle

Passons à la réponse indicielle. Rappelons que la réponse indicielle est la réponse d'un système dynamique à une entrée échelon unitaire. Les graphiques ci-dessous montrent la réponse indicielle de chaque moteur du quadrirotor en fonction du temps en boucle ouverte. Les réponses varient selon les moteurs, ce qui peut indiquer des différences dans les caractéristiques ou la performance de chaque moteur. Par exemple, si un moteur a une réponse en z qui augmente constamment, cela pourrait signifier qu'il est plus puissant ou moins amorti que les autres. Ces graphiques sont essentiels pour évaluer comment chaque moteur individuel contribue à la dynamique globale du système et pour identifier les performances dynamiques du système en analysant des comportements tels que le dépassement, le temps de montée, ou le temps de stabilisation.
On remarque que la réponse indicielle n'est pas bornée car par exemple pour z on voit qu'elle diverge vers l'infini. Pour les x y et $\psi$, le système diverge également pour certains moteurs. On peut donc conclure que le système est **instable**.

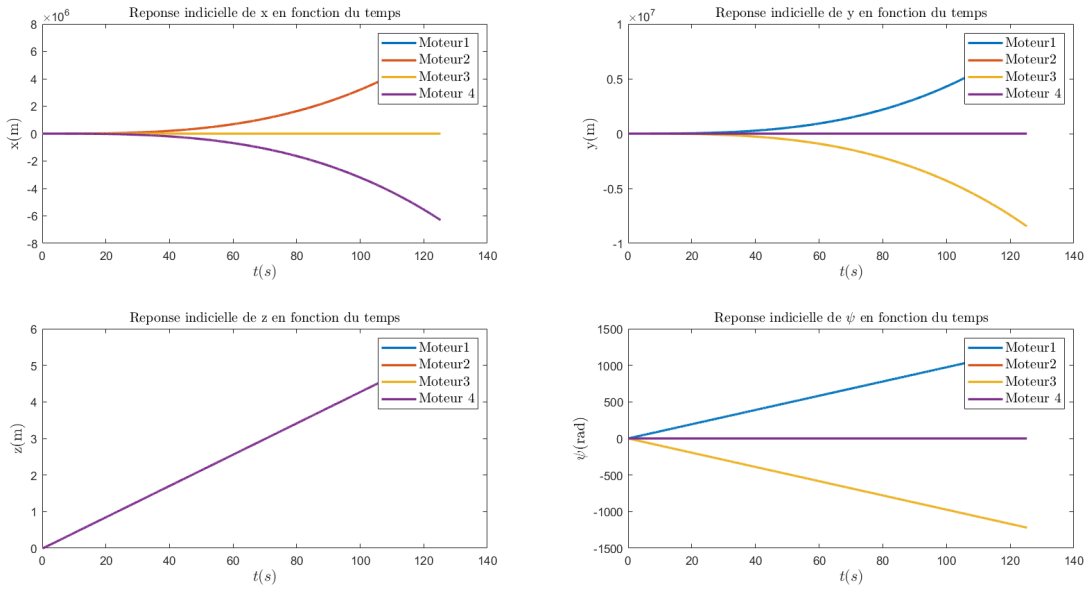


FIG. 3.3 – *Réponse indicielle en boucle ouverte*

## 3.9 Réponse impulsionnelle

Les graphiques suivants de la réponse impulsionnelle montrent la réaction des moteurs à une impulsion de dirac. Ces graphiques sont utilisés pour analyser la stabilité, la rapidité et la précision avec lesquelles les systèmes répondent aux entrées données. Sur ces graphes ci-dessous représentant la réponse impulsionnelle des 4 moteurs pour les positions en x, y et z ainsi que pour l'angle $\psi$ Nous pouvons remarquer que les réponses impulsionnelles ne convergent pas toutes vers 0, donc notre système est **instable**.

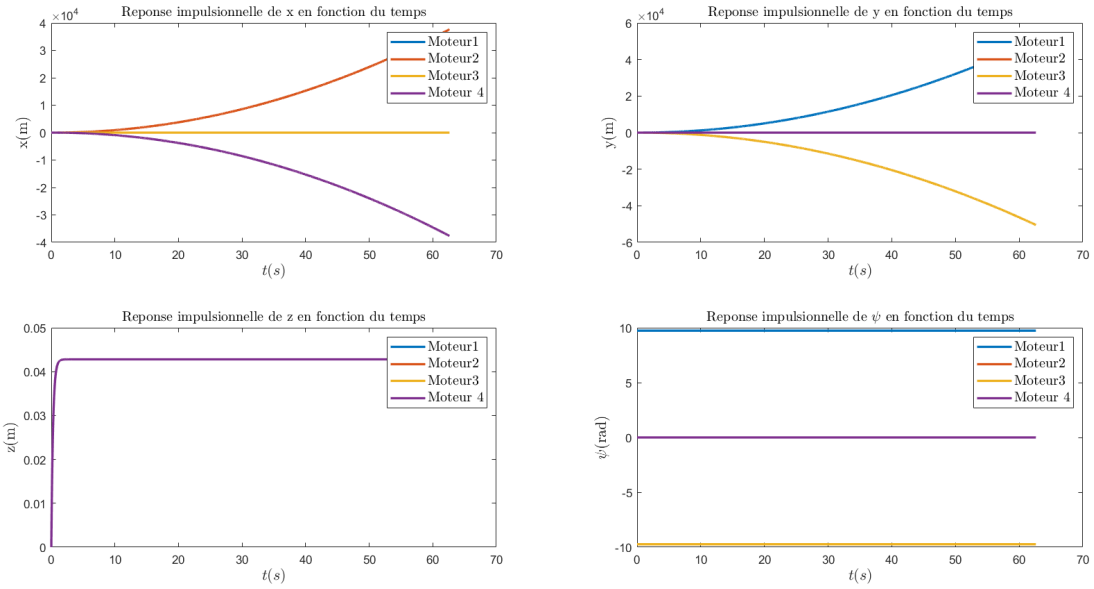


FIG. 3.4 – *Réponse impulsionnelle en boucle ouverte*

## 3.10 Choix de C optimale

En prenant la matrice qu'il nous a été donnée, nous nous sommes rendus compte que l'observabilité et la détectabilité n'étaient pas respectées. C'est pour celà que nous avons dû trouver une matrice C optimale qui respecterait ceci. Nous l'avons trouvé en s'inspirant de la source [] et en considérant les 6 capteurs du drône "Parrot MiniDrone Rolling Spider":

– Un capteur ultrasonique qui mesure la distance entre le drone et le sol ou d'autres obstacles en envoyant des ondes sonores qui rebondissent et reviennent.

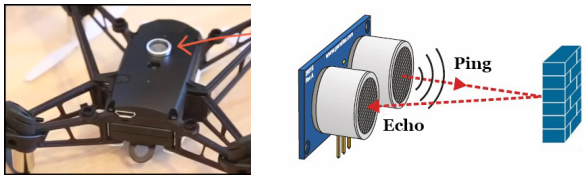


FIG. 3.5 – *Le capteur ultrasonique*

– Une centrale à inertie qui comporte:
  – Un gyroscope à 3 axes mesurant l'orientation du drone.
  – Un accéléromètre à 3 axes qui mesure l'accélération du drone dans toutes les directions.

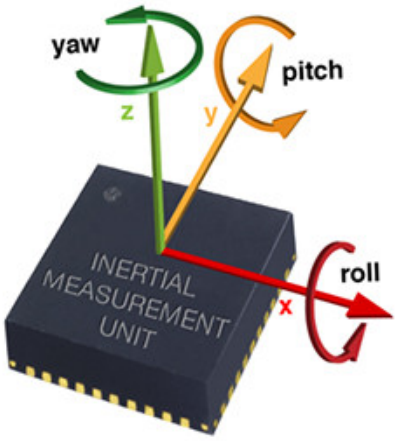


FIG. 3.6 – *La centrale d'inertie*

– Un capteur de pression qui mesure la pression atmosphérique pour déterminer l'altitude du drone.
– Une caméra pour déterminer le mouvement horizontal et rotationnel.

FIG. 3.7 – *La caméra*

Ainsi, on utilise le capteur ultrasonique et le capteur de pression pour déterminer l'altitude du drone et on utilise la centrale d'inrtie(IMU) et la caméra pour déterminer le mouvement horizontal et rotationnel.

Pour garantir l'observabilité de notre système on a choisi cette matrice C avec 4 sorties $x$, $y$, $z$ et $\psi$, car pour estimer les 12 états du drone il suffit de connaître ces 4 variables de sortie qu'on va mesurer avec les capteurs.. Les 8 autres variables on les obtient à partir de la dynamique du système (équations différentielles du mouvement)

$$C = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix}$$

Maintenant avec cette nouvelle matrice C, notre système est observable.

# 4 Asservissement d'état

Dans cette partie, nous allons concevoir des lois de contrôle d'asservissement d'état pour stabiliser notre drone. Nous allons le faire en premier lieu par l'assignation spectrale via la commande "place" sur matlab et d'autre part via l'assignation spectrale par la méthode des LMIs. Nous allons ensuite évaluer les performances du système en boucle fermée pour chaque méthode, en étudiant les trajectoires d'état, les réponses impulsionnelle et indicielle, ainsi que la robustesse du système face aux perturbations.

## 4.1 Définition

Commençons par rappeler ce qu'est un asservissement d'état. C'est un contrôle u s'écrivant sous la forme:

$$u = Kx + v$$

où K est la matrice de gain et $v$ l'entrée extérieure. Donc notre but va être de trouver une matrice K de sorte que A + BK soit stable dans le système à boucle fermée.

## 4.2 Assignabilité spectrale par le placement de pôles

La première méthode est le placement de pôles qui calcule la matrice de gain appropriée pour garantir la stabilité du système en fixant des pôles. En changeant les pôles, nous pouvons affecter la façon dont l'énergie est dissipée. Nous avons choisis nos nouveaux pôles comme suit: on cherche à ce que les valeurs propres soient désormais toutes à partie réelle négative. De plus, si on prend un pôle proche de 0, la convergence risque d'être très rapide. Si les valeurs propres sont à partie complexe, il y aura beaucoup d'oscillations. Et si les pôles sont éloignées de l'axe imaginaire ça va créer des overshoots. Afin de satisfaire ces spécifications, on a assigné ces nouveaux pôles à notre système:

```
new_poles=[-3.1000,-3.2000,-3.3000,-0.6765,-0.6765,-0.6765,
 -3.4000,-3.5000,-3.6000,-3.7000,-3.8000,-3.9000]
```

On a trouvé une matrice d'asservissement K qui stabilise le système et qui satisfait les performances dynamiques souhaitées, le choix de ces pôles a assuré la stabilité du système et a diminué les oscillations tout en ayant des temps de montée et des overshoots raisonnables. On tient aussi à préciser que le contrôle qui résulte de l'assignabilité spectrale u s'écrit sous la forme:

$$u = v - K_{place}x$$

où $K_{place}$ est la matrice d'asservissement obtenu avec l'assignabilité spectrale

## 4.3 Assignabilité spectrale par la méthode des LMIs

Les inégalités matricielles linéaires jouent un rôle important dans de nombreux problèmes de théorie de contrôle. Le but est de réécrire le problème d'assignabilité spectrale à l'aide des LMIs (Inéquations Matricielles Linéaires).
Dans ce contexte, les LMI sont utilisées pour définir les contraintes sur les matrices de gain afin de garantir la stabilité et d'autres propriétés souhaitées du système en boucle fermée. L'objectif est généralement d'assigner les pôles du système en boucle fermée à des emplacements spécifiques dans le plan complexe pour répondre à des critères de performance prédéfinis. Les LMI sont un ensemble de conditions formulées sous forme d'inégalités matricielles. Elles sont exprimées comme suit :

Soient $X_1 \dots X_n$ des variables matricielles. Soit H($X_1 \dots X_n$) une fonction matricielle. L'inégalité matricielle H($X_1 \dots X_n$)$<0$ ou $>0$ est une LMI si H est symétrique et affine par rapport aux $X_1 \dots X_n$ qui sont appelées variables de décisions.

Ainsi, la condition selon laquelle les pôles d'un système sont situés dans une région donnée du plan complexe peut être formulée comme six contraintes LMI qu'on a implémenté sur Matlab à l'aide de **LMI editor** :

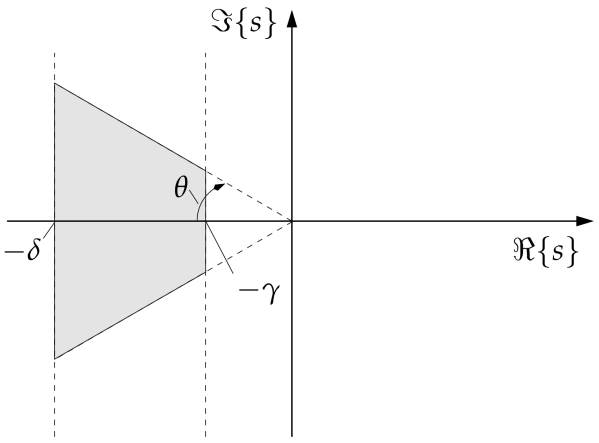


FIG. 4.1 – *La région des pôles dans le plan complexe*

LMI 1: elle est typiquement utilisée pour assurer la stabilité d'un système

$$AW + WA^T + BY + Y^T B + 2\alpha W \geq 0$$

LMI 2: elle assure que la matrice W est semi-définie positive

$$W \geq 0$$

LMI 3:

$$\begin{pmatrix} -r * W & WA^T + Y^T B^T \\ AW + BY & -r * W \end{pmatrix} \geq 0$$

LMI 4: elle est relative à la fréquence d'oscillation

$$\begin{pmatrix} sin(\theta)(AW + WA^T + BY + Y^T B^T) & cos(\theta)(AW - WA^T + BY - Y^T B^T) \\ cos(\theta)(-AW + WA^T - BY + Y^T B^T) & sin(\theta)(AW + WA^T + BY + Y^T B^T) \end{pmatrix} \geq 0$$

LMI 5: elle impose une contrainte sur les entrées avec la condition initiale

$$\begin{pmatrix} 1 & x_0^T \\ x_0 & W \end{pmatrix} \geq 0$$

LMI 6: elle impose une contrainte sur le voltage u = 3.7

$$\begin{pmatrix} u_{\max}^2 * I_n & Y \\ Y^T & W \end{pmatrix} \geq 0$$

Pour celà, on utilise le théorème de Lyapunov qui nous dit que si on a un système $\dot{x} = Ax$ alors on peut garantir qu'il existe une matrice P symétrique définie positive tel que $A^T P$ est défini positif. Ainsi, le contrôle $u$ qui résulte de la méthode des LMIs s'écrit sous la forme:

$$u = v + K_{lmi} x$$

où $K_{lmi}$ est la matrice d'asservissement obtenu avec la méthode des LMIs

## 4.4 Nos graphes

**Trajectoire d'états** :

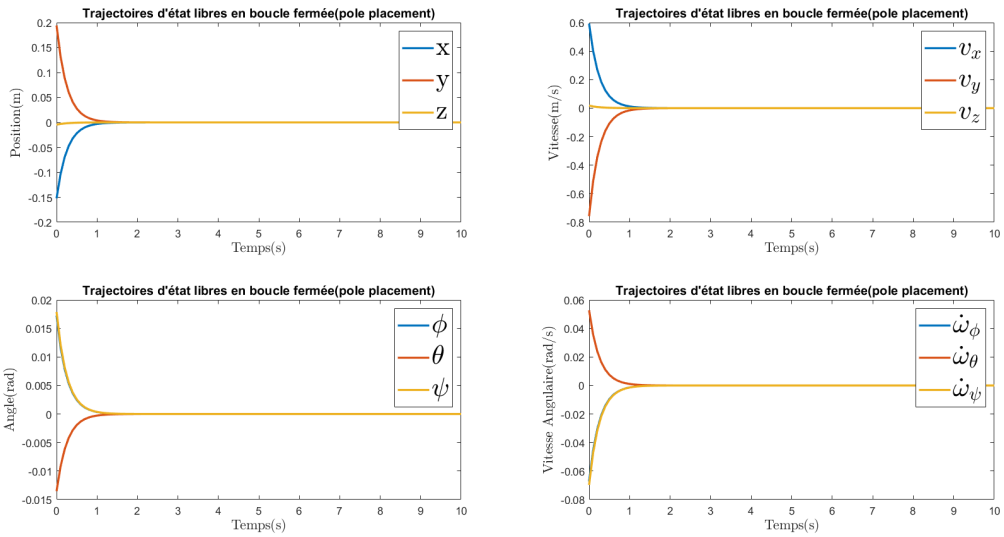


FIG. 4.2 – *Trajectoires d'états en boucle fermée avec placement de pôles*

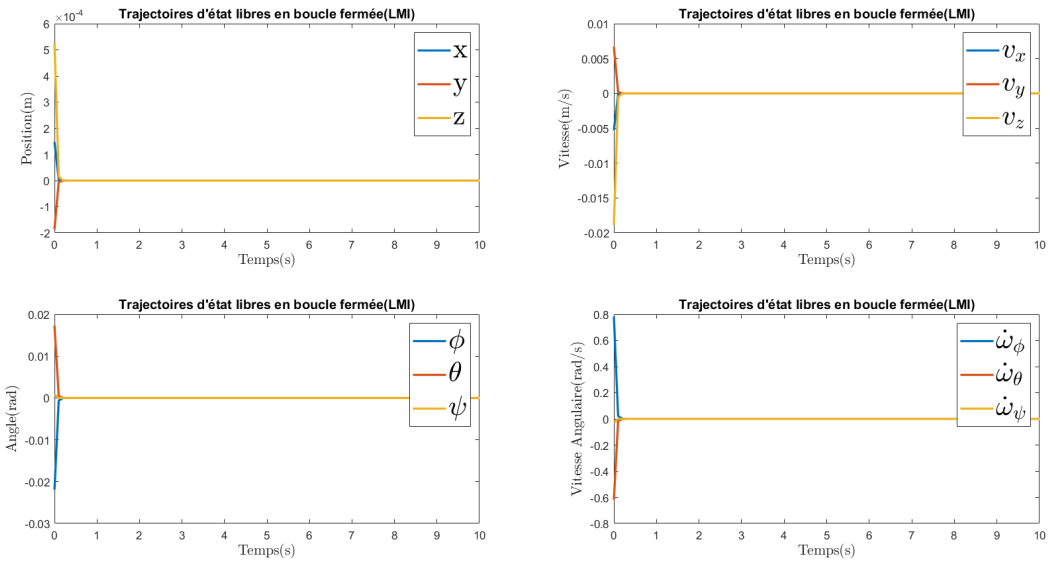


FIG. 4.3 – *Trajectoires d'états en boucle fermée avec LMI*

On remarque que les trajectoires d'état en boucle fermée avec pole placement et LMI convergent toutes vers 0 car les pôles choisis se trouvent dans le demi-plan complexe à gauche ouvert. Le système est maintenant **stable**. De plus, on constate également que les trajectoires d'état libres avec LMI convergent plus rapidement vers 0 par rapport à celles du Pôle Placement, cela est dû au fait que les LMI's nous permettent d'assigner des pôles dans une région spécifique contrairement au "pole placement" qui assigne des pôles stables mais sans aucune contrainte supplémentaire.

**Réponse indicielle** :

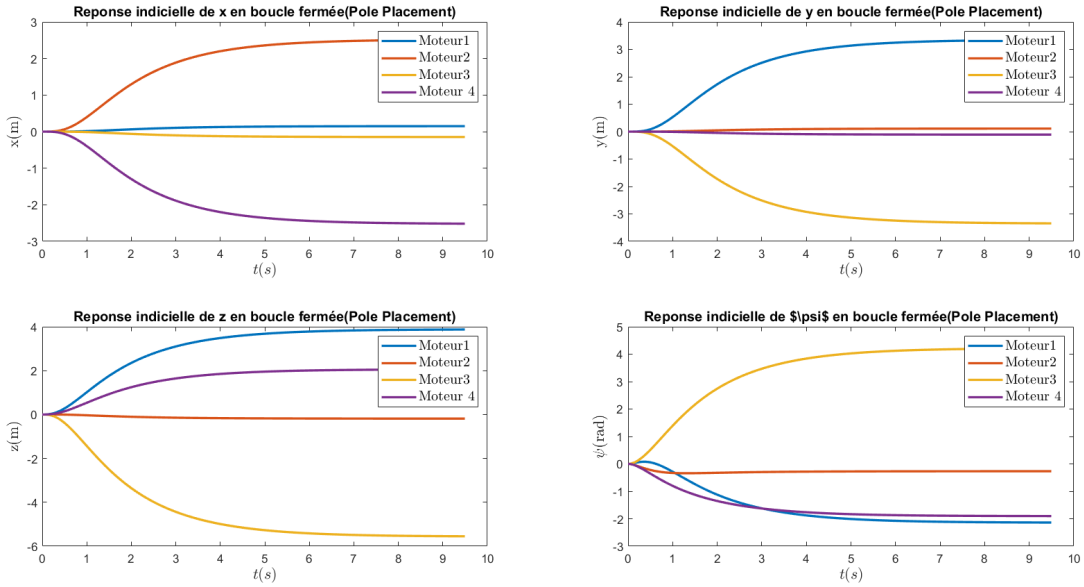


FIG. 4.4 – *Réponse indicielle en boucle fermée avec le placement de pôles*

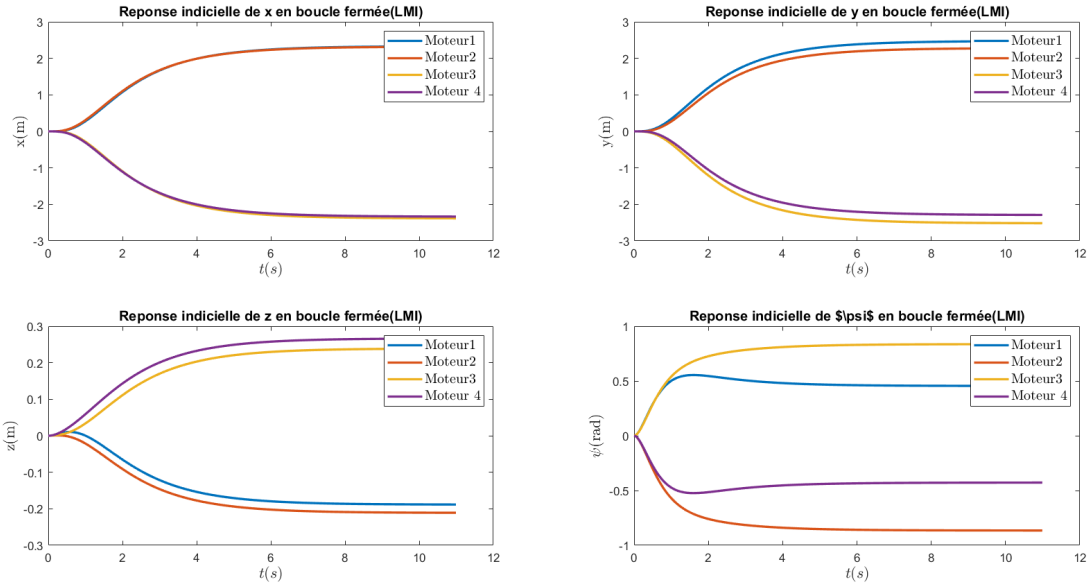


FIG. 4.5 – *Réponse indicielle en boucle fermée avec LMI*

Sur ces 2 graphes ci-dessus, on remarque que les réponses indicielles avec LMI et Pole Placement sont bornées car elles convergent toutes sans oscillations vers leurs gains statiques. Le système est maintenant **stable**. Rappelons qu'en boucle ouverte, avant l'asservissement, la réponse indicielle n'était pas bornée car elle divergeait vers l'infini (voir figure 3.3).

**Réponse impulsionnelle** :

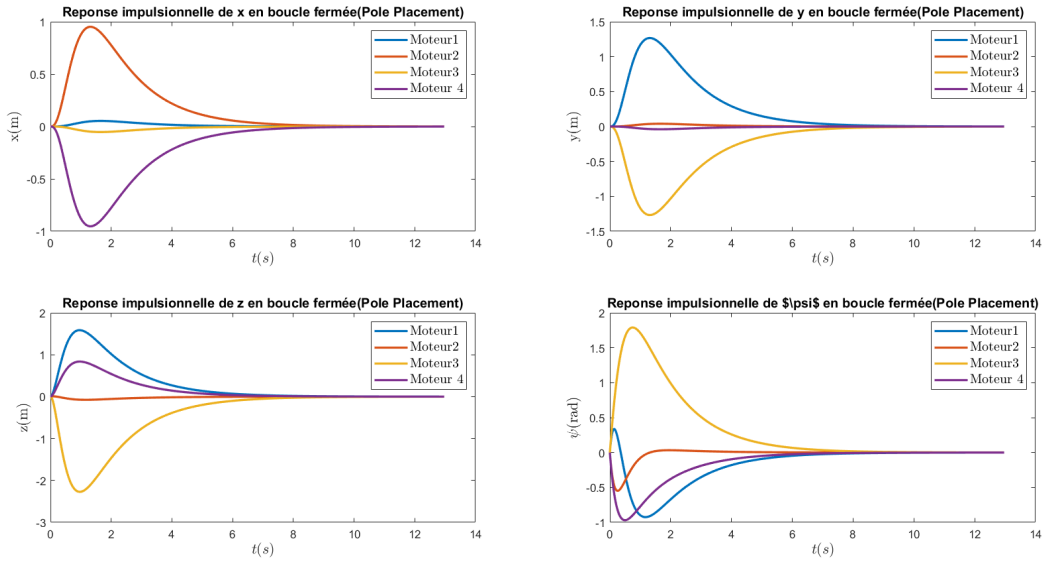


FIG. 4.6 – *Réponse impulsionnelle en boucle fermée avec le placement de pôles*

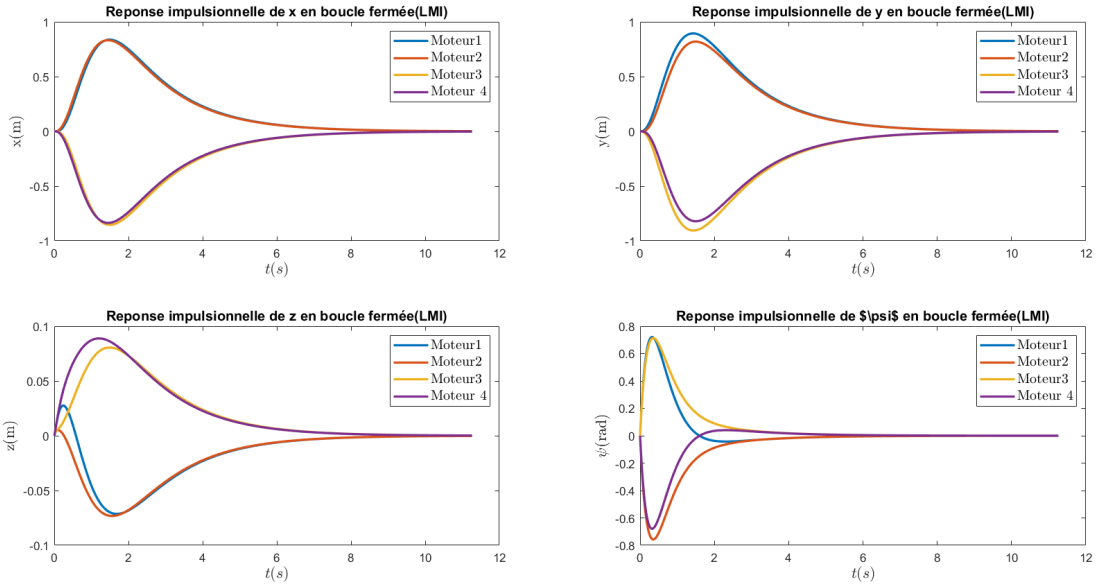


FIG. 4.7 – *Réponse impulsionnelle en boucle fermée avec les LMI's*

Nous remarquons ici qu'avec les LMI et le pole placement, les réponses impulsionnelles convergent toutes vers 0 car notre système est stable. Notons qu'avant l'asservissement, la réponse impulsionelle était instable(voir figure 3.4).

**Justification des choix de pôles et des paramètres $\alpha$, r et $\theta$ :**
On a choisi les $\alpha$ et r entre 2.5 et 3 et $\theta$ à $\frac{\pi}{6}$, ce choix semble efficace car il empêche les oscillations, diminue l'overshoot et permet d'avoir des temps de montée et de stabilisation qui sont raisonnables. Ces choix sont fait par "essais-erreurs".

**Comparaison des performances :**
En général la méthode des LMI est plus puissante que la méthode du Pole Placement, car elle permet d'assigner des pôles dans une région spécifique du plan complexe, mais elle permet aussi de créer des modèles de contrôle très robustes et performants.

# 5 Modèle non linéaire

Lors de cette section, nous allons tester une des lois de contrôle sur le modèle non linéarisé. Nous commencerons par détailler la dynamique du système en utilisant Simulink pour construire un modèle simplifié du quadrirotor, en se basant sur les équations non linéaires sous-jacentes.

## 5.1 Dynamique

Pour coder le modèle non linéaire sur Simulink, nous allons tout d'abord commencer par construire un simple modèle pour notre dynamique du quadrirotor à partir des équations non-linéaires, voir Section 2. Pour cela, nous avons créé un sous-système pour la modélisation de la dynamique avec les angles roll $\phi$, pitch $\theta$ et yaw $\psi$, ainsi que les positions $x$, $y$ et $z$ comme variables de sortie et les commandes du moteur U1, U2, U3 et U4 comme variables d'entrées(5.1). Ce sous-système va nous permettre par la suite de créer une loi de contrôle pour le modèle non linéaire.

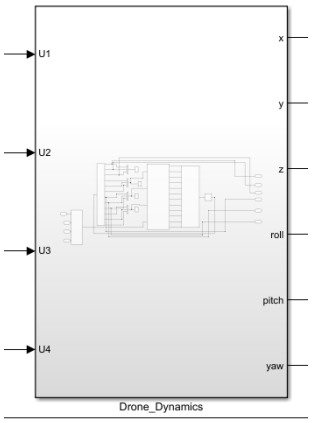


FIG. 5.1 – *Sous-système du modèle non linéaire*

Voici, ci-dessous, à quoi ressemble l'intérieur de notre sous-système qui contient aussi plusieurs sous-systèmes en lien avec les équations non linéaires.

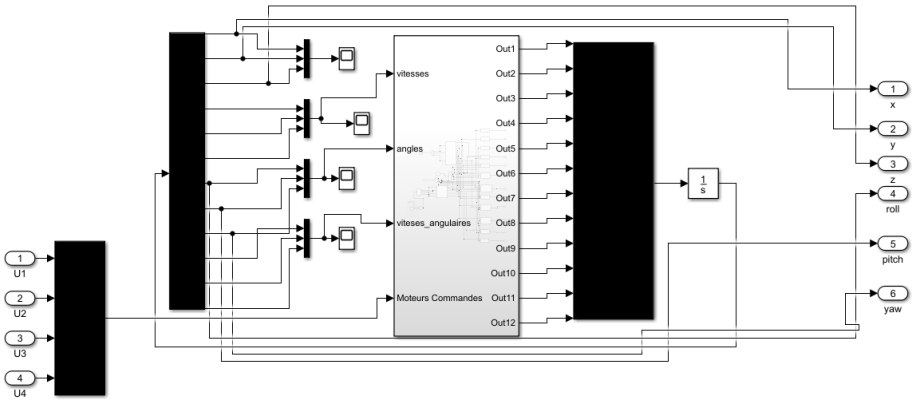


FIG. 5.2 – *L'intérieur de notre Sous-système*

### 5.1.1 Commandes des moteurs

Afin de rendre simple et lisible notre code simulink, on a crée un sous-système à l'intérieur de la dynamique qui prend en entrée les 4 commandes des moteurs U1=$V1^2$, U2=$V2^2$, U3=$V3^2$ et U4=$V4^2$, pour les transformer en la force de portance W1 et les 3 moments W2, W3 et W4 qui vont ensuite contrôler le mouvement du drone.

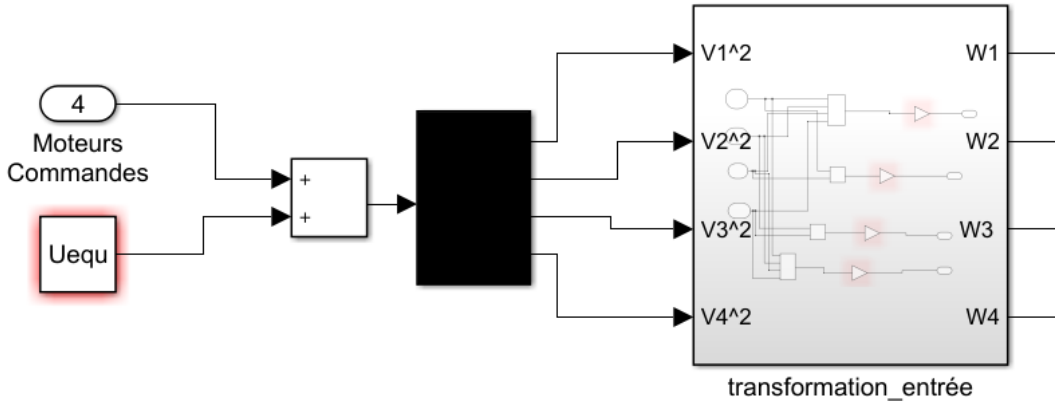


FIG. 5.3 – *Sous-système des moteurs*

Avec:

W1 = $c_t(V_1^2 + V_2^2 + V_3^2 + V_4^2)$ est la force totale de portance

W2 = $M_x = rc_p(V_4^2 - V_2^2)$ est le moment dus aux forces de traînée autour de l'axe x

W3 = $M_y = rc_p(V_3^2 - V_1^2)$ est le moment dus aux forces de traînée autour de l'axe y

W4 = $M_z = c_t(V_1^2 - V_2^2 + V_3^2 - V_4^2)$ est le moment dus aux forces de traînée autour de l'axe z.

Afin que la commande à l'entrée des moteurs soit effectivement la commande appliquée sur le drone, il ne faut pas oublier d'ajouter le $U^*$ = Uequ sur la figure 5.3 ci-dessus, qui est la valeur de l'entrée à l'équilibre obtenue dans la partie linéarisation, c'est-à-dire la valeur de la commande nécessaire pour maintenir le drone stable. Nous avons $\delta_U(t) = U(t) - U^*$ et nous voulons que l'entrée $U(t)$ soit effectivement celle qui sera appliquée sur le drône, il faut donc compenser l'entrée avec $U^*$. Ainsi on aura $\delta_U(t) = U(t) - U^* + U^* = U(t)$. Avec $U^* = \begin{bmatrix} \frac{mg}{4c_p} & \frac{mg}{4c_p} & \frac{mg}{4c_p} & \frac{mg}{4c_p} \end{bmatrix}$ indiquant que la commande d'entrée d'équilibre pour chaque moteur est la même et est proportionnelle à la masse $m$ du drone et à l'accélération gravitationnelle $g$, divisée par quatre fois la constante $c_p$. Cette uniformité suggère que, à l'état d'équilibre, tous les moteurs doivent fournir la même quantité de force de portance pour contrebalancer exactement le poids du drone.

### 5.1.2 Expressions trigonométriques

Nous avons également crée un sous-système qui prend en entrée les 3 angles $\phi$, $\theta$ et $\psi$ et donnent en sortie 9 expressions trigonométriques qui apparaissent dans les équations. Ce sous-système nous a aidé à rendre lisible notre code simulink et d'éviter des répétitions et des erreurs d'implémentations. Par exemple, l'expression trigonométrique ang1 est égale à $\sin(\psi)\sin(\phi)+\cos(\psi)\cos(\phi)\sin(\theta)$ et on va l'utiliser dans l'équation 4.

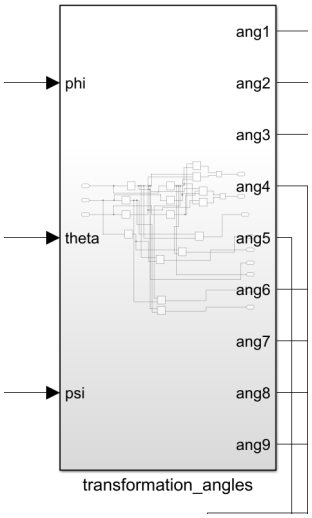


FIG. 5.4 – *Les expressions trigonométriques*

### 5.1.3 Dynamique linéaire et rotationnelle

Pour ce qui concerne la dynamique linéaire, nous avons créé 6 sous-systèmes où nous avons implémenté les six équations qui donnent les accélérations et les vitesses linéaires

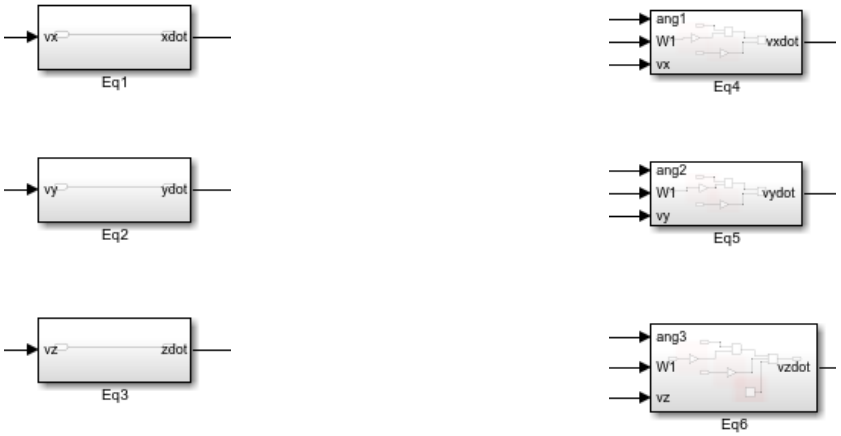


FIG. 5.5 – *Les vitesses linéaires à gauche - Les accélérations linéaires à droite*

De façon similaire, nous avons implémenté les six autres équations de la dynamique rotationnelle qui donnent les accélérations et vitesses angulaires.

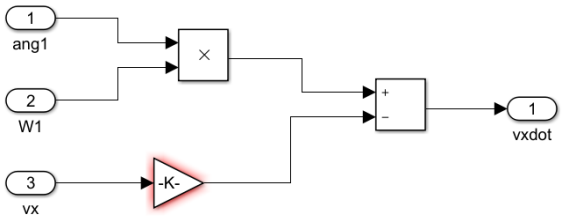


FIG. 5.6 – *Sous-système de $\dot{v_x}$*

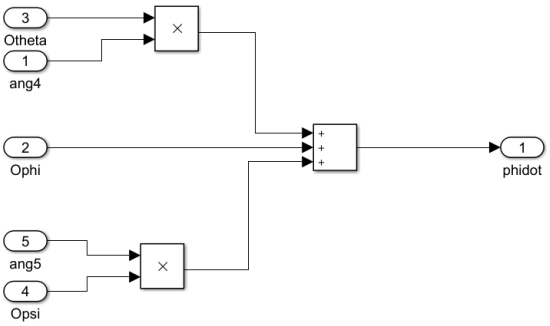


FIG. 5.7 – *Sous-système de $\dot{\phi}$*

## 5.2 Test d'une loi de contrôle sur le modèle non-linéaire

### 5.2.1 La méthode des LMIs

Nous allons maintenant tester la loi de contrôle avec la méthode des LMIs, car elle permet d'assigner les pôles dans des régions très spécifiques et prend en considération les contraintes imposées tels que la condition initiale ainsi que le voltage de la batterie, ce qui n'est pas le cas avec la méthode Pole Placement.

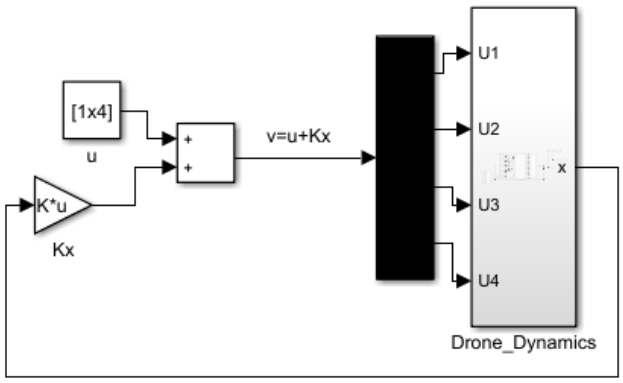


FIG. 5.8 – *Stabilisation par asservissement statique d'état*

On tient à rappeler qu'avec cette méthode on a réussi à stabiliser le drone autour de son équilibre avec la linéarisation. La question qui se pose maintenant est: Est ce que la méthode des LMIs va réussir à stabiliser le drone dans le modèle non-linéaire ?

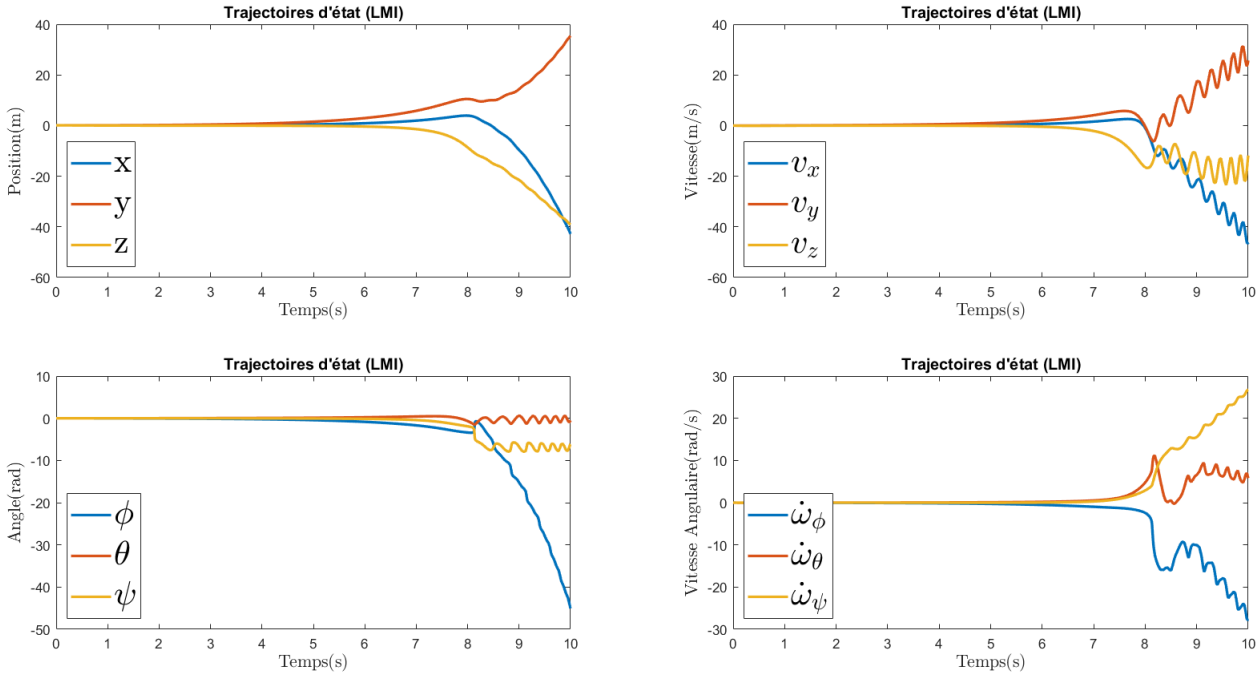


FIG. 5.9 – *Modèle non linéaire: Trajectoires d'état avec LMIs*

### 5.2.2 Résultats et interprétations

Pour répondre à la question précédente, nous allons lancer une simulation de cette loi de contrôle sur le modèle non linéaire en choisissant un contrôle $u(t) = 0$ et une condition initiale $x_0$ différente de l'état $x^*$ à l'équilibre, afin de vérifier si la trajectoire d'état libre $x(t)$ va converger à l'équilibre. La condition initiale choisie pour cette simulation est $U =$ [0, 0, 0, 0] et $x =$ [1,1,1,0,0,0,0,0,0,0,0,0] c'est à dire qu'on a éloigné le drone d'1 mètre des 3 axes x, y et z liées au référentiel terrestre qui est fixe.

D'après les résultats obtenues, nous pouvons clairement voir que la trajectoire d'état libre ne converge pas vers l'état nul. Ce qui signifie que le système n'est pas stable. Cela était bien évidemment prévu, et peut s'expliquer par le fait que notre matrice de contrôle K ne fonctionne que pour une dynamique linéarisée autour d'un équilibre, et par conséquent dès qu'on passe au modèle non linéaire la matrice d'asservissement K ne vas plus pouvoir stabiliser le système qui est non-linéaire. Nous pouvons remarquer également qu'au début de la simulation les trajectoires d'état étaient presque nulles mais à partir d'un certain instant $t$, elles commencent à diverger ou à s'éloigner du zéro. Cela vient du fait qu'aux premiers instants le drône était proche de l'équilibre mais après quelques secondes il va s'éloigner de cet équilibre et la matrice K n'a plus aucun effet.

# 6 Commande PID

Dans cette partie, notre objectif est de corriger le comportement du drône afin qu'il suit une trajectoire de référence qu'on lui donne, avec des perturbations extérieures et pour celà, on va concevoir une loi de contrôle PID (Proportionelle, Integral, Dérive) appliquée au modèle non linéaire via simulink.

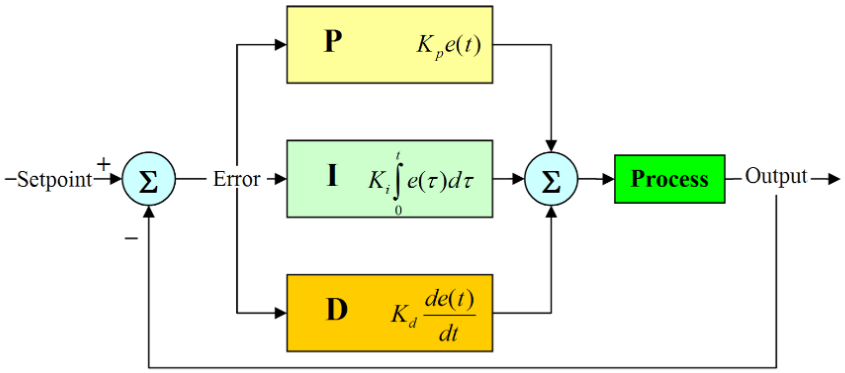


FIG. 6.1 – *Le régulateur PID*

Rappelons la définition du compensateur PID :

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau)d\tau + K_d \frac{de(t)}{dx}$$

où e(t) est la différence entre la valeur souhaitée et la mesure. On a la somme de 3 gains. Le premier est le compensateur proportionelle et celui-ci n'est pas suffisant dans notre cas. Donc on ajoute le deuxième gain intégrale qui se base sur l'erreur passée. Si l'on utilise juste un compensateur PI, on pourrait dépasser la référence donc on introduit par la suite le gain dérivé qui se base sur l'erreur future qui va être anticipée pour améliorer la situation. Ce dernier calcule le taux de variation de l'erreur.

## 6.1 Fonctionnement et modèle de la loi de contrôle

Notons que pour réaliser ce compensateur PID dans notre modèle, nous nous sommes basées sur la vidéo en [5] (voir bibliographie).

Tout d'abord, la force totale de portance joue un rôle essentiel dans le maintien de l'altitude du drone. Elle peut soit augmenter l'altitude si elle dépasse le poids du drone, soit la réduire dans le cas contraire. Cependant, lorsque le drone présente un taux élevé de tangage (Pitch) ou de roulis (Roll), la force de portance a un impact simultané sur les mouvements vertical et horizontal, ce qui nécessite une gestion spécifique. Dans le cadre de notre modèle de contrôle, nous supposons que les taux de Roll et de Pitch sont très faibles, limitant ainsi l'effet de la portance principalement à l'altitude, c'est-à-dire au mouvement vertical.

Le principe de ce contrôle consiste à mesurer en permanence l'altitude, à la comparer avec une altitude de référence, puis à transmettre cette erreur à un contrôleur qui l'utilise pour calculer la force de portance nécessaire afin de corriger cette erreur. Cependant, il existe une autre contrainte à prendre en compte : les perturbations extérieures telles que le vent, qui peuvent induire de petites déviations dans le Roll ou le Pitch au fur et à mesure du mouvement du drone. Ces déviations entraînent une influence indésirable de la portance sur les mouvements verticaux et horizontaux.

Pour résoudre ce problème, nous introduisons un contrôle qui couple les contraintes de position en $x$ et $y$ avec celles des angles de tangage (Pitch) et de roulis (Roll). Ce couplage se traduit par un contrôle qui ajuste les positions $x$ et $y$ à (0,0) (boucle externe / Outer Loop), puis maintient les angles de tangage et de roulis à (0,0) (boucle interne / Inner loop). Ce type de contrôle est appelé **contrôle en cascade**, et il permet de corriger les mouvements horizontaux dus aux perturbations extérieures tout en maintenant la portance affectant uniquement l'altitude.

Notre système comporte donc deux blocs PID qui corrigent les erreurs de mouvement horizontal en $x$ et $y$, ainsi que quatre autres blocs PID qui jouent un rôle central dans le modèle en calculant les commandes de contrôle en fonction de la différence entre la sortie actuelle du système et la valeur de référence souhaitée. Ensuite, le bloc MMA (mixeur des moteurs) reçoit les commandes (thrust-roll-pitch-yaw) générées par les différents régulateurs PID et les combine de manière spécifique pour contrôler les vitesses des quatre rotors.

Enfin, les sorties du système sont les réponses en termes de position ($x$, $y$, $z$) et d'angles ($\phi$, $\theta$, $\psi$) du modèle par rapport à la commande de contrôle (U1, U2, U3, U4). Ces sorties sont comparées à la référence pour déterminer l'erreur, qui est ensuite renvoyée aux blocs PID pour un ajustement. Cette boucle de rétroaction permet au système de s'ajuster en fonction de l'erreur entre la sortie et la référence. Voici ci-dessous notre modèle.

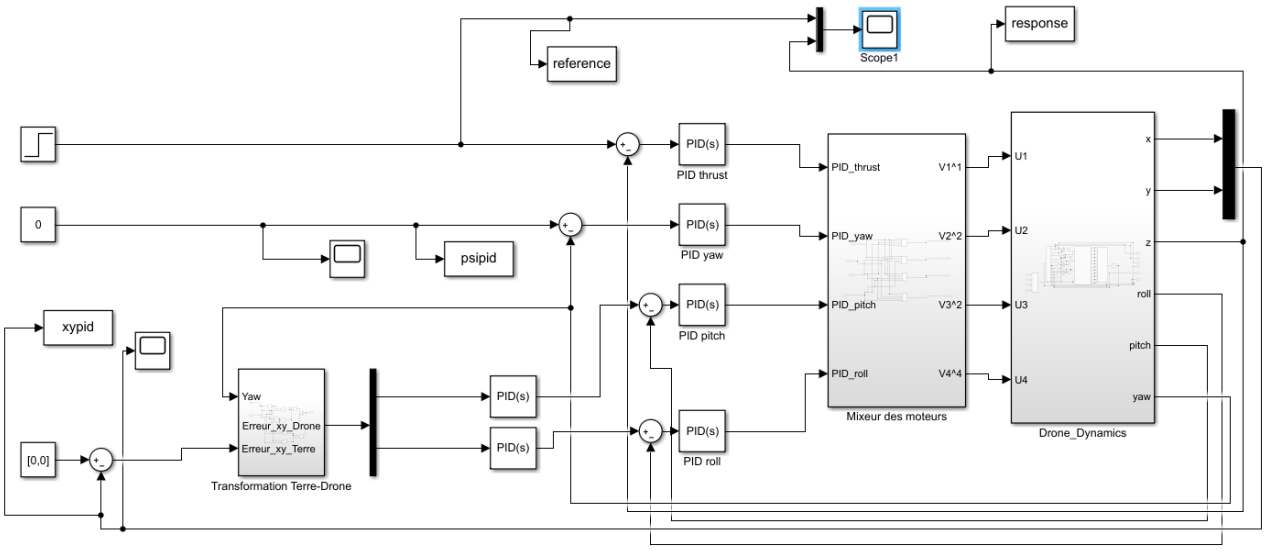


FIG. 6.2 – *Commandes d'entrée et loi de contrôle PID sur simulink*

## 6.2 Choix des valeurs des gains

Pour choisir les valeurs des gains on a utilisé **le PID Tuner** sur simulink avec l'option **reference tracking** pour suivre la valeur de référence de l'altitude $z$. Le pid tuner fait une linéarisation de la dynamique du modèle non-linéaire puis calcule les meilleurs gains. Nous avons récupéré les gains et puis nous les avons testés sur notre modèle simulink, tout en vérifiant bien les performances et la robustesse de notre modèle.

## 6.3 Résultats et interprétations

Pour tester notre modèle de controle et nos gains PID, nous avons choisi comme condition initiale l'état nul $x = [0,0,0,0,0,0,0,0,0,0,0,0]$ et nous souhaitons faire monter le drone jusqu'à 1 mètre d'altitude. Voici ci-dessous le graphe qu'on obtient ainsi qu'un tableau résumant les performances:

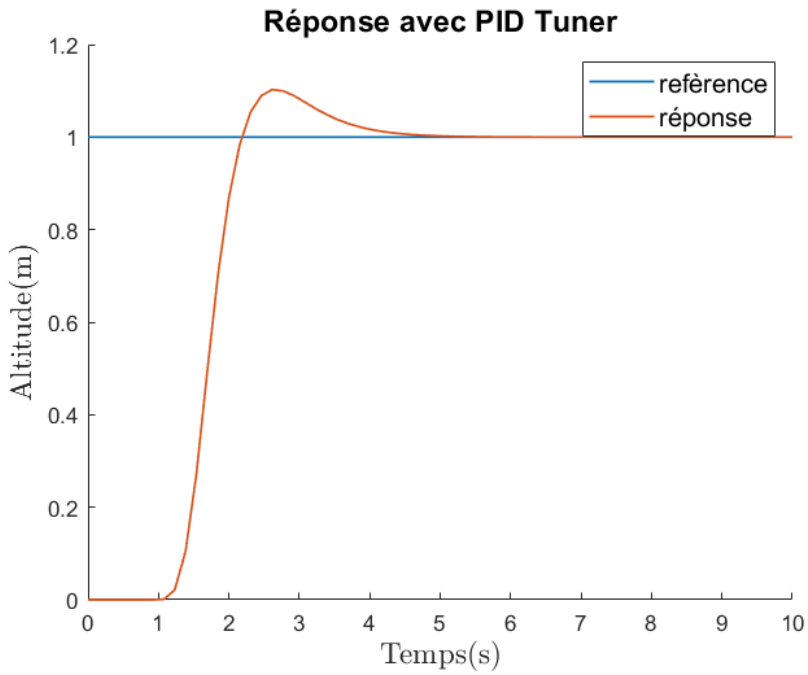


FIG. 6.3 – *Réponse à une référence échelon*

| Paramètre | Valeur |
|---|---|
| Temps de montée | 0.201 sec |
| Temps de stabilisation | 2.21 sec |
| Dépassement | 10.3 % |
| Valeur Finale | 1 m |

TAB. 1 – *Résumé des performances de la réponse à une référence(fonction échelon)*

Le graphique ci-dessus représente la réponse d'un système à une référence échelon en utilisant un régulateur PID. Le graphique montre deux courbes : une pour la valeur de référence et une pour la réponse du système. La courbe bleue est donc la référence que le système est censé suivre. Ici, on voit que la référence est une fonction échelon car

elle est bien égale à 1 sur les positifs. La courbe rouge est la réaction réelle du système face à la référence. On observe une montée rapide, le système réagit rapidement à la référence. Après ceci, on peut observer un dépassement, ie "overshoot" de 10.3 %, le système dépasse la référence, ce qui est typique dans les réponses aux régulateurs PID sans un réglage parfait. Suite à cet overshoot, le système se stabilise à partir d'environ 2.21 secondes. L'objectif du réglage PID est de minimiser l'overshoot pour obtenir une réponse qui atteint rapidement la consigne et y reste stable sans oscillation excessive. Dans notre cas, même si la réponse montre un overshoot, elle finit par se stabiliser sur la valeur de consigne, ce qui indique que le régulateur PID est raisonnablement bien réglé.

# 7 Simulation et expérimentation

Dans cette dernière partie du projet, nous avons, à l'aide d'un code fourni tester, en simulation et expérimentalement, notre loi de contrôle PID sur un drone.

## 7.1 Explication brève du code

Après avoir conçu un modèle de contrôle pour notre drone, il nous reste encore une étape très importante qui va permettre par la suite au drone d'effectuer le vol en toute sécurité.
Ainsi, le code qui a été fourni à savoir le **Flight Control System** contient la logique de contrôle qu'on va intégrer dans le drone.
Cette logique de contrôle reprend à l'instar du contrôleur 3 autres parties supplémentaires indispensables:

- **Un estimateur d'état/State Estimator**: Ce bloc récupère les données des capteurs et effectue les calculs et les conversions nécessaires pour estimer les variables états dont le contrôleur a besoin.
- **Enregistreur des données/Data Logging**: Ce bloc permet d'enregistrer toutes les informations du vol du drone et de les retourner dans un fichier MAT pour les utilisateurs.
- **Protège Pannes/Fault Protection**: Ce bloc permet à l'aide des shutdown flags/drapeaux d'arrêt d'arrêter l'exécution du drone en cas de risques pour assurer la sécurité de l'environnement du drone(les observateurs par exemple).

## 7.2 Réglages des paramètres PID

Il s'agit de la dernière étape de notre travail, qui consiste à régler les paramètres PID du contrôleur.
Pour ajuster les valeurs des gains des différentes PID, nous avons commencé par choisir des valeurs de manière aléatoire, sans grand succès. Finalement, nous nous sommes basés sur un modèle simulink utilisée dans une compétition **Parrot Minidrone Competition, 2019** [6].
La logique derrière ce choix de ces paramètres consiste à ajuster la valeur de chaque paramètre en fixant les autres à 0(par exemple on fixe $K_d$ et $K_i$ à 0 puis on ajuste le $K_p$ et ainsi de suite) et continue de faire la même opération pour tous les blocs pid jusqu'à obtenir la réponse désirée et les performances dynamiques souhaitées..

## 7.3 Résultats

Voici ci-dessous nos graphes de simulation.Ils présentent les résultats générés à partir de simulations informatiques:

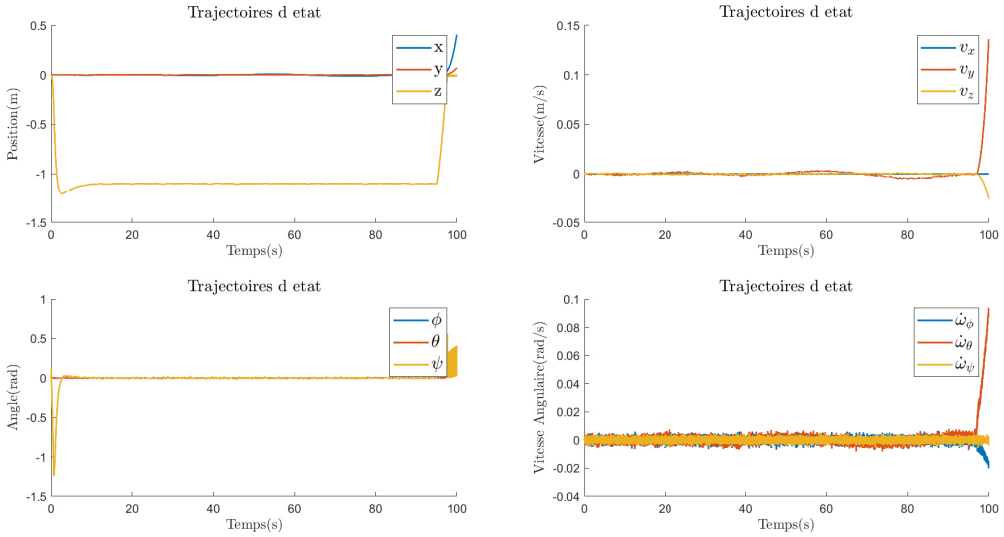


FIG. 7.1 – *Graphes en simulation*

Ensuite, nous avons tester expérimentalement le code fourni sur le drone en conditions réelles, et à l'aide du **Data Logging** on récupéré le **flight log.MAT** qui est le fichier matlab contenant toutes les données du vol. Les graphes suivants montrent les trajectoires d'état recueillies à partir d'expériences réelles:

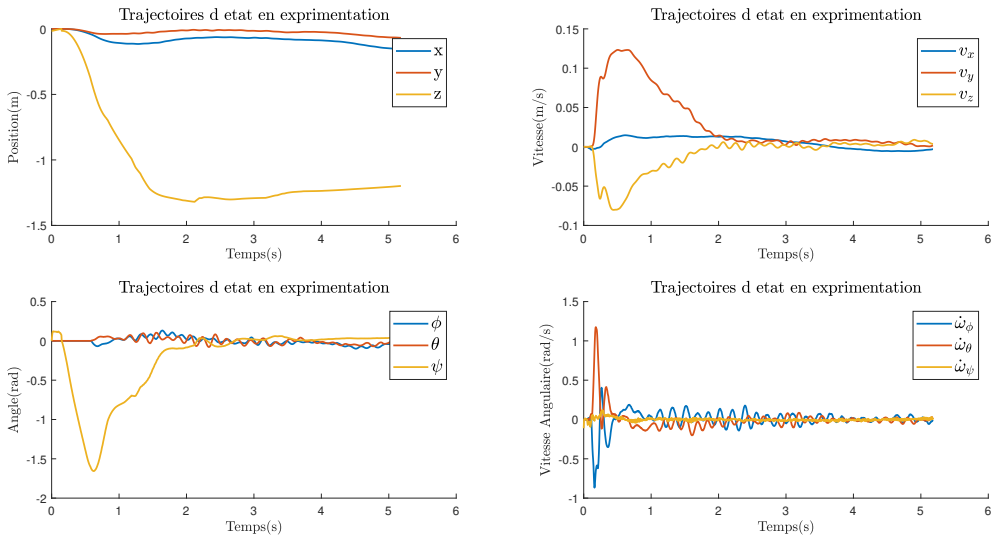


FIG. 7.2 – *Graphe en expérimentation*

## 7.4 Interprétations

### 7.4.1 Interprétation: Graphes de simulation

Tout d'abord, il est important de noter que sur les graphiques l'altitude est représentée de manière inversée sur l'axe z, c'est-à-dire qu'une altitude plus élevée correspond à une valeur plus basse sur l'axe. On constate que la trajectoire d'état de z suit bien la valeur de référence souhaitée qui est de 1 mètre.
La valeurs de x et y sont approximativement égales à 0 ce qui correspond bien à nos attentes.
Ces graphes de simulation montre des tendances idéalisées car ils ne sont pas affectés par les perturbations extérieures comme le vent, les erreurs de mesure des capteurs, ect.
En examinant les vitesses affichées dans le coin supérieur droit, nous remarquons que toutes les valeurs sont assez constantes, ce qui est rassurant. Dans le coin inférieur gauche, Les angles roulis $\phi$ et le tangage $\theta$ sont maintenues très proches de 0 tandis que le lacet $\psi$ montre une brusque montée vers -1.25 (rad) due à la différence de vitesses entre les rotors (1,3) et (2,4) du drone et puis qui se stabilise rapidement en 0.

### 7.4.2 Interprétation: Graphes d'expérimentation

Les graphes d'experimentation montrent les données recueillies à partir d'expériences réelles. Ces graphiques montrent plus de variabilité en raison de facteurs de **perturbations extérieures**, les **imperfections des capteurs**, et la **condition initiale**.

# 8 Conclusion

En conclusion, ce projet a démontré une approche rigoureuse dans l'étude et la conception d'un système de contrôle pour un quadrirotor. Grâce à une modélisation, à l'analyse de la dynamique du système et à la mise en place de commandes PID, nous avons réussi à stabiliser le quadrirotor et à le faire répondre précisément à des commandes de vol spécifiques. La combinaison des techniques d'assignabilité spectrale et des LMI a démontré son efficacité pour la stabilisation en boucle fermée du système, même en présence de perturbations liées au comportement non linéaire du drone. Les simulations réalisées avec Matlab et Simulink ont validé nos modèles et notre approche de contrôle, en révélant des performances satisfaisantes en termes de réponse indicielles et impulsionnelle, ainsi que des trajectoires d'état en adéquation avec les références désirées. Les tests expérimentaux ont complété la validation en fournissant des données réelles qui confirment le comportement attendu du système. Ces essais ont également souligné l'importance de la commande PID pour ajuster dynamiquement le comportement du quadrirotor, garantissant ainsi son adaptabilité dans un environnement variable. Finalement, les résultats obtenus démontrent la puissance de l'ingénierie de contrôle et ouvrent de grande perspectives dans le domaine de l'aéronautique. Les applications futures de ce travail sont illimitées.

# 9 Bibliograhie

[1] Azouz Mustapha, *Modélisation et commande d'un quadrirotor : Etude comparative de la commande floue et PID*, Université Mouloud Mammeri Detizi-Ouzou, 2016

[2] Oualid Araar, *Full linear control of a quadrotor UAV, LQ vs H </inf*, 2014

[3] Zaid Tahir et Waleed Tahir, *State Space System Modelling of a Quad Copter UAV*

[4] KHEBBACHE Hicham, *Tolérance aux défauts via la méthode backstepping des systèmes non linéaires* P25

[5] Playlist Youtube: MATLAB, *Drone Simulation and Control* , 2019

[6] Parrot Minidrone Competition, *https://github.com/abdelhakim96/MathWorks-Mini-Drone-Competition-2020/tree/main*, 2019