

# Modélisation et Simulation de la Dynamique d'un Hexacoptère : Formulation par Angles d'Euler et Quaternions

**TAF: Robotique et Interactions**

**Cours: Dynamique des robots**

*Élèves :*

Marco LLONTOP HERRERA  
Jordan DELGADO GUTIERREZ

*Enseignants :*

Frédéric BOYER  
Vincent LEASTARD

10 décembre 2025

## Table des matières

<b>1 Introduction</b>	<b>3</b>
1.1 Contexte et Problématique . . . . .	3
1.2 Objectif du Projet et Méthodologie . . . . .	4
<b>2 Modélisation Mathématique</b>	<b>6</b>
2.1 Systèmes de Référence . . . . .	6
2.1.1 Système de Référence Terrestre (Repère 0) . . . . .	7
2.1.2 Système de Référence du Corps (Repère 1) . . . . .	7
2.2 Matrice de Transformation . . . . .	7
2.3 Transformation des Vitesses Angulaires . . . . .	7
<b>3 Dynamique (Équations de Mouvement)</b>	<b>10</b>
3.1 Dynamique Translationnelle . . . . .	10
3.2 Dynamique Rotationnelle . . . . .	11
3.2.1 Matrice d'Inertie du Corps Rigide . . . . .	12
3.2.2 Mouvement de Lacet (Yaw - $\tau_\psi$ ) : . . . . .	12
3.2.3 Mouvement de Roulis (Roll - $\tau_\phi$ ) . . . . .	14
3.2.4 Mouvement de Tangage (Pitch - $\tau_\theta$ ) . . . . .	15
3.2.5 Le Vecteur des Couples Extérieurs . . . . .	16
3.2.6 L'Effet Gyroscopique ( $\Gamma$ ) . . . . .	16
3.2.7 Équation Finale du Mouvement Angulaire . . . . .	16
<b>4 Vecteur d'État</b>	<b>17</b>
4.1 Modélisation avec des angles d'Euler . . . . .	18
4.2 Modélisation avec des quaternions . . . . .	19
<b>5 Implémentation et Structure du Code</b>	<b>21</b>
5.1 Organisation Générale des Fichiers . . . . .	21
5.2 Le Lanceur de Simulation : <code>test_drone.m</code> . . . . .	22
5.3 Le Générateur de Scénarios : <code>generate_test_case.m</code> . . . . .	23
5.4 Les Modèles Dynamiques : <code>dynamics_model_*.m</code> . . . . .	24
5.5 Visualisation Unifiée : <code>animations.m</code> . . . . .	25
<b>6 Résultats et Analyse des Scénarios</b>	<b>27</b>
6.1 Cas 0 : Vol Stationnaire (Hover) . . . . .	27
6.2 Cas 1 : Rotation de Lacet (Yaw) . . . . .	28
6.3 Cas 2 : Décollage Vertical . . . . .	29
6.4 Cas 3 : Mouvement de Tangage . . . . .	30
6.5 Cas 4 : Mouvement de Roulis . . . . .	31
6.6 Cas 5 : Étude de la Singularité (Gimbal Lock) . . . . .	33
6.7 Cas 6 : Manœuvre Combinée (Décollage + Lacet) . . . . .	34
<b>7 Conclusion</b>	<b>36</b>

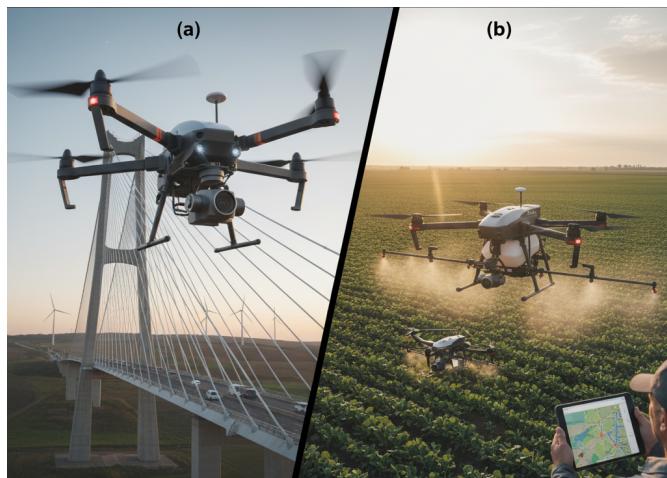
## 8 Bibliographie

37

# 1 Introduction

## 1.1 Contexte et Problématique

Au cours de la dernière décennie, les véhicules aériens sans pilote (UAV), communément appelés drones, ont transcendé le domaine de la recherche militaire pour devenir des outils omniprésents et transformationnels dans une multitude de secteurs civils. De la logistique (livraison de colis) à la cinématographie (prises de vue aériennes), en passant par l'inspection d'infrastructures et l'agriculture de précision, leur polyvalence n'est plus à démontrer. Parmi la grande famille des UAV, les **multirotors** se distinguent par leur capacité de décollage et d'atterrissement vertical (VTOL) ainsi que leur aptitude au vol stationnaire. Notre étude se concentre spécifiquement sur l'**hexacoptère**, un multirotor à six hélices. Cette configuration offre un compromis remarquable entre agilité, capacité d'emport et redondance (un atout de sécurité majeur, l'appareil pouvant potentiellement rester stable même après la défaillance d'un moteur).



**FIGURE 1** – Applications des drones : (a) Inspection d'infrastructures à grande échelle ; (b) Utilisation en agriculture de précision pour la cartographie et l'épandage.

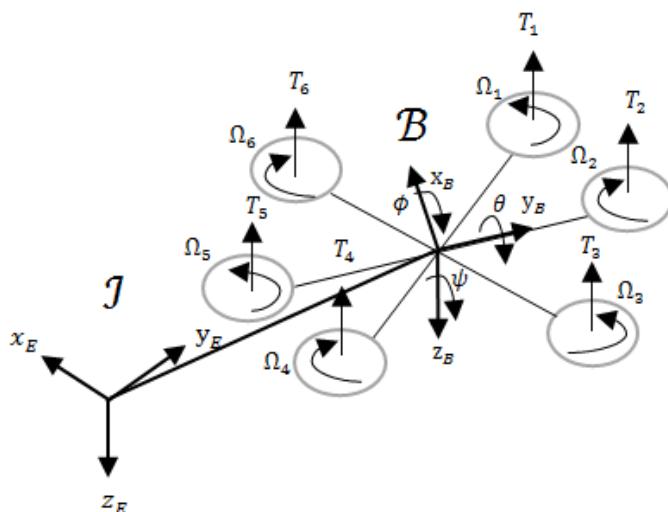
Cependant, cette manœuvrabilité exceptionnelle a pour contrepartie une complexité dynamique redoutable. Un hexacoptère est, par nature, un système **intrinsèquement instable** ; sans une boucle de contrôle active permanente (des milliers de corrections par seconde), il ne peut maintenir son équilibre. De plus, sa dynamique est :

- **Fortement non linéaire** : Par exemple, la force de portance (poussée) générée par une hélice n'est pas linéaire par rapport à la vitesse de commande du moteur, mais est proportionnelle au carré de sa vitesse de rotation ( $T \propto \omega^2$ ).
- **Fortement couplée** : C'est un système à entrées et sorties multiples (MIMO). Le simple fait de vouloir "avancer" (translation) implique une manœuvre de "piquer du nez" (rotation), modifiant ainsi simultanément l'altitude et l'orientation. Chaque action affecte l'ensemble de l'état du véhicule.

## 1.2 Objectif du Projet et Méthodologie

Face à cette complexité, concevoir un contrôleur de vol robuste (le "cerveau" du drone) qui soit à la fois performant et sûr, ne peut se faire par simple intuition ou essais-erreurs sur le terrain. Il est indispensable de disposer d'un environnement de test fiable, reproductible et sans risque. L'objectif principal de ce projet est donc de **développer un simulateur dynamique de haute fidélité pour un hexacoptère**, en utilisant l'environnement MATLAB. Ce simulateur, véritable jumeau numérique du drone, remplit plusieurs fonctions critiques :

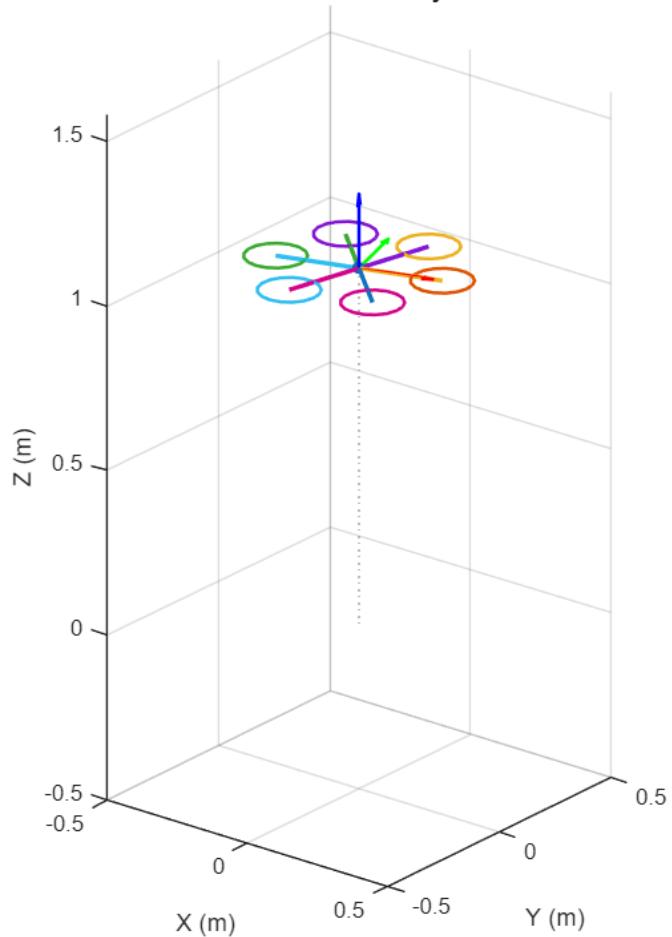
1. Valider le modèle mathématique : S'assurer que les équations de la physique (découlant des lois de Newton-Euler) décrivent avec précision le comportement réel du drone.
2. Concevoir et tester des contrôleurs : Permettre l'implémentation et le "tuning" d'algorithmes de contrôle (PID, LQR, etc.) dans un environnement sûr avant de les déployer sur un appareil physique.
3. Analyser la performance : Simuler des scénarios complexes (pannes moteur, rafales de vent) pour comprendre les limites du système.



**FIGURE 2** – Représentation schématique de la configuration de l'hexacoptère. Les moteurs impairs (1, 3, 5) tournent dans un sens anti-horaire, tandis que les moteurs pairs (2, 4, 6) tournent en sens horaire pour compenser le couple de lacet (Yaw).

Pour atteindre cet objectif, ce rapport détaille le processus complet de modélisation et de simulation. On commence par dériver les équations fondamentales du mouvement, en séparant la translation (mouvement dans l'espace) de la rotation (changement d'orientation). On se concentre d'abord sur la méthode de représentation d'orientation la plus intuitive : les **Angles d'Euler (Roll, Pitch, Yaw)**. On implémentera cette physique dans la fonction `dynamics_model_euler.m` et la validera à l'aide d'un banc de test `test_drone.m`. L'analyse de ces simulations, visualisées par notre script `animations.m`, mettra en lumière les comportements attendus du drone (décollage, rotation, translation) ainsi que les limites de cette approche.

Frame 201 / 201 — x=0.00 y=0.00 z=1.08



**FIGURE 3** – Aperçu de l'environnement de simulation 3D développé sous MATLAB. L'animation affiche la trajectoire calculée (ligne pointillée), la position du centre de masse, et les axes du corps (RGB) en temps réel, permettant une validation visuelle immédiate du comportement du modèle.

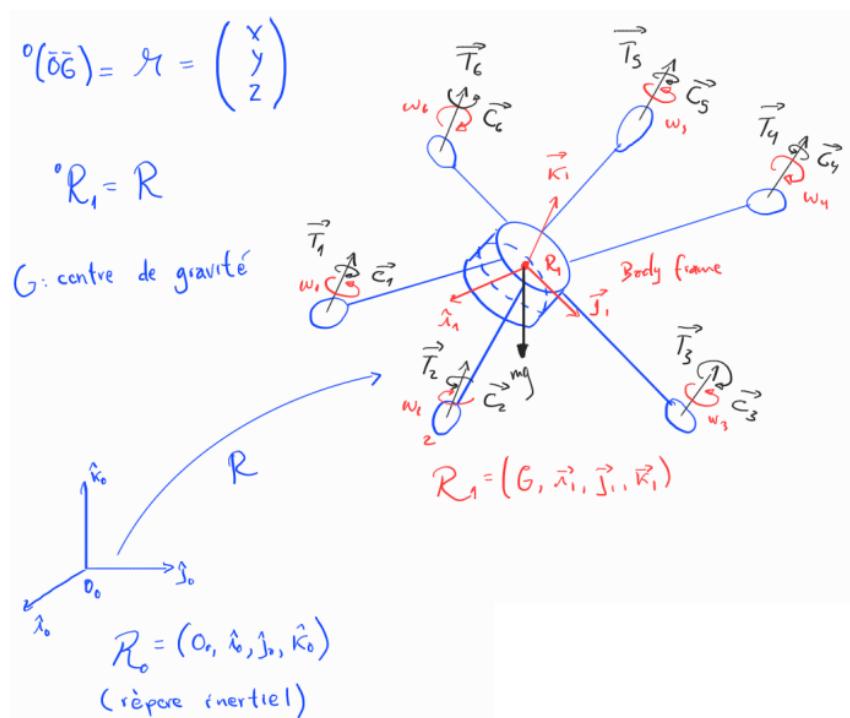
Cette analyse nous conduira à exposer une faiblesse fondamentale de la représentation par Angles d'Euler : le verrouillage de cardan ([Gimbal Lock](#)). Ce problème de singularité mathématique, qui survient lors de manœuvres à forte inclinaison, rend le modèle instable et inutilisable pour des simulations acrobatiques. Cette limitation ouvrira la voie à la seconde partie du projet, qui explorera une méthode de représentation mathématique plus robuste et plus complexe pour surmonter ce défi : un modèle utilisant des [Quaternions](#).

## 2 Modélisation Mathématique

La précision de notre simulateur repose entièrement sur la fidélité de sa représentation mathématique des phénomènes physiques. Cette section détaille les équations de mouvement (ÉDM) qui régissent le comportement d'un hexacoptère, considéré comme un corps rigide, en utilisant la formulation des angles d'Euler pour décrire son orientation.

### 2.1 Systèmes de Référence

Afin de décrire le mouvement complet du drone dans l'espace, il est indispensable de définir deux systèmes de coordonnées distincts : un système de référence fixe par rapport à la Terre (que nous appellerons le système 0) et un système attaché au drone lui-même (que nous appellerons le système 1).



**FIGURE 4** – Représentation des deux systèmes de référence utilisés pour la modélisation : le repère Terrestre  $\mathcal{R}_0$  (fixe), et le repère du Corps  $\mathcal{R}_1$  (mobile) attaché au centre de masse du drone. La transformation entre ces deux repères est cruciale pour l'analyse des forces et des mouvements.

### 2.1.1 Système de Référence Terrestre (Repère 0)

Ce repère, noté  $\mathcal{R}_0 = (O_0, \mathbf{x}_0, \mathbf{y}_0, \mathbf{z}_0)$ , est considéré comme fixe et non accéléré par rapport à la surface terrestre (repère inertiel). L'origine  $O_0$  est fixée arbitrairement à un point du sol (point de décollage). L'axe  $\mathbf{z}_0$  pointe vers le haut (opposé à la gravité). Les axes  $\mathbf{x}_0$  et  $\mathbf{y}_0$  définissent le plan horizontal. C'est dans ce repère qu'on mesure la position absolue du drone  $\boldsymbol{\xi} = [x \ y \ z]^T$  et son orientation  $\boldsymbol{\eta} = [\varphi \ \theta \ \psi]^T$ .

### 2.1.2 Système de Référence du Corps (Repère 1)

Ce repère, noté  $\mathcal{R}_1 = (O_1, \mathbf{x}_1, \mathbf{y}_1, \mathbf{z}_1)$ , est attaché au centre de masse (ou centre de gravité, C.G.) du drone et se déplace et pivote avec lui. L'origine  $O_1$  coïncide avec le centre de masse du drone (G). L'axe  $\mathbf{x}_1$  pointe vers l'avant (nez) du drone. L'axe  $\mathbf{y}_1$  pointe vers l'aile gauche du drone. L'axe  $\mathbf{z}_1$  pointe vers le haut du drone (perpendiculaire aux bras). Dans ce repère, les forces de poussée des moteurs et les couples qu'ils génèrent sont plus facilement définis et les vitesses angulaires  $\boldsymbol{\nu} = [p \ q \ r]^T$  sont également exprimées ici.

## 2.2 Matrice de Transformation

Pour exprimer un vecteur du repère inertiel vers le repère drone, on utilise une matrice de rotation définie par les angles d'Euler :  $\psi$  (yaw),  $\theta$  (pitch) et  $\phi$  (roll). Cette matrice correspond à une composition de rotations dans l'ordre Z–Y–X (yaw → pitch → roll).

$$R = R_z(\psi)R_y(\theta)R_x(\phi)$$

$$\mathbf{R}_{0 \leftarrow 1} = \begin{pmatrix} \cos \theta \cos \psi & \sin \phi \sin \theta \cos \psi - \cos \phi \sin \psi & \cos \phi \sin \theta \cos \psi + \sin \phi \sin \psi \\ \cos \theta \sin \psi & \sin \phi \sin \theta \sin \psi + \cos \phi \cos \psi & \cos \phi \sin \theta \sin \psi - \sin \phi \cos \psi \\ -\sin \theta & \sin \phi \cos \theta & \cos \phi \cos \theta \end{pmatrix}$$

Cette matrice permet de projeter la vitesse linéaire, forces et orientations définies dans le repère Terre vers le repère du drone, indispensable pour le contrôle d'attitude et la dynamique du vol. Il convient de mentionner que pour passer du repère 0 au repère 1, on peut utiliser l'inverse de cette matrice qui est orthogonale  $R^{-1} = R^T$ .

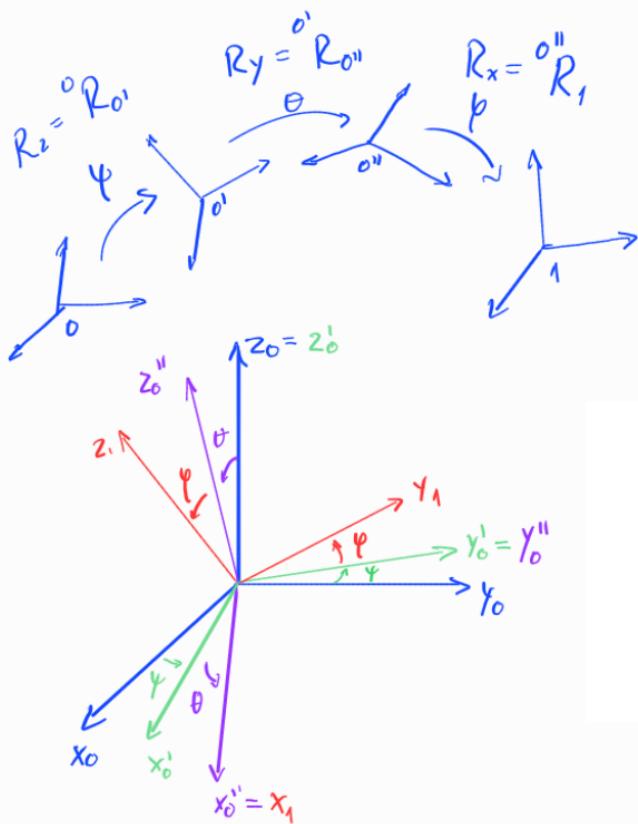
## 2.3 Transformation des Vitesses Angulaires

Les vitesses angulaires  $p, q, r$  sont les vitesses de rotation instantanées autour des axes du corps  $\mathbf{x}_1, \mathbf{y}_1, \mathbf{z}_1$ . Elles sont exprimées dans le repère du corps  $\mathcal{R}_1$ . Cependant, la dérivée des angles d'Euler  $(\dot{\phi}, \dot{\theta}, \dot{\psi})$  représente la vitesse de changement de l'orientation du corps, exprimée par rapport au repère terrestre  $\mathcal{R}_0$ . Il existe une relation non triviale entre ces deux ensembles de vitesses. La relation est donnée par la matrice de transformation cinématique  $\mathbf{T}_e$  :



$$\begin{pmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{pmatrix} = \mathbf{T}_e(\phi, \theta) \begin{pmatrix} p \\ q \\ r \end{pmatrix}$$

Pour dériver  $\mathbf{T}_e$ , on considère que la vitesse angulaire totale dans le repère 0 est la somme des vitesses angulaires de chaque rotation élémentaire, projetées dans le repère du corps 1. La démonstration détaillée commence par la visualisation des rotations à l'aide de la figure suivante :



**FIGURE 5** – Illustration de la séquence de rotations pour définir l'orientation d'un corps rigide par les angles d'Euler (Z-Y-X). (a) Lacet ( $\psi$ ) autour de l'axe Z. (b) Tangage ( $\theta$ ) autour du nouvel axe Y. (c) Roulis ( $\varphi$ ) autour du nouvel axe X. Cette méthode offre une interprétation intuitive de l'orientation du drone.

Les vitesses angulaires  $p, q, r$  sont les vitesses de rotation instantanées autour des axes du corps  $\mathbf{x}_1, \mathbf{y}_1, \mathbf{z}_1$ . Elles sont exprimées dans le repère du corps  $\mathcal{R}_1$ . Cependant, la dérivée des angles d'Euler  $(\dot{\phi}, \dot{\theta}, \dot{\psi})$  représente la vitesse de changement de l'orientation du corps, exprimée par rapport au repère terrestre  $\mathcal{R}_0$ . Pour relier ces deux concepts, on applique la **loi de composition des vitesses angulaires**. La vitesse angulaire totale  $\boldsymbol{\omega}$  (notée  $\nu$ ) est la somme vectorielle des vitesses angulaires relatives entre chaque repère intermédiaire créé par les rotations successives :

$${}^1(\boldsymbol{\omega}_{1/0}) = {}^1(\boldsymbol{\omega}_{1/0''} + \boldsymbol{\omega}_{0''/0'} + \boldsymbol{\omega}_{0'/0})$$

Où les indices  $0'$  et  $0''$  représentent les repères intermédiaires après la rotation de lacet ( $\psi$ ) et de tangage ( $\theta$ ) respectivement. En développant cette somme, la vitesse angulaire s'exprime comme la somme des taux de changement des angles d'Euler, portés par leurs axes de rotation respectifs :

$$\boldsymbol{\omega} = \dot{\phi}\boldsymbol{x}_1 + \dot{\theta}\boldsymbol{y}' + \dot{\psi}\boldsymbol{z}_0$$

Ici :

- $\dot{\phi}\boldsymbol{x}_1$  est la vitesse de roulis autour de l'axe final du corps (noté  $\boldsymbol{i}_1$  ou  $\boldsymbol{x}_1$ ).
- $\dot{\theta}\boldsymbol{y}'$  est la vitesse de tangage autour de l'axe intermédiaire (noté  $\boldsymbol{j}_0''$  ou  $\boldsymbol{y}'$ ).
- $\dot{\psi}\boldsymbol{z}_0$  est la vitesse de lacet autour de l'axe terrestre initial (noté  $\boldsymbol{k}_0$  ou  $\boldsymbol{z}_0$ ).

Le vecteur final  $\boldsymbol{\nu} = [p \ q \ r]^T$  correspond à la projection de cette somme vectorielle dans le repère final du corps  $\mathcal{R}_1$ . Cela peut s'écrire sous forme matricielle :

$$\boldsymbol{\nu} = \begin{pmatrix} p \\ q \\ r \end{pmatrix} = \dot{\phi} \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} + \dot{\theta} {}^1(\boldsymbol{y}') + \dot{\psi} {}^1(\boldsymbol{z}_0)$$

Pour obtenir la matrice de transformation finale  $\boldsymbol{T}_e$ , il faut exprimer les vecteurs  $\boldsymbol{y}'$  et  $\boldsymbol{z}_0$  dans la base du corps  $\mathcal{R}_1$  en utilisant les matrices de rotation inverses. En projetant ces vecteurs unitaires dans le repère 1 (corps) après les rotations ZYX, on obtient :

$$\begin{aligned} \boldsymbol{x}_1 &= \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}_1 \\ \boldsymbol{y}' &= \boldsymbol{R}_x(-\phi) \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}_1 = \begin{bmatrix} 0 \\ \cos \phi \\ -\sin \phi \end{bmatrix}_1 \\ \boldsymbol{z}_0 &= \boldsymbol{R}_x(-\phi) \boldsymbol{R}_y(-\theta) \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}_1 = \begin{bmatrix} -\sin \theta \\ \cos \theta \sin \phi \\ \cos \theta \cos \phi \end{bmatrix}_1 \end{aligned}$$

En regroupant les termes :

$$\begin{aligned} \boldsymbol{\nu} &= {}^1(\boldsymbol{\omega}_{1/0}) = \dot{\phi} \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} + \dot{\theta} \begin{pmatrix} 0 \\ \cos \phi \\ -\sin \phi \end{pmatrix} + \dot{\psi} \begin{pmatrix} -\sin \theta \\ \sin \phi \cos \theta \\ \cos \phi \cos \theta \end{pmatrix} \\ \boldsymbol{\nu} &= \begin{pmatrix} 1 & 0 & -\sin \theta \\ 0 & \cos \phi & \sin \phi \cos \theta \\ 0 & -\sin \phi & \cos \phi \cos \theta \end{pmatrix} \begin{pmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{pmatrix} \end{aligned}$$

$$W_\eta = \begin{pmatrix} 1 & 0 & -\sin(\theta) \\ 0 & \cos(\phi) & \sin(\phi) \cos(\theta) \\ 0 & -\sin(\phi) & \cos(\phi) \cos(\theta) \end{pmatrix}$$

On a besoin de l'inverse de cette matrice pour trouver  $\mathbf{T}_e$  :

$$\boldsymbol{\nu} = W_\eta \dot{\boldsymbol{\eta}} \quad \rightarrow \quad \dot{\boldsymbol{\eta}} = W_\eta^{-1} \boldsymbol{\nu}$$

$$\begin{pmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{pmatrix} = \begin{pmatrix} 1 & \sin(\phi) \tan(\theta) & \cos(\phi) \tan(\theta) \\ 0 & \cos(\phi) & -\sin(\phi) \\ 0 & \sin(\phi) / \cos(\theta) & \cos(\phi) / \cos(\theta) \end{pmatrix} \begin{pmatrix} p \\ q \\ r \end{pmatrix} \quad (1)$$

Cette matrice relie directement les vitesses angulaires exprimées dans le repère 1 aux taux de changement des angles d'Euler. Elle est une composante essentielle du calcul de  $\mathbf{x}_{dot}$  dans `dynamics_model_euler.m` : `deuler = Wn_inv * omega;`.

### 3 Dynamique (Équations de Mouvement)

Les Équations de Mouvement (ÉDM) décrivent comment le drone accélère (linéairement et angulairement) en réponse aux forces et couples appliqués. On utilise la [Seconde Loi de Newton](#) pour la translation et [les équations d'Euler](#) pour la rotation.

#### 3.1 Dynamique Translationnelle

Pour établir l'équation de translation, on part de la dynamique générale d'un corps rigide. Dans le repère mobile du corps ( $\mathcal{R}_1$ ), la loi de Newton s'écrit initialement :

$$\frac{d(m\mathbf{v}_1)}{dt} + \boldsymbol{\nu} \times (m\mathbf{v}_1) = {}^1\mathbf{F}_{ext}$$

Cependant, l'objectif est d'exprimer le mouvement dans le repère inertiel ( $\mathcal{R}_0$ ). Puisque ce repère est fixe ("ne roule pas"), les termes de forces centrifuges et de Coriolis ( $\boldsymbol{\nu} \times m\mathbf{v}$ ) s'annulent. L'équation se simplifie alors considérablement. De cette façon, on peut procéder par l'analyse des forces.

→ [Poussée ou Thrust](#) : la force générée par chaque propulseur  $i$  est dirigée selon l'axe  $\mathbf{k}_1$  (axe Z du corps). Sa magnitude dépend des caractéristiques physiques de l'hélice et de l'air :

$$\mathbf{T}_i = T_i \mathbf{k}_1 \quad \text{avec} \quad T_i = C_T \rho \pi l^4 \omega_i^2 = k \omega_i^2$$

D'après les données du système :

- $C_T = 10^{-5}$  (Coefficient de poussée)
- $\rho = 1.293 \text{ kg/m}^3$  (Densité de l'air)
- $l = 0.25 \text{ m}$  (Longueur de pale)



La poussée totale exprimée dans le repère 1 est la somme des contributions des 6 moteurs :

$${}^1\mathbf{T}_B = \sum_{i=1}^6 {}^1\mathbf{T}_i = \begin{pmatrix} 0 \\ 0 \\ T_B \end{pmatrix} \quad \text{où} \quad T_B = k \sum_{i=1}^6 \omega_i^2$$

→ Gravité : exprimée directement dans le repère 0 :

$${}^0\mathbf{F}_g = \begin{pmatrix} 0 \\ 0 \\ -mg \end{pmatrix}$$

Si l'on suppose que la masse est constante, l'équation initiale devient

$$m\ddot{\mathbf{v}}_1 + \boldsymbol{\nu} \times (m\mathbf{v}_1) = \mathbf{R}^T {}^0\mathbf{F}_g + {}^1\mathbf{T}_B$$

Pour obtenir l'équation finale du mouvement (l'accélération  $\ddot{\boldsymbol{\xi}}$ ), on change au repère inertiel en projetant la poussée dans ce repère à l'aide de la matrice de rotation  $\mathbf{R}_{0 \leftarrow 1}$  et en annulant la force centrifuge. L'équation fondamentale de la dynamique de translation devient :

$$m\ddot{\boldsymbol{\xi}} = {}^0\mathbf{F}_g + \mathbf{R}_{0 \leftarrow 1} {}^1\mathbf{T}_B \quad (2)$$

En développant cette équation sous forme matricielle, on obtient la relation utilisée dans la simulation pour calculer les accélérations linéaires :

$$\begin{pmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ -g \end{pmatrix} + \frac{k}{m} \mathbf{R}_{0 \leftarrow 1} \begin{pmatrix} 0 \\ 0 \\ \omega_1^2 + \omega_2^2 + \omega_3^2 + \omega_4^2 + \omega_5^2 + \omega_6^2 \end{pmatrix} \quad (3)$$

Ceci est implémenté dans le code comme

```
dV=[0;0;-PARAMS.g]+(1/PARAMS.m)*(R*[0;0;sum(T_i)]);
```

En développant le produit de la matrice de rotation par la force, on peut également exprimer cette équation matricielle de la manière suivante :

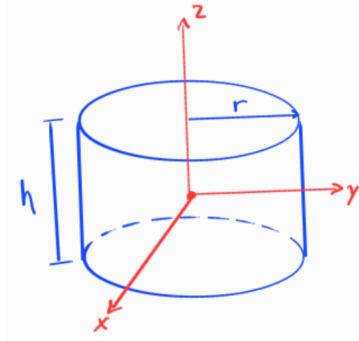
$$\begin{pmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ -g \end{pmatrix} + \frac{T_B}{m} \begin{pmatrix} \cos \phi \sin \theta \cos \psi + \sin \phi \sin \psi \\ \cos \phi \sin \theta \sin \psi - \sin \phi \cos \psi \\ \cos \phi \cos \theta \end{pmatrix} \quad (4)$$

### 3.2 Dynamique Rotationnelle

L'analyse de la rotation débute par la modélisation de la structure physique du drone et des couples générés par les rotors.

### 3.2.1 Matrice d'Inertie du Corps Rigide

L'hexacoptère est modélisé comme une structure symétrique à six bras. Pour simplifier le calcul des moments d'inertie, on approxime le corps central et l'ensemble de la structure comme un cylindre plein homogène de masse  $m$ , de rayon  $r$  et de hauteur  $h$ .



**FIGURE 6** – Représentation du drone comme un cylindre

La matrice d'inertie  $\mathbf{I}$  exprimée dans le repère du corps  $\mathcal{R}_1$  est diagonale en raison de la symétrie du corps :

$$\mathbf{I} = \begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix}$$

Les moments d'inertie principaux sont calculés selon les formules du cylindre plein. L'inertie autour de l'axe vertical ( $Z_1$ ) est calculée comme suit :

$$I_{zz} = \frac{1}{2}mr^2$$

Et l'inertie autour des axes horizontaux ( $X_1, Y_1$ ) :

$$I_{xx} = I_{yy} = \frac{1}{12}m(3r^2 + h^2)$$

Avec les paramètres physiques du système ( $m = 2 \text{ kg}$ ,  $r = 0,1 \text{ m}$ ,  $h = 0,04 \text{ m}$ ), cette matrice définit la résistance du drone aux accélérations angulaires.

### 3.2.2 Mouvement de Lacet (Yaw - $\tau_\psi$ ) :

→ **Couple de Traînée (Drag Torque)** : Chaque hélice  $i$  génère un couple de traînée aérodynamique  $\mathbf{C}_i$  opposé à son sens de rotation. Ce couple est proportionnel au carré de la vitesse de rotation  $\omega_i$  :

$$\begin{aligned} \mathbf{C}_i &= -C_i \operatorname{sign}(\omega_i) \mathbf{z}_1 \\ C_i &= C_D \rho \pi l^4 \omega_i^2 = b \omega_i^2 \end{aligned}$$

Où le coefficient de traînée  $b$  regroupe les constantes aérodynamiques :

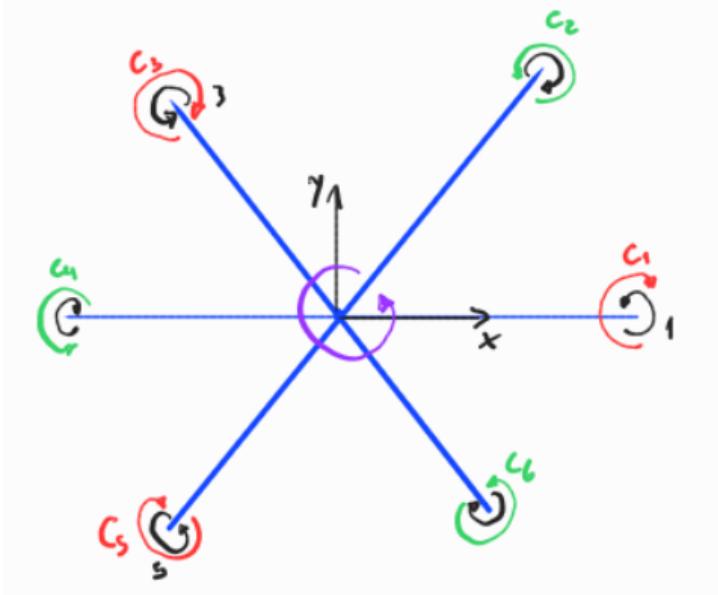


- $C_D = 10^{-7}$  (Coefficient de traînée adimensionnel)
- $\rho = 1.293 \text{ kg/m}^3$  (Densité de l'air)
- $l = 0.25 \text{ m}$  (Longueur de pale)

→ **Couple Total par Moteur :** Le couple total  $\tau_{M_i}$  généré par un ensemble moteur-hélice par rapport à l'axe Z comprend le couple de traînée aérodynamique et l'effet d'inertie du rotor lui-même ( $I_{M_i}$ ) lors des changements de vitesse :

$$\boxed{\tau_{M_i} = b\omega_i^2 + I_{M_i}\dot{\omega}_i}$$

Pour un hexacoptère, la configuration des rotors alterne les sens de rotation pour annuler le couple net en vol stationnaire :



**FIGURE 7 –** Mouvement de Lacet (Yaw -  $\tau_\psi$ )

Les moteurs impairs (1, 3, 5) tournent dans un sens anti-horaire et les moteurs pairs (2, 4, 6) tournent dans le sens opposé horaire. Le couple total de lacet  $\tau_\psi$  est la somme algébrique des contributions de chaque moteur. En supposant une convention de signes alternés :

$$\tau_\psi = \sum \tau_{M_i} = \sum C_i + \sum I_{M_i}\dot{\omega}_i$$

L'équation finale du couple de lacet, incluant les effets aérodynamiques et inertIELS des rotors, est :

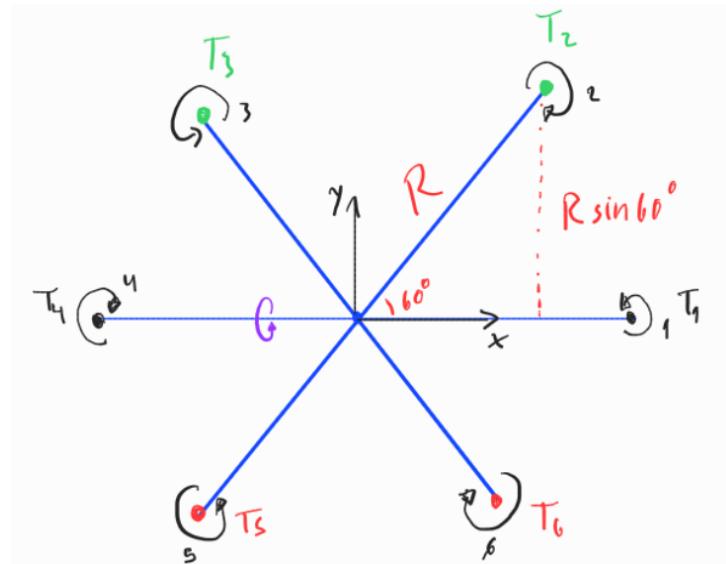
$$\boxed{\tau_\psi = b(-\omega_1^2 + \omega_2^2 - \omega_3^2 + \omega_4^2 - \omega_5^2 + \omega_6^2) + I_{M_i} \sum_{i=1}^6 \dot{\omega}_i}$$

Note : Le terme  $I_{M_i} \sum \dot{\omega}_i$  représente l'effet gyroscopique interne des moteurs accélérant, souvent négligé dans les modèles simplifiés mais inclus ici pour la modélisation.



### 3.2.3 Mouvement de Roulis (Roll - $\tau_\phi$ )

Le couple de roulis provoque une rotation autour de l'axe longitudinal  $x_1$ . Pour générer ce mouvement, le contrôleur applique une stratégie différentielle :



**FIGURE 8 –** Mouvement de Roulis (Roll -  $\tau_\phi$ )

Les moteurs 2 et 3 augmentent leur vitesse angulaire ( $\omega \uparrow$ ), les moteurs 5 et 6 diminuent leur vitesse angulaire ( $\omega \downarrow$ ) et les moteurs 1 et 4 ne changent pas, car ils sont situés sur l'axe  $x_1$  et ont un bras de levier nul pour ce mouvement. Le bras de levier pour les moteurs diagonaux par rapport à l'axe X est la projection  $R \sin(60^\circ)$ . Le couple total est la somme des couples générés par chaque force de poussée  $T_i$  multipliée par sa distance perpendiculaire à l'axe X :

$$\tau_\phi = \sum \mathbf{r}_y \times \mathbf{T}_B$$

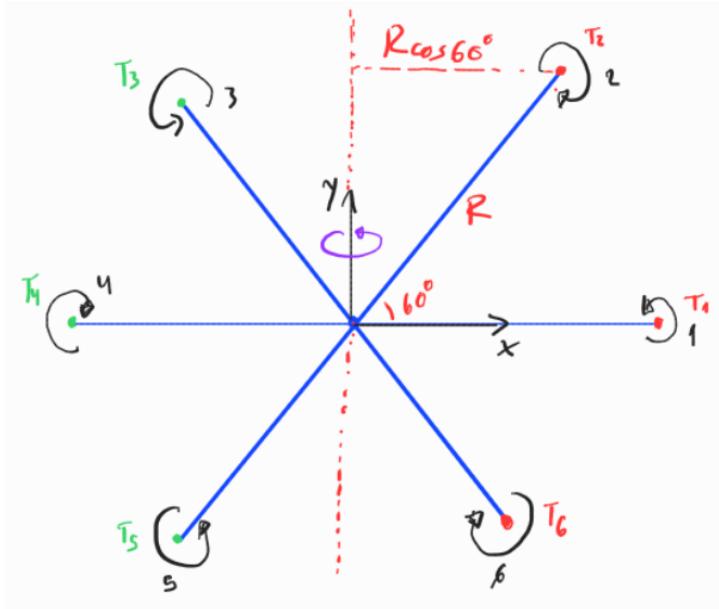
$$\tau_\phi = (R \sin 60^\circ)T_2 + (R \sin 60^\circ)T_3 - (R \sin 60^\circ)T_5 - (R \sin 60^\circ)T_6$$

En simplifiant l'expression précédente, l'équation finale du couple de roulis est :

$$\boxed{\tau_\phi = \frac{\sqrt{3}}{2} R k (\omega_2^2 + \omega_3^2 - \omega_5^2 - \omega_6^2)}$$

### 3.2.4 Mouvement de Tangage (Pitch - $\tau_\theta$ )

Le couple de tangage provoque une rotation autour de l'axe transversal  $y_1$ .



**FIGURE 9 –** Mouvement de Tangage (Pitch -  $\tau_\theta$ )

Les moteurs 1 et 4 ont un bras de levier maximal égal à  $R$  et les moteurs 2, 3, 5 et 6 ont un bras de levier réduit, projeté sur l'axe X :  $R \cos(60^\circ) = 1/2R$ . Selon la convention de signe, une force à l'arrière (les moteurs 3, 4 et 5) crée un couple positif, et une force à l'avant (les moteurs 1, 2 et 6) crée un couple négatif :

$$\tau_\theta = \sum \mathbf{r}_x \times \mathbf{T}_B$$

$$\tau_\theta = -RT_1 + RT_4 - (R \cos 60^\circ)T_2 + (R \cos 60^\circ)T_3 + (R \cos 60^\circ)T_5 - (R \cos 60^\circ)T_6$$

En factorisant par  $Rk$  et en remplaçant  $\cos(60^\circ) = 1/2$  :

$$\boxed{\tau_\theta = Rk \left( -\omega_1^2 + \omega_4^2 - \frac{1}{2}\omega_2^2 + \frac{1}{2}\omega_3^2 + \frac{1}{2}\omega_5^2 - \frac{1}{2}\omega_6^2 \right)}$$

Ces équations montrent que le tangage est plus "sensible" aux moteurs 1 et 4 (coefficients 1) qu'aux autres moteurs (coefficients 0.5), ce qui est logique géométriquement. La valeur du bras  $R$  du drone qui est prise en compte est 0.3 m

### 3.2.5 Le Vecteur des Couples Extérieurs

En regroupant les équations de  $\tau_\phi$ ,  $\tau_\theta$  et  $\tau_\psi$  dérivées précédemment, on forme le vecteur des moments extérieurs appliqués au centre de gravité :

$${}^1(\boldsymbol{\tau}_{ext}^G) = \begin{pmatrix} \tau_\phi \\ \tau_\theta \\ \tau_\psi \end{pmatrix} = \begin{pmatrix} \frac{\sqrt{3}}{2}Rk(\omega_2^2 + \omega_3^2 - \omega_5^2 - \omega_6^2) \\ Rk(-\omega_1^2 - \frac{1}{2}\omega_2^2 + \frac{1}{2}\omega_3^2 + \omega_4^2 + \frac{1}{2}\omega_5^2 - \frac{1}{2}\omega_6^2) \\ b(-\omega_1^2 + \omega_2^2 - \omega_3^2 + \omega_4^2 - \omega_5^2 + \omega_6^2) + I_{M_i} \sum \dot{\omega}_i \end{pmatrix}$$

### 3.2.6 L'Effet Gyroscopique ( $\Gamma$ )

L'hexacoptère possède 6 rotors tournant à haute vitesse. Lorsque le drone lui-même tourne, l'axe de rotation de ces hélices change, ce qui crée un couple de réaction gyroscopique  $\Gamma$ . Soit  $I_r$  le moment d'inertie du rotor (partie tournante du moteur + hélice) et  $\omega_\Gamma$  la vitesse résiduelle des hélices :

$$\omega_\Gamma = \omega_1 - \omega_2 + \omega_3 - \omega_4 + \omega_5 - \omega_6$$

Pour la simulation du drone, on a pris la valeur de  $I_r = 3.357 \cdot 10^{-5} \text{ kg} \cdot \text{m}^2$  de [1]. Le couple gyroscopique est défini par le produit vectoriel entre la vitesse angulaire du drone ( $\boldsymbol{\nu}$ ) et le moment cinétique des rotors ( $H_i$ ) :

$$\Gamma = \sum \boldsymbol{\nu} \times \mathbf{H}_i = \sum \boldsymbol{\nu} \times (I_r \omega_i \mathbf{z}_1)$$

En développant le produit vectoriel :

$$\Gamma = I_r \begin{bmatrix} p \\ q \\ r \end{bmatrix} \times \begin{bmatrix} 0 \\ 0 \\ \omega_\Gamma \end{bmatrix} = I_r \omega_\Gamma \begin{bmatrix} q \\ -p \\ 0 \end{bmatrix}$$

### 3.2.7 Équation Finale du Mouvement Angulaire

L'équation d'Euler complète, incluant l'inertie du corps, le couplage inertiel, l'effet gyroscopique et les couples extérieurs, s'écrit :

$${}^1\mathbf{I}\dot{\boldsymbol{\nu}} + \boldsymbol{\nu} \times ({}^1\mathbf{I}\boldsymbol{\nu}) + \Gamma = {}^1\boldsymbol{\tau}_{ext}^G \quad (5)$$

L'objectif est d'isoler l'accélération angulaire  $\dot{\boldsymbol{\nu}} = [\dot{p} \ \dot{q} \ \dot{r}]^T$  pour la simulation numérique. En inversant la matrice d'inertie  $\mathbf{I}$  (diagonale) et en passant les autres termes à droite :

$$\begin{pmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{pmatrix} = \begin{pmatrix} 1/I_{xx} & 0 & 0 \\ 0 & 1/I_{yy} & 0 \\ 0 & 0 & 1/I_{zz} \end{pmatrix} \left[ \begin{pmatrix} \tau_\phi \\ \tau_\theta \\ \tau_\psi \end{pmatrix} - \begin{pmatrix} p \\ q \\ r \end{pmatrix} \times \begin{pmatrix} I_{xx}p \\ I_{yy}q \\ I_{zz}r \end{pmatrix} - I_r \omega_\Gamma \begin{pmatrix} q \\ -p \\ 0 \end{pmatrix} \right]$$



En développant ligne par ligne, on obtient le système d'équations différentielles couplées implémenté dans `dynamics_model_euler.m` :

$$\begin{aligned}\dot{p} &= \frac{1}{I_{xx}} ((I_{yy} - I_{zz})qr - I_r\omega_\Gamma q + \tau_\phi) \\ \dot{q} &= \frac{1}{I_{yy}} ((I_{zz} - I_{xx})pr + I_r\omega_\Gamma p + \tau_\theta) \\ \dot{r} &= \frac{1}{I_{zz}} ((I_{xx} - I_{yy})pq + \tau_\psi)\end{aligned}\tag{6}$$

Ces équations montrent que les mouvements sont fortement liés : une rotation en lacet ( $r$ ) combinée à un tangage ( $q$ ) induira naturellement un roulis ( $p$ ) à cause du terme  $(I_{yy} - I_{zz})qr$ .

## 4 Vecteur d'État

La dynamique de l'hexacoptère est régie par les équations fondamentales de Newton-Euler. Avant de définir l'état du système, on pose les équations régissant le mouvement de translation et de rotation, telles qu'illustrées dans l'analyse théorique par les équations 2 et 5 :

$$m\ddot{\xi} = {}^0(\mathbf{R}_{ext})$$

où :

- $m$  est la masse totale du drone,
- $\ddot{\xi}$  est son accélération linéaire,
- ${}^0(\mathbf{R}_{ext})$  est la résultante des forces extérieures exprimées dans le repère terrestre,

$${}^1\mathbf{I}\dot{\nu} + \boldsymbol{\nu} \times ({}^1\mathbf{I}\boldsymbol{\nu}) + \boldsymbol{\Gamma} = {}^1\boldsymbol{\tau}_{ext}^G$$

où :

- $\mathbf{I}$  représente la matrice d'inertie du drone,
- $\boldsymbol{\nu}$  est la vitesse angulaire exprimée dans le repère du corps ( $\mathcal{R}_1$ ),
- $\boldsymbol{\Gamma}$  est la couple de réaction gyroscopique
- ${}^1\boldsymbol{\tau}_{ext}^G$  est la somme des moments extérieurs appliqués au centre de gravité  $G$  exprimés dans  $\mathcal{R}_1$  ;

Ces équations fondamentales, précédemment développées à l'aide d'un modèle avec des angles d'Euler, serviront à définir le vecteur d'état et donc à construire l'équation différentielle des états.

## 4.1 Modélisation avec des angles d'Euler

Pour décrire la cinématique du drone par rapport au repère de référence  $\mathcal{R}_0$ , on définit le [vecteur de position linéaire](#) :

$$\boldsymbol{\xi} = [x \ y \ z]^T$$

Ce vecteur représente les coordonnées cartésiennes du centre de masse dans l'espace. Et pour représenter l'orientation du drone par rapport au repère de référence  $\mathcal{R}_0$ , où  $\varphi$  est le roulis (roll),  $\theta$  est le tangage (pitch), et  $\psi$  est le lacet (yaw), on définit le [vecteur de rotation \(angles d'Euler\)](#) :

$$\boldsymbol{\eta} = [\varphi \ \theta \ \psi]^T$$

En combinant ces grandeurs cinématiques et leurs dérivées, on construit le vecteur d'état complet  $\mathbf{x}$  de dimension 12 utilisé pour la simulation :

$$\mathbf{x} = \begin{bmatrix} \boldsymbol{\xi} \\ \dot{\boldsymbol{\xi}} \\ \boldsymbol{\nu} \\ \boldsymbol{\eta} \end{bmatrix} = \begin{bmatrix} x \\ y \\ z \\ v_x \\ v_y \\ v_z \\ p \\ q \\ r \\ \phi \\ \theta \\ \psi \end{bmatrix}$$

En utilisant les équations 1, 4, 6, l'équation différentielle d'état peut être écrite comme suit :

$$\dot{\mathbf{x}} = \begin{bmatrix} v_x \\ v_y \\ v_z \\ \left( \begin{array}{c} 0 \\ 0 \\ -g \end{array} \right) + \frac{T_B}{m} \begin{pmatrix} \cos \phi \sin \theta \cos \psi + \sin \phi \sin \psi \\ \cos \phi \sin \theta \sin \psi - \sin \phi \cos \psi \\ \cos \phi \cos \theta \end{pmatrix} \\ \frac{1}{I_{xx}} ((I_{yy} - I_{zz})qr - I_r \omega_\Gamma q + \tau_\phi) \\ \frac{1}{I_{yy}} ((I_{zz} - I_{xx})pr + I_r \omega_\Gamma p + \tau_\theta) \\ \frac{1}{I_{zz}} ((I_{xx} - I_{yy})pq + \tau_\psi) \\ \begin{pmatrix} 1 & \sin(\phi) \tan(\theta) & \cos(\phi) \tan(\theta) \\ 0 & \cos(\phi) & -\sin(\phi) \\ 0 & \sin(\phi)/\cos(\theta) & \cos(\phi)/\cos(\theta) \end{pmatrix} \begin{pmatrix} p \\ q \\ r \end{pmatrix} \end{bmatrix}$$



## 4.2 Modélisation avec des quaternions

Il est crucial de noter que la matrice inverse de transformation des vitesses angulaires  $W_\eta^{-1}$ , qui fait partie de l'équation 1, ne peut être définie que si et seulement si  $\theta \neq \pi/2 + k\pi$ , ( $k \in \mathbb{Z}$ ). Cette restriction est la principale conséquence de la formulation par les angles d'Euler et conduit au phénomène de blocage de cardan (**gimbal lock**), situation typique dans laquelle le système perd un degré de liberté angulaire.

Pour contourner ce problème comme indiqué dans [2], il est possible d'opter pour une autre représentation de l'orientation spatiale du hexacoptère. La rotation de l'aéronef d'un repère à un autre peut être identifiée par quatre paramètres appelés quaternions. L'avantage principal d'une approche basée sur les quaternions réside non seulement dans l'absence de singularités, mais aussi dans la simplicité des calculs.

La représentation par quaternions présentée dans l'article [3] repose sur le **théorème de rotation d'Euler**, qui stipule que tout déplacement d'un corps rigide, lorsqu'un point est maintenu fixe, est équivalent à une rotation. Par conséquent, si  $\alpha$  représente l'angle de rotation autour du vecteur unitaire  $\mathbf{u}$ , il est possible de définir un quaternion  $\mathbf{q}$  comme suit :

$$\mathbf{q} = \begin{bmatrix} \cos(\alpha/2) \\ \mathbf{u} \sin(\alpha/2) \end{bmatrix} = \begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix}$$

Contrairement aux angles d'Euler, les rotations définies par des quaternions ne nécessitent pas un ensemble d'axes de rotation prédéfinis, car l'axe de rotation unique peut être ajusté de manière continue. Étant donné que cette méthode de rotation s'effectue autour d'une direction arbitraire, elle n'implique qu'un seul axe de rotation. Il est donc impossible de perdre un degré de liberté, ce qui garantit qu'un blocage de cardan ne peut pas survenir. Par ailleurs, un quaternion est défini par quatre composantes décrivant la rotation dans l'espace tridimensionnel. Pour qu'il représente correctement une rotation, les éléments du quaternion  $\mathbf{q}$  doivent satisfaire une équation de contrainte essentielle, appelée condition de normalisation (ou condition de normalité), qui s'écrit :

$$q_0^2 + q_1^2 + q_2^2 + q_3^2 = 1$$

De cette manière, en utilisant les quaternions, il est possible de construire une matrice  $\mathbf{Q}$ , qui est l'analogie de la matrice de rotation  $\mathbf{R}$  des angles d'Euler, permettant la transformation du système de référence 1 vers le système de référence 0. Cette matrice  $\mathbf{Q}$  est également orthogonale et est définie par les composantes du quaternion  $\mathbf{q} = [q_0 \ q_1 \ q_2 \ q_3]^T$  comme suit :

$$\mathbf{Q} = \begin{bmatrix} q_0^2 + q_1^2 - q_2^2 - q_3^2 & 2(q_1 q_2 - q_0 q_3) & 2(q_1 q_3 + q_0 q_2) \\ 2(q_1 q_2 + q_0 q_3) & q_0^2 - q_1^2 + q_2^2 - q_3^2 & 2(q_2 q_3 - q_0 q_1) \\ 2(q_1 q_3 - q_0 q_2) & 2(q_2 q_3 + q_0 q_1) & q_0^2 - q_1^2 - q_2^2 + q_3^2 \end{bmatrix}$$

Et en utilisant les quaternions, la relation entre le vecteur de vitesse angulaire du corps  $\nu$  et la dérivée du quaternion  $\dot{\mathbf{q}}$  est linéaire et ne souffre pas de singularité. Elle est donnée par l'équation suivante :

$$\dot{\mathbf{q}} = \frac{1}{2} \boldsymbol{\Omega}(\nu) \mathbf{q}$$

Où  $\boldsymbol{\Omega}(\mathbf{v})$  est une matrice antisymétrique construite à partir des composantes de la vitesse angulaire  $\nu$ , et  $\mathbf{q}$  est le quaternion.

$$\boldsymbol{\Omega}(\mathbf{v}) = \begin{bmatrix} 0 & -p & -q & -r \\ p & 0 & r & -q \\ q & -r & 0 & p \\ r & q & -p & 0 \end{bmatrix}$$

En utilisant cette modélisation par quaternions, il devient simple d'établir une analogie avec le modèle précédent basé sur les angles d'Euler et d'en déduire le vecteur d'état  $\mathbf{x}$  de dimension 13 utilisé dans le fichier de simulation `dynamics_model_quat.m`.

$$\mathbf{x} = \begin{bmatrix} \xi \\ \dot{\xi} \\ \nu \\ \mathbf{q} \end{bmatrix} = \begin{bmatrix} x \\ y \\ z \\ v_x \\ v_y \\ v_z \\ p \\ q \\ r \\ q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix}$$

$$\dot{\mathbf{x}} = \begin{bmatrix} v_x \\ v_y \\ v_z \\ \left( \begin{array}{c} 0 \\ 0 \\ -g \end{array} \right) + \frac{k}{m} \mathbf{Q} \begin{pmatrix} 0 \\ 0 \\ \omega_1^2 + \omega_2^2 + \omega_3^2 + \omega_4^2 + \omega_5^2 + \omega_6^2 \end{pmatrix} \\ \frac{1}{I_{xx}} ((I_{yy} - I_{zz})qr - I_r \omega_\Gamma q + \tau_\phi) \\ \frac{1}{I_{yy}} ((I_{zz} - I_{xx})pr + I_r \omega_\Gamma p + \tau_\theta) \\ \frac{1}{I_{zz}} ((I_{xx} - I_{yy})pq + \tau_\psi) \\ \frac{1}{2} \begin{pmatrix} 0 & -p & -q & -r \\ p & 0 & r & -q \\ q & -r & 0 & p \\ r & q & -p & 0 \end{pmatrix} \begin{pmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{pmatrix} \end{bmatrix}$$

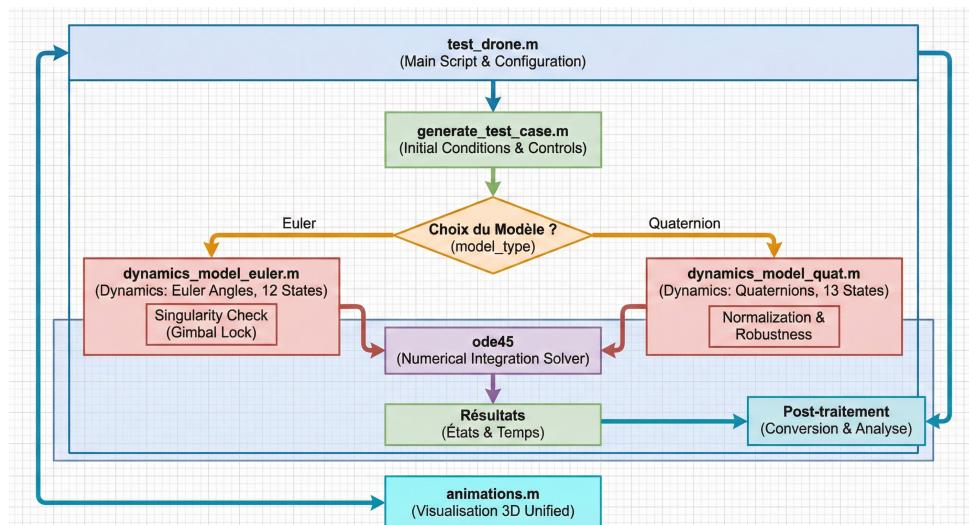
## 5 Implémentation et Structure du Code

La mise en œuvre de la simulation est réalisée via une suite de scripts MATLAB modulaires. Cette architecture permet de comparer directement deux approches de modélisation (Angles d'Euler vs Quaternions) au sein du même environnement de test. Cette section détaille la fonction et la structure de chaque composant logiciel.

### 5.1 Organisation Générale des Fichiers

Le projet s'articule autour de cinq fichiers principaux, séparant la logique de test, la génération de scénarios, les moteurs physiques et la visualisation :

1. `test_drone.m` : Le script principal. Il configure les paramètres, choisit le modèle dynamique (Euler ou Quaternion), lance l'intégration numérique et gère l'affichage.
2. `generate_test_case.m` : Une fonction auxiliaire qui génère les conditions initiales ( $\mathbf{x}_0$ ) et les commandes de poussée ( $\mathbf{T}_B$ ) pour différents scénarios de vol standardisés.
3. `dynamics_model_euler.m` : Le moteur physique utilisant la convention des angles d'Euler (ZYX). Il gère 12 états et inclut une détection de singularité.
4. `dynamics_model_quat.m` : Le moteur physique utilisant les quaternions unitaires. Il gère 13 états et offre une robustesse totale pour toutes les orientations.
5. `animations.m` : Le module de visualisation 3D capable d'interpréter et d'animer les données issues des deux types de modèles.



**FIGURE 10** – Diagramme montrant le flux de données. `test_drone.m` initialise les paramètres et appelle `generate_test_case.m`. Selon le choix de l'utilisateur, `ode45` intègre soit `dynamics_model_euler.m` soit `dynamics_model_quat.m`. Les résultats sont unifiés pour être visualisés par `animations.m`.

## 5.2 Le Lanceur de Simulation : test\_drone.m

Ce script est le point d'entrée unique. Il permet à l'utilisateur de configurer rapidement l'expérience (choix du cas, choix du modèle mathématique, choix de lancer ou non l'animation et le temps de simulation) et d'exécuter la simulation. Les fonctionnalités clés sont les suivantes :

- **Configuration Utilisateur** : Des variables claires permettent de choisir le scénario (`test_case`), le type de modèle (`model_type = 'euler'` ou `'quat'`) et d'activer ou non l'animation.
- **Définition des Paramètres** : Initialise la structure `PARAMS` avec les constantes physiques (Masse  $m = 2.0\text{kg}$ , Inertie  $\mathbf{I}$ , coefficients aérodynamiques  $k, b$ ).
- **Préparation du Scénario** : Appelle `generate_test_case` pour obtenir les vecteurs de poussée cibles (`TB`) et l'état initial. Convertit ensuite les poussées en vitesses de rotation moteurs au carré (`u`), qui sont les entrées réelles du système dynamique.
- **Sélection Dynamique du Modèle** : Utilise une structure `switch` pour définir le pointeur de fonction (`dyn_f`) vers le modèle physique approprié. Si le modèle quaternion est choisi, il convertit automatiquement les angles initiaux en quaternions.
- **Post-traitement** : Après la simulation (`ode45`), il uniformise les données de sortie (conversion Quaternions → Euler) pour faciliter le traçage des courbes.

**Code 1 : Extrait de test\_drone.m**

matlab

```
% --- User Settings ---
test_case    = 1; % 0:Hover, 1:Yaw, 2:Takeoff, 3:Pitch, 4:Roll,
                % 5:GimbalLock
model_type   = 'euler';% 'euler' or 'quat'
tf = 2; dt = 1e-2;

% ... Définition des PARAMS (m, I, k, b) ...

% --- Generate Inputs ---
[TB, x0] = generate_test_case(test_case, PARAMS);
u = 1/PARAMS.k * TB; % Conversion Poussée -> Vitesse Moteur^2

% --- Select Dynamics ---
switch model_type
    case 'euler'
        dyn_f = @(t,x) dynamics_model_euler(t, x, u, PARAMS);
    case 'quat'
        dyn_f = @(t,x) dynamics_model_quat(t, x, u, PARAMS);
        % Conversion état initial Euler -> Quaternion si nécessaire
        % ...
end
```

```
% --- Simulation ---
[t, X] = ode45(dyn_f, 0:dt:tf, x0);
```

### 5.3 Le Générateur de Scénarios : generate\_test\_case.m

Cette fonction isole la logique de définition des manœuvres. Elle retourne les forces cibles et l'état initial, permettant de tester le drone dans des configurations standardisées. Ses fonctionnalités clés sont :

- **Calcul du Stationnaire** : Calcule automatiquement la force de poussée par moteur nécessaire pour maintenir le drone en vol stationnaire, en équilibrant la force de gravité totale. ( $T_{\text{hover}} = mg/6$ ).
- **Bibliothèque de Cas** :
  - **Cas 0-4** : Tests unitaires classiques (Stationnaire, Lacet, Décollage, Tangage, Roulis).
  - **Cas 5 (Gimbal Lock)** : Un cas critique où le drone est initialisé avec un angle de tangage  $\theta = \pi/2$  ( $90^\circ$ ). Ce cas est spécifiquement conçu pour faire échouer le modèle Euler et valider le modèle Quaternion.
  - **Cas 6** : Manœuvre combinée (Décollage + Rotation).

**Code 2 : Extrait de generate\_test\_case.m**

matlab

```
function [TB,x0] = generate_test_case(test_case, PARAMS)
Fg = PARAMS.m * PARAMS.g;
T_hover = Fg / 6;
x0 = zeros(12,1); % État par défaut

switch test_case
  case 0 % Hover
    TB = T_hover * ones(6,1);
  case 1 % Yaw rotation (Déséquilibre horaire/anti-horaire)
    T_yaw = 0.3;
    TB = [T_hover+T_yaw; T_hover-T_yaw; ...];
  case 5 % Gimbal lock test
    % Initialisation à 90 degrés de tangage
    x0 = [zeros(9,1); 0; pi/2; 0];
    TB = ones(6,1)*T_hover; % Poussée simple
    % ...
end
end
```

## 5.4 Les Modèles Dynamiques : dynamics\_model\_\*.m

Le cœur physique de la simulation est divisé en deux fichiers distincts pour permettre une comparaison rigoureuse.

### dynamics\_model\_euler.m

Ce modèle utilise la représentation minimale de 12 états : Position, Vitesse, Vitesse Angulaire, Angles d'Euler.

- **Matrice de Rotation** :  $R = \text{rotz}(\psi) * \text{roty}(\theta) * \text{rotx}(\phi);$  est construite pour transformer les vecteurs du repère corps au repère inerte. Il est à noter que Les fonctions `rotz`, `roty`, `rotx` ont été définies même si elles l'étaient déjà via une boîte à outils afin de permettre une plus grande personnalisation.
- **Cinématique** : Utilise la matrice  $W_{\dot{\eta}}^{-1}$  pour relier les vitesses angulaires du corps ( $\boldsymbol{\nu}$ ) aux dérivées des angles d'Euler ( $\dot{\boldsymbol{\eta}}$ ).
- **Singularité** : Inclut une vérification explicite `if abs(cos(theta)) < 1e-6` pour détecter le risque de verrouillage de cardan et émettre un avertissement.

### dynamics\_model\_quat.m

Ce modèle utilise la représentation par quaternions (13 états).

- **Matrice  $Q$**  : Dans ce cas, la fonction `quatToRotm(quat)` a été définie pour convertir les quaternions en matrice  $Q$ , qui sera utilisée pour transférer les forces du repère du corps au repère terre.
- **Cinématique** : Utilise l'équation linéaire  $\dot{\boldsymbol{q}} = \frac{1}{2}\boldsymbol{\Omega}(\boldsymbol{\omega})\boldsymbol{q}$ , qui est inconditionnellement stable.
- **Normalisation** : Assure que le quaternion reste unitaire à chaque pas de temps.
- **Robustesse** : Permet de simuler des acrobaties ou des orientations verticales sans erreur numérique.

### Code 3 : Comparaison Cinématique (Extraits)

matlab

```
% --- DANS dynamics_model_euler.m ---
% Matrice cinématique Euler (Singulière si theta = pi/2)
Wn_inv = [ 1, sin(phi)*tan(theta), cos(phi)*tan(theta);
            0,      cos(phi),           -sin(phi);
            0, sin(phi)/cos(theta), cos(phi)/cos(theta) ];
deuler = Wn_inv * omega;

% --- DANS dynamics_model_quat.m ---
% Cinématique Quaternion (Jamais singulière)
OMEGA = [ 0, -p, -q, -r;
            p,  0,  r, -q;
```

```

q, -r, 0, p;
r, q, -p, 0 ];
dq = 0.5 * OMEGA * quat;

```

## 5.5 Visualisation Unifiée : animations.m

Ce script gère l'affichage 3D quel que soit le modèle utilisé. Il détecte le format des données d'entrée (Euler Nx3 ou Quaternions Nx4) et adapte le calcul de la matrice de rotation pour l'affichage. Ses fonctionnalités clés sont :

- **Polymorphisme** : Accepte indifféremment des angles d'Euler ou des quaternions grâce à une structure `switch type`.
- **Rendu Graphique** : Dessine le corps, les 6 bras, les rotors (cercles) et les repères locaux ( $\mathbf{x}_B$ ,  $\mathbf{y}_B$ ,  $\mathbf{z}_B$ ).
- **Optimisation** : Utilise `drawnow limitrate` et met à jour les propriétés XData/YData des objets graphiques au lieu de les redessiner, assurant une animation fluide.
- **Calcul des limites dynamiques des axes** : Les limites `xmin`, `xmax`, `ymin`, `ymax`, `zmin`, `zmax` sont calculées à partir de l'étendue des positions simulées, avec un rembourrage (`pad`) pour s'assurer que le drone reste visible sans toucher les bords de la fenêtre.
- **Création des objets graphiques** : Tous les éléments visuels du drone (corps central, 6 bras, 6 rotors, 3 flèches d'axes du corps) sont créés une seule fois au début de la fonction. Des "handles" (`body_h`, `arm_h`, `rotor_h`, `qx`, `qy`, `qz`) sont stockés pour ces objets. Une ligne pointillée `traj_h` visualise la trajectoire complète du drone.
- **Boucle d'animation optimisée** : Le cœur de l'animation est une boucle `for` qui parcourt les points de données de la simulation. Pour maintenir la fluidité, elle utilise un pas de temps adaptatif (`step`) pour limiter le nombre d'images affichées (`max_frames`).
- **Mise à jour des objets** : À chaque itération, au lieu de recréer les objets, leurs propriétés (XData, YData, ZData) sont mises à jour à l'aide de la fonction `set`. La matrice de rotation  $\mathbf{R}$  est recalculée pour l'orientation actuelle du drone, puis appliquée aux points du corps et des rotors pour les positionner correctement dans le monde 3D.
- **Axes du corps** : Des flèches (quivers) représentent les axes  $\mathbf{x}_B$  (rouge),  $\mathbf{y}_B$  (vert) et  $\mathbf{z}_B$  (bleu) du drone, une aide précieuse pour visualiser son orientation.
- **Contrôle du dessin** : `drawnow limitrate` assure une mise à jour fluide de l'affichage sans surcharger le processeur. Le test `if ishandle(fh), return; end` permet à l'animation de s'arrêter proprement si la fenêtre de la figure est fermée manuellement.

#### Code 4 : Extrait de Code de animations.m

matlab

```

function animations(positions, data, PARAMS, type)
    % Définition de la géométrie du drone
    angles_rot = (0:5) * 2*pi/6;
    rb = [L*cos(angles_rot); L*sin(angles_rot); zeros(1,6)];

    % Création des objets graphiques (Corps, Bras, Rotors)
    body_h = scatter3(ax, 0, 0, 0, 30, 'filled');
    arm_h = gobjects(6,1);
    for i=1:6
        arm_h(i) = plot3(ax, [0 0], [0 0], [0 0], 'k-', 'LineWidth', 2);
    end

    %% ----- BOUCLE D'ANIMATION -----
    for k = 1:step:N
        pos = positions(k,:).';
        % --- Sélection de la Matrice de Rotation ---
        if strcmpi(type, 'euler')
            phi = data(k,1); theta = data(k,2); psi = data(k,3);
            R = rotz(psi) * roty(theta) * rotx(phi);
        elseif strcmpi(type, 'quat')
            q = data(k,:).';
            q = q / norm(q); % Normalisation essentielle
            R = quatToRotm(q);
        end

        % --- Mise à jour des graphiques ---
        set(body_h, 'XData', pos(1), 'YData', pos(2), 'ZData', pos(3));

        for i=1:6
            % Transformation : Rotation + Translation
            rb_i = rb(:,i);
            r_in = R * rb_i + pos;

            % Mise à jour des bras
            set(arm_h(i), 'XData', [pos(1), r_in(1)], ...
                'YData', [pos(2), r_in(2)], ...
                'ZData', [pos(3), r_in(3)]);
        end

        drawnow limitrate % Force l'affichage
        if ~ishandle(fh), return; end % Arrêt si fenêtre fermée
    end
end

```



## 6 Résultats et Analyse des Scénarios

Cette section présente les résultats obtenus par la simulation pour les différents scénarios de vol définis dans le fichier `generate_test_case.m`. Chaque cas a pour but de valider une composante spécifique de la dynamique du drone. Les résultats visuels sont générés par deux fonctions :

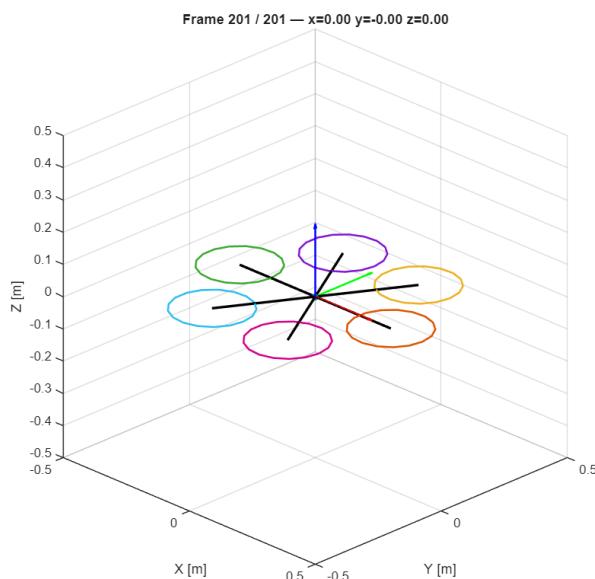
- Graphiques 2D (`plot_all.m`) : Présentent l'évolution temporelle des positions  $(x, y, z)$ , vitesses  $(v_x, v_y, v_z)$ , vitesses angulaires  $(p, q, r)$  et angles d'Euler  $(\phi, \theta, \psi)$ .
- Visualisation 3D (`animations.m`) : Permet de valider qualitativement le comportement spatial du drone et son orientation.

### 6.1 Cas 0 : Vol Stationnaire (Hover)

Le drone est initialisé à l'origine avec une vitesse nulle. La commande moteur est calculée pour compenser exactement le poids du drone :

$$\sum T_i = m \cdot g = 19,62 \text{ N}$$

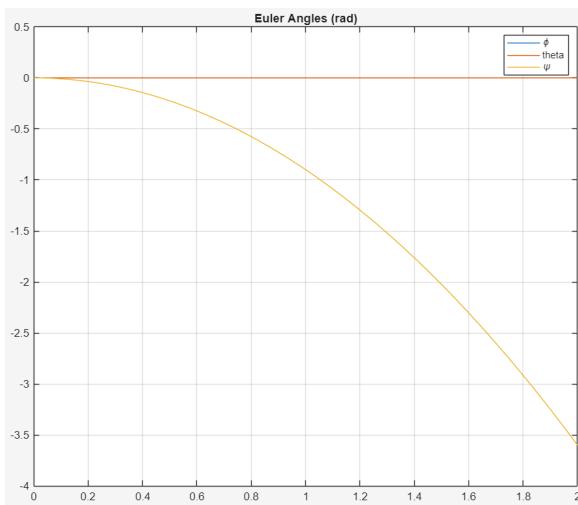
Les couples  $(\tau_\phi, \tau_\theta, \tau_\psi)$  sont nuls et comme illustré dans la Figure 14, le drone maintient parfaitement sa position. La position reste constante à  $(0, 0, 0)$ , les vitesses restent nulles, confirmant l'équilibre statique des forces et aucune rotation n'est observée, validant l'absence de couples parasites.



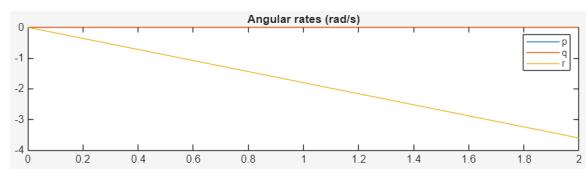
**FIGURE 11** – Résultats du Cas 0 : Les graphiques montrent une stabilité parfaite. La position  $z$  et les angles d'Euler restent à 0, validant le modèle de poussée stationnaire.

## 6.2 Cas 1 : Rotation de Lacet (Yaw)

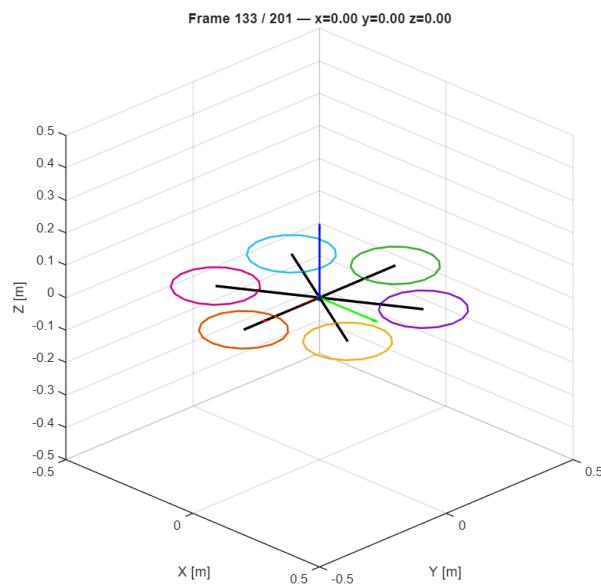
Une perturbation est introduite dans les commandes moteurs : les rotors tournant dans le sens anti-horaire voient leur poussée augmentée, tandis que les autres sont diminués d'autant ( $T_{yaw} = 0.3$ ). La poussée totale reste constante ( $F_z = mg$ ), mais un couple net  $M_z$  est créé. Le lacet ( $\psi$ ) diminue quadratiquement (accélération angulaire constante) comme on peut le voir sur la Figure 12, confirmant la génération du couple  $M_z$  et l'attitude ( $z$ ) reste stable, prouvant que la manœuvre de lacet n'affecte pas la portance globale.



**FIGURE 12** – Courbe parabolique : Évolution de l'angle de lacet  $\psi$ .



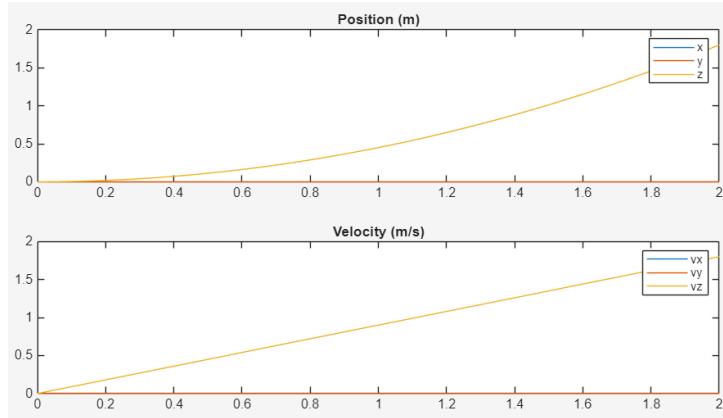
**FIGURE 13** – La vitesse angulaire  $q$  diminue linéairement.



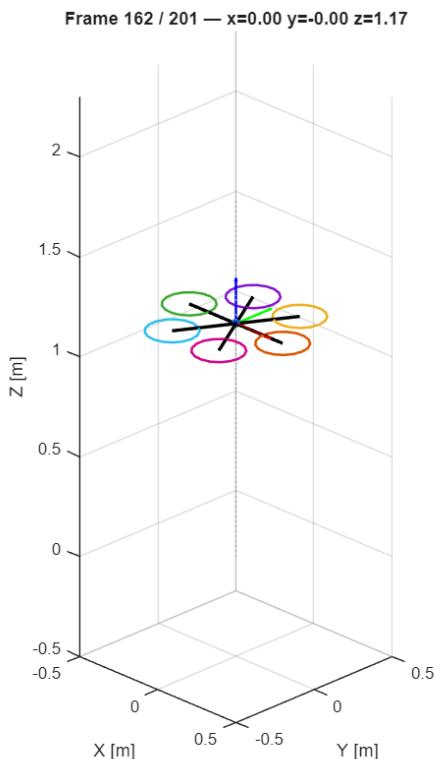
**FIGURE 14** – Résultat du Cas 1 : Drone à plat, tournant.

### 6.3 Cas 2 : Décollage Vertical

La poussée de chaque moteur est augmentée uniformément de 0.3 par rapport au stationnaire. La force totale dépasse le poids :  $F_{tot} > mg$ . La courbe de position décrit une parabole ascendante ( $z(t) \propto t^2$ ), caractéristique d'un mouvement uniformément accéléré selon la loi de Newton  $\sum F_z = ma_z$ . Les angles de roulis ( $\phi$ ) et tangage ( $\theta$ ) restent nuls, indiquant une montée parfaitement verticale sans dérive latérale.



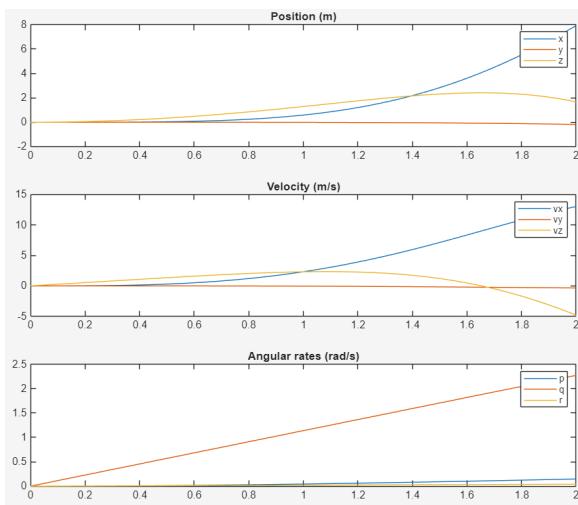
**FIGURE 15** – L'altitude augmente quadratiquement, la vitesse verticale croît linéairement.



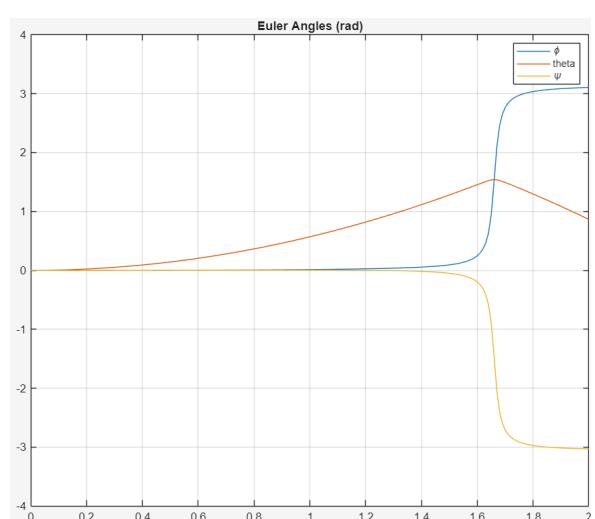
**FIGURE 16** – Cas 2 : Décollage vertical.

## 6.4 Cas 3 : Mouvement de Tangage

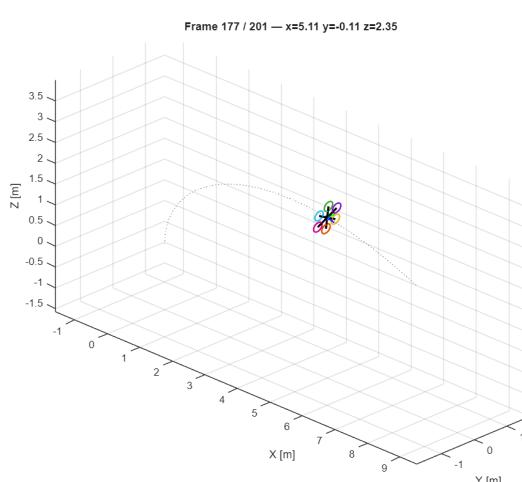
Des déséquilibres de poussée sont appliqués entre l'avant et l'arrière (Pitch). Ce cas valide le couplage rotation-translation fondamental des multirotors. Le couple induit une inclinaison du drone ( $\theta > 0$ ). Le vecteur poussée  $\mathbf{T}$  s'incline et c'est pourquoi une force de poussée supérieure à celle utilisée pour compenser la gravité est appliquée à chaque moteur afin de ne pas « tomber » directement. Une composante horizontale de la poussée apparaît ( $F_x = T \sin \theta$ ), provoquant une accélération linéaire. La Figure 17 montre clairement qu'une augmentation de l'angle  $\theta$  précède immédiatement l'augmentation de la vitesse  $v_x$ .



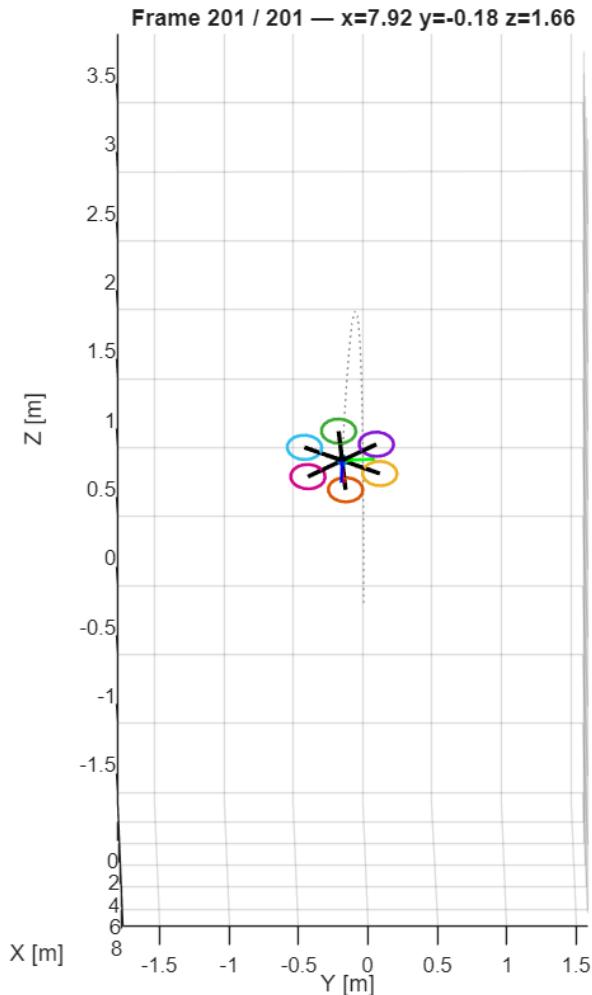
**FIGURE 17** – Graphiques illustrant le mouvement selon les axes  $x$  et  $z$ , et l'augmentation linéaire de la vitesse angulaire  $q$ .



**FIGURE 18** – L'évolution de l'angle  $\theta$  montre la rotation vers l'avant du drone.



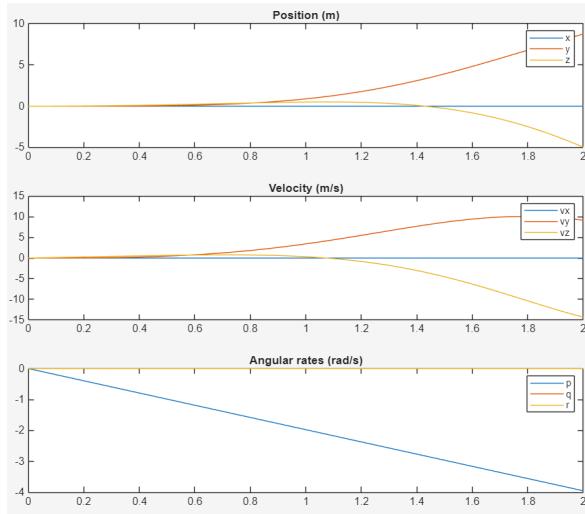
**FIGURE 19** – Translation induite par le tangage. On observe le lien de causalité direct entre l'inclinaison  $\theta$  et le déplacement longitudinal  $x$ .



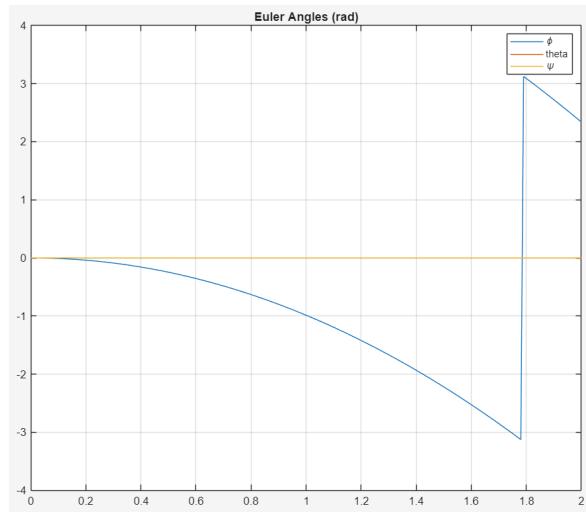
**FIGURE 20** – Effet gyroscopique. Le mouvement devrait être purement dans la direction  $x$ , mais en raison de l'effet gyroscopique, il y a une légère déviation sur l'axe  $y$ .

## 6.5 Cas 4 : Mouvement de Roulis

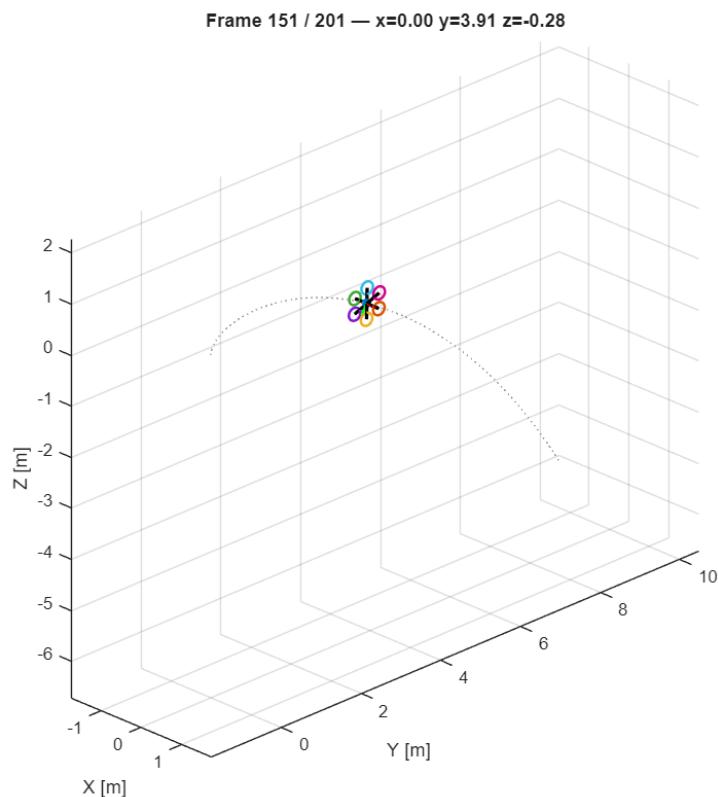
Des déséquilibres de poussée sont appliqués entre la gauche et la droite (Roll). Le couple induit une inclinaison du drone ( $\phi < 0$ ). Le drone s'incline vers la gauche et donc le vecteur poussée  $\mathbf{T}$ , solidaire du corps, s'incline aussi. Une composante horizontale de la poussée apparaît ( $F_y = T \sin \phi$ ), provoquant une accélération linéaire. La Figure 17 montre clairement qu'il y a une augmentation de la vitesse  $v_y$ .



**FIGURE 21** – Graphiques illustrant le mouvement selon les axes  $y$  et  $z$ , et l'augmentation linéaire de la vitesse angulaire  $p$ .



**FIGURE 22** – L'évolution de l'angle  $\phi$  montre la rotation vers la gauche du drone.



**FIGURE 23** – Translation induite par le roulis. On observe le lien de causalité direct entre l'inclinaison  $\phi$  et le déplacement latéral  $y$ .

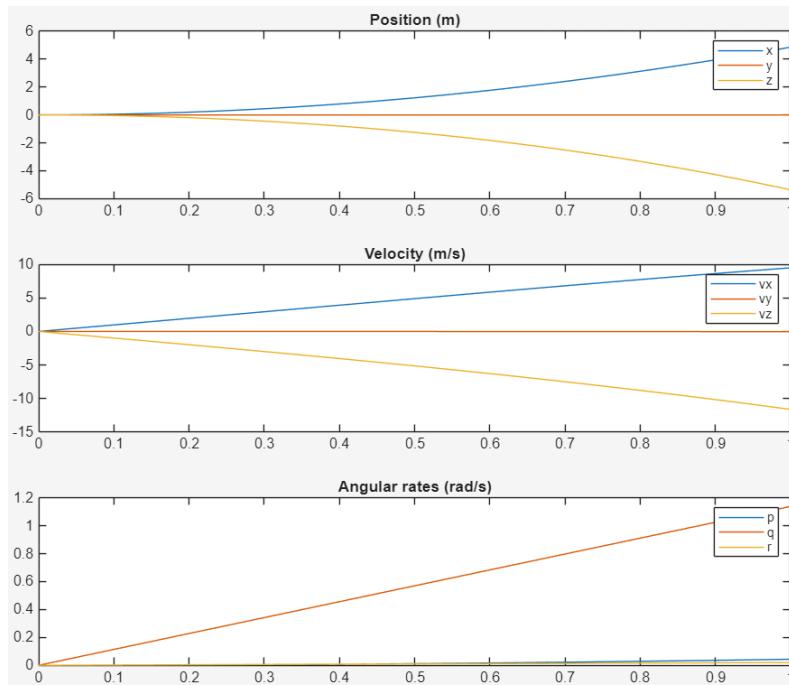
## 6.6 Cas 5 : Étude de la Singularité (Gimbal Lock)

Ce test critique initialise le drone avec une orientation de tangage de  $90^\circ$  ( $\theta = \pi/2$ ). C'est un point de singularité mathématique pour la représentation en angles d'Euler (ZYX), où la matrice cinématique devient indéfinie ( $\tan(\pi/2) \rightarrow \infty$ ). La Figure ?? illustre la différence fondamentale de robustesse :

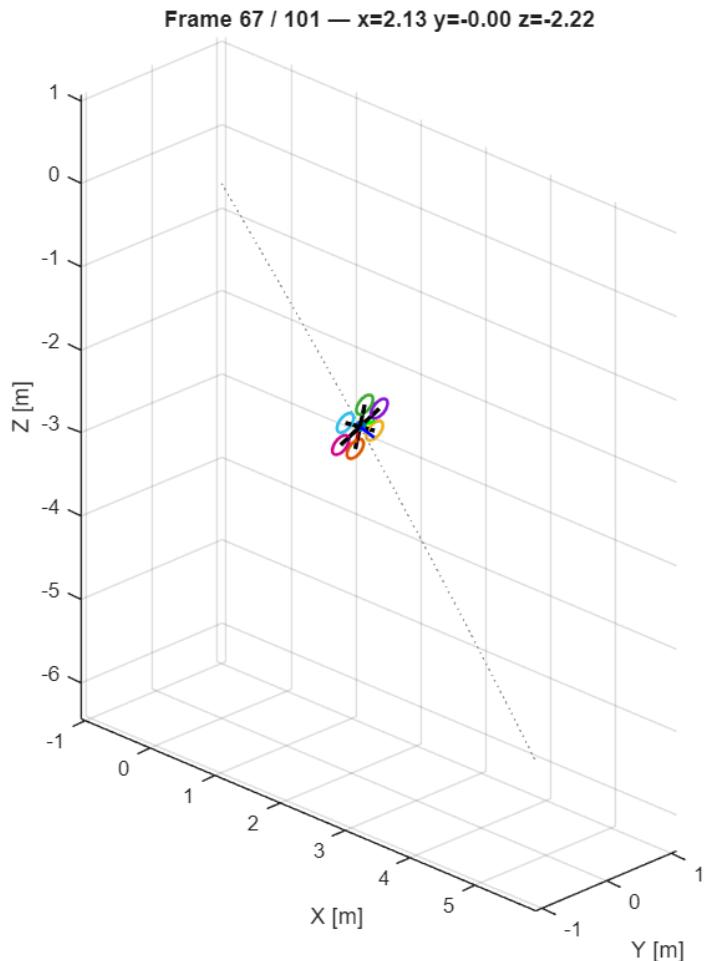
- **Modèle Euler** : La simulation échoue ou produit des oscillations numériques aberrantes dès les premiers instants, car le calcul des dérivées angulaires  $\dot{\phi}$  et  $\dot{\psi}$  implique une division par  $\cos(\theta) \approx 0$ . Figure 24.
- **Modèle Quaternions** : La simulation se déroule de manière fluide. Le drone tombe sous l'effet de la gravité (puisque sa poussée est horizontale), tout en pivotant correctement selon la dynamique gyroscopique, sans aucune erreur numérique. Figure 26.

```
Command Window
In ode45 (line 298)
In test_drone (line 80)
Warning: Gimbal-lock risk: theta ~= +pi/2. Using pseudo-inverse for Euler kinematics.
> In dynamics_model_euler (line 74)
In test_drone>@(t,x)dynamics_model_euler(t,x,u,PARAMS) (line 68)
In ode45 (line 302)
In test_drone (line 80)
Warning: Gimbal-lock risk: theta ~= +pi/2. Using pseudo-inverse for Euler kinematics.
```

**FIGURE 24** – Échec de la simulation Euler à  $\theta = 90^\circ$ .



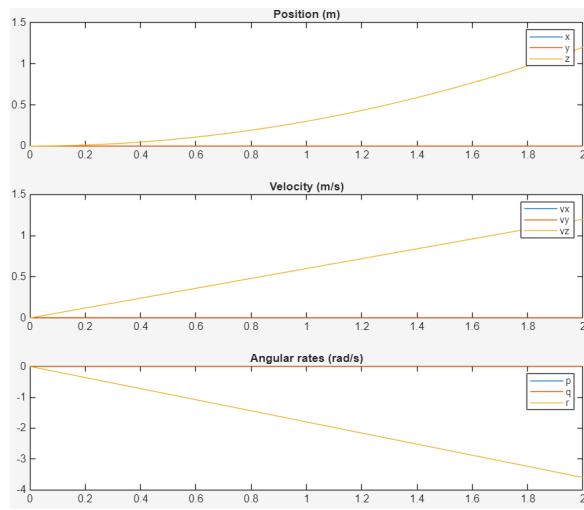
**FIGURE 25** – Graphiques illustrant le mouvement selon les axes  $x$  et  $z$ , et l'augmentation linéaire de la vitesse angulaire  $q$  en utilisant le modèle Quaternion.



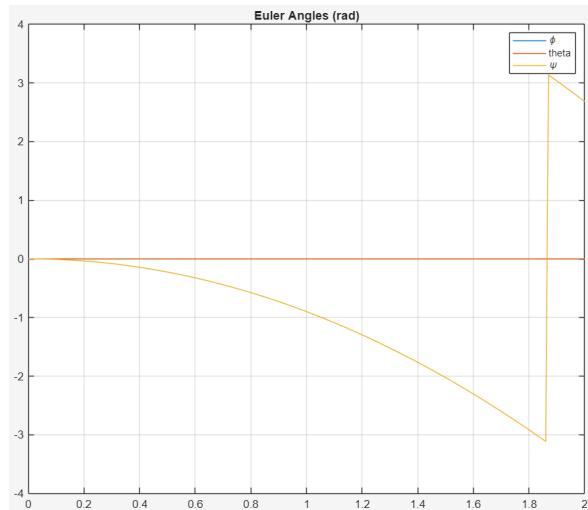
**FIGURE 26** – Simulation réussie avec Quaternions. Mouvement diagonal, vers le bas et vers l'avant

## 6.7 Cas 6 : Manœuvre Combinée (Décollage + Lacet)

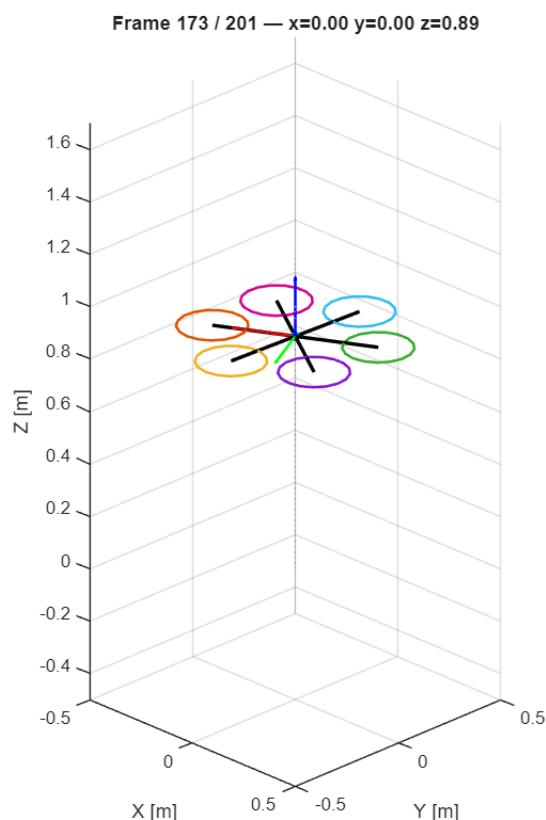
Ce scénario final valide la capacité du modèle à gérer des dynamiques couplées complexes. Comme le montre la trajectoire 3D (Figure 29), le drone s'élève tout en effectuant une rotation. Cela confirme que les équations de mouvement gèrent correctement la superposition des forces verticales et des couples de rotation sans interférence numérique inattendue.



**FIGURE 27** – Graphiques illustrant le mouvement selon l'axe  $z$ , et l'augmentation linéaire de la vitesse angulaire  $r$ .



**FIGURE 28** – L'évolution de l'angle  $\psi$  montre la rotation du drone selon l'axe  $z$ .



**FIGURE 29** – La trace (en pointillés gris) montre le mouvement vertical et il y a également un mouvement de rotation dû au couple de lacet.

## 7 Conclusion

Ce projet s'est attaché à développer, implémenter et simuler le modèle dynamique complet d'un hexacoptère, en mettant l'accent sur la problématique de la représentation de l'attitude. L'objectif principal était de fournir une plateforme de simulation robuste capable de comparer deux approches mathématiques distinctes : la formulation par angles d'Euler et la formulation par quaternions.

L'étude comparative menée à travers les différents scénarios de vol a permis de mettre en évidence les forces et les limites intrinsèques de chaque représentation :

- **Équivalence en Vol Nominal** : Pour la majorité des cas testés (Cas 0 à 4 et Cas 6), couvrant le vol stationnaire et les trajectoires, les deux modèles ont produit des résultats strictement identiques. Cela valide la cohérence physique des équations de mouvement dérivées (Newton-Euler) et confirme que, loin des singularités, la représentation par angles d'Euler est tout aussi précise que celle par quaternions.
- **La Limite des Angles d'Euler** : Le modèle basé sur les angles d'Euler (ZYX) offre l'avantage indéniable d'être intuitif et directement interprétable par un opérateur humain (roulis, tangage, lacet). Cependant, la simulation du **Cas 5** a confirmé sa vulnérabilité majeure : le blocage de cardan (*Gimbal Lock*). Lorsque l'angle de tangage atteint  $\pm 90^\circ$ , la matrice cinématique devient singulière, entraînant l'échec de l'intégration numérique. Ce modèle est donc suffisant pour des vols stabilisés mais inadapté pour des manœuvres acrobatiques.
- **La Robustesse des Quaternions** : Le modèle basé sur les quaternions unitaires s'est révélé inconditionnellement stable. Bien que moins intuitive (nécessitant une conversion pour la visualisation), cette approche élimine totalement les singularités mathématiques. Elle a permis de simuler sans erreur le comportement du drone dans des configurations extrêmes où le modèle d'Euler échouait.

L'architecture logicielle développée sous MATLAB et que l'on peut trouver sur [GitHub repo](#) constitue un atout majeur de ce travail. La structure modulaire du code a permis :

1. Une **ségrégation claire** entre la physique du drone (`dynamics_model`), la logique de test (`generate_test_case`) et la visualisation (`animations`).
2. Une **comparaison "plug-and-play"** des modèles mathématiques grâce à un script principal unifié (`test_drone.m`), permettant de basculer instantanément d'une formulation à l'autre.
3. Une **validation visuelle immédiate** grâce à un moteur d'animation 3D polymorphe, capable d'interpréter indifféremment les sorties des deux modèles.

En conclusion, ce travail a démontré que si la formulation par angles d'Euler reste pertinente pour l'analyse de données et le pilotage basique, l'utilisation des quaternions est impérative pour le noyau de calcul dynamique d'un simulateur de vol ou d'un contrôleur embarqué fiable. La plateforme de simulation ainsi créée offre une base solide pour des travaux futurs, tels que l'implémentation de lois de commande en boucle fermée (PID, Backstepping) ou l'intégration de capteurs bruités.

## 8 Bibliographie

- [1] Kuldeep SINGH. « Modelling and Controls of a Hexacopter ». Major Subject : Mechanical Engineering. Master of Science Thesis. Kingsville, Texas : Texas A&M University - Kingsville, College of Graduate Studies, déc. 2018.
- [2] V. ARTALE, C.L.R. MILAZZO et A. RICCIARDELLO. « Mathematical Modeling of Hexacopter ». In : *Applied Mathematical Sciences* 7.97 (2013), p. 4805-4811. DOI : 10.12988/ams.2013.37385.
- [3] Dang-Khanh LE et Taek-Kun NAM. « A study on the modeling of a hexacopter ». In : *Journal of the Korean Society of Marine Engineering* 39.10 (2015), p. 1023-1030. ISSN : 2234-7925. DOI : 10.5916/jkosme.2015.39.10.1023.