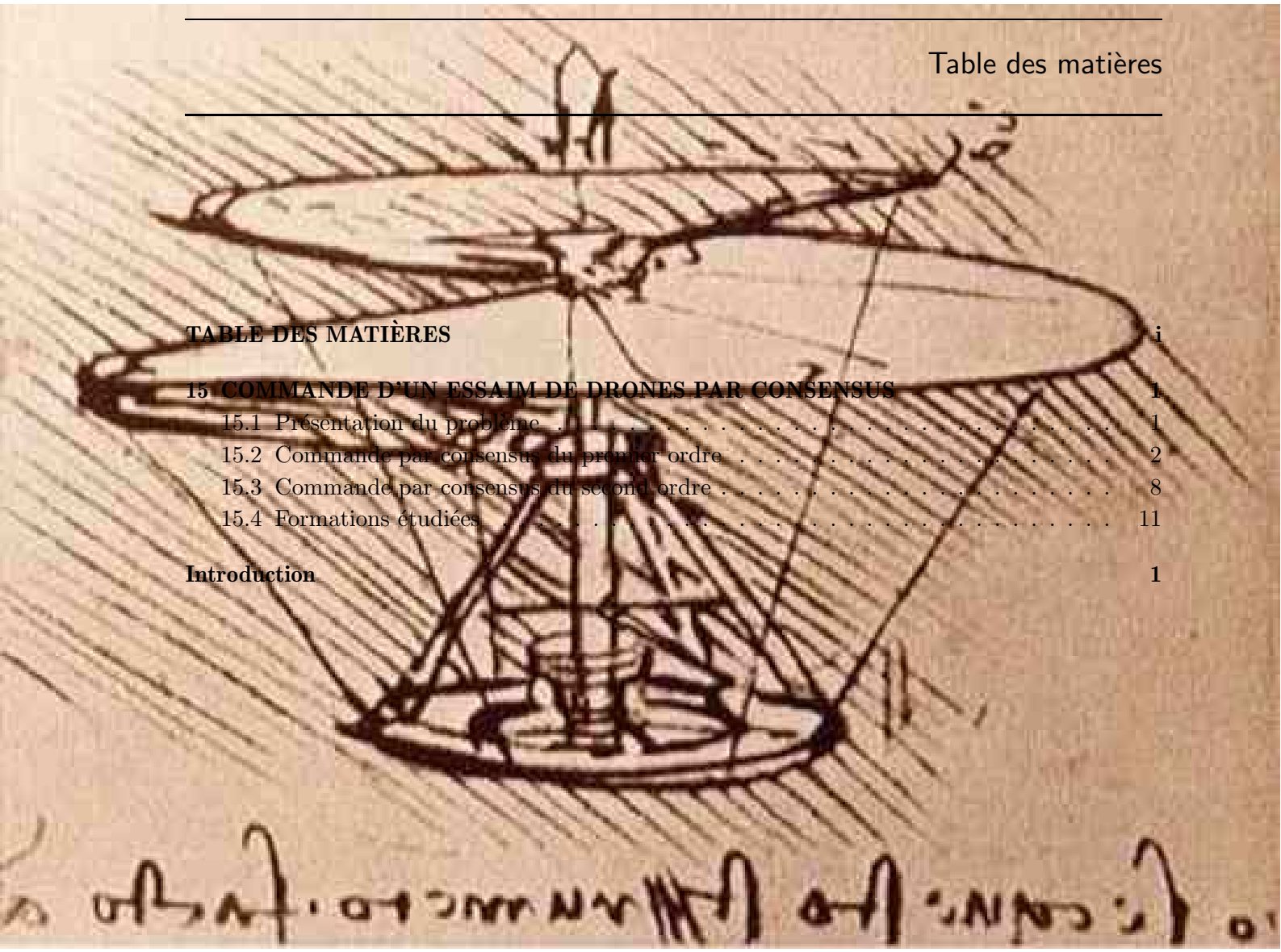

Table des matières

TP 15

COMMANDE D'UN ESSAIM DE DRONES PAR CONSENSUS

15.1 Présentation du problème

Ce TP traite de la *commande par consensus* qui est une stratégie de commande étudiée depuis le début des années 2000. Celle-ci permet à un ensemble d'agents de coopérer entre eux en échangeant des informations sur leur état dans le but d'atteindre un même état final, et cela quelque soit leur état initial. Il s'agit d'une technique de commande non supervisée en ce sens où la génération des commandes n'est pas opérée par un agent externe au groupe mais relève de l'action de chacun des agents. La figure 15.1 illustre une expérience simple mettant en œuvre cette stratégie de commande avec un essaim de deux drones qui doivent rejoindre une même hauteur.

Cette activité comporte deux études, la première porte sur la commande par consensus du premier ordre, la seconde sur la commande par consensus au second ordre. Chacune se décomposant en trois parties. Dans la première partie, vous étudierez les propriétés de la commande d'un essaim de drones par consensus. Dans la seconde partie, vous simulerez le comportement de l'essaim ainsi contrôlé. Ces simulations seront restituées dans un environnement d'animation 3D. Dans la troisième partie, vous mettrez expérimentalement en œuvre un essaim de plusieurs drones dans l'arène de vol pour réaliser un consensus sur la hauteur, puis sur la position dans un plan horizontal.

15.1.1 Liste du matériel

Pour les simulations, vous êtes invités à utiliser les postes informatiques munis de MATLAB 2019 et de la *toolbox Simulink Animation 3D*. Pour les expérimentations, vous utiliserez les matériels et logiciels suivants :

— Système de capture de mouvement,
— Logiciel Qualisys Track Manager installé sur le PC attenant à la volière de drones,
— Plusieurs drones Crazyflie,
— 1 Routeur,

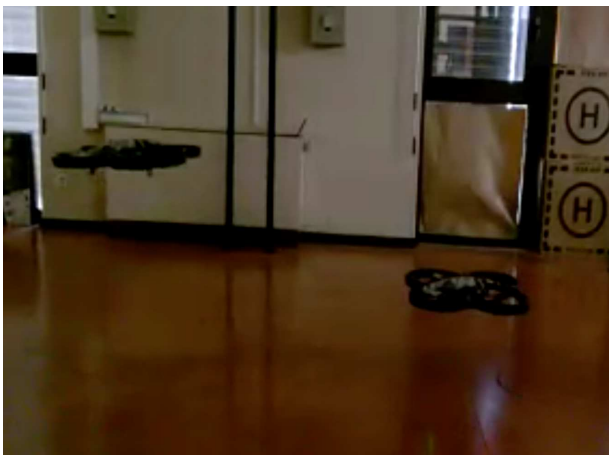

Figure 15.1 – Consensus pour un essaim de 2 ARDrones - EPRE EA2015

— 1 Joystick
— 1 PC Linux avec un IDE Python,

Les fichiers utiles pour la partie simulation sont dans le dossier TP_Consensus_2020 sur le bureau et sous Moodle dans l'espace dédié à ce TME.

Pour les parties expérimentales, vous trouverez le dossier correspondant à ce TME sur le bureau du PC Linux. Il contient le programme Python de gestion des drones que vous aurez à modifier, ainsi que la documentation qui lui est associée.

15.2 Commande par consensus du premier ordre

Dans cette partie du TME, on s'intéresse à une stratégie de commande dite par *consensus du premier ordre*, qui vise à amener les $N = 4$ drones d'un essaim à une même hauteur. Tous les drones de l'essaim sont identiques et l'équation d'état de la hauteur du drone i est celle d'un système du premier ordre :

$$\dot{z}_i = u_i \tag{15.1}$$

Où u_i est la commande du drone i. La hauteur de chaque drone est mesurée, soit z_{i_m} cette mesure de hauteur. Cette dernière est supposée parfaite *c-à.d* $z_{i_m} = z_i$.

15.2.1 Cas d'un graphe des liaisons non-orienté

Pour cette stratégie de commande, chaque agent i de l'essaim élabore sa commande u_i d'après les informations de hauteur que lui transmettent les drones avec lesquels il communique. Le réseau de communication de ces données est décrit par un graphe tel que celui représenté sur la figure 15.2 et la loi de commande afférente pour le drone i s'écrit :

$$u_i = \sum_{j=1}^{N} k_{ij}(z_j - z_i) \tag{15.2}$$

où $k_{ij} = 1$ si le robot j communique sa position au robot i
$k_{ij} = 0$ sinon

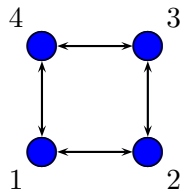

Figure 15.2 – Graphe non orienté décrivant les liaisons entre les drones

Question *0* : *Pour le graphe des liaisons de données de la figure 15.2, écrivez les équations d'état de l'essaim ainsi contrôlé et montrez que ces équations ont la même structure que celle d'un système commandé par retour d'état pour lequel il n'y a pas d'entrée de consigne. Vous préciserez en particulier l'expression de la matrice d'état corrigée qu'on notera* $-\mathbf{L}$. *La matrice* $\mathbf{L}$ *est aussi appelée en théorie des graphes matrice laplacienne.*

Question *1* : *Au vu de la structure de la matrice laplacienne, montrez que zéro est une valeur propre de cette matrice et proposez une expression possible pour le vecteur propre associé.*

Notez que la matrice laplacienne est réelle et symétrique, que par conséquent elle est toujours diagonalisable sur $\mathbb{R}$. Qu'en outre, la matrice de passage constituée des vecteurs propres associés aux valeurs propres de $\mathbf{L}$ est orthogonale.

Question *2* : *Pour le graphe des liaisons de données inter-drones qui vous a été communiqué, saisissez sous* MATLAB *les matrices d'état et de commande de la représentation d'état corrigée. Vous vérifierez que zéro est bien une des valeurs propres de la matrice d'état corrigée et que la matrice de passage dans la base modale est orthogonale.*

Question *3* : *En utilisant la matrice de passage dans la base modale et les valeurs propres de la matrice d'état corrigée calculées à la question précédente, établissez les solutions des équations d'état du drone corrigé dans la base modale en fonction des hauteurs initiales* $\mathbf{Z}^t(0) = \begin{pmatrix} z_1(0) & \dots & z_N(0) \end{pmatrix}$.

Question *4* :

1. *Montrez qu'il y a consensus c-à.d que les hauteurs convergent toutes vers une même valeur non définie a priori et fonction des seules conditions initiales.*
2. *Que représente dans ces conditions le noyau de* $\mathbf{L}$ *?*
3. *Précisez en justifiant votre réponse quelle sera la nature des modes et l'ordre de grandeur du temps de réponse pour que l'essaim réalise le consensus.*

Ouvrez le fichier *Consensus_4_drones.slx*. Ce fichier contient les modèles de quatre drones dont les entrées de commande sont les consignes de gîte ϕ_{ref}, de tangage θ_{ref}, de vitesse de lacet r_{ref} et de vitesse ascensionnelle v_{zref}. Vous pouvez accéder aux conditions initiales de chaque

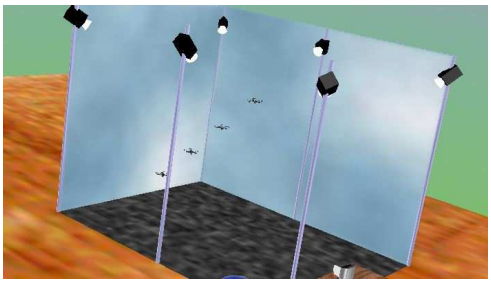

Figure 15.3 – Animation 3D du consensus sous SIMULINK 3D

drone en double-cliquant sur le bloc représentant le drone considéré.
Dans ce qui suit, la consigne de vitesse ascensionnelle vz_{refi} du drone i joue le rôle de la commande u_i.

Question 5 : *Réalisez la commande par consensus et testez-là. Visualisez et enregistrez les chronogrammes des hauteurs et des commandes des drones. Relevez les caractéristiques des régimes transitoires et comparez-les à ceux établis dans la partie théorique.*

Question 6 : *Sur des essais successifs, appliquez comme conditions initiales les vecteur propres associés aux valeurs propres non nulles. Que constatez-vous ? Ce résultat était-il prévisible ?*

Question 7 : *Donnez une interprétation qualitative des résultats obtenus en étudiant la structure du graphe et la manière dont l'information est partagée pour réaliser le consensus.*

15.2.2 Cas d'un graphe des liaisons orienté

Question 8 : *Pour le graphe des liaisons de la figure 15.4, reprenez l'étude précédente sans refaire les calculs théoriques. Vous observerez que la matrice* $\mathbf{L}$ *n'est pas symétrique et n'a donc plus les propriétés de celle obtenue pour un graphe non-orienté. Ceci a évidemment des conséquences que vous mettrez en évidence sur les simulations que vous ne manquerez pas de réaliser sur le même modèle que précédemment. Cette fois cependant, la variable sur laquelle le consensus opère pour le drone i est le cap* ψ_i *et la variable de commande est la vitesse de lacet de consigne* r_{refi}.

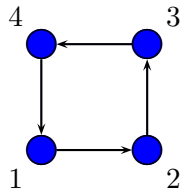

Figure 15.4 – Graphe non orienté décrivant les liaisons entre les drones

15.2.3 EXPÉRIMENTATION

L'objet de cette expérimentation est de mettre en œuvre une commande par consensus pour un essaim constitué de plusieurs drones [1]

Vous devrez d'abord vous familiariser avec l'environnement de programmation Python et les programmes déjà existants qui vous permettront d'implémenter vos propres fonctions par la suite.

Puis, vous coderez sous Python une stratégie de commande par consensus du premier ordre dans le but de réaliser un asservissement sur la vitesse ascensionnelle des drones. Enfin, vous mettrez en œuvre cette stratégie expérimentalement.

Acquisition de la position des drones

Démarrez le PC relié aux Caméras ainsi que le PC Linux, et vérifiez qu'il sont tous deux reliés en réseau via un routeur fonctionnel. Branchez ensuite les caméras Infra-Rouge, et lancez le programme Qualisys Track Manager (QTM). Double-cliquez sur le projet "TME_consensus" (tout le paramétrage nécessaire à ce TME a été réalisé en amont).

Une fois le projet ouvert, cliquez sur "File", puis "New" et une liste de fenêtres apparaît. Ces fenêtres correspondent au champ de vision de chacune des caméras infra-rouge de l'arène de vol. Sélectionnez alors la vue 3D avec l'icône correspondante dans la barre d'outils située à gauche de ces fenêtres.

Une représentation en 3D de l'arène de vol est désormais affichée : elle est le résultat de la fusion des données 2D de toutes les caméras par triangulation. Désormais, le logiciel QTM est capable d'observer les marqueurs infra-rouge présents dans l'arène, de calculer leur position par rapport à un repère terrestre au centre de l'arène, et de partager cette position avec d'autres programmes et d'autres appareils connectés au routeur, le tout en temps réel.

Préparation du programme Python

Sur le PC Linux, ouvrez le dossier "TME_consensus" présent sur le bureau. Ce dossier contient tous les fichiers utiles à cette partie expérimentale. Pour la suite, on appellera ce dossier le répertoire de travail.

Démarrez le logiciel PyCharm et ouvrez le projet qui correspond au répertoire de travail. Dans l'arborescence à gauche, repérez les fichiers du répertoire de travail et ouvrez les scripts Python suivants : "main.py" et "consensus_control_law.py". Il s'agit des deux seuls fichiers Python que vous aurez à consulter ou modifier lors de ce TME.

Ensuite, connectez les antennes radio qui permettront d'assurer la connexion aux drones. Ces antennes se présentent sous la forme de Dongles USB regroupés sur un hub à brancher sur l'un des ports USB du PC Linux. Connectez également une manette ou un joystick qui vous permettra de contrôler les drones pendant le vol.

1. Dans la limite des capacités temps réels du PC et du volume de vol offert par l'arène.

Documentation

Une documentation détaillée des programmes et des drones est disponible sous forme de Notebook interactif à ouvrir avec JupyterLab.

Démarrez JupyterLab à l'aide du raccourci sur le bureau. Cela provoque l'ouverture d'une fenêtre Firefox qui sert d'interface graphique. Sur cette fenêtre, accédez à l'arborescence sur la gauche et parcourez les dossiers de l'ordinateur pour vous placer dans le répertoire de travail.

Double-cliquez alors sur le fichier "Documentation.ipynb" pour l'afficher. Ce fichier comporte de plus amples informations sur le fonctionnement du programme Python que vous serez amené à utiliser, mais aussi sur les drones Crazyflie.

Note : servez-vous de la table des matières pour trier les informations, la documentation se veut très généraliste et certaines parties ne vous concerneront pas particulièrement.

Validation expérimentale

Programmation de la loi de commande Sur PyCharm, dans le script Python "consensus_control_law.py", identifiez la fonction "z_consensus_control_law()". Cette fonction est appelée en boucle par le programme principal pour chacun des drones en vol, et retourne une commande de vitesse ascensionnelle.

A l'aide des indications présentes sur le fichier Python, implémentez votre propre code dans la partie balisée "A remplir" de la fonction afin d'assurer le consensus des drones selon z.

Une fois que votre loi de commande est implémentée, munissez-vous de plusieurs drones Crazyflie avec leurs batteries chargées.

Commencez à vous renseigner sur les caractéristiques principales du modèle de drone utilisé (cf. Documentation) : son fonctionnement, ses particularités.

Important : lien entre QTM et les drones Les lois de commande utilisées pour contrôler les drones font appel à une mesure de leur position. Ainsi, lorsque plusieurs drones sont en vol en même temps, il est nécessaire de correctement faire le lien entre les marqueurs infra-rouge repérés par QTM et les drones réels (et donc d'envoyer la bonne mesure de position au bon drone).

Pour réaliser ce lien, le programme Python attend que l'utilisateur renseigne la position initiale de chaque drone avant le décollage, afin qu'il puisse regrouper chaque marqueur vu par QTM avec la liaison radio du drone qui lui correspond.

Faites donc preuve d'une grande rigueur lors de la configuration du programme : une erreur dans le paramétrage des drones entraîne à coup sûr une perte de contrôle suivie d'un crash violent dès les premières secondes de vol.

Lancement du programme Python Sur Pycharm, placez-vous dans le script "main.py" et exécutez le programme principal en cliquant sur la flèche verte associée à la ligne "if __name__ == '__main__'" (tout en bas du script), puis sur "Run main".

Une interface graphique apparaît : il s'agit de la fenêtre de réglage des paramètres de vol.

Dans la documentation, accédez à la partie qui concerne la configuration et l'exécution du programme principal pour avoir accès à la procédure de réglage et d'installation des drones dans l'arène.

Paramétrez et installez tous les drones que vous souhaitez faire voler.

Connexion aux drones Une fois que tous les drones sont paramétrés, correctement installés à leur positions initiales respectives, et correctement orientés, vous pouvez les mettre sous tension.

Après avoir vérifié que personne ne se trouve dans l'arène de vol, appuyer sur le bouton "Valider" de l'interface graphique. Ce bouton ferme la fenêtre de paramétrage et lance la procédure de connexion aux drones.

Suivez la documentation pour vous assurer que la connexion aux drones se passe normalement, et attendez la fin de l'étape de connexion.

Si la procédure s'est terminée sans problème particulier, alors les drones sont en attente, prêts à décoller.

Affectation des touches Les affectations des touches du Joystick ou de la manette sont présentées sous forme de tableau dans la documentation.

Décollage Avant de donner l'ordre aux drones de décoller, repérez la touche correspondant à l'arrêt d'urgence : le décollage est le moment le plus critique du vol, donc soyez attentifs aux comportements des drones, et restez prêts à appuyer sur l'arrêt d'urgence à la moindre anomalie.

Faites alors décoller les drones.

Test de la commande de consensus sur z Une fois que le décollage est terminé, les drones se mettent en phase d'attente en vol stationnaire. Vous pouvez alors activer le mode de consensus en z et vérifier la validité de votre loi de commande. Le bouton de retour en position initiale vous permet de rétablir des écarts de hauteur entre les drones, et donc de répéter plusieurs fois la convergence de hauteur en mode consensus.

Lorsque le mode de consensus est activé, il est possible de contrôler manuellement l'un des drones selon le paramétrage que vous avez effectué. Lorsque le mode manuel est activé, le drone concerné n'exécute plus votre loi de commande de consensus sur z mais il obéit à vos commandes de déplacement à la manette ou au joystick, tandis que les autres drones gardent leur comportement de consensus.

Question *9* : *Observez et comparez le temps de convergence des drones vers une hauteur commune, d'abord lorsque tous les drones sont en mode consensus, puis lorsqu'un drone est piloté manuellement et que tous les autres sont en mode consensus. Pourquoi le temps de convergence est-il différent entre ces deux cas ?*

Enregistrement et visualisation des données de vol Une fois que le vol est terminé et que les drones sont déconnectés, repérez le fichier "logs.csv" dans le répertoire de travail et renommez-le.

Ouvrez alors le Notebook "Flight_logs_dashboard.ipynb" sur JupyterLab. Ce Notebook interactif contient des cellules de code Python qui restituent les données contenues dans le fichier de logs sous forme de graphiques.

Dans la toute dernière cellule de code, entrez le nom de votre fichier de logs, puis cliquez sur l'onglet "Run", et "Run all cells".

Vous pouvez ensuite enregistrer les graphiques obtenus en tant qu'images si besoin.

15.3 Commande par consensus du second ordre

Le but dans cette partie est d'opérer à un consensus des drones et de leur faire tenir une formation géométrique au choix parmi celles décrites sur la figure 15.5. Les échanges d'informations entre les drones se font au sens du graphe des liaisons représenté sur la figure 15.2. Dans le cas présent, les drones communiquent leur position mais aussi leur vitesse.
Les équations du drone $i \in \{1 \dots N\}$ sont :

$$\ddot{x}_i = v_i \tag{15.3}$$

$$\ddot{y}_i = w_i \tag{15.4}$$

Comme ce modèle est décrit par des équations différentielles d'ordre deux, le consensus est dit du second ordre. Pour un drone i, les coordonnées de sa position (x_i, y_i) et de sa vitesse (v_{xi}, v_{yi}) sont mesurées et ces mesures sont supposées parfaites.

Question *10* : *Etablissez les équations d'état et d'observation pour un essaim de $N = 4$ drones. Vous adopterez comme vecteur d'état $\mathbf{X}^t = (x_1 \dots x_n \ , \ v_{x1} \dots v_{xn})$ que l'on notera de façon condensée $\begin{pmatrix} \mathbf{x}^t & \mathbf{v_x}^t \end{pmatrix}$. Même démarche sur l'axe $\vec{y}$ avec le vecteur d'état $\mathbf{Y}^t$.*

15.3.1 Lois de commande

Pour le drone i, la loi de commande par consensus dite du second ordre a pour expression :

$$\ddot{x}_i = \sum_{j=1}^{n} k_{ij}(x_j - x_i) - \xi k_{ij}(\dot{x}_i - \dot{x}_j) \tag{15.5}$$

$$\ddot{y}_i = \sum_{j=1}^{n} k_{ij}(y_j - y_i) - \xi k_{ij}(\dot{y}_i - \dot{y}_j) \tag{15.6}$$

où ξ est un paramètre homogène à un facteur d'amortissement.

Question *11* : *Pour le graphe des liaisons de la figure 15.2 et pour le vecteur d'état $\begin{pmatrix} \mathbf{x}^t & \mathbf{v_x^t} \end{pmatrix}$, exprimez l'équation d'état des $N = 4$ drones ainsi commandés. Il est recommandé d'adopter une notation synthétique en utilisant la matrice laplacienne $\mathbf{L}$ définie dans la première partie.*

15.3.2 Etude et simulation du consensus

On montre que les drones, initialement dans des positions quelconques, font consensus, c-à.d qu'ils se retrouvent tous en un même point de l'espace.

Le calcul des valeurs propres de la matrice d'état montre l'existence de valeurs propres ayant un ordre de multiplicité égal à deux et pour lesquelles les vecteurs propres associés sont colinéaires. Dans ces conditions, la matrice de passage n'est pas inversible. On peut néanmoins faire apparaître les modes propres en opérant à une réduction Jordan de la matrice d'état. On exprime alors les équations d'état dans la base de Jordan en utilisant les mêmes formules de changement de base que celles utilisées pour passer dans la base modale. On note $\boldsymbol{\zeta}$ le vecteur d'état dans la base de Jordan.

Question *12* : *Avec MATLAB et la fonction* ***jordan****, calculez pour votre essaim la forme de jordan de matrice d'état et la matrice de passage afférente.*

Question *13* : *Exprimez les équations d'état dans la base de Jordan, intégrez ces équations. Donnez les expressions des $\zeta_i(t)$ en régime permanent et montrez que les drones font consensus.*

Question *14* : *Pour des conditions initiales $\mathbf{x}(0)$ de votre choix et $\mathbf{v}(0) = \mathbf{0}$, calculez numériquement, au moyen de la matrice de passage dans la base de Jordan et des coordonnées de $\boldsymbol{\zeta}$, le vecteur d'état $\mathbf{X}$ en régime permanent et concluez sur l'existence d'un consensus.*

Question *15* : *Sur le fichier Consensus_4_drones.slx de l'essaim réalisé sous SIMULINK, implémentez la commande par consensus au second ordre. La commande $-\theta_{refi}$ joue ici le rôle de v_i pour le contrôle de x_i et ϕ_{refi} joue ici le rôle de v_i pour le contrôle de y_i. Faites attention à prendre le signe $-$ qui apparaît dans le contrôle de l'assiette ; vous devrez multiplier par -1 la commande calculée par consensus avant de l'appliquer à θ_{refi}.*

15.3.3 Evolution en formation

On ajoute des entrées de consigne en vue de contrôler les écarts entre les drones afin de réaliser la formation désirée pour le graphe des liaisons donné.

$$\ddot{x}_i = \sum_{j=1}^{n} k_{ij}(x_j - x_i + r_{ij}) - \xi k_{ij}(\dot{x}_i - \dot{x}_j) \tag{15.7}$$

$$\ddot{y}_i = \sum_{j=1}^{n} k_{ij}(y_j - y_i + \rho_{ij}) - \xi k_{ij}(\dot{y}_i - \dot{y}_j) \tag{15.8}$$

où r_{ij} (resp. ρ_{ij}) est l'écart souhaité entre les drones i et j selon $\vec{x}$ (resp. $\vec{y}$). On peut reformuler ces équations :

$$\ddot{x}_i = r_i + \sum_{j=1}^{n} k_{ij}(x_j - x_i) - \xi k_{ij}(\dot{x}_i - \dot{x}_j) \tag{15.9}$$

$$\ddot{y}_i = \rho_i + \sum_{j=1}^{n} k_{ij}(y_j - y_i) - \xi k_{ij}(\dot{y}_i - \dot{y}_j) \tag{15.10}$$

où $r_i = \sum_{j=1}^{n} r_{ij}$ et $\rho_i = \sum_{j=1}^{n} \rho_{ij}$

Question *16* : *Modifiez le modèle sous SIMULINK pour réaliser cette mise en formation. Vérifier qu'en régime permanent les écarts entre les drones sont ceux désirés.*

15.3.4 EXPÉRIMENTATION

On cherche à mettre en œuvre expérimentalement une opération de consensus du second ordre sur les axes $\vec{x}$ et $\vec{y}$ pour un essaim de drones.

Servez-vous des éléments de la première partie expérimentale pour la mise en place des systèmes.

Sur le PC Linux, sur PyCharm et dans le script "consensus_control_law.py", repérez la fonction "xy_consensus_control_law()" et implémentez votre propre loi de commande dans cette fonction.

Commencez par valider votre loi de commande avec deux drones et affinez par expérimentation la valeur de ξ.

Comme pour la première partie expérimentale, une fois que les drones sont en mode consensus, il est possible de prendre le contrôle d'un drone manuellement.

Une fois que votre loi de commande est au point, augmentez le nombre de drones (jusqu'à 4, au-delà de 4 drones, demandez l'accord du responsable de séance), et choisissez un motif de formation pour ces drones. Des exemples de formations sont donnés dans la partie 15.4, mais libre à vous de créer la votre.

Attention En mode consensus, les drones n'assurent pas les fonctions d'anti-abordage. Faites donc attention à ce que les positions initiales des drones soient compatibles avec la formation que vous demandez (pas de croisement). Par mesure de sécurité, étagez les drones à des hauteurs différentes pour éviter les abordages.

Ensuite, calculez pour chaque drone les valeurs d'offset sur $\vec{x}$ et $\vec{y}$ correspondant aux variables r et ρ sur l'équation 15.9 et son homologue sur $\vec{y}$.

Puis, vérifiez expérimentalement votre implémentation de formation.

15.3.5 Pour aller plus loin

Faites voler une formation de seulement deux drones, puis prenez manuellement le contrôle de l'un d'eux alors que l'autre est en mode consensus sur x et y.

Les drones en mode consensus sont asservis en lacet pour recopier le cap du drone piloté manuellement. Faites alors varier légèrement le cap du drone que vous pilotez manuellement (voir assignation des touches dans la documentation).

Question 17 : *Qu'observez-vous sur le comportement du drone en mode consensus ? Expliquez ce phénomène, proposez une solution pour le corriger et modifiez la fonction "xy_consensus_control_law() pour implémenter votre solution. Enfin, validez expérimentalement votre travail.*

15.4 Formations étudiées

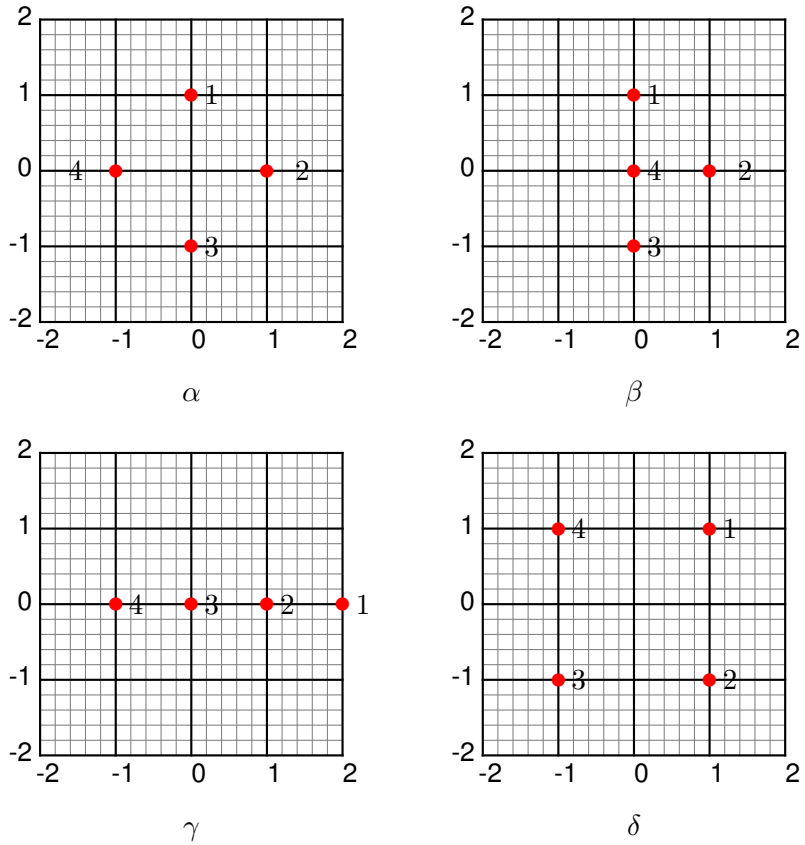

Figure 15.5 – Exemple de formation