

République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université Djilali Bounaama - Khemis Miliana



Faculté des Sciences et de la Technologie
Département de la Technologie

Mémoire du Projet de Fin d'Etudes
Pour l'obtention de diplôme

Master

En

« Génie électrique »

Option :

« Automatique des systèmes de production »

Titre :

**Optimisation métaheuristique d'un système
d'inférence floue pour la commande d'un drone de
type Quadrotor**

Réalisé par :

ZATOUB Mohamed Siddiq

Encadré par :

Dr. REZOUG AMAR

Dr. HIMOUR Yacine

Année Universitaire 2015/2016

REMERCIEMENTS

Avant tout nous remercions Dieu Le tout puissant de nous avoir donné le courage, la volonté, la patience, et la santé durant toutes ces années et que grâce à lui ce travail a pu être réalisé.

En premier lieu, nous tenons à remercier vivement notre encadreur **Dr. REZOUG AMAR** et Co-encadreur **Dr. HIMOUR Yacine** pour les orientations et les conseils judicieux qu'il n'a cessé de nous prodiguer. Ces derniers nous ont été d'un grand soutien tout au long de l'élaboration de ce travail.

Egalement un grand merci à nos enseignants qui tout au long de notre scolarité n'ont ménagé ni leur temps ni leurs efforts contribuant de la manière la plus efficace à notre formation.

Nous n'oublierons pas tous ceux qui nous ont aidés de près ou de loin durant notre étude.

Dédicace

A ma mère.

A mon père.

A mes frères et sœurs.

A tous la famille ZATOUT

A mes amis et mes collègues de la promotion 2015/2016

« Automatique des systèmes des production ».

Sommaire

1. Introduction générale	1
---------------------------------	----------

Chapitre I : Etat de l'art sur les UAVs et Modélisation du quadrotor

I.1 Introduction	3
I.2 Généralités sur les UAV	3
I.2.1 Définition d'UAV	3
I.2.2 Bref Historique	3
I.2.3 Classification des drones	5
I.2.4 Les applications des drones	6
I.3 Modélisation dynamique du Quadrotor	7
I.3.1 Les possibilités de vol du Quadrotor	7
I.3.2 Modèle dynamique du quadrotor	8
I.3.3 Matrice de rotation	9
I.3.4 Effets physiques agissants sur le quadrotor	10
I.3.4.1 Les force	11
I.3.4.2 les Moment	11
I.3.4.3 Effet gyroscopique	12
I.3.5 Développement du modèle mathématique selon Newton-Euler	12
I.3.5.1 Equations de mouvement de translation	14
I.3.5.2 Equations de mouvement de rotation	14
I.3.6 La représentation d'état du système	16
I.3.7 Les valeurs des paramètres des quadrotor	17
I.4 Conclusion	18

Chapitre II : Logique Flou & Optimisation métaheuristique

II.1 Introduction	19
II.2 La logique floue	19
II.2.1 Logique floue vs logique classique	19
II.2.2 Les bases théoriques de la logique floue	20
II.2.2.1 Les sous-ensembles flous	20
II.2.2.2 Fonction d'appartenance	20

II.2.2.3	Notions caractéristiques	20
II.2.2.4	Opérations sur les sous-ensembles flous	21
II.2.3	Systeme d'inférence floue (SIF)	22
II.2.3.1	Fuzzification	22
II.2.3.2	Moteur d'inférence	22
II.2.3.2.1	Méthode de MAMDANI	22
II.2.3.2.2	Méthode de Sugeno	23
II.2.3.3	Déffuzification	23
II.3	Optimisation métaheuristique	24
II.3.1	Principales caractéristiques	24
II.3.2	L'algorithme d'Optimisation par Essaim Particulaire (OEP)	24
II.3.2.1	Principe de fonctionnement	24
II.3.2.2	Les éléments de l'OEP	25
II.3.2.3	Principe fondamental	25
II.3.2.4	Pseudo code d'algorithme	26
II.3.3	L'Algorithme BAT	27
II.3.3.1	Echolocation	27
II.3.3.2	Principe fondamental de l'algorithme	27
II.3.3.3	Pseudo code de l'algorithme	28
II.3.4	La Recherche Coucou (CS)	29
II.3.4.1	Le comportement et la reproduction des coucous	29
II.3.4.2	Le principe de la recherche coucou	29
II.3.4.3	Le vol de Lévy (Lévy flight)	30
II.3.4.4	Pseudo code de la recherche coucou (CS)	31
II.4	Conclusion	31

Chapitre III : Optimisation métaheuristique d'un système d'inférence floue pour la commande d'un drone de type quadrotor

III.1	Introduction	32
III.2	Commande d'un quadrotor	32
	III.2.1 Commande d'attitude	32
	III.2.2 Commande d'altitude	33
	III.2.4 Contrôleurs flous proposé pour commande d'un quadrotor	33
III.3	optimisation méthaheuristique des contrôleurs flous développés	36
	III.3.1 Critères à optimisés	38
	III.3.1.1 Intégral de l'erreur absolue	38
	III.3.1.2 Intégral de l'erreur quadratique	38
	III.3.1.3 Intégrale du temps multiplié par la valeur absolue de l'erreur	39
	III.3.2 optimisation méthaheuristicques des controleur flou	39
	III.3.2.1 PSO appliqué pour l'optimisation de la distribution des FAs des sorties	40
	III.3.2.2 BAT appliqué pour l'optimisation de la distribution des FAs des sorties	41
	III.3.3.3 CS appliqué pour l'optimisation de la distribution des FAs des sorties	43
III.4	Conclusion	45
	Chapitre 4 : Résultats de simulation et Interprétations	
IV.1	Introduction	47
	IV.2.1 Résultats de simulation de controle Flou de l'attitude sans optimisation	47
	IV.2.2 Résultats optimisés en attitude	49
	IV.2.3 Résultat de simulation en altitude	56
	IV.3.1 Application des commandes floues optimisées pour le contrôle de l'attitude d'un quadrotor :	58
	IV.3.2 Les résultats de commande floue optimale de l'altitude et de l'attitude à la fois	64
	IV.4 Conclusion	67
	Conclusion Général	68

List des Tableaux

Tableau I.1 : Projets de conception des Quadrotor	4
Tableau I.2 : Paramètres du quadrotor utilisé.	17
Tableau III.1 : La base des règles du Contrôleur	35
Tableau III.2 : Paramètres de PSO	40
Tableau III.3 : Paramètres de BAT	42
Tableau III.4 : Paramètres de CS	44
Tableau IV.1 : Valeurs des critères par flou conventionell	48
Tableau IV.2 : Résultats obtenus par minimisation le critère IAE.	50
Tableau IV.3 : Résultats obtenus par minimisation le critère ISE	52
Tableau IV.4 : Résultats obtenus par minimisation le critère ITAE	54
Tableau IV.5 : Moyenne de la somme de temps de traitement	55
Tableau IV.6 : Résultats obtenus par minimisation le critères en altitude	57
Tableau IV.7 : Temps de simulation en altitude	58
Tableau IV.8 : Comparaison entre les algorithmes	66

List des figures

Figure I.1 : Les mouvements possibles du quadrotor	8
Figure I.2 : Géométrie du Quadrotor	9
Figure I.3 : Quadrotor OS4	18
Figure II.1 : logique floue et logique classique.	20
Figure II.2 : Notions caractéristiques d'une fonction d'appartenance	21
Figure II.3 : Système d'inférence floue.	22
Figure II.4 : Les différents modèles d'inférence floue	23
Figure II.5 : Déplacement de la particule.	26
Figure III.1 : Commande d'attitude d'un quadrotor	33
Figure III.2 : Les fonctions d'appartenances de L'erreur	34
Figure III.3 : Les fonctions d'appartenances de la dérivée de L'erreur	34
Figure III.4 : Les singletons de la sortie	35
Figure III.5 : Structure d'optimisation	37
Figure IV.1 : Résultats obtenus par flou conventionnel	48
Figure IV.2 : Diminution de critère IAE	51
Figure IV.3 : Diminution de critère ISE	53
Figure IV.4 : Diminution de critère ITAE	55
Figure IV.5 : Somme de diminution des critères IAE, ISE et ITAE	56
Figure IV.6 : Somme de diminution des critères ISE, IAE et ITAE en altitude	58
Figure IV.7 : les graphes obtenus par minimisation ISE	59
Figure IV.8 : les graphes obtenus par minimisation d'IAE	61
Figure IV.9 : les graphes obtenus par minimisation d'ITAE	62
Figure IV.10 : les graphes en altitude	64
Figure IV.11 : les graphes en altitude	66

Les premiers développements en commande floue ont été initialisés par Mamdani [16]. L'idée de base consistait à exploiter l'expérience des opérateurs humains pour construire une loi de commande. Un jeu de règles floues traduit alors le comportement des opérateurs en termes de stratégie de commande. Le fonctionnement d'un contrôleur flou dépend principalement des caractéristiques de trois sous-systèmes : la fuzzification, les règles floues et la défuzzification. La phase de fuzzification est parfaitement spécifiée lorsque sont définies les fonctions d'appartenance des termes linguistiques décrivant les entrées. Les règles floues sont issues d'une analyse des connaissances fournies par un expert humain et de certaines caractéristiques du système commandé. La méthode utilisée pour la défuzzification doit permettre de transformer une fonction d'appartenance en une valeur de commande envoyée au système.

L'expertise humaine ne permet pas la synthèse des contrôleurs flous optimaux par conséquent en termes de performances et robustesse ces contrôleurs sont limités. La distribution optimale des fonctions d'appartenances d'un système d'inférence flou permet à la fois d'obtenir de meilleures performances et robustesse. Une multitude de méthodes d'optimisation des contrôleurs flous ont été proposées et appliquées dans la littérature dont on peut les classer en deux catégories. La première consiste à l'optimisation des SIF en basant sur les méthodes analytiques telle le descend de gradient par exemple. La deuxième catégorie consiste à utiliser les méthodes d'optimisation inespérées de la nature pour trouver le SIF optimal on peut citer les algorithmes génétiques par exemple.

Plusieurs méthodes métaheuristiques ont été développées et appliquées ces dernières années sur les systèmes technologiques. Elles sont pour la plupart basées sur un le codage en population et sur une recherche guidée et itérative de la valeur optimale d'une fonction fitness dans cette population. Pour le cas des SIF, les chercheurs se sont concentrés principalement sur l'optimisation : des fonctions d'appartenance, des règles floues ou simultanément les fonctions d'appartenance et les règles floues.

Dans ce travail l'optimisation des contrôleurs flous par des méthodes métaheuristiques telles que : la méthode d'essaim de particules (PSO), l'algorithme BAT et la recherche Coucou (CS) a été réalisé pour objectif de commander en attitude et en altitude un système avionique de type quadrotor. Le deuxième objectif de ce travail est d'extraire à travers

l'application de ces algorithmes sur le quadrotor le quel est le meilleur en terme de performances.

Ce travail comprendra quatre chapitres :

- Le premier chapitre est consacré aux généralités sur les drones et particulièrement sur le quadrotor ainsi qu'à la modélisation dynamique de ce dernier.
- Le deuxième chapitre est dédié à la présentation de la logique floue et les algorithmes d'optimisation métaheuristiques tels que : l'algorithme optimisation par essaim particulaire (PSO), l'algorithme BAT et la recherche coucou (CS).
- Le troisième chapitre est consacré à synthèse des contrôleurs flous optimisés pour la commande du Quadrotor.
- Le quatrième chapitre est pour objectif de présenter les résultats de simulations des commandes optimisées.

I.1 Introduction

Un drone est un engin volant qui capable de voler et d'effectuer une mission sans présence humain à bord, il peut voler d'une façon autonome ou à l'assistance d'un pilote à une station. Les drones sont capables de transporter des caméras, des capteurs, des équipements de communication ou d'autres dispositifs. Ils sont utilisés pour réaliser des missions de reconnaissance, de recherche d'informations ou des opérations de combat, etc.

Afin de concevoir un contrôleur de vol, on doit d'abord comprendre profondément les mouvements de drone, sa dynamique et par conséquent établir les équations qui décrivent sa dynamique. Cette compréhension est nécessaire pas simplement pour la conception du contrôleur, mais afin de s'assurer que les simulations du véhicule dépeindront un comportement aussi proche que possible à la réalité quand la commande est appliquée.

Ce chapitre contient deux parties, la première concerne des généralités sur les drones quant à la deuxième partie elle concerne la modélisation dynamique d'un drone de type quadrotor.

I.2 Généralités sur les UAV

I.2.1 Définition d'UAV

Un UAV (Unmanned Aerial Vehicle) ou drone est : un véhicule aérien motorisé, qui ne transporte pas d'opérateur humain, utilise la force aérodynamique pour assurer sa portance, peut voler de façon autonome ou être piloté à distance, être réutilisable ou récupérable et qui emporte une charge utile létale ou non létale [1].

I.2.2 Bref Historique

Malgré que la configuration "Quadrotor" n'a pas obtenue beaucoup d'attention jusqu'au début des années 80. Depuis, plusieurs chercheurs ont commencé à s'intéresser à cette configuration pour des raisons de simplicité, de capacité à supporter une charge utile et son coût réduit.

Depuis 2001, plusieurs centres de recherche et de groupes de spécialistes en aéronautique ont commencé à publier les premiers résultats à propos de la modélisation et de la commande de cet hélicoptère à quatre rotors tels que : les travaux de l'université de Compiègne et le projet 'Robovolint' en France, ceux de l'université de Lakehead au Canada

et beaucoup d'autres. Le tableau suivant dénombre quelques premiers projets de conception de Quadrotor [1] :

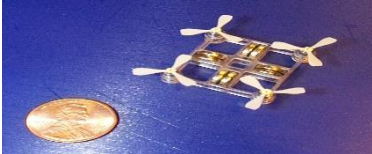
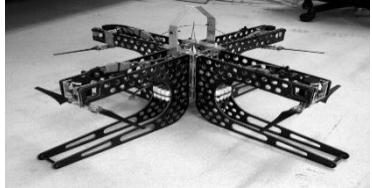





Projets	Université/Ecole	Conception
Le Mesicopter (1999-2001)	Stanford aux USA	
Le X4-Flyer(2002-2004)	Université Nationale Australienne	
Le X4-Flyer	CEA en France	
OS4 (2003-2007)	EPFL en suisse	
Quentin H4	EITA en France	
Le Quadrotor de Pennsylvanie	Pennsylvanie en USA	
STARMAC	L'université de Stanford USA	

Tableau I.1 : Projets de conception des Quadrotor

I.2.3 Classification des drones :

Les drones peuvent être répartis selon plusieurs critères : la taille, l'altitude, les systèmes de contrôle, etc.

A. Selon la taille

On distingue les drones Haute Altitude Longue Endurance (HALE), Moyenne Altitude Longue Endurance (MALE), micro drones et mini drones. Ces catégories sont résumées ci-après :

- **HALE** : Ce sont des drones de grande taille, le plus souvent à voilure fixe. Ils sont capables de rester très longtemps en vol et de collecter des informations sur de très longues périodes (entre 12 et 48 heures).
- **MALE** : sont utilisés pour des vols de longue durée à moyenne altitude opérationnelle, ayant une grande autonomie.

Ces deux types de drones font partie de la classe de grande taille. Ils peuvent embarquer des armes, ce qui nécessite généralement d'avoir un humain dans la boucle, ce dernier doit garder la décision de tir et pouvoir à tout moment annuler la mission.

- **Mini drones** : Ce sont des drones légers et de taille réduite (jusqu'à quelques kilogrammes et d'une envergure jusqu'à 1 à 2 mètres) ils ont une autonomie relativement faible (de 10 à 30 minutes) et généralement utilisés pour l'observation de zones difficiles à l'accès.
- **Micro drones** : Ce sont des drones ayant des tailles variant du centimètre à quelques dizaines de centimètres. Généralement propulsés électriquement. Ils permettent de faire des vols à l'intérieur. ils s'emportent de faibles charges.

B. Selon le mode de propulsion

On peut aussi classer les drones selon le fonctionnement aérodynamiques dont on trouve :

- Les drones à voilures fixes : sont des drones utilisant les ailes fixes dans leur mode de déplacement, qui sont soit :
 - Plus lourd que l'air : type avion.
 - Plus léger que l'air : type Dirigeable.
- Les drones à ailes battantes : de type oiseau ou insecte.
- Drones à voilures tournantes : Ce type présente les avantages suivants

- Un décollage et atterrissage vertical.
- Sont capables d'effectuer un vol stationnaire à basse vitesse et à faible altitude.

Les drones à voilure tournante se subdivisent en plusieurs sous-classes dont on trouve entre autres : Mono-rotor, birotor, tris-rotor, Quadrotor.

I.2.4 Les applications des drones

Les drones peuvent être faire des centaines d'applications, parmi ceux-ci [2] :

- Services publics et collectivités (Assistance police, Enquête écologique...)
- Agriculture (Surveillance des zones agricoles, Analyse des cultures...)
- Cartographie (géomètre expert, Analyse numérique des terrains, reconstitution 3D...)
- Architectures et urbanisme (Etude de chantier ou de grands ouvrages sur des angles de vue permettant une meilleure implantation).
- Intervention en sécurité civile (Intervention dans des situations d'urgence afin d'apporter un soutien aux autorités locales pour la surveillance de zones sinistrée).
- Tourisme et suivi d'évènements (Films et photos touristiques avec des angles de vue ne peuvent être effectués avec du matériel standard).
- Missions dangereuses (détection de gaz toxiques, radiations).
- Surveillance :
 - Surveillance de territoire réglementée (chasse, pêche...)
 - Surveillance de chantiers routiers, ferroviaires, bâtiments et travaux publiques...)
 - Surveillance des frontières
 - Surveillance du trafic routier et du transport de matières dangereuses.
- Industrie :
 - Contrôles de zones à risques
 - Surveillance d'installations (panneaux solaires, éoliennes, antennes, etc.).

I.3 Modélisation dynamique du Quadrotor

Beaucoup de tentatives de modélisation du drone sont enregistrées dans la littérature, telle que celle établie par Lozano [3] en utilisant la méthode Euler-Lagrange et celle de Bouabdallah [4], ou encore, le modèle présenté par Hamel [5] basé sur le formalisme de Newton a été obtenu à partir de la dynamique d'un corps rigide associé au fuselage auquel sont ajoutées les forces aérodynamiques générées par les rotors, ce modèle en plus intègre la dynamique des actionneurs. Notre approche de modélisation est basée sur le formalisme de Newton.

I.3.1 Les possibilités de vol du Quadrotor

Le Quadrotor est doté de quatre rotors dont les hélices sont à pas fixe qui lui permettent la sustentation par une variation de la vitesse angulaire de ses rotors. Deux hélices de même pas sont montées en opposée et tournent dans le même sens et les deux autres hélices tournent dans le sens opposé.

Les mouvements de base de Quadrotor sont réalisés en variant la vitesse de chaque rotor changeant de ce fait la poussée produite. Le quadrotor incline vers la direction du rotor plus lent, qui tient compte alors de translation le long de cet axe. Par conséquent, les mouvements sont couplés, signifiant que le Quadrotor ne peut pas réaliser la translation sans roulement ou tangage, ce qui signifie qu'un changement de la vitesse d'un rotor se traduit dans un mouvement en au moins trois degrés de liberté.

Les mouvements possibles du Quadrotor sont :

- Le mouvement vertical (sustentation) s'obtient de la contribution des quatre hélices au même temps.
- Le déplacement suivant l'axe X se produit suite à une rotation autour de l'axe Y, cette dernière se crée à cause de la différence de portance des rotors 1-3 (**Tangage θ**).
- Le déplacement suivant l'axe Y se produit suite à une rotation autour de l'axe X, cette dernière se crée à cause de la différence de portance des rotors 2-4 (**Roulis ϕ**).
- Le mouvement en lacet nécessite que deux rotors du même axe tournent dans un sens tandis que les deux autres dans l'autre sens (**Lacet ψ**).

La figure suivant expliquée les possibilités de vol du Quadrotor

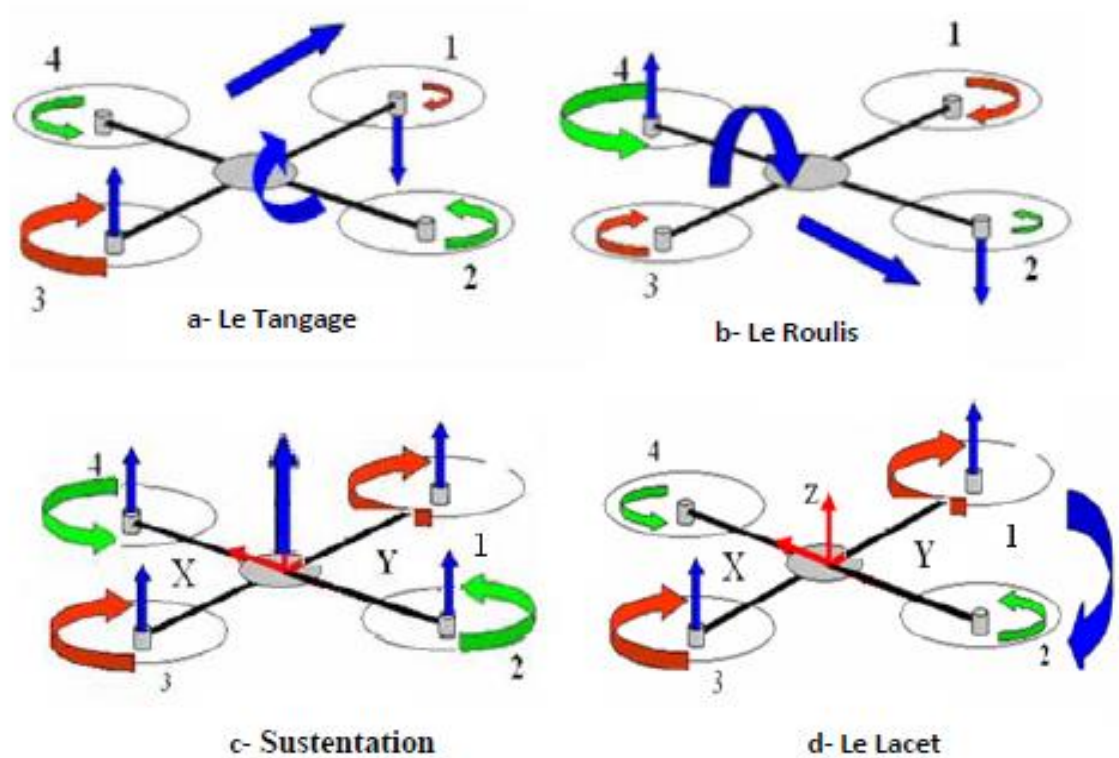


Figure I.1 : Les mouvements possibles du quadrotor

I.3.2 Modèle dynamique du quadrotor

La modélisation des quadrotor est très difficile puisque la dynamique du système est fortement non linéaire et couplée. Afin de pouvoir comprendre au mieux le modèle dynamique développé ci-dessous, nous considérons les hypothèses suivantes :

- La structure du quadrotor est supposée rigide et symétrique, ce qui induit que la matrice d'inertie sera supposée diagonale,
- Les hélices sont supposées rigides pour pouvoir négliger l'effet de leur déformation lors de la rotation.
- Le centre de masse et l'origine du repère lié à la structure coïncident.
- Les forces de portance et de traînée sont proportionnelles aux carrés de la vitesse de rotation des rotors, ce qui est une approximation très proche du comportement aérodynamiques du système.

Pour évaluer le modèle mathématique du quadrotor on utilise deux repères, un repère fixe lié à la terre R^b et un autre mobile R^m lié au centre de gravité de quadrotor. Le passage entre le repère mobile et le repère fixe est donné par une matrice dite matrice de transformation T qui contient l'orientation et la position de repère mobile par rapport au repère fixe.

On choisit le placement des axes comme suit :

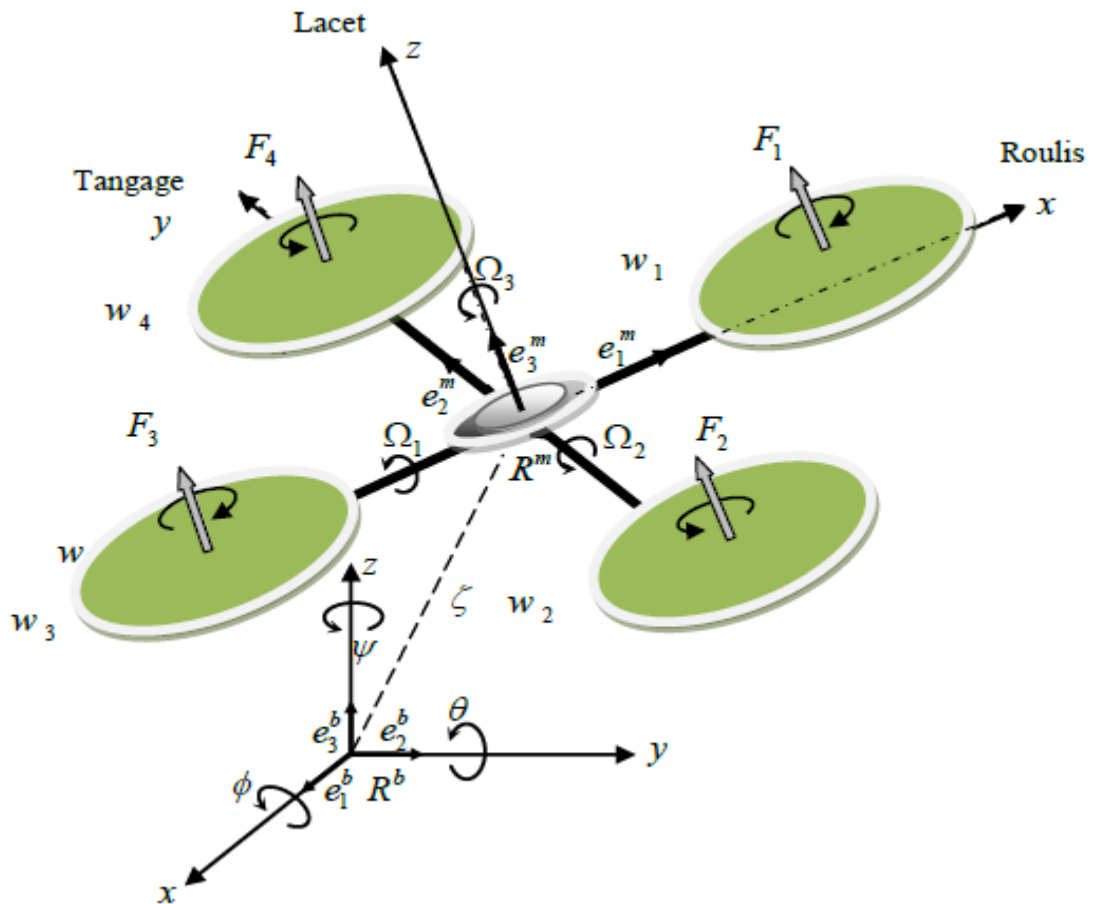


Figure I.2 : Géométrie du Quadrotor.

$$T = \begin{bmatrix} R & \zeta \\ 0 & 1 \end{bmatrix} \quad (I.1)$$

Avec R la matrice de rotation (décrit l'orientation de l'objet mobile), $\zeta = [x \ y \ z]^T$ est le vecteur de position. Pour déterminer les éléments de la matrice de R , on utilise les angles d'Euler.

I.3.3 Matrice de rotation

En utilisant les angles d'Euler les matrices des rotations sont :

- La rotation autour de x par l'angle ϕ traduite par la matrice :

$$R(x, \phi) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & c\phi & -s\phi \\ 0 & s\phi & c\phi \end{pmatrix} \quad (I.2)$$

- La rotation autour de y par l'angle θ traduite par la matrice :

$$R(y, \theta) = \begin{pmatrix} c\theta & 0 & s\theta \\ 0 & 1 & 0 \\ -s\theta & 0 & c\theta \end{pmatrix} \quad (\text{I.3})$$

- La rotation autour de z par l'angle ψ traduite par la matrice :

$$R(z, \psi) = \begin{pmatrix} c\psi & s\psi & 0 \\ -s\psi & c\psi & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad (\text{I.4})$$

La matrice globale de rotation entre le repère R^b et R^m est

$R = R(\phi, \theta, \psi) = R(x, \phi) * R(y, \theta) * R(z, \psi)$. En effectuant le produit de ces trois matrices en obtient :

$$R(\phi, \theta, \psi) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & c\phi & -s\phi \\ 0 & s\phi & c\phi \end{pmatrix} \begin{pmatrix} c\theta & 0 & s\theta \\ 0 & 1 & 0 \\ -s\theta & 0 & c\theta \end{pmatrix} \begin{pmatrix} c\psi & s\psi & 0 \\ -s\psi & c\psi & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad (\text{I.5})$$

$$R(\phi, \theta, \psi) = \begin{pmatrix} c\theta c\psi & c\psi s\theta s\phi - s\psi c\phi & c\psi s\theta s\phi + s\psi s\phi \\ c\theta s\psi & s\psi s\theta s\phi + c\psi c\phi & s\psi s\theta c\phi - c\psi s\phi \\ -s\theta & c\theta s\phi & c\theta c\phi \end{pmatrix} \quad (\text{I.6})$$

Avec $c = \cos$, et $s = \sin$;

I.3.4 Effets physiques agissants sur le Quadrotor

I.3.4.1 Les forces

Le poids : il donnée par $= mg$, ou m :est la masse et g la gravité.

Les forces de poussée : qui sont des forces provoquées par la rotation des moteurs, elles sont perpendiculaires sur le plan des hélices.

$$F_i = b\omega_i^2 \quad (\text{I.7})$$

Avec $i = 1: 4$, et b le coefficient de portance.

Les forces de trainée : la force de trainée est le couplage entre une force de pression et la force de frottement visqueux, dans ce cas on a deux forces de trainée agissant sur le système qu'elles sont :

- La traînée dans les hélices : elle agisse sur les pales, elle est proportionnelle à la densité de l'air, à la forme des pales et au carré de la vitesse de rotation de l'hélice, elle est donnée par la relation suivante :

$$T_h = d\omega^2 \quad (\text{I.8})$$

Avec d est le coefficient de drag.

- La traînée selon les axes (x y z) : elle est due au mouvement du corps du quadrotor

$$F_t = K_{ft}v \quad (\text{I.9})$$

Avec K_{ft} le coefficient de traînée de translation et v la vitesse linéaire.

I.3.4.2 les Moment

Moment dus aux forces de poussée :

- La rotation autour de l'axe x : elle est due au moment créé par la différence entre les forces de portance des rotors 2 et 4, ce moment est donné par la relation suivante :

$$M_x = l(F_4 - F_2) = lb(\omega_4^2 - \omega_2^2) \quad (\text{I.10})$$

Avec l est la longueur du bras entre le rotor et le centre de gravité du quadrotor.

- La rotation autour de l'axe y : elle est due au moment créé par la différence entre les forces de portance des rotors 1 et 3, ce moment est donné par la relation suivante :

$$M_y = l(F_3 - F_1) = lb(\omega_3^2 - \omega_1^2) \quad (\text{I.11})$$

Moments dus aux forces de traînée :

- La rotation autour de l'axe z : elle est due à un couple réactif provoqué par les couples de traînée dans chaque hélice, ce moment est donné par la relation suivante :

$$M_z = d(\omega_1^2 - \omega_2^2 + \omega_3^2 - \omega_4^2) \quad (\text{I.12})$$

- Moment résultant des frottements aérodynamiques, il est donné par :

$$M_a = K_{fa}\Omega^2 \quad (\text{I.13})$$

Avec K_{fa} le coefficient des frottements aérodynamiques et Ω est la vitesse angulaire.

I.3.4.3 Effet gyroscopique

L'effet gyroscopique se définit comme la difficulté de modifier la position ou l'orientation du plan de rotation d'une masse tournante. Dans notre cas il y a deux moments gyroscopiques, le premier est le moment gyroscopique des hélices, l'autre est le moment gyroscopique dû aux mouvements de quadrotor.

- Moment gyroscopique des hélices : il est donné par la relation suivante :

$$M_{gh} = \sum_1^4 \Omega \wedge J_r [0 \ 0 \ (-1)^{i+1} \ \omega_i]^T \quad (\text{I.14})$$

Avec J est l'inertie des rotors.

- Moment gyroscopique dû aux mouvements de quadrotor, il est donné par la relation suivante :

$$M_{gh} = \Omega \wedge J\Omega \quad (\text{I.15})$$

Avec J est l'inertie du système.

I.3.5 Développement du modèle mathématique selon Newton-Euler [6]

En utilisant la formulation de Newton-Euler, les équations sont écrites sous la forme suivante :

$$\left\{ \begin{array}{l} \dot{\zeta} = v \\ m\ddot{\zeta} = F_f + F_t + F_g \\ \dot{R} = RS(\Omega) \\ J\dot{\Omega} = -\Omega \wedge J\Omega + M_f - M_a - M_{gh} \end{array} \right. \quad (\text{I.16})$$

Avec ζ : est le vecteur de position du quadrotor,

m : La masse totale du quadrotor ,

Ω : La vitesse angulaire exprimée dans le repère fixe,

R : La matrice de rotation,

\wedge : Le produit vectoriel,

J : Matrice d'inertie symétrique de dimension (3x3), elle est donnée par :

$$J = \begin{bmatrix} I_x & 0 & 0 \\ 0 & I_y & 0 \\ 0 & 0 & I_z \end{bmatrix} \quad (\text{I.17})$$

$S(\Omega)$: est la matrice antisymétrique, pour un vecteur de vitesse $\Omega = [\Omega_1 \ \Omega_2 \ \Omega_3]^T$ elle est donnée par :

$$S(\Omega) = \begin{bmatrix} 0 & -\Omega_3 & \Omega_2 \\ \Omega_3 & 0 & -\Omega_1 \\ -\Omega_2 & \Omega_1 & 0 \end{bmatrix} \quad (\text{I.18})$$

F_f : est la force totale générée par les quatre rotors, elle est donnée par :

$$F_f = R * \begin{bmatrix} 0 & 0 & \sum_{i=1}^4 F_i \end{bmatrix}^T \quad (\text{I.19})$$

$$F_i = b\omega_i^2$$

F_t : est la force de trainée selon les axes (x y z) , elle est donnée par :

$$F_t = \zeta \begin{bmatrix} -K_{ftx} & 0 & 0 \\ 0 & -K_{fity} & 0 \\ 0 & 0 & -K_{ftz} \end{bmatrix} \quad (\text{I.20})$$

K_{ftx} , K_{fity} , K_{ftz} : Les coefficients de trainée de translation.

F_g : Force de gravité, elle est donnée par :

$$F_g = \begin{bmatrix} 0 \\ 0 \\ -mg \end{bmatrix} \quad (\text{I.21})$$

M_f : Moment provoqué par les forces de poussée et de trainée

$$M_f = \begin{bmatrix} l(F_4 - F_2) \\ l(F_3 - F_1) \\ d(\omega_1^2 - \omega_2^2 + \omega_3^2 - \omega_4^2) \end{bmatrix} \quad (\text{I.22})$$

M_a : Moment résultant des frottements aérodynamiques, il est donnée par

$$M_a = \begin{bmatrix} K_{fax}\dot{\phi}^2 \\ K_{fay}\dot{\theta}^2 \\ K_{faz}\dot{\psi}^2 \end{bmatrix} \quad (\text{I.23})$$

K_{fax} , K_{fay} , K_{faz} : Les coefficients des frottements aérodynamiques.

I.3.5.1 Equations de mouvement de translation

On a :

$$m\ddot{\zeta} = F_f + F_t + F_g \quad (I.24)$$

On remplace chaque force par sa formule (19.I, 20.I et 21.I), on trouve :

$$m \begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix} = \begin{bmatrix} c\psi s\theta c\phi + s\psi s\phi \\ s\psi s\theta c\phi - s\psi c\phi \\ c\theta c\phi \end{bmatrix} \sum_{i=1}^4 F_i - \begin{bmatrix} K_{f_{tx}}\dot{x} \\ K_{f_{ty}}\dot{y} \\ K_{f_{tz}}\dot{z} \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ mg \end{bmatrix} \quad (I.25)$$

On obtient alors les équations différentielles qui définissent la dynamique de translation :

$$\begin{cases} \ddot{x} = \frac{1}{m} (\psi s\theta c\phi + s\psi s\phi) (\sum_{i=1}^4 F_i) - \frac{K_{f_{tx}}}{m} \dot{x} \\ \ddot{y} = \frac{1}{m} (s\psi s\theta c\phi - s\psi c\phi) (\sum_{i=1}^4 F_i) - \frac{K_{f_{ty}}}{m} \dot{y} \\ \ddot{z} = \frac{1}{m} (c\theta c\phi) (\sum_{i=1}^4 F_i) - \frac{K_{f_{tz}}}{m} \dot{z} - g \end{cases} \quad (I.26)$$

I.3.5.2 Equations de mouvement de rotation

On a :

$$J\dot{\Omega} = -M_{gm} - M_{gh} + M_f - M_a \quad (I.27)$$

On remplace chaque moment par la formule correspondante, on trouve :

$$\begin{bmatrix} I_x & 0 & 0 \\ 0 & I_y & 0 \\ 0 & 0 & I_z \end{bmatrix} = - \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \wedge \left(\begin{bmatrix} I_x & 0 & 0 \\ 0 & I_y & 0 \\ 0 & 0 & I_z \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \right) - \begin{bmatrix} J_r \bar{\Omega}_r \dot{\theta} \\ -J_r \bar{\Omega}_r \dot{\phi} \\ 0 \end{bmatrix} - \begin{bmatrix} K_{fax} \dot{\phi}^2 \\ K_{fay} \dot{\theta}^2 \\ K_{faz} \dot{\psi}^2 \end{bmatrix} + \begin{bmatrix} l(F_4 - F_2) \\ l(F_3 - F_1) \\ d(\omega_1^2 - \omega_2^2 + \omega_3^2 - \omega_4^2) \end{bmatrix} \quad (I.28)$$

On obtient alors les équations différentielles définissant la dynamique de la rotation :

$$\left\{ \begin{array}{l} I_x \ddot{\phi} = -\dot{\theta} \dot{\psi} (I_z - I_y) - J_r \bar{\Omega}_r \dot{\theta} - K_{fax} \dot{\phi}^2 + lb(\omega_4^2 - \omega_2^2) \\ I_y \ddot{\theta} = \dot{\phi} \dot{\psi} (I_z - I_x) + J_r \bar{\Omega}_r \dot{\phi} - K_{fay} \dot{\theta}^2 + lb(\omega_3^2 - \omega_1^2) \\ I_z \ddot{\psi} = -\dot{\phi} \dot{\theta} (I_y - I_x) - K_{faz} \dot{\psi}^2 + d(\omega_1^2 - \omega_2^2 + \omega_3^2 - \omega_4^2) \end{array} \right. \quad (\text{I.29})$$

Avec :

$$\bar{\Omega}_r = \omega_1 - \omega_2 + \omega_3 - \omega_4 \quad (\text{I.30})$$

En conséquence, le modèle dynamique complet régit la dynamique du quadrotor est :

$$\left\{ \begin{array}{l} \ddot{\phi} = \dot{\theta} \dot{\psi} \frac{(I_y - I_z)}{I_x} - \frac{J_r}{I_x} \bar{\Omega}_r \dot{\theta} - \frac{K_{fax}}{I_x} \dot{\phi}^2 + \frac{l}{I_x} u_2 \\ \ddot{\theta} = \dot{\phi} \dot{\psi} \frac{(I_z - I_x)}{I_y} + \frac{J_r}{I_y} \bar{\Omega}_r \dot{\phi} - \frac{K_{fay}}{I_y} \dot{\theta}^2 + \frac{l}{I_y} u_3 \\ \ddot{\psi} = -\dot{\phi} \dot{\theta} \frac{(I_x - I_y)}{I_z} - \frac{K_{faz}}{I_z} \dot{\psi}^2 + \frac{1}{I_z} u_4 \\ \ddot{x} = -\frac{K_{ftx}}{m} \dot{x} + \frac{1}{m} u_x u_1 \\ \ddot{y} = -\frac{K_{fty}}{m} \dot{y} + \frac{1}{m} u_y u_1 \\ \ddot{z} = -\frac{K_{ftz}}{m} \dot{z} - g + \frac{\cos(\phi) \cos(\theta)}{m} u_1 \end{array} \right. \quad (\text{I.31})$$

Avec

$$\left\{ \begin{array}{l} u_x = (c\phi c\psi s\theta + s\phi s\psi) \\ u_y = (c\phi s\theta s\psi - s\phi c\psi) \end{array} \right. \quad (\text{I.32})$$

Et :

$$\begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} = \begin{bmatrix} b & b & b & b \\ 0 & -lb & 0 & lb \\ -lb & 0 & lb & 0 \\ d & -d & d & -d \end{bmatrix} \begin{bmatrix} \omega_1^2 \\ \omega_2^2 \\ \omega_3^2 \\ \omega_4^2 \end{bmatrix} \quad (\text{I.33})$$

À partir de (I.32), on trouve :

$$\begin{cases} \phi_d = \arcsin(u_x \sin(\psi_d) - u_y \cos(\psi_d)) \\ \theta_d = \arcsin\left(\frac{u_x \cos(\psi_d) + u_y \sin(\psi_d)}{\cos(\phi_d)}\right) \end{cases} \quad (\text{I.34})$$

I.3.6 La représentation d'état du système

On choisit le vecteur d'état suivant [6]

$$\begin{aligned} X &= [\phi \quad \dot{\phi} \quad \theta \quad \dot{\theta} \quad \psi \quad \dot{\psi} \quad x \quad \dot{x} \quad y \quad \dot{y} \quad z \quad \dot{z}]^T \\ &= [x_1 \quad x_2 \quad x_3 \quad x_4 \quad x_5 \quad x_6 \quad x_7 \quad x_8 \quad x_9 \quad x_{10} \quad x_{11} \quad x_{12}]^T \end{aligned}$$

On obtient la représentation d'état suivante :

$$\begin{cases} \dot{x}_1 = x_2 \\ \dot{x}_2 = a_1 x_4 x_6 + a_2 x_2^2 + a_3 \bar{\Omega}_r x_4 + b_1 u_2 \\ \dot{x}_3 = x_4 \\ \dot{x}_4 = a_4 x_2 x_6 + a_5 x_4^2 + a_6 \bar{\Omega}_r x_2 + b_2 u_3 \\ \dot{x}_5 = x_6 \\ \dot{x}_6 = a_7 x_2 x_4 + a_8 x_6^2 + b_3 u_4 \\ \dot{x}_7 = x_8 \\ \dot{x}_8 = a_9 x_8 + \frac{1}{m} u_x u_1 \\ \dot{x}_9 = x_{10} \\ \dot{x}_{10} = a_{10} x_{10} + \frac{1}{m} u_y u_1 \\ \dot{x}_{11} = x_{12} \\ \dot{x}_{12} = a_{11} x_{12} + \frac{\cos(\phi) \cos(\theta)}{m} u_1 - g \end{cases} \quad (\text{I.35})$$

Avec :

$$\left[\begin{array}{l} a_1 = \frac{(I_y - I_z)}{I_x}, a_2 = -\frac{K_{fax}}{I_x}, a_3 = -\frac{J_r}{I_x}, a_4 = \frac{(I_z - I_x)}{I_y}, a_5 = -\frac{K_{fay}}{I_y}, a_6 = \frac{J_r}{I_y}, \\ a_7 = \frac{(I_x - I_y)}{I_z}, a_8 = -\frac{K_{faz}}{I_z}, a_9 = -\frac{K_{ftx}}{m}, a_{10} = -\frac{K_{fty}}{m}, a_{11} = -\frac{K_{ftz}}{m}, b_1 = \frac{l}{I_x}, \\ b_2 = \frac{l}{I_y}, b_3 = \frac{1}{I_z} \end{array} \right. \quad (I.36)$$

I.3.7 Les valeurs des paramètres des quadrotor

Nous allons utiliser les paramètres du quadrotor OS4 illustré dans la figure (I.3), ses paramètres sont donnés par le tableau suivant [7] :

Paramètre	Valeur	Paramètre	Valeur
m	0.650 kg	I_z	$1.3 \times 10^{-2} \text{ kg.m}^2$
g	9.806 m/s ²	K_{fax}	$5.5670 \times 10^{-4} \text{ N/rad/s}$
l	0.23 m	K_{fay}	$5.5670 \times 10^{-4} \text{ N/rad/s}$
b	$3.13 \times 10^{-5} \text{ N/rad/s}$	K_{faz}	$6.3540 \times 10^{-4} \text{ N/rad/s}$
d	$7.5 \times 10^{-5} \text{ N.m/rad/s}$	K_{ftx}	$5.5670 \times 10^{-4} \text{ N/rad/s}$
J_r	$6 \times 10^{-5} \text{ kg.m}^2$	K_{fty}	$5.5670 \times 10^{-4} \text{ N/rad/s}$
I_x	$7.5 \times 10^{-3} \text{ kg.m}^2$	K_{ftz}	$6.3540 \times 10^{-4} \text{ N/rad/s}$
I_y	$7.5 \times 10^{-3} \text{ kg.m}^2$		

Tableau I.2 : Paramètres du quadrotor utilisé.



Figure I.3 : Quadrotor OS4

I.4 Conclusion

Dans ce chapitre nous avons donné des généralités sur le monde de drone et leurs domaines d'applications, on a constaté que, l'une des configurations qu'a connue un développement et une exploitation considérables, cette dernière décennie, que ce soit à l'échelle scientifique ou industrielle est le quadrotor.

Sa modélisation a été établie par l'usage du formalisme Newton-Euler, ce formalisme nous a permis d'établir le modèle dynamique du quadrotor. À partir du modèle obtenu, nous concluons que le quadrotor est un système sous actionné. De plus la complexité du modèle, la non linéarité, et l'interaction entre les états du système, peuvent se voir clairement.

II.1 Introduction

Le nombre de travaux de recherche sur le développement des contrôleurs par logique floue ainsi que le nombre d'applications industrielles de la commande floue a augmenté considérablement dans les trois dernières décennies. Les bases théoriques de la logique floue ont été réellement introduites en 1965 par le professeur Lotfi A. Zadeh de l'université de Berkeley en Californie.

D'autres parts, les métaheuristiques sont des méthodes d'optimisation utilisées pour résoudre des problèmes difficiles. Elles explorent aléatoirement l'espace de recherche pour trouver les meilleures solutions.

En première partie de ce chapitre nous allons introduire le concept de base de la logique floue (la théorie et la commande), puis en deuxième partie nous allons présenter les algorithmes métaheuristiques et leurs principes d'optimisation. Trois algorithmes que nous allons appliquer pour l'optimisation des contrôleurs flous tels que : PSO, BAT, CS seront détaillés.

II.2 La logique floue

La logique floue est une technique permet de traiter des connaissances imprécises basée sur l'usage des termes linguistique. Elle donne les moyens de convertir une commande linguistique basée sur le raisonnement humain, en une commande automatique permettant ainsi la commande des systèmes complexes dont les informations sont exprimées d'une façon vague et mal définie [8]. La logique floue est la tentative de créer une technique fonctionne comme le cerveau humain.

II.2.1 la logique floue et la logique classique

Contrairement à la logique classique dédiée au fonctionnement de la machine. La logique floue est une tentative de créer une technique fonctionne comme le cerveau humain.

Exemple

Supposant que la limite est 1.80m, je mesure 1.75m. Suis-je vraiment petit ?

Logique classique : oui je suis petit 100 %

Logique floue : à la fois je suis petit et grand

II.2.2 Les bases théoriques de la logique floue

II.2.2.1 Les sous-ensembles flous

La logique floue repose sur la théorie des ensembles flous, qui sont une généralisation de la théorie des ensembles classiques. Dire que la théorie des ensembles flous est une généralisation de la théorie des ensembles classiques signifie que cette dernière n'est qu'un cas particulier la théorie des ensembles flous. Plus précis la théorie des ensembles classiques n'est qu'un sous-ensemble de la théorie des ensembles flous. Comme la figure suivant représente :

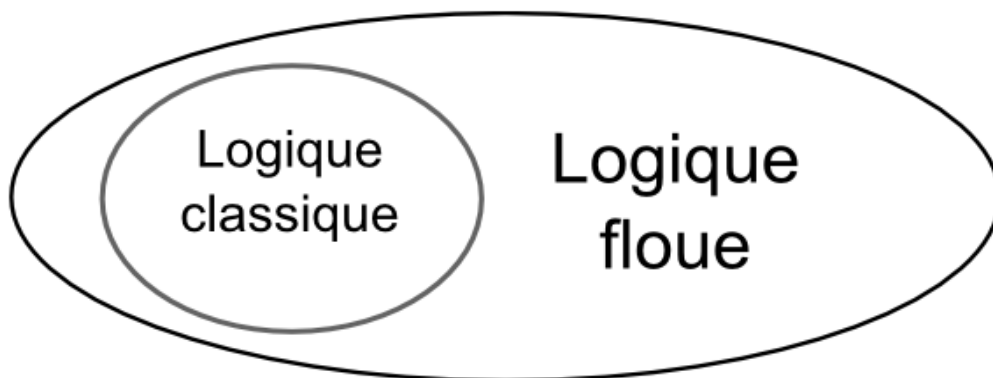


Figure II.1 : logique floue et logique classique.

II.2.2.2 Fonction d'appartenance

DEFINITION : Soit X un ensemble. Un *sous-ensemble flou* A de X est caractérisé par une *fonction d'appartenance* $f^a: X \rightarrow [0,1]$

Il existe plusieurs formes des fonctions d'appartenance, le plus souvent on utilise les formes trapézoïdales, sigmoïde, tangente hyperbolique, exponentielle, gaussienne.

II.2.2.3 Notions caractéristiques

Soit A un sous-ensemble flou de X

- DEFINITION 1: La **hauteur** de A , notée $h(A)$, correspond à la borne supérieure de l'ensemble d'arrivée de sa fonction d'appartenance : $h(A) = \sup\{\mu_A(x) / x \in X\}$.
- DEFINITION 2: A est dit **normalisé** si et seulement si $h(A) = 1$. En pratique, il est extrêmement rare de travailler sur des ensembles flous non normalisés.
- DEFINITION 3: Le **support** de A est l'ensemble des éléments de X appartenant au moins un peu à A . Autrement dit, c'est l'ensemble $\text{supp}(A) = \{x \in X / \mu_A(x) > 0\}$.

- DEFINITION 4 : Le *noyau* de A est l'ensemble des éléments de X appartenant totalement à A . Autrement dit, c'est l'ensemble $noy(A)=\{x \in X / \mu_A(x)=1\}$. Par construction, $noy(A) \subseteq supp(A)$.
- DEFINITION 5: Une α -coupe de A est le sous-ensemble classique des éléments ayant un degré d'appartenance supérieur ou égal à α : α -coupe(A)= $\{x \in X / \mu_A(x) \geq \alpha\}$.

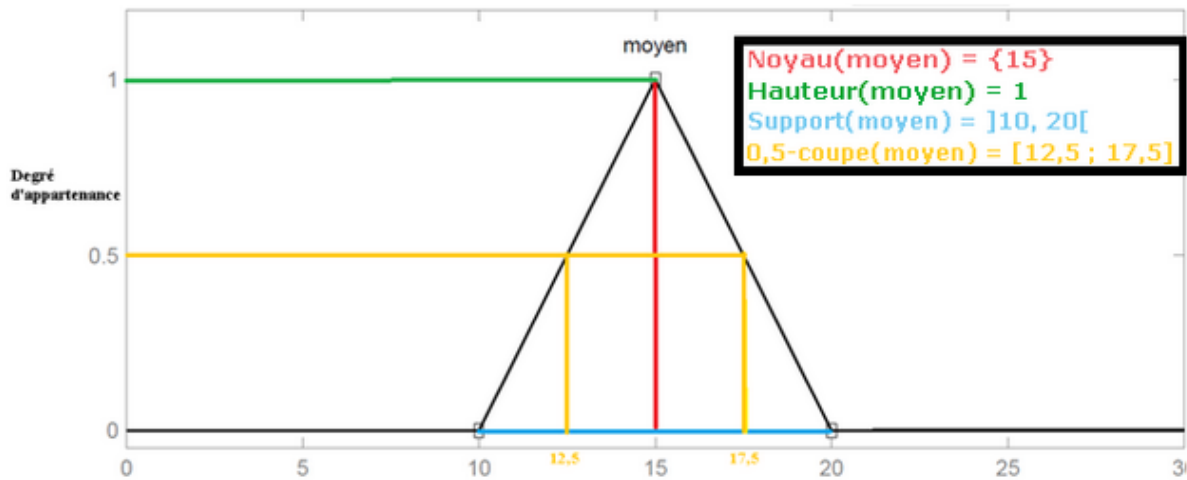


Figure II.2 : Notions caractéristiques d'une fonction d'appartenance

II.2.2.4 Opérations sur les sous-ensembles flous

La définition des opérateurs sur les ensembles flous est choisie, à l'instar des fonctions d'appartenance. Voici les deux ensembles d'opérateurs pour le complément (NON), l'intersection (ET) et l'union (OU) utilisés le plus couramment :

Dénomination	Intersection ET : $\mu_{A \cap B}(x)$	Réunion OU : $\mu_{A \cup B}(x)$	Complément NON $\mu_{\bar{A}}(x)$
Opérateur de Zadeh MIN/MAX	$\min(\mu_A(x), \mu_B(x))$	$\max(\mu_A(x), \mu_B(x))$	$1 - \mu_A(x)$
Probabiliste PROD/PROBOR	$\mu_A(x) \times \mu_B(x)$	$\mu_A(x) + \mu_B(x) - \mu_A(x) \times \mu_B(x)$	$1 - \mu_A(x)$

Note :

$$\mu_{A \cap \bar{A}}(x) \neq 0 \quad (\text{II.1})$$

$$\mu_{A \cup \bar{A}}(x) \neq 1 \quad (\text{II.2})$$

II.2.3 Système d'inférence floue (SIF)

Un système d'inférence floue se constitue par trois blocs (Figure II.3) :

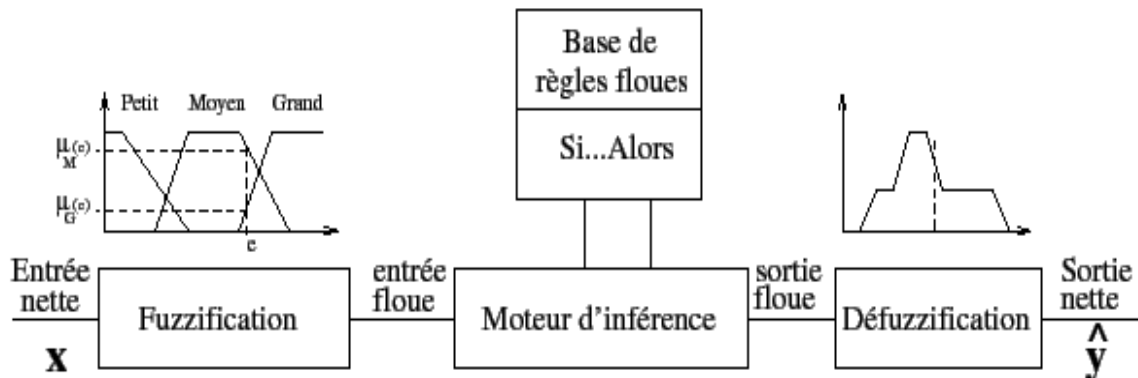


Figure II.3 : Système d'inférence floue.

II.2.3.1 Fuzzification :

C'est l'opération de transformation des entrées/sorties (grandeurs physiques) en grandeurs floues (variables linguistiques). On définit des fonctions d'appartenance de toutes les variables d'entrées et de sorties.

II.2.3.2 Moteur d'inférence

Constitué de l'ensemble des règles qui liées les variables d'entrées (exprimées comme des variables linguistiques) et les variables de sortie (également exprimées comme des variables linguistique). En logique floue on peut définir plusieurs méthodes d'inférence Mamdani, Tsukumoto, Takagi Sugeno, etc.

II.2.3.2.1 Méthode de MAMDANI [9]

Dans le modèle de Mamdani, les prémisses et les conclusions des règles sont symboliques ou linguistiques. Cette méthode se base sur l'utilisation de l'opérateur *min* pour l'implication floue et l'opérateur *max* pour l'agrégation des règles. La sortie nécessite l'utilisation généralement de la méthode de *Centre de Gravite* pour la défuzzification. Une autre variante du modèle de Mamdani consiste à remplacer l'opérateur *min* de l'implication floue par le *produit algébrique*.

II.2.3.2 Méthode de Sugeno [9]

Elle s'appelle aussi méthode de Takagi-Sugeno. Dans ce cas, des règles floues de type Sugeno sont utilisées. En effet, les conclusions des règles floues sont des polynômes ou plus généralement des fonctions des variables d'entrée. L'implication floue est réalisée par l'opérateur *min* ou par le *produit algébrique*. La sortie finale est égale à la moyenne pondérée des conclusions des règles.

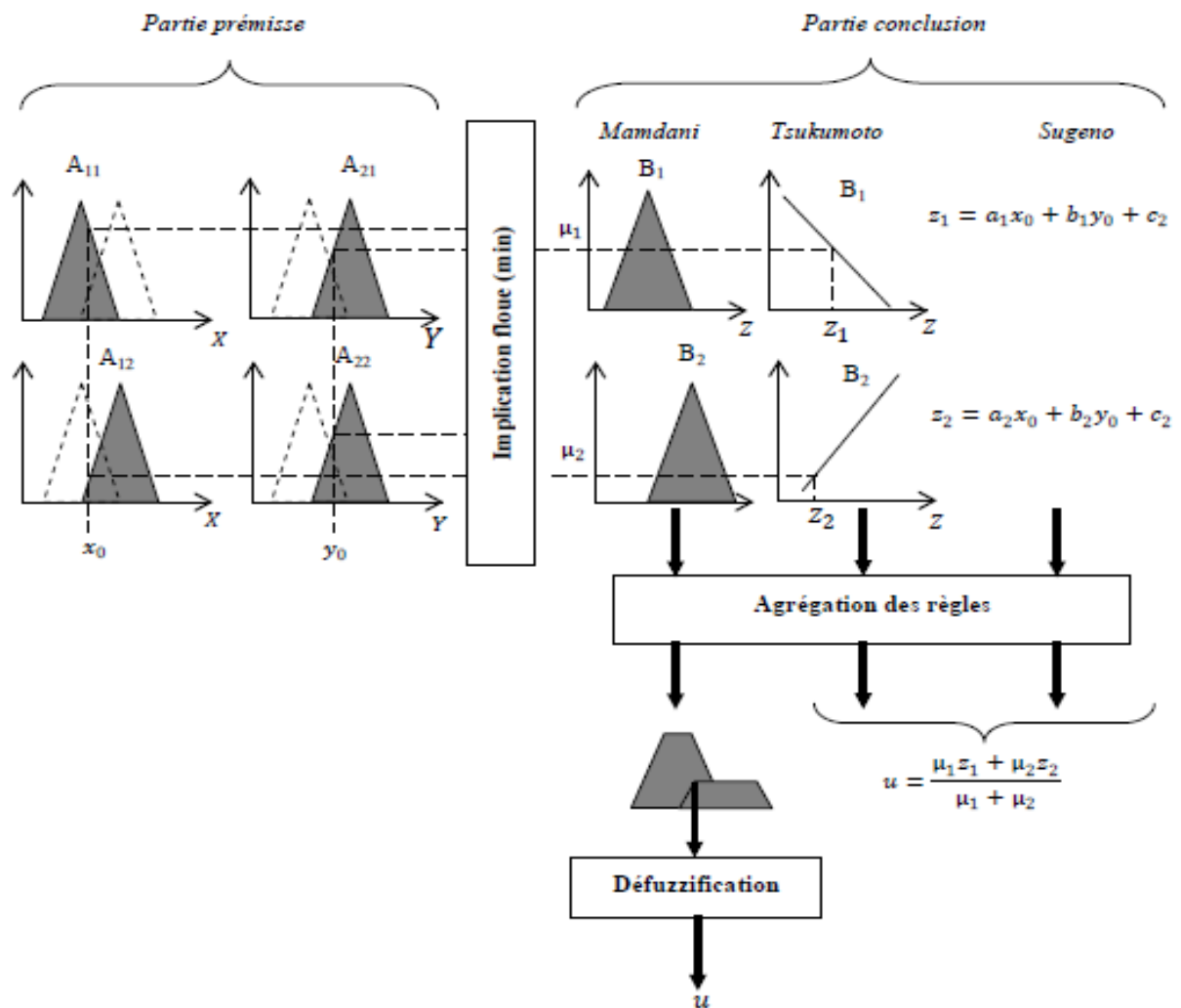


Figure II.4 : Les différents modèles d'inférence floue

II.2.3.3 Défuzzification :

Les méthodes d'inférence donnent une fonction d'appartenance résultante pour la variable de sortie. Il s'agit donc d'une information floue qu'il faut transformer en grandeur physique. Plusieurs méthodes existent, moyenne des maxima, centroïde, somme pondérée, etc.

II.3 Optimisation métaheuristique

Le terme métaheuristique vient des mots grecs *meta* (au-delà) et *heuriskein* (trouver) [10]. Les métaheuristicques sont apparues dans les années 1980 et forment une famille d'algorithmes d'optimisation visant à résoudre des problèmes d'optimisation difficile (souvent issus des domaines de la recherche opérationnelle, de l'ingénierie ou de l'intelligence artificielle). Elles sont généralement utilisées comme des méthodes génériques pouvant optimiser une large gamme de problèmes différents, sans nécessiter de changement profonds dans l'algorithme employé. L'un des intérêts majeurs des métaheuristicques est leur facilité d'utilisation dans des problèmes concrets sans nécessité de connaissances particulières sur le problème d'optimisation à résoudre.

Les métaheuristicques sont souvent inspirées par des systèmes naturels, qu'ils soient pris en physique (les méthodes de voisinage comme le recuit simulé et la recherche tabou), en biologie de l'évolution (les algorithmes évolutifs comme les algorithmes génétiques) ou encore en éthologies (les algorithmes de colonies de fourmis et d'optimisation par essaim particulaire).

II.3.1 Principales caractéristiques [10]

- Les métaheuristicques sont des stratégies qui permettent de guider la recherche d'une solution optimale.
- Le but visé par les métaheuristicques est d'explorer l'espace de recherche efficacement afin de déterminer des solutions optimales.
- Les métaheuristicques peuvent contenir des mécanismes qui permettent d'éviter d'être bloqué dans des régions de l'espace de recherche.
- Les métaheuristicques peuvent faire appel à des heuristiques qui tiennent compte de la spécificité du problème traité, mais ces heuristiques sont contrôlées par une stratégie de niveau supérieur.
- Les métaheuristicques peuvent faire usage de l'expérience accumulée durant la recherche de l'optimum, pour mieux guider la suite du processus de recherche.

II.3.2 L'algorithme d'Optimisation par Essaim Particulaire (OEP)

II.3.2.1 Principe de fonctionnement

La méthode d'Optimisation par Essaim Particulaire (OEP) a été proposée en 1995 par James Kennedy et Russel Eberhart [11] qui cherchaient à simuler la capacité des oiseaux à

voler de façon synchrone et leur aptitude à changer soudainement de direction, tout en restant en formation optimale. Le fonctionnement de l'OEP fait qu'elle peut être classée parmi les méthodes itératives (approche progressive de la solution) et stochastiques dans le but d'améliorer la situation existante en se déplaçant partiellement au hasard et partiellement selon des règles prédéfinies, en vue d'atteindre la solution globale souhaitée.

La méthode d'optimisation par essaim particulaire, est une procédure de recherche basée sur une population d'individus, appelés particules, qui changent leur position (état) avec le temps. Dans un système d'OEP, les particules se déplacent à l'intérieur d'un espace de recherche. Pendant le déplacement, chaque particule ajuste sa position selon sa propre expérience, et selon l'expérience des particules voisines, se servant de sa meilleure position produite et de celle de ses voisines.

II.3.2.2 Les éléments de l'OEP [11]

Pour appliquer l'OEP, il faut définir un espace de recherche constitué de particules et une fonction coût (fonction '*objectif*') à optimiser. Le principe de l'algorithme est de déplacer ces particules afin qu'elles trouvent l'optimum. Chacune de ces particules est dotée :

- D'une position, c'est-à-dire ses coordonnées dans l'ensemble de définition.
- D'une vitesse qui permet à la particule de se déplacer. De cette façon, au cours des itérations, chaque particule change de position. Elle évolue en fonction de son meilleur voisin, de sa meilleure position, et de sa position précédente. C'est cette évolution qui permet de tomber sur une particule optimale.
- D'un voisinage, c'est-à-dire un ensemble de particules qui interagissent directement sur la particule, en particulier celle qui a le meilleur critère.

Dans le cas d'un problème d'optimisation, la qualité d'un site de l'espace de recherche est déterminée par la valeur de la fonction coût en ce point.

II.3.2.3 Principe fondamental

L'algorithme de base de l'OEP travaille sur une population appelée *essaim* de solutions possibles, elles-mêmes appelées *particules*. Ces particules sont placées aléatoirement dans l'espace de recherche de la fonction *objectif*.

A chaque itération, chaque particule se déplace en prenant en compte sa meilleure position (p_{best}) ainsi que la meilleure position de son voisinage (g_{best}). On calcule alors la nouvelle vitesse de chaque particule par la formule (II.3), La nouvelle position sera

déterminée par la somme de la position précédente et la nouvelle vitesse comme l'indique l'équation suivante :

$$V(t + 1) = w * V(t) + c_1 * R_1 * (p_{best}(t) - p(t)) + c_2 * R_2 * (g_{best}(t) - p(t)) \quad (II.3)$$

$$p(t + 1) = p(t) + V(t + 1) \quad (II.4)$$

Où V est la vitesse de la particule, w est en général une constante appelée, *coefficient d'inertie*, p est la position (solution) actuelle, c_1 et c_2 sont deux constantes, appelées *coefficients d'accélération*, R_1 et R_2 sont deux nombres aléatoires tirés uniformément dans l'intervalle $[0,1]$.

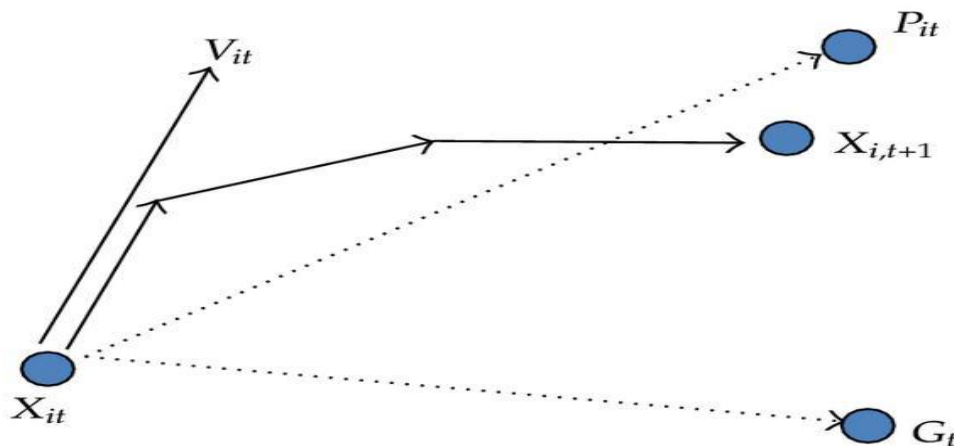


Figure II.5 : Déplacement de la particule.

II.3.2.4 Pseudo code d'algorithme

Algorithme II.1 : Algorithme OEP (PSO)

Entrée Fonction objective, Nombre de particules

Sortie $f(g_{best})$ Solution optimal

Initialisation : Générer aléatoirement x^i, v^i, R_1^i, R_2^i

Pour chaque particule $i, p_{best}^i = p^i$

Tant que le critère d'arrêt n'est pas atteint **faire**

Pour $i = 1$ à N **faire**

Modification de la position et la vitesse de chaque particule par (II.3)et (II.4)

Evaluation des positions

Si $f(p^i) < f(p_{best}^i)$

$p_{best}^i = p^i$

Fin Si

Si $f(p_{best}^i) < f(g_{best})$

$g_{best} = p_{best}^i$

Fin Si

Fin Pour

Fin tant que

Le critère d'arrêt peut être différent suivant le problème posé. Si l'optimum global est connu a priori, on peut définir une erreur acceptable comme critère d'arrêt. Sinon, il est commun de fixer un nombre maximum d'évaluations de la fonction *objectif* ou un nombre maximum d'itérations comme critère d'arrêt.

II.3.3 L'Algorithme BAT [12]

II.3.3.1 Echolocalisation

L'écholocalisation consiste à envoyer des sons et à écouter leur écho pour localiser, et dans une moindre mesure identifier, les éléments d'un environnement. Elle est utilisée par certains animaux. Les chauves-souris sont les exemples les plus connus d'animaux utilisant l'écholocation.

Les microchiroptères génèrent des ultrasons via leur larynx. Ce son est émis par leur nez et leur bouche. La fréquence d'émission varie entre 14 000 et 100 000 hertz, bien au-delà de ce que l'oreille humaine peut percevoir (l'oreille humaine moyenne perçoit les sons entre 20 Hz et 20000 Hz). Un groupe de vocalisations émises permet d'explorer l'environnement.

II.3.3.2 Principe fondamental de l'algorithme

L'optimisation par la technique des chauvesouris (BAT) est une approche inspirée du comportement de chasse des chauvesouris. Cette méthode d'optimisation a été inventée par Xin-She Yang en 2010 [12]. Au cours de leur vol et afin d'éviter les obstacles et cibler leur proies, chaque individu (bat) émet un bisonar à travers son environnement le retour de l'écho permet d'identifier les divers objets de son entourage. Les études développées dans ce domaine ont montré que la résonance de l'onde émise varie d'une valeur élevée au cours d'un vol de prospection à une valeur assez faible avec une augmentation de fréquence quand la chauvesouris détecte et oblique en direction d'une proie. Il est supposé par intuition que ces animaux sont en mesure de différencier leurs proies des autres obstacles proches, notamment les chauves-souris avoisinantes.

Similaire à l'OEP, l'algorithme BAT peut être implémenté pour la résolution de problèmes d'optimisation continue où les solutions possibles peuvent être représentées par les positions géographiques des bats.

Le principe de l'algorithme est décrit comme suit :

- Dans un espace \mathbb{R}^n , à l'instant t , chaque bat i a une position x_i^t se déplace à une vitesse $v_i^t \in \mathbb{R}^n$ en émettant une pulsation de fréquence constante f_{min}
- A la perception d'une proie, les paramètres de position et vitesse seront ajustés selon les relations suivantes :

$$f_i = f_{min} + \beta(f_{max} - f_{min}) \quad (\text{II.5})$$

$$v_i^{t+1} = v_i^t + f_i(x_i^t - x_{best}^t) \quad (\text{II.6})$$

$$x_i^{t+1} = x_i^t + v_i^{t+1} \quad (\text{II.7})$$

Où β un vecteur aléatoire $\in [0,1]$ et x_{best} est la position du meilleur bat du groupe.

- En chaque itération on met à jour la fréquence A_i et l'amplitude r_i par les équations suivantes :

$$A_i^{t+1} = \alpha A_i^t \quad (\text{II.8})$$

$$r_i^{t+1} = r_i^0 [1 - \exp(-\gamma t)] \quad (\text{II.9})$$

Où α et γ sont des constants $0 < \alpha < 1$ et $\gamma > 0$.

II.3.3.3 Pseudo code de l'algorithme

Algorithme II.2 : Algorithme BAT

Entrée Fonction objective, Nombre de bats

Sortie $f(x_{best})$ Solution optimal

Initialisation : Générer aléatoirement x_i, v_i, β_i

Initialiser $x_{best} = x_i$ telque $f(x_i) = \min(f(x_{i...N}))$

Tant que le critère d'arrêt n'est pas atteint **faire**

Pour $i = 1$ à N **faire**

Modification de la position et la vitesse de chaque bat par (II.6) et (II.7)

Evaluation des positions

Si (rand > r)

$x_i = x_{best} + rand$

Fin Si

Evaluer $f(x_i)$

Si $f(x_i) < f(x_{best})$ & (rand < A)

$x_{best} = x_i$

$f(x_{best}) = f(x_i)$

Fin Si

Fin Pour

 accroître la fréquence de l'onde et diminuer l'amplitude par (II.8) et (II.9)

Fin tant que

II.3.4 La Recherche Coucou (CS)

II.3.4.1 Le comportement et la reproduction des coucous [13-14]

Les coucous sont d'oiseaux parasites de couvée. Ils ne construisent pas leur propre nid pour pondre leurs œufs. Quelques espèces entre eux s'engagent à pondre leurs œufs dans des nids communautaires. D'autres espèces pondent leurs œufs dans des nids d'autres espèces puis ils se sauvent. Certains coucou ne couvent pas et ne nourrissent jamais leurs poussins. De nombreuses espèces d'oiseaux apprennent à reconnaître les œufs coucous déversés dans leurs propres nids. Dans le cas où l'oiseau hôte arrive à découvrir l'œuf coucou dans son nid, il va soit jeter l'œuf étranger hors son nid ou abandonner son propre nid.

En général, les œufs du coucou se développent plus rapidement et éclosent plus tôt que ceux de l'oiseau d'accueil. Dès son éclosion, le poussin coucou dupe ses parents adoptifs. Il éjecte les œufs d'accueil par aveuglement ce qui lui offre la monopolisation des soins de ses parents adoptifs et de la nourriture destinée à toute une couvée. Il incite ses parents adoptifs à suivre le rythme de son taux de croissance élevé par son appel de mendicité rapide. Qui imite l'appel des poussins d'accueil et aussi sa bouche ouverte qui construit un fort stimulus. En effet, il grandit plus vite au détriment des poussins d'accueil.

II.3.4.2 Le principe de la recherche coucou [14-15]

La recherche coucou est une métaheuristique très récente, proposé par Xin-She et Deb en 2009. En s'inspirant du comportement des coucous dans leur reproduction, Yang et Deb se sont basés sur trois principes pour proposer leur nouvelle métaheuristique

- Chaque coucou pond un seul œuf à la fois. Il le dépose dans un nid qu'il choisit aléatoirement.
- Les meilleurs nids qui incluent des œufs (solutions) de bonnes qualités vont être les élus qui construisent les membres de la nouvelle génération.
- Le nombre des nids hôtes valides est fixé. L'oiseau hôte peut détecte le coucou étranger avec une probabilité $P_a \in [0,1]$ Dans ce cas-là, l'oiseau hôte tranche entre écarte le coucou de son nid en lui éjecter hors nid ou abandonner son nid pour aller construire un autre dans une nouvelle position.

La probabilité P_a représente la fraction de N nids qui vont être remplacés par de nouveaux nids (avec de nouvelles solutions aléatoires dans de nouvelles positions dans l'espace de

recherche). La qualité d'un nid ou d'une solution est mesurée en fonction de la fonction fitness qui se varie d'un problème à un autre.

Afin de générer une nouvelle solution $x(t + 1)$ pour un coucou i , Yang et Deb ont intégré le vol de Lévy de la manière suivante :

$$x_i(t + 1) = x_i(t) + \alpha \oplus Lévy(\lambda) \quad (II.10)$$

Où $\alpha > 0$ est la taille du pas, elle est liée au problème traité.

II.3.4.3 Le vol de Lévy (Lévy flight)

Le vol de Lévy ou Lévy flight a été proposé par le mathématicien français Paul Pierre Lévy, un des fondateurs de la théorie moderne de probabilités. Depuis sa création, le vol de Lévy a donné des interprétations théoriques à plusieurs phénomènes physiques, chimiques, biologiques et naturels. En fait, le vol de Lévy permet de modéliser des marches aléatoires composées d'un grand nombre de pas où les transitions sont basées sur des probabilités. En terminologie mathématique, le vol de Lévy est une *marche aléatoire* (une formalisation mathématique d'une trajectoire composée d'un ensemble de pas aléatoires), dans laquelle la distance entre les pas a une *distribution probabilitaire* (une fonction qui représente la probabilité d'un nombre aléatoire de prendre une valeur donnée), à *queue-lourde* (dont les queues ne sont pas bornées de façon exponentielle).

La distribution de Lévy

$$Lévy \sim u = t^{-\lambda}, (1 < \lambda \leq 3) \quad (II.11)$$

La longueur de pas calculé par l'équation suivant :

$$S = \frac{u}{|v|^{1/\beta}} \quad (II.12)$$

Où u et v sont tiré par la distribution normal

$$u \sim N(0, \sigma_u^2), \quad v \sim N(0, \sigma_v^2) \quad (II.13)$$

Où

$$\sigma_u = \left\{ \frac{\Gamma(1+\beta)\sin(\frac{\pi\beta}{2})}{\Gamma[\frac{1+\beta}{2}]\beta 2^{(\beta-1)/2}} \right\}^{1/\beta}, \quad \sigma_v = 1 \quad (II.14)$$

II.3.4.4 Pseudo code de la recherche coucou (CS)

Algorithme II.3 : Algorithme CS

Entrée Fonction objective, la population

Sortie Solution optimal

Tant que le critère d'arrêt n'est pas atteint **faire**

 Générer une nouvelle solution x_j par Lévy flight

Pour $i = 1$ à N **faire**

 Calculer leur fitness

Si ($F(x_j) < F(x_i)$)

$x_i = x_j$

Fin Si

Fin Pour

 Modifier une fraction P_a de son contenu pour obtenir des nouvelles solutions s par Lévy flight

Pour $i = 1$ à N **faire**

 Calculer leur fitness

Si ($F(x_i) < F(x_{best})$)

$x_{best} = x_i$

$F(x_{best}) = F(x_i)$

Fin Si

Fin Pour

Fin tant que

II.4 Conclusion

Dans ce chapitre, les fondations nécessaires à la maîtrise de quelques techniques intelligentes sont établies. Après avoir introduit des concepts de base sur lesquelles reposent les systèmes d'inférence floue, la structure générale d'un contrôleur flou ainsi que ses différents composants ont été décrits.

Les métaheuristiques notamment la méthode d'optimisation par essaim particulaire (PSO), L'algorithme BAT et la recherche coucou (CS), sont des stratégies qui permettent de guider la recherche à une solution optimale. Grâce à la simplicité et la souplesse de leurs principes, ils peuvent être un outil d'optimisation et de conception des systèmes de contrôle de processus complexes dont la dynamique n'est toujours pas maîtrisable.

III.1 Introduction

Les métaheuristiques, particulièrement les algorithmes évolutionnaires tels que les algorithmes génétiques ont été largement utilisés pour la conception des contrôleurs flous voir par exemple [23]. Cependant, les algorithmes génétiques sont coûteux en temps de calcul, car ils gèrent simultanément plusieurs solutions comme ils peuvent être piégés dans des minima locaux [24]. Dans ce travail, nous avons appliqué trois algorithmes métaheuristiques tels que PSO, BAT, CS sur des contrôleurs flous pour objectif d'améliorer les résultats de commande d'un quadrotor. Ces algorithmes ont des avantages par rapport au cas des algorithmes génétiques.

La première partie de ce chapitre concerne le développement des contrôleurs flous pour la commande de l'attitude du quadrotor en premier lieu et en second lieu concerne la synthèse d'un contrôleur flou pour la stabilisation en altitude du quadrotor. La deuxième partie du chapitre est consacré à l'optimisation des sorties des contrôleurs flous par l'usage les algorithmes PSO, BAT, CS.

III.2 Commande floue d'un Quadrotor

La commande d'un Quadrotor peut être subdivisée en deux principales étapes. La première consiste en la stabilisation de l'attitude. La deuxième consiste en la stabilisation de l'altitude du Quadrotor. La stratégie de commande est basée principalement sur deux boucles (boucles interne et boucle externe), la boucle interne contient quatre lois de commande : commande de roulis u_2 , commande de tangage u_3 , commande de lacet u_4 et commande d'altitude u_1 . Dans ce qui suit ces des tâches seront détaillées.

III.2.1 Commande floue de l'attitude

La Commande d'attitude est la tâche la plus importante dans la commande du drone, l'objectif principal de cette tâche est la synthèse de lois de commande stabilisantes robustes en termes d'orientation du Quadrotor. Généralement les angles de roulis et de tangage sont forcés à zéro pour permettre un vol stationnaire. L'idée de l'application de la commande sur le Quadrotor en attitude est de fixer la première commande u_1 en une valeur constante ($mg \pm \delta U$) et de calculer les autres commandes (u_2, u_3, u_4). La figure III.1 montre le schéma Simulink des commandes réalisées:

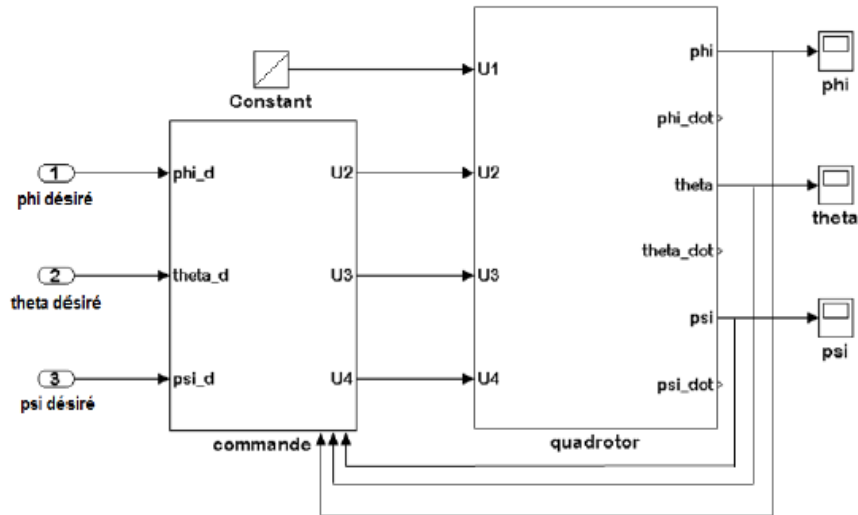


Figure III.1 : Commande d'attitude d'un quadrotor

III.2.2 Commande d'altitude

Le contrôleur de l'altitude dont nous avons se limité au déplacement du système el long de l'axe Z permet de garder le Quadrotor à une altitude désiré au-dessus de la terre.

III.2.4 Contrôleurs flous proposés pour commande d'un quadrotor

Nous avons considéré des contrôleurs flous de type Takagi Sugino d'ordre zéro. Chaque contrôleur est composé de deux entrées et une sortie, les entrées sont l'erreur en position angulaire et sa dérivée. Ces deux informations fournissent une description suffisamment détaillée sur la dynamique des états du système à commander. Dans ce cas, les règles floues associées à la précision et la vitesse de convergence sont formulées. La sortie consiste en les sommes et différences des vitesses à appliquer sur les moteurs du quadrotor. Les variables d'entrée / sortie sont définies par onze (11) valeurs floues dénommées : $\{P$ (Positive), Z (Zero), et N (Négative) $\}$ pour l'erreur et $\{P\dot{e}$ (Positive), $Z\dot{e}$ (Zéro), et $N\dot{e}$ (Négative) $\}$ pour la dérivée de l'erreur, $\{NG$ (Négatif Grand), NM (Négatif Moyenne), Z (Zéro), PM (Positif Moyenne), PG (Positif Grand) $\}$ pour la sortie. Les distributions des fonctions d'appartenances sur les univers des discours pour toutes les entrées sorties sont normalisées entre $[-1,1]$ pour l'erreur et les sorties. Pour la variation de l'erreur il est entre $[-3,3]$. Les fonctions d'appartenance utilisées ainsi que leurs distributions sur les univers de discours sont présentées dans les figures et suivantes :

- Les fonctions d'appartenances de l'entrée erreur sont :
- $\mu_N(e) = \text{trapzoid}(-1, -0.15, 0)$;
 $\mu_Z(e) = \text{triangle}(-0.15, 0, 0.15)$;
 $\mu_P(e) = \text{trapzoid}(0, 0.15, 1)$;
- Les fonctions d'appartenances de l'entrée dérivée de l'erreur sont :
 $\mu_N(de) = \text{trapzoid}(-3, -1.5, 0)$;
 $\mu_Z(de) = \text{tirangle}(-1.5, 0, 1.5)$;
 $\mu_P(de) = \text{trapzoid}(0, 1.5, 3)$;

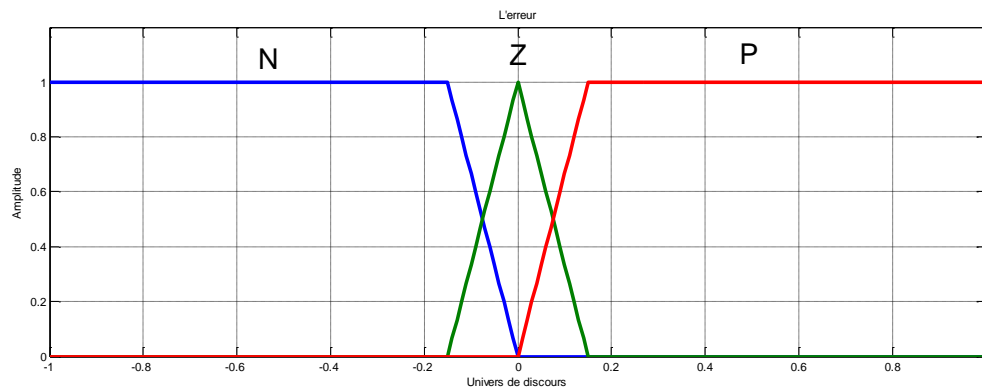


Figure III.2 : Les fonctions d'appartenances de L'erreur

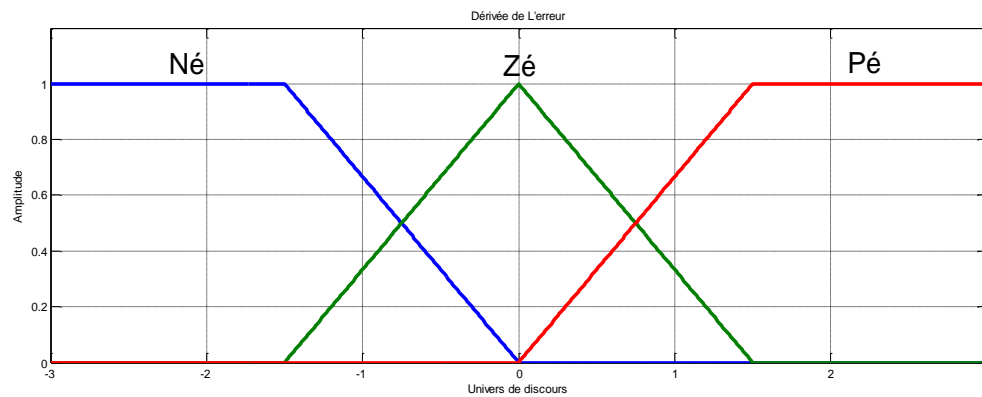


Figure III.3 : Les fonctions d'appartenances de la dérivée de L'erreur

Les fonctions d'appartenances de la sortie sont des singletons dont leurs valeurs sont

NG=-1

NM=-0.25

Z=0

PM=0.25

PG=1

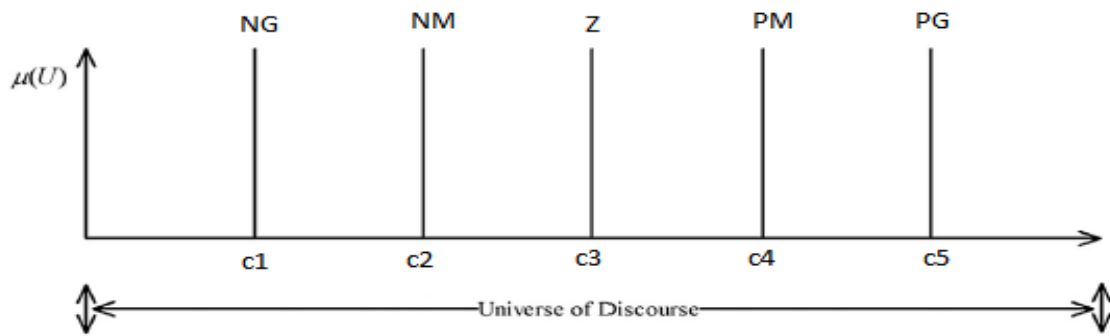


Figure III.4 : Les singletons de la sortie

La base des règles :

Notre choix de base des règles respecte les deux conditions principales des bases des règles la complétude et la consistance telles que :

- La Complétude : une base de règles d'un système flou est dite complète si, pour chaque vecteur d'entrée, il existe au moins une règle floue activée. Afin d'assurer cette propriété, les fonctions d'appartenance doivent couvrir toutes les plages possibles des variables d'entrée. L'utilisation de fonctions d'appartenance triangulaires régulièrement réparties respecte la propriété de complétude.
- La consistance : une base de règles d'un système flou est dite inconsistante, s'il existe deux règles floues ayant la même prémisse mais des conclusions différentes. La propriété de consistance permet d'éviter les contradictions dans une base de règles.

La table de règles elle est de forme anti-diagonale, et antisymétrique.

		L'erreur(e)		
		N	Z	P
Dérivée De L'erreur	<i>Né</i>	NG	NM	Z
	<i>Zé</i>	NM	Z	PM
	<i>Pé</i>	Z	PM	PG

Tableau III.1 : La base des règles du Contrôleur

- **Exemple1** : Si l'erreur est N (c'est-à-dire qu'on est au-dessus de la référence) et la dérivée de l'erreur est $N\dot{e}$ (c'est-à-dire que la vitesse est en train d'augmenter) alors la commande est NG (il faut diminuer les forces des quatre moteurs pour descendre).
- **Exemple2** : Si l'erreur est P (qu'on est au-dessous de la référence) et la dérivée de l'erreur est $P\dot{e}$ (la vitesse est entrain de diminuer) alors la commande est PG (il faut augmenter les forces davantage pour atteindre la référence).
- **Exemple3** : Si l'erreur est Z (la référence est atteinte) et la dérivée de l'erreur est Z (la vitesse est nulle) alors la commande est $Z\dot{e}$ (on maintient les forces constantes, la portance est égal au poids).

La méthode d'implication agrégation est celle de Zadeh *min-max*. le max dans cette méthode représente l'opérateur logique OU et le min représente l'opérateur ET. La conclusion dans chaque règle est introduite par le ALORS, elle permet de relier le facteur d'appartenance de la condition avec la fonction d'appartenance de la variable de sortie en utilisant le min. l'agrégation est définie généralement par l'opérateur OU et donc par conséquent il est remplacé par le max. pour la défuzzification sachant qu'on a un contrôleur flou de type Takagi-Sugeno d'ordre zéro. Etant donné que chaque règle possède une conclusion numérique, et de cette manière, le temps consommé par la procédure de défuzzification est considérablement réduit comparant le cas d'un contrôleur de type Mamdani. En fait, la sortie du contrôleur flou de type Takagi Sugino d'ordre zéro consiste en le calcul du barycentre dont elle est donnée par la relation suivante :

$$y(x) = \frac{\sum_{k=1}^N \mu_k(x) * f_k(x)}{\sum_{k=1}^N \mu_k(x)} \quad (\text{III.3})$$

III.3 optimisation métaheuristique des contrôleurs flous développés

La figure suivante illustre la stratégie d'optimisation adoptée, elle consiste à trouver la distribution la plus optimale possible dans la base de recherche des valeurs linguistiques des sorties des contrôleurs flous.

Le choix d'optimiser uniquement les sorties des contrôleurs (c1, c2, c3, c4, c5) est justifié par plusieurs raisons telles que :

- La modélisation de la dynamique de l'entrée par deux variables floues est supposé suffisante, en revanche l'usage des sorties de type singletons engendre des performances et une robustesse à améliorer.
- L'optimisation de la sortie permet d'avoir les meilleures performances possibles par rapport à la commande notamment en énergie.
- La minimisation du nombre de paramètres à optimiser est aussi une raison importante sachant que le temps de traitement est lié étroitement au matériel disponible pour effectuer l'optimisation.

Le schéma d'optimisation des sorties des contrôleurs est donné par la figure ci-dessous. Le principe de cette optimisation peut être résumé comme suit :

A chaque pas de simulation on calcule l'erreur et sa dérivée pour les injecter dans les contrôleurs, en premier itérations l'algorithme d'optimisation donne une distribution aléatoires pour les singletons de les sorties des contrôleurs flous, cinq singletons pour chaque contrôleur (c_1, c_2, c_3, c_4, c_5). Le contrôleur donne une loi de commande à partir de cette distribution, la loi de commande attaque directement le système (Quadrotor) pour prendre une sortie de système. D'après cette sortie on calcule la valeur numérique de fonction objective (critère à minimiser). La sortie de la fonction objectif utilisé par l'algorithme d'optimisation pour modifier et prendre une nouvelle distribution des sorties (c_1, c_2, c_3, c_4, c_5). On répète cette opération jusqu'à la condition d'arrête.

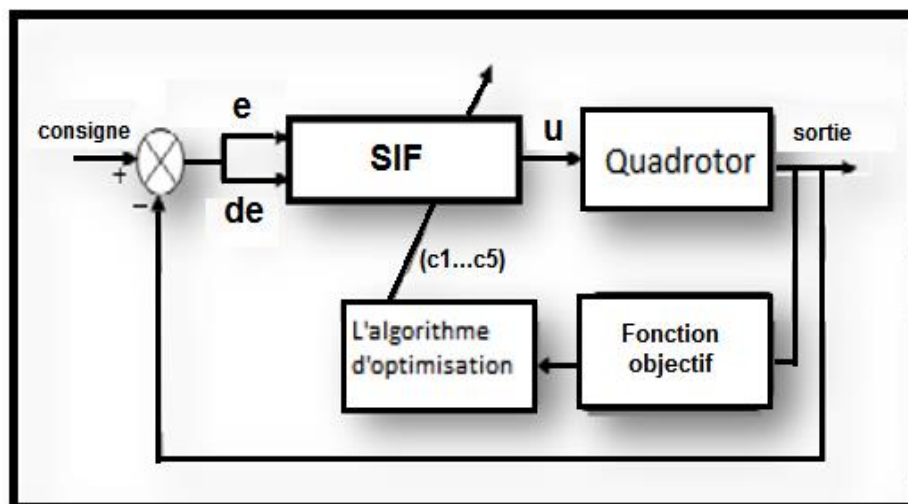


Figure III.5 : Structure d'optimisation

III.3.1 Critères d'optimisation utilisés

Pour dire qu'une commande est optimale si seulement si elle minimise un critère imposé. Le critère est un indice de performance qui est généralement représenté par une fonction monotone montante ou descendante. Dans ce travail, nous avons choisi trois critères utilisés fréquemment pour examiner les performances obtenues par les techniques d'optimisations. Ces critères sont : intégrale de l'erreur absolue (IAE), Intégral de l'erreur quadratique (ISE) et l'intégrale du temps multiplié par la valeur absolue de l'erreur (ITAE). L'usage de chaque critère est détaillé dans les sous sections suivantes.

III.3.1.1 Intégrale de l'erreur quadratique

L'intégrale de l'erreur quadratique (Integral Squared Error ISE en Anglais) permet de visualiser les erreurs obtenues en appliquant transitoire dans une application de commande. L'usage de ce critère comme fonction fitness dans une application d'optimisation méthaheuristique permet de minimiser l'erreur en régime transition et à la fois examiner les méthodes d'optimisation utilisées par rapport à cet indice de performance. Mathématiquement ce critère standard est donné par l'équation suivante :

$$ISE = \int_0^T e^2 dt \quad (III.4)$$

III.3.1.2 Intégrale du temps multiplié par la valeur absolue de l'erreur

L'intégrale du temps multiplié par la valeur absolue de l'erreur (Integral Time-weighted Absolute Error ITAE en Anglais), consiste à calculé la somme de erreur instantané multiplié par le temps de l'instant. L'usage de ce critère permet de quantifier les erreurs en régime permanent, car le temps dans ce cas augmente en fut et à mesure et si l'erreur n'égale pas à zéro alors la valeur de ce critère augmente aussi. L'usage de l'ITAE dans un algorithme d'optimisation méthaheuristique permet de minimiser l'erreur en régime transitoire. Et dans notre cas comparer les résultats des trois algorithmes PSO, CouCou et Bat. Mathématiquement ITAE est donnée comme suit :

$$ITAE = \int_0^T (t * |e|) dt \quad (III.5)$$

III.3.1.3 Intégrale de l'erreur absolue

L'intégral de l'erreur absolue (Integral Absolute Error IAE en Anglais), ce critère prend uniquement la somme des valeurs absolue de l'erreur entre les valeurs mesurées et les valeurs désirées au cours de l'exécution d'une application. Ce critère permet de quantifier toutes les erreurs obtenues en régimes transitoire et en régime permanent dans une application de commande. L'usage de l'IAE comme fonction fitness permet de favoriser la minimisation de l'erreur en générale. On peut dire que ce critère est au milieu entre l'ISE et l'ITAE. L'équation mathématique de l'IAE est donnée :

$$IAE = \int_0^T |e| dt \quad (III.6)$$

III.3.2 optimisation méthaheuristiques des controleur flou

Pour appliquer algorithmes méthaheuristique tels que : optimisation par essaim particulaire PSO, l'algorithme BAT et la recherche coucou CS pour l'optimisation des contrôleurs flous. Nous devons tout d'abord définir les entrées et les sorties de ces algorithmes. L'entrée est unique pour tous les cas elle consiste en le critère à minimiser. Dans ce travail nous avons testé les trois critères présentés dans la section précédente tels que : IAE, ISE et ITAE. La sortie est la distribution des singletons de chaque contrôleur flou qui corresponde au réglage de l'un des angles ou bien le réglage de l'altitude.

Le choix des paramètres des algorithmes méthaheuristiques est une tâche primordiale pour l'amélioration de la performance de l'algorithme. Cependant, le choix exige plusieurs expériences parce que chaque problème exige un réglage de paramètres selon la complexité du problème traité et les moyens de calcul disponible. Dans les cas que nous avons traités, nous avons choisi la plupart des paramètres d'une façon arbitraire. Tous les algorithmes démarrent avec des distributions aléatoires des fonctions de chaque sortie.

III.3.2.1 PSO appliqué pour l'optimisation de la distribution des FAs des sorties

Le choix des paramètres de l'algorithme PSO est imposé de façon arbitraire dont le nombre de particules doit être fixé entre 10 et 40 particules. Réellement 10 particules suffisent pour résoudre la plupart des problèmes [9]. Le changement de la distribution des singletons est obtenu à chaque itération en utilisant les équations (II.3) et (II.4). Les coefficients d'accélération C1 et C2, généralement sont égaux et de valeurs entre 0 et 2. Afin de réaliser une accélération maximale. R1 et R2 sont des valeurs aléatoires générées à chaque itération, ils représentent une distribution uniforme. Le nombre d'itération est une information importante pour le fonctionnement du PSO. Le tableau suivant résume les paramètres utilisés pour l'optimisation de la distribution des singletons.

PSO	
Paramètre	Valeur
Nombre de particule	10
Coefficient d'accélération pour le composant cognitif C1	2
Coefficient d'accélération pour le composant social C2	2
R1, R2	Aléatoire
Dimensions de particule	Selon le problème, (5) pour chaque contrôleur
Itérations	20

Tableau III.2 : Paramètres de PSO

Le pseudocode ci-dessous résume l'application de PSO pour l'optimisation de des distributions des singletons.

Algorithme II.1 : Algorithme OEP (PSO)

Entrée Fonction objective (IAE) , Nombre de particules

Sortie $IAE((c1, c2, c3, c4, c5)_{best})$ Solution optimal

Initialisation : Générer aléatoirement $:x^i, v^i, R_1^i, R_2^i$

Pour chaque particule $i, (c1, c2, c3, c4, c5)_{best}^i = (c1, c2, c3, c4, c5)^i$

Tant que le critère d'arrêt n'est pas atteint **faire**

Pour $i = 1$ à N **faire**

Modification de la position et la vitesse de chaque particule par (II.3)et (II.4)

Evaluation des positions

Si $IAE((c1, c2, c3, c4, c5)^i) < IAE((c1, c2, c3, c4, c5)_{best}^i)$

$$(c1, c2, c3, c4, c5)_{best}^i = (c1, c2, c3, c4, c5)^i$$

Fin Si

Si $IAE((c1, c2, c3, c4, c5)_{best}^i) < IAE((c1, c2, c3, c4, c5)_{best})$

$$(c1, c2, c3, c4, c5)_{best} = (c1, c2, c3, c4, c5)_{best}^i$$

$$IAE((c1, c2, c3, c4, c5)_{best}) = IAE((c1, c2, c3, c4, c5)_{best}^i)$$

Fin Si

Fin Pour

Fin tant que

III.3.2.2 BAT appliqué pour l'optimisation de la distribution des FAs des sorties

Plusieurs paramètres doivent être donnés à l'algorithme BAT pour qu'il fonctionne. Tel que f_{min} et f_{max} , l'amplitude de démarrage, la fréquence de démarrage est entre (0 et 1). Et un nombre de BAT et d'itération. L'algorithme BAT se déroule selon les équations (II.6) et (II.7). Le tableau ci-dessous résume les paramètres utilisés :

BAT	
Paramètre	Valeur
Nombre de BAT	10
Valeur initial de l'amplitude $A(0)$	0.9
Valeur initial de la fréquence $R(0)$	0.9
Les coefficients de l'amplitude et de la fréquence $\gamma = \sigma$	0.9
Borne inférieur de la fréquence f_{min}	0
Borne supérieur de la fréquence f_{max}	0.5
Coefficient de bornes de la fréquence	Aléatoire
Dimensions de particule	Selon le problème, (5) pour chaque contrôleur
Itérations	20

Tableau III.3 : Paramètres de BAT

le choix de $f_{min}=0$ et $f_{max}=0.5$ est dicté par le fait que si l'intervalle entre f_{min} et f_{max} est grand le changement au niveau de la distribution est grand a chaque optimisation ce qui va influencer les résultats de commande du quadrotor. L'amplitude de démarrage $A(0)$ décroît à chaque itération il est choisi d'une façon standard égale à 0.9. La fréquence R doit avoir une valeur initiale dont dans la majorité des cas égale à 0.9. La (Nombre nous avons choisi une population minimale égale à 10 afin d'éviter la lourdeur de traitement. Le pseudocode ci-dessous résume l'application de BAT pour l'optimisation de des distributions des singletons.

Algorithme II.2 : Algorithme BAT

Entrée Fonction objective(IAE), Nombre de bats

Sortie $IAE((c1, c2, c3, c4, c5)_{best})$ Solution optimal

Initialisation : Générer aléatoirement $(c1..c5)_i, v_i, \beta_i$

Initialiser $(c1 \dots c5)_{best} = (c1..c5)_i$ telque $IAE((c1..c5)_i) = \min(IAE((c1..c5)_{i..N}))$

Tant que le critère d'arrêt n'est pas atteint **faire**

Pour $i = 1$ à N **faire**

Modification de la position et la vitesse de chaque bat par (II.6) et (II.7)

Evaluation des positions

Si ($\text{rand} > r$)

$(c1..c5)_i = (c1 \dots c5)_{best} + \text{rand}$

Fin Si

Evaluer $IAE((c1..c5)_i)$

Si $IAE((c1..c5)_i) < IAE((c1..c5)_{best})$ & ($\text{rand} < A$)

$(c1..c5)_{best} = (c1..c5)_i$

$IAE((c1..c5)_{best}) = IAE(c1..c5)_i$

Fin Si

Fin Pour

 accroître la fréquence de l'onde et diminuer l'amplitude par (II.8) et (II.9)

Fin tant que

III.3.3.3 CS appliqué pour l'optimisation de la distribution des FAs des sorties

Le choix des paramètres de l'algorithme CS est imposé de façon arbitraire dont le nombre de particules doit être fixé entre 10 et 40 particules. Réellement 10 particules suffisent pour résoudre la plupart des problèmes. Le changement de la distribution des singletons est obtenu à chaque itération en utilisant le vol de Lévy. Les paramètres utilisés sont résumés dans la table ci-dessous :

CS	
Paramètre	Valeur
Nombre de particule	10
Probabilité P_a	0.25
Coefficient de la distribution de vol de lèvy β	1.5
Dimensions de particule	Selon le problème, (5) pour chaque contrôleur
Itérations	20

Tableau III.4 : Paramètres de CS

Nous avons pris 10 comme nombre de particules qu'est le cas le plus simple, le coefficient de la distribution de vol de lèvy est toujours entre $0 < \beta < 2$, c'est pour cette raison que nous avons utilisé une valeur vaut $3/2$. Le pseudocode ci-dessous résume l'application de l'algorithme CS pour l'optimisation de des distributions des singletons.

Algorithme II.3 : Algorithme CS

Entrée Fonction objective IAE , la taille de population

Sortie $IAE((c1, c2, c3, c4, c5)_{best})$ Solution optimal

Tant que le critère d'arrêt n'est pas atteint **faire**

Générer $(c1, c2, c3, c4, c5)$ par Lévy flight

Pour $i = 1$ à N **faire**

Calculer $IAE((c1..c5)_i)$

Si $IAE((c1..c5)_i) < IAE((c1..c5)_{best})$

$(c1..c5)_{best} = (c1..c5)_i$

Fin Si

Fin Pour

Modifier une fraction P_a de son contenu pour obtenir des nouvelles solutions s par Lévy flight

Pour $i = 1$ à N **faire**

Calculer $IAE((c1..c5)_i)$

Si $IAE((c1..c5)_i) < IAE((c1..c5)_{best})$

$(c1..c5)_{best} = (c1..c5)_i$

$$IAE((c1..c5)_{best} = IAE((c1..c5)_i$$

Fin Si

Fin Pour

Fin tant que

III.4 Conclusion

Dans ce chapitre, nous avons proposé une méthodologie d'optimisation d'une loi de commande simple, efficace, stable et robuste pour une classe de systèmes non linéaires. La démarche proposée dans ce travail, repose sur la synthèse des contrôleurs flous pour la commande de l'attitude et de l'altitude et puis leurs optimisations par trois algorithmes metaheuristiques tels que : PSO, BAT et CS.

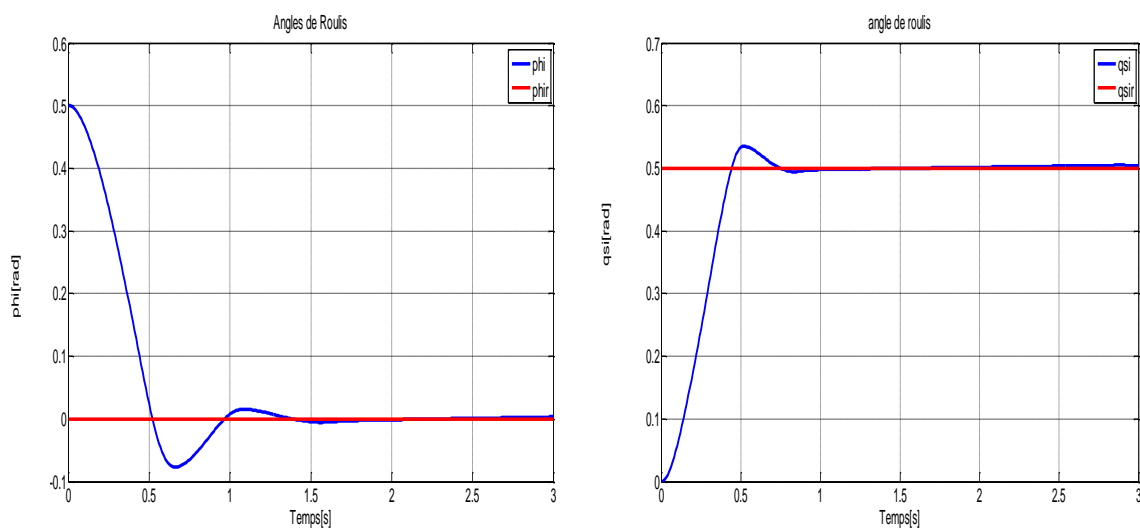
IV.1 Introduction

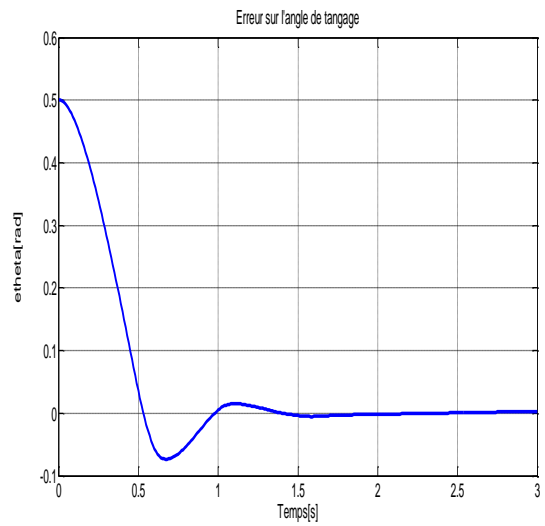
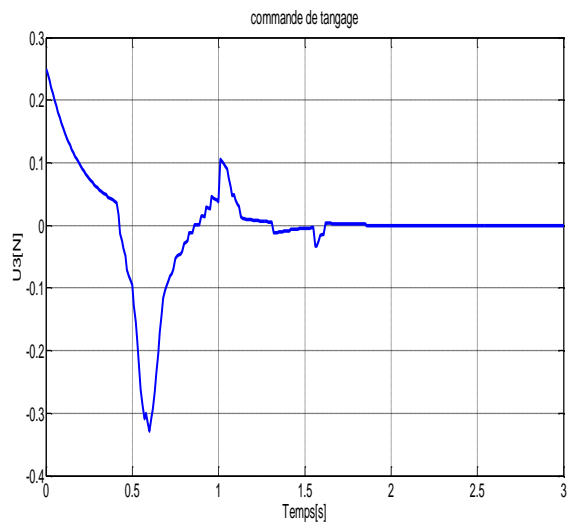
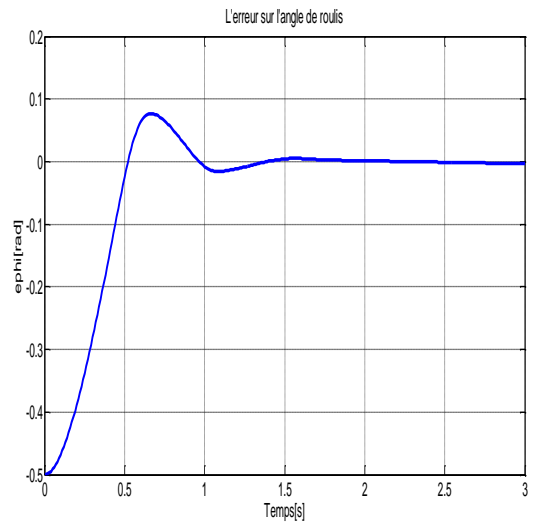
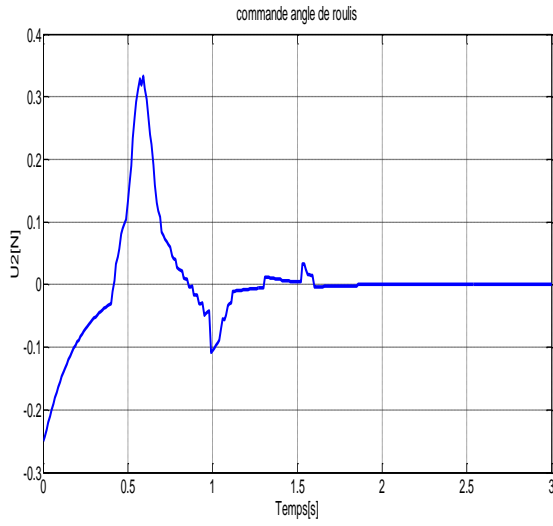
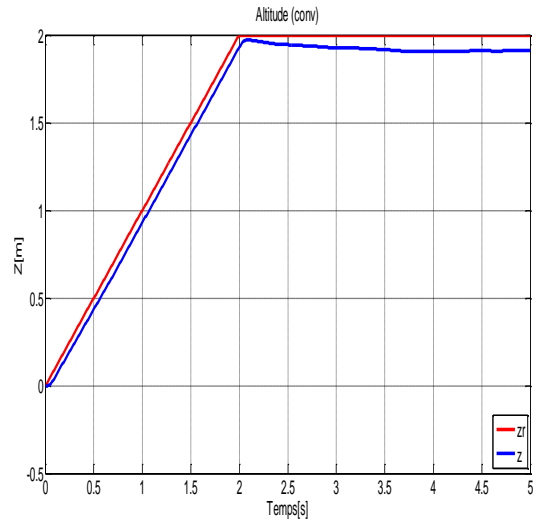
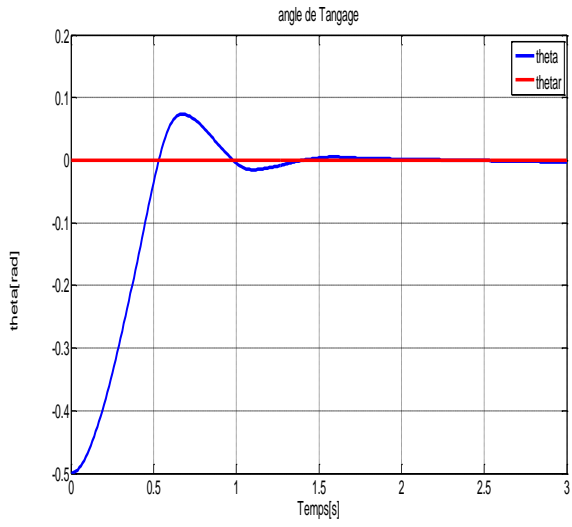
Ce chapitre présente les résultats de simulation issus de l'application des algorithmes métaheuristiques proposés précédemment appliqués pour l'optimisation de la distribution des fonctions d'appartenance des sorties des contrôleurs flous. Ces contrôleurs optimisés sont utilisés par la suite pour la commande d'un quadrotor. Les simulations ont été effectuées en d'attitude ainsi qu'en d'altitude par les trois algorithmes métaheuristiques, optimisation par essaim des particules (PSO), algorithme BAT et la recherche de cuckoo (CS) avec l'usage de différentes fonctions fitness. Les résultats de simulation ont été obtenus en utilisant logiciel MATLAB.

IV.2.1 Résultats de simulation de contrôle Flou de l'attitude sans optimisation

En attitude les angles de départ pour l'angles de roulis, tangage et de lacet sont 0.5 radians, 0.5 radians et 0 radians, respectivement. Les angles désirés sont 0 radians pour le roulis et le tangage, et 0.5 pour le lacet. La durée de simulation est de 3 secondes avec un pas d'échantillonnage de 0.01. En altitude, l'altitude de départ est 0 mètre et l'altitude désirés est 2 m, la durée de simulation est 5 secondes avec un pas de 0.01s.

Les résultats ci-dessous représentent les réponses et les commandes obtenues par l'application des contrôleurs flous sans optimisation sur le quadrotor en attitude et en altitude.





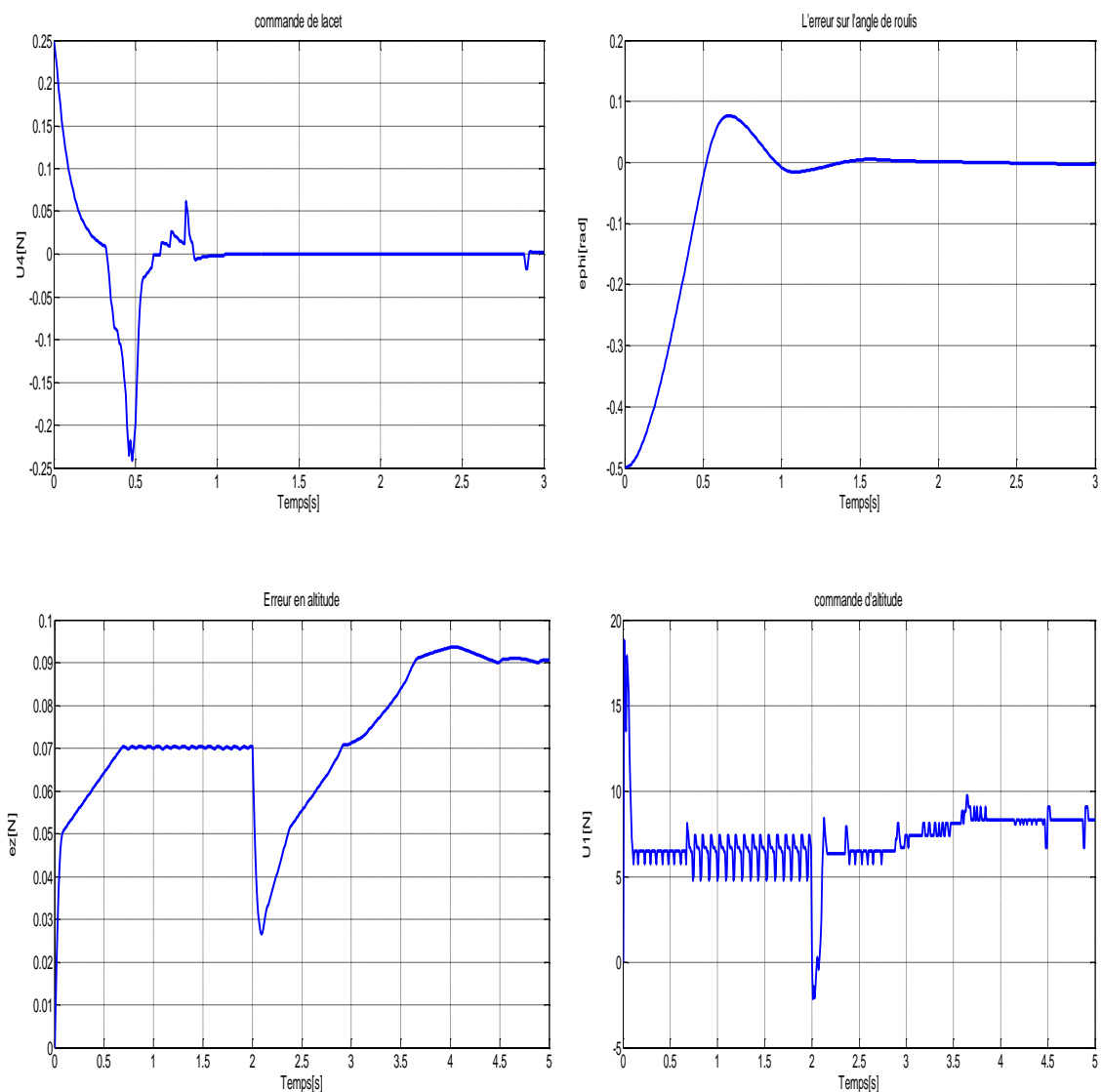


Figure IV.1 : Résultats obtenus par flou conventionnel.

Le tableau ci-dessous représente les valeurs des critères obtenues après application

Résultats de flou conventionnel			
critère	ISE	IAE	ITAE
$\phi + \theta + \psi$	0.1691	0.5138	0.1453
z	0.0272	0.3594	0.9809

Tableau IV.1 : Valeurs des critères par flou conventionnel

Interprétation

A travers des résultats ci-dessus on peut remarquer que : l'application des contrôleurs flous pour le contrôle de la position et de l'orientation du quadrotor donne des résultats acceptables mais pas optimaux, on remarque un grand dépassement au niveau des réponses de l'attitude et une erreur statique considérable en altitude. L'optimisation va permettre d'améliorer les erreurs en régime transitoire.

IV.2.2 Résultats optimisés en attitude

Dans ce passage on présente les résultats de simulation de l'application des algorithmes PSO, BAT et CS sur les contrôleurs flous pour la stabilisation de l'attitude de quadrotor à savoir l'angle de roulis (ϕ), de tangage (θ) et de lacet (ψ)

- **Condition de simulation**

Les angles de départ pour l'angles de roulis, tangage et de lacet sont 0.5 radian, 0.5 radian et 0 radian respectivement, les angles désirés sont 0 radian pour le roulis et le tangage, et 0.5 radian pour le lacet, la durée de simulation est de 3 second avec un pas d'échantillonnage de 0.01 S.

En première partie de ce passage on va présenter les distributions obtenues par minimisation des critères que nous avons les expliqués en chapitre 3 (IAE, ISE, ITAE), en utilisant les trois algorithmes (PSO, CS, BAT). Pour chaque critère on présente les statistiques de simulation avec les courbes de diminution des critères. En deuxième partie on va présenter les courbes des réponses obtenues par l'application de ces distributions sur les contrôleurs flous.

- **Résultats obtenus par la minimisation de l'intégrale de la valeur absolue de l'erreur :**

Les tableaux (IV.2-2,3 et 4) représentent les distributions des singletons obtenus après application de PSO, de CS et de BAT avec la prise en considération de la fonction objectif l'intégrale de la valeur absolue de l'erreur. Nous avons effectué cinq tests défirants des algorithmes PSO, de CS et de BAT pour objectif de garder le meilleur résultat obtenu sachant que ces algorithmes utilisent des valeurs aléatoires pour démarrer la simulation. L'application de ces cinq expériences pour la commande floue de quadrotor a abouti aux résultats quantitatifs du critère IAE qui sont donnés par le meilleur, le moyen et le pire résultat. Le

tableau (IV.2-1) ci-dessous résume ces résultats ainsi que le temps de traitement pour chaque algorithme.

IAE				
Algorithme	Meilleur	Pire	Moyenne	Temps de calcul moyenne
PSO	0.2563	0.4443	0.3550	322.464 s
CS	0.4087	0.4988	0.4451	712.588 s
BAT	0.2253	0.3877	0.3018	386.380 s

1 : statistiques du simulation

Les trois tableaux ci-dessous donnent les distributions des fonctions d'appartenances.

IAE Angles (ϕ)					
Algorithme	<i>NG</i>	<i>NM</i>	<i>Z</i>	<i>PM</i>	<i>PG</i>
PSO	-7.1304	-3.9591	0.0446	3.2595	6.7896
CS	-1.6129	-0.5237	-0.0030	0.7077	0.7342
BAT	-9.9318	-9.8608	0.5375	0.8404	9.1098

2 : les angles de roulis

IAE Angles (θ)					
Algorithme	<i>NG</i>	<i>NM</i>	<i>Z</i>	<i>PM</i>	<i>PG</i>
PSO	-5.3872	-2.6239	0.0046	2.0163	2.7943
CS	-1.3273	-0.4326	0.0641	0.5473	0.9252
BAT	-8.2474	-3.9849	0.0628	7.3107	9.2441

3 : les angles de tangage

IAE Angles (ψ)					
Algorithme	<i>NG</i>	<i>NM</i>	<i>Z</i>	<i>PM</i>	<i>PG</i>
PSO	-2.0471	-1.1021	-0.0924	5.1012	5.7984
CS	-0.8061	-0.5946	-0.0163	0.7469	1.0283
BAT	-4.2472	-0.8443	-0.5332	5.6435	12.0253

4 : les angles de lacet

Tableau IV.2 : Résultats obtenus par minimisation le critère IAE.

La figure ci-dessous montre la meilleure évolution du critère IAE avec l'usage des trois algorithmes (PSO, CS et BAT) pour un nombre d'itérations égale à 20.

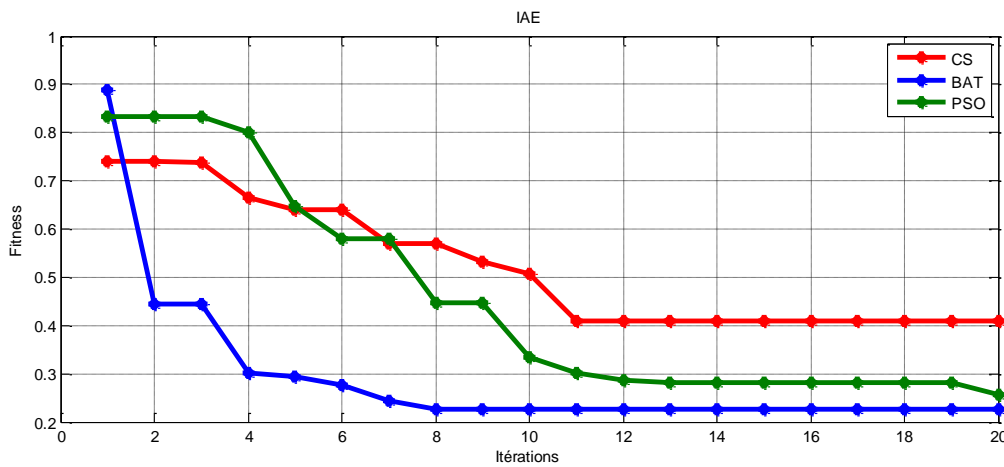


Figure IV.2 : Diminution de critère IAE

Interprétation : on remarque que l'algorithme BAT converge à partir de la 8ème itération à son minimum. Le pire des résultats en termes de rapidité de convergence à son minimum est celui de PSO. En revanche en termes de minimisation de la fonction fitness le PSO est nettement meilleur que le CS, mais le BAT a donné une valeur minimale de la fonction fitness par rapport aux deux autres algorithmes. En termes de temps de traitement le PSO et le BAT ont consommés pratiquement le même temps avec une légère supériorité du PSO par rapport au BAT. Cependant, le CS a consommé un temps très important comparé aux deux autres algorithmes.

- **Résultats obtenus par la minimisation de ISE :**

Les tableaux (IV.3-2, 3 et 4) représentent les distributions des singletons obtenus après application de PSO, de CS et de BAT avec la prise en considération de la fonction objectif qu'est ici ISE. Comme dans le premier cas, nous avons effectué cinq tests défirants des algorithmes PSO, de CS et de BAT pour objectif de garder le meilleur résultats. L'application de ces cinq expériences pour la commande floue de quadrotor a aboutie aux résultats quantitatifs du critère IAE qui sont donnés par le meilleur, le moyen et le pire résultat. Le tableau (IV.3-1) ci-dessous résume ces résultats ainsi que le temps de traitement pour chaque algorithme.

ISE				
Algorithme	Meilleur	Pire	moyenne	Temps de calcul moyenne
PSO	0.0993	0.1207	0.1101	317.711 s
CS	0.1135	0.1241	0.1195	604.420 s
BAT	0.0656	0.0982	0.0833	357.674 s

1 : statistiques du simulation

Les trois tableaux ci-dessous donnent les distributions des fonctions d'appartenances obtenues après optimisation.

ISE Angles (ϕ)					
Algorithme	<i>NG</i>	<i>NM</i>	<i>Z</i>	<i>PM</i>	<i>PG</i>
PSO	-1.9096	-1.7980	-0.0314	1.2675	3.3400
CS	-1.7541	-0.9719	0.0020	0.1680	1.7713
BAT	-25.7885	-10.2600	0.2891	4.3108	17.2902

2 : les angles de roulis

ISE Angles (θ)					
Algorithme	<i>NG</i>	<i>NM</i>	<i>Z</i>	<i>PM</i>	<i>PG</i>
PSO	-1.1936	-0.6818	-0.2765	2.2233	2.7796
CS	-1.5115	-0.3029	-0.1257	1.0218	1.3366
BAT	-8.1312	-3.3771	-2.1762	11.6309	11.8729

3 : les angles de tangage

ISE Angles (ψ)					
Algorithme	<i>NG</i>	<i>NM</i>	<i>Z</i>	<i>PM</i>	<i>PG</i>
PSO	-0.7334	-0.5448	0.1658	0.6258	1.0392
CS	-0.4026	-0.2079	-0.1936	2.0567	2.1841
BAT	-14.7692	-0.4605	-0.3203	4.7718	14.1775

4 : les angles de lacet

Tableau IV.3 : Résultats obtenus par minimisation le critère ISE

La figure ci-dessous montre la meilleure évolution du critère ISE avec l'usage des trois algorithmes (PSO, CS et BAT) pour un nombre d'itérations égale à 20.

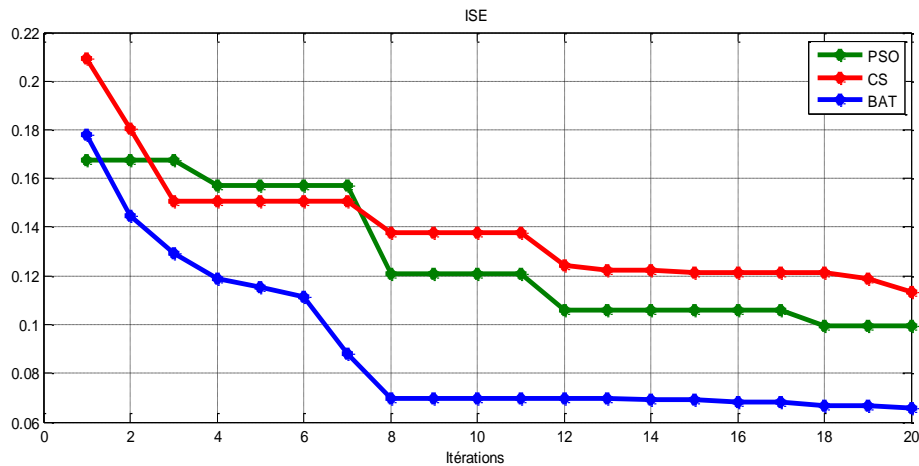


Figure IV.3 : Diminution de critère ISE

Interprétation : l'algorithme BAT converge à partir de la 8ème itération à son minimum. Le pire des résultats en termes de rapidité de convergence à son minimum est celui de CS. Le minimum des valeurs des fitness est obtenu par le BAT le pire est celui de CS. En termes de temps de traitement le PSO et le BAT ont consommés pratiquement le même temps avec une légère supériorité du PSO par rapport au BAT. Cependant, le CS a consommé un temps très important comparé aux deux autres algorithmes.

- **Résultats obtenus par la minimisation de ITAE :**

Maintenant la fonction fitness utilisée pour trouver les distribution optimale des singletons est l' ITAE, les résultats obtenus sont résumés dans les tableaux (IV.4-2, 3 et 4) après application de PSO, de CS et de BAT. Comme dans le premier et le deuxième cas nous avons effectué cinq tests défirants des algorithmes PSO, de CS et de BAT pour objectif de garder le meilleur résultats. L'application de ces cinq expériences pour la commande floue de quadrotor a abouti aux résultats quantitatifs du critère IAE qui sont donnés pour le meilleur, le moyen et le pire résultat. Le tableau (IV.4-1) ci-dessous résume ces résultats ainsi que le temps de traitement pour chaque algorithme.

ITAE				
Algorithme	Meilleur	pire	moyenne	Temps de calcul moyenne
PSO	0.0947	0.2457	0.1345	372.041 s
CS	0.1448	0.2616	0.2110	648.453 s
BAT	0.0914	0.1906	0.1911	340.786 s

1 : statistiques du simulation

Les trois tableaux ci-dessous donnent les distributions des fonctions d'appartenances.

ITAE Angles (ϕ)					
Algorithme	<i>NG</i>	<i>NM</i>	<i>Z</i>	<i>PM</i>	<i>PG</i>
PSO	-6.9410	-1.4799	-0.0305	0.4749	14.1483
CS	-2.3947	-0.8057	0.0343	0.4212	0.9982
BAT	-1.8953	-0.8663	-0.0058	2.2885	4.4938

2 : les angles de roulis

ITAE Angles (θ)					
Algorithme	<i>NG</i>	<i>NM</i>	<i>Z</i>	<i>PM</i>	<i>PG</i>
PSO	-4.6671	-0.7540	0.0032	1.2759	1.3083
CS	-0.5621	-0.2879	0.0046	0.3424	0.8822
BAT	-5.0435	-2.1560	0.1582	1.1345	4.3485

3 : les angles de tangage

ITAE Angles (ψ)					
Algorithme	<i>NG</i>	<i>NM</i>	<i>Z</i>	<i>PM</i>	<i>PG</i>
PSO	-1.5904	-1.5350	0.0494	0.9144	3.4818
CS	-2.1994	-0.2271	0.0154	0.3358	4.4491
BAT	-4.3186	-3.6961	0	2.0908	2.8464

4 : les angles de lacet

Tableau IV.4 : Résultats obtenus par minimisation le critère ITAE.

La figure ci-dessous montre la meilleure évolution du critère ITAE avec l'usage des trois algorithmes (PSO, CS et BAT) pour un nombre d'itérations égale à 20.

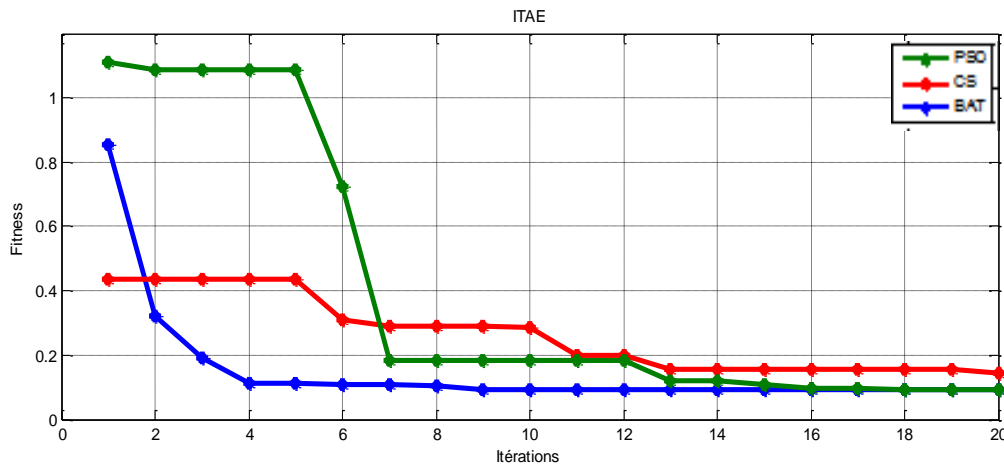


Figure IV.4 : Diminution de critère ITAE

Interprétation : l'algorithme BAT converge à partir de la 4ème itération à son minimum. En termes de rapidité de convergence à son minimum CS est le mauvais. Le minimum des valeurs des fitness est obtenu par le BAT le pire est celui de PSO. En termes de temps de traitement le PSO et le BAT ont consommés pratiquement le même temps avec une légère supériorité au PSO par rapport au BAT. Cependant, le CS a consommé un temps très important comparé aux deux autres algorithmes.

Examen de meilleur algorithme entre le PSO, le BAT et le CS

Afin de fixer les idées concernant le meilleur algorithme d'optimisation métaheuristique à appliquer pour l'optimisation des singletons des sorties des contrôleurs flous dans le cas de la commande de l'attitude d'un quadrotor, nous avons effectué deux opérations simples :

- La représentation de l'évolution de la somme des fonctions fitness utilisées (IAE, ITAI et ISE) en fonction des itérations (la figure IV.5).
- La moyenne de la somme de temps de traitement pour toutes les fonctions fitness dont les résultats sont donnés dans la table (IV.5).

Algorithme	IAE	ISE	ITAE	Moyenne
PSO	317.711 s	372.041 s	322.464 s	337.405 s
CS	604.420 s	648.453 s	712.588 s	655.153 s
BAT	357.674 s	340.786 s	386.380 s	361.613 s

Tableau IV.5 : Moyenne de la somme de temps de traitement

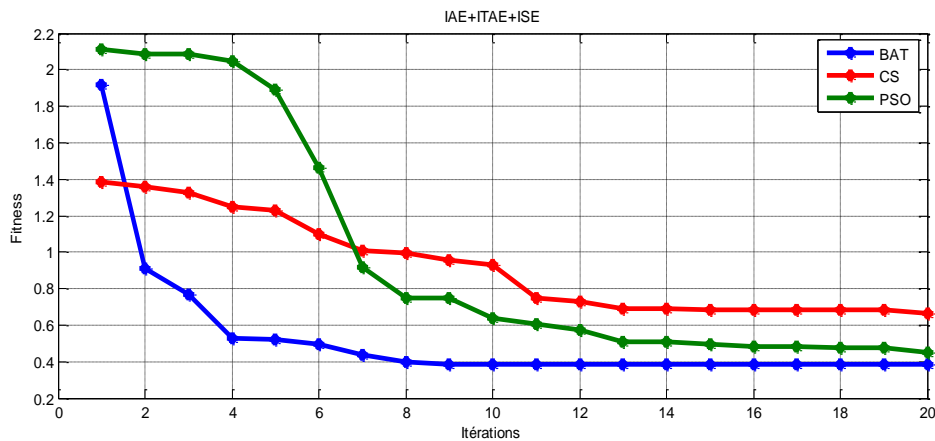


Figure IV.5 : Somme de diminution des critères IAE, ISE et ITAE

Interprétation concluante sur le meilleur algorithme entre le PSO, le BAT et le CS

A travers les résultats donnés par la figure et les tableaux ci-dessus on peut remarquer que : L'algorithme BAT a un temps de convergence avec un temps de traitement relativement rapide. Le BAT a pu aussi donner le minimum en termes de valeur de la somme des fitness. Le PSO utilise un temps de traitement relativement bon et une valeur moyenne en termes de fitness. Le pire des trois algorithmes est le CS ce dernier consomme un temps de traitement élevé, pratiquement c'est le temps pour tester les deux autres algorithmes, sa convergence est lente en plus le minimum de la fonction fitness est le plus élevé. Jusqu'à ici, vu ces résultats on peut dire que le BAT est le meilleur algorithme.

IV.2.3 Résultat de simulation en altitude

Dans ce passage on présente les résultats de simulation de l'application des algorithmes optimisations par PSO, BAT et CS sur le quadrotor en altitude. On prend l'état initial nul $z(0)=0$, la consigne est $z_d = 2$, la durée de simulation est de 5 secondes avec un pas d'échantillonnage de 0.01s. Il faut souligner que les angles sont laissés libres sans aucune commande. Nous avons effectué trois tests défirants sur les algorithmes afin de garder le milleur résultat obtenu. L'application de ces trois expériences pour la commande floue de quadrotor a abouti aux milleurs résultats quantitatifs des critères qui sont résumés dans la table ci-dessous.

L'algorithme	IAE	ISE	ITAE
PSO	0.0316	$4.7 * 10^{-4}$	0.0818
CS	0.0443	$5.6 * 10^{-4}$	0.0871
BAT	0.0274	$2.5 * 10^{-4}$	0.0750

1 : statistiques de simulation

Les trois tableaux ci-dessous donnent les distributions des fonctions d'appartenances dans le cas de l'optimisation des contrôleurs de l'altitude.

IAE Altitude (z)					
Algorithme	<i>NG</i>	<i>NM</i>	<i>Z</i>	<i>PM</i>	<i>PG</i>
PSO	-0.8591	-0.5153	0.0625	1.0018	1.1295
CS	-3.6640	-0.5295	0.0435	1.1895	1.5889
BAT	-9.4921	-3.5812	-0.1014	0.9278	5.3519

2 : Minimisation d'IAE

ISE Altitude (z)					
Algorithme	<i>NG</i>	<i>NM</i>	<i>Z</i>	<i>PM</i>	<i>PG</i>
PSO	-2.4196	-0.1082	0.0079	1.2701	1.7786
CS	-3.8020	-0.4893	0.0623	0.5381	0.6242
BAT	-6.2741	-3.0895	0.0145	0.5375	7.6523

3 : Minimisation d'ISE

ITAE Altitude (z)					
Algorithme	<i>NG</i>	<i>NM</i>	<i>Z</i>	<i>PM</i>	<i>PG</i>
PSO	-0.5175	-0.2775	0.0633	0.6513	2.3551
CS	-9.0232	-2.2449	-0.0149	0.4894	7.0971
BAT	-2.9235	-2.5001	-0.0307	1.9062	5.4611

4 : Minimisation d'ITAE

Tableau IV.6 : Résultats obtenus par minimisation le critères en altitude

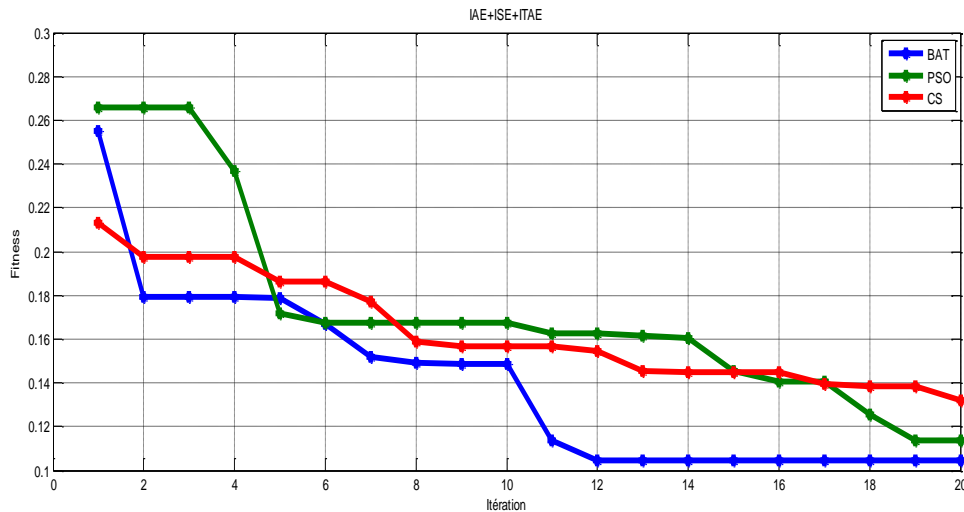


Figure IV.6 : Somme de diminution des critères ISE, IAE et ITAE en altitude

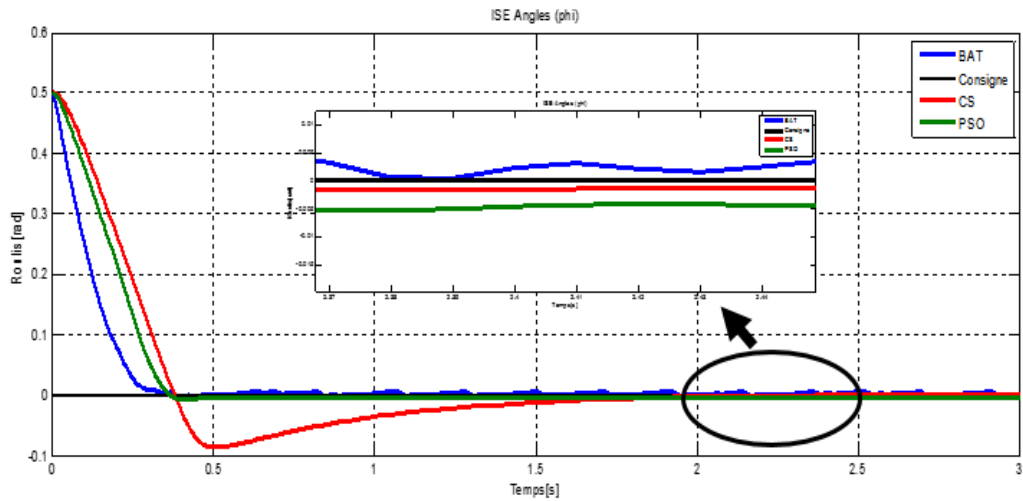
Algorithme	IAE	ISE	ITAE	Moyenne
PSO	523.806 s	464.693 s	418.969 s	337.405 s
CS	1491.084 s	1299.00 s	959.624 s	655.153 s
BAT	590.107 s	386.789 s	565.573 s	361.613 s

Tableau IV.7 : Temps de simulation en altitude

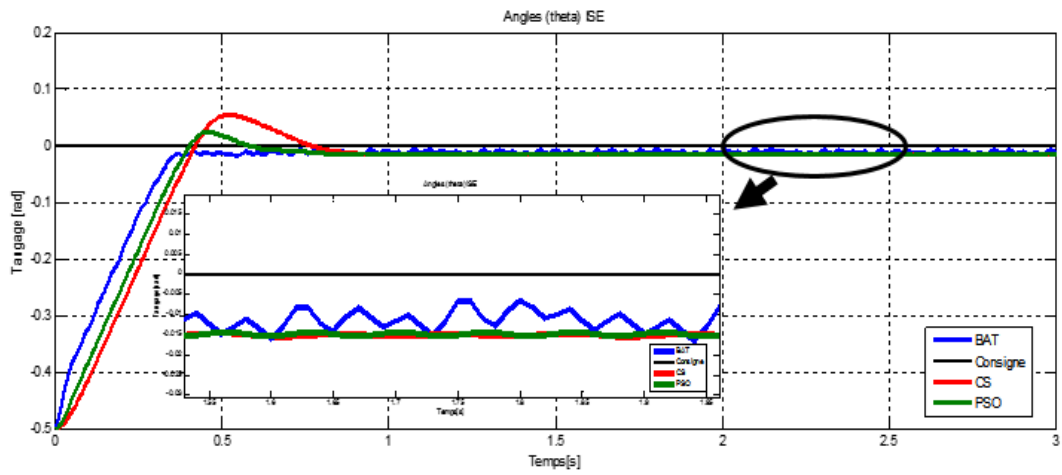
Interprétation : les résultats de l'optimisation en altitude sont quasiment les mêmes que ceux obtenus dans le cas de l'attitude. L'algorithme BAT converge à partir de la 12ème itération à son minimum de valeur de la fonction fitness. Le PSO a gardé sa supériorité par rapport au CS en termes de convergence et de précision.

IV.3.1 Application des commandes floues optimisées pour le contrôle de l'attitude d'un quadrotor :

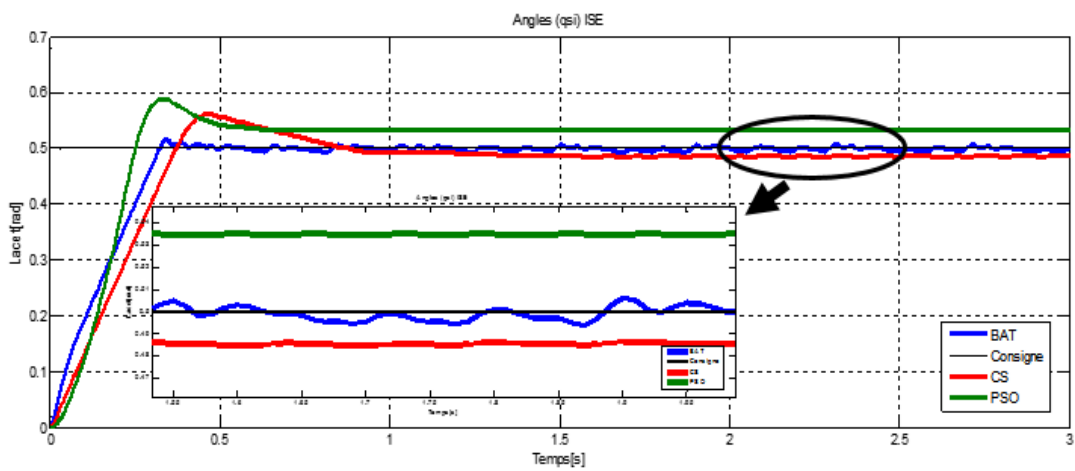
En attitude les angles de départ pour les angles de roulis, tangage et de lacet sont 0.5 radians, 0.5 rad et 0 rad, respectivement. Les angles désirés sont 0 rad pour le roulis et le tangage, et 0.5 rad pour le lacet. La durée de simulation est de 3 secondes avec un pas d'échantillonnage de 0.01s. Les paramètres utilisés sont les meilleurs obtenus par les PSO, le BAT et CS dans le cas de minimisation de chaque fonction fitness. Les résultats ci-dessous représentent les réponses après l'application des contrôleurs flous.



1 : les angles de roulis



2 : les angles de tangage

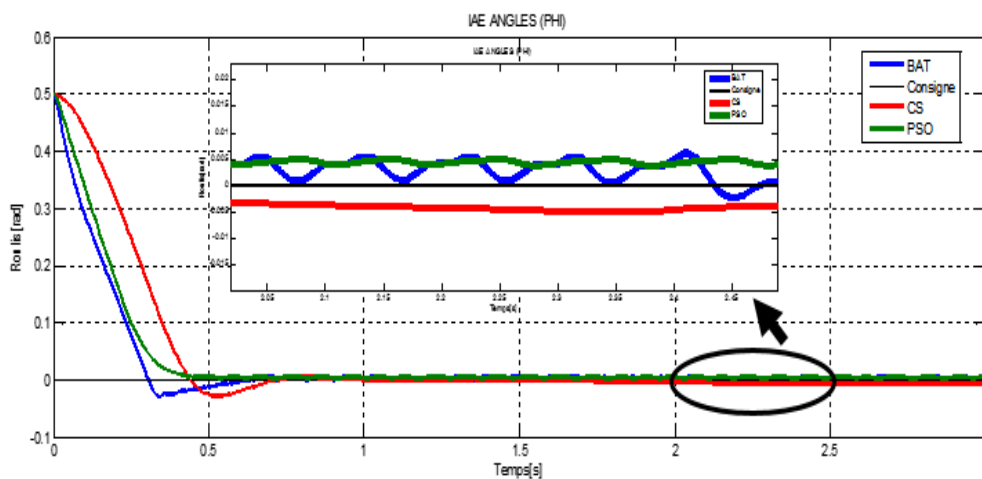


3 : les angles de lacet

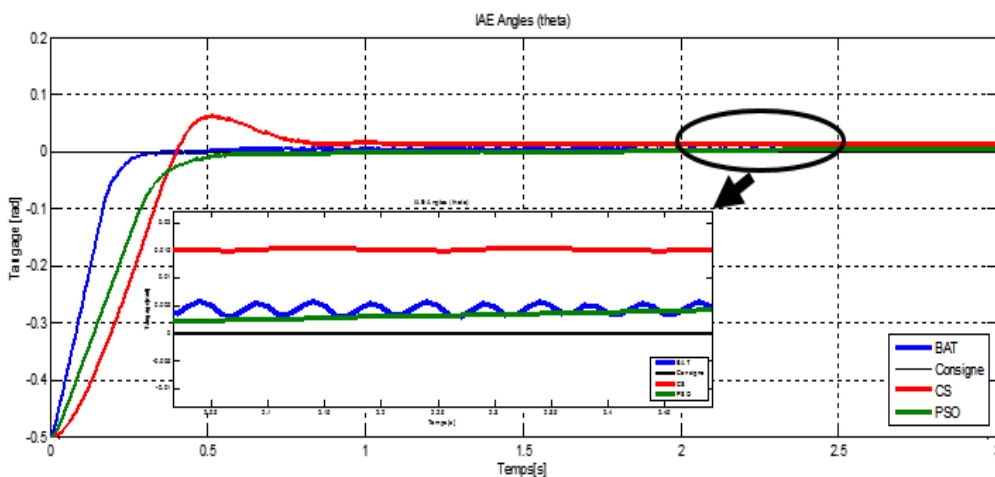
Figure IV.7 : les graphes obtenus par minimisation ISE

Interprétation :

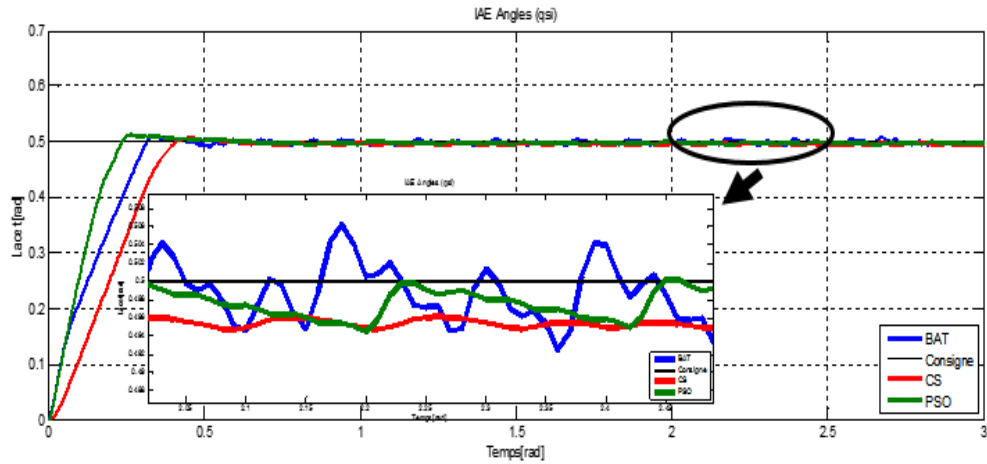
Le critère ISE a été adopté dans ce travail afin d'améliorer la réponse transitoire du système. A travers les résultats obtenus présentés dans les figures (IV.7-1, 2 et 3) On voit bien que le régime transitoire est nettement meilleur dans le cas des commandes optimisées par l'algorithme BAT. Le temps de montée est aussi meilleur dans le cas du BAT. En régime permanent le BAT a donné des résultats avec une erreur faible mais ces réponses présentent de légères oscillations acceptables. Le PSO a donné aussi des résultats acceptables en régime transitoire mais le cas du lacet l'erreur en régime permanent est grande. Les contrôleurs optimisés par l'algorithme CS ont donnés des réponses lentes.



1 : les angles de roulis



2 : les angles de tangage

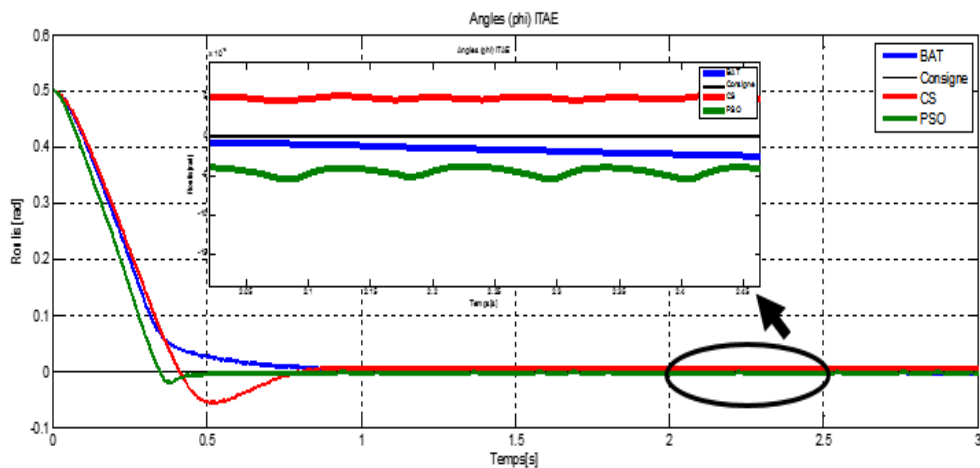


3 : les angles de lacet

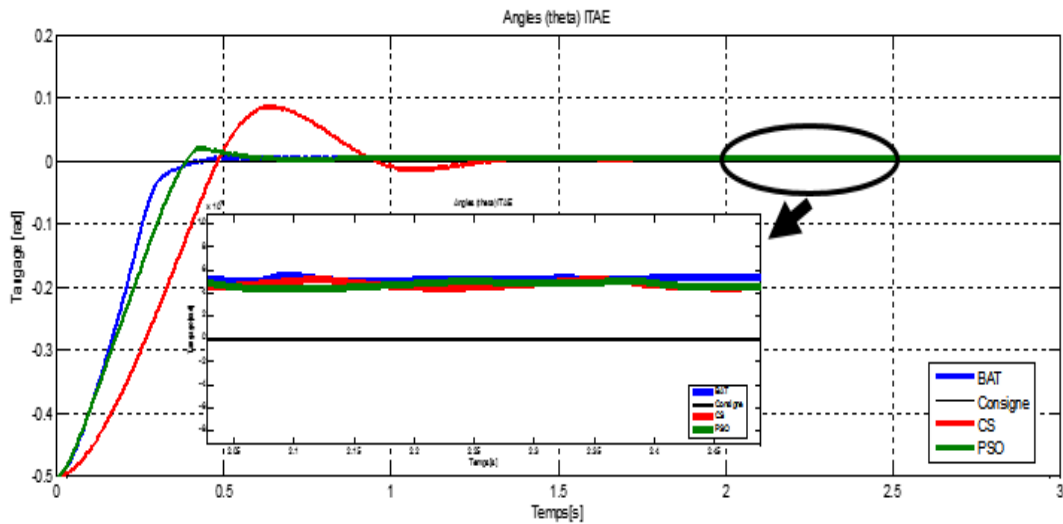
Figure IV.8 : les graphes obtenus par minimisation d'IAE

Interprétation :

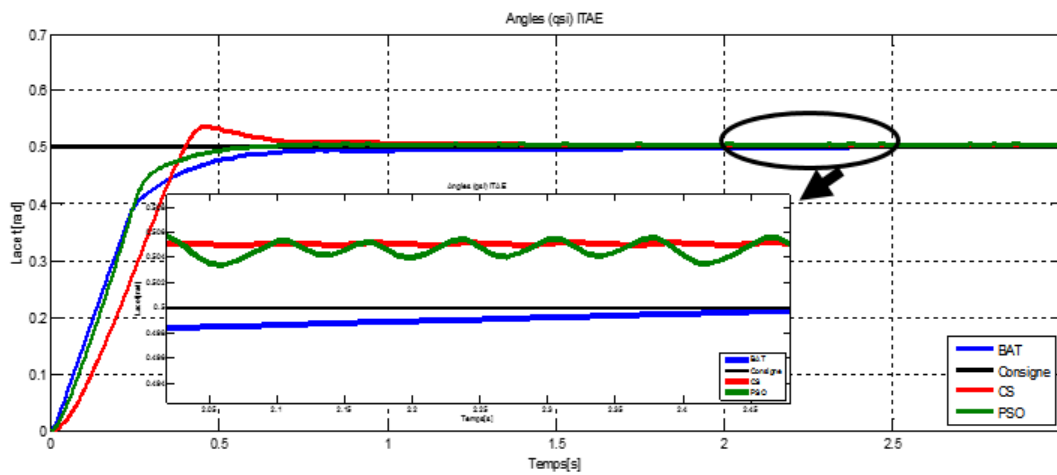
Le critère IAE a été adopté dans ce travail afin d'améliorer la réponse à la fois transitoire et permanente du système. A travers les résultats obtenus présentés dans les figures (IV.8-1, 2 et 3). Que ce soit en régime transitoire ou permanent le BAT est le meilleur malgré que le PSO été légèrement mieux que le BAT dans le cas du lacet. Les petites oscillations remarquées dans le cas de l'usage de premier critère (ISE) ont été enlevés dans le cas de l'usage de ce critère. Le pire des résultats est celui obtenus par le CS en régime transitoire notamment pour le tangage et le lacet et en régime permanent notamment le roulis et tangage.



1 : les angles de roulis



2 : les angles de tangage



3 : les angles de lacet

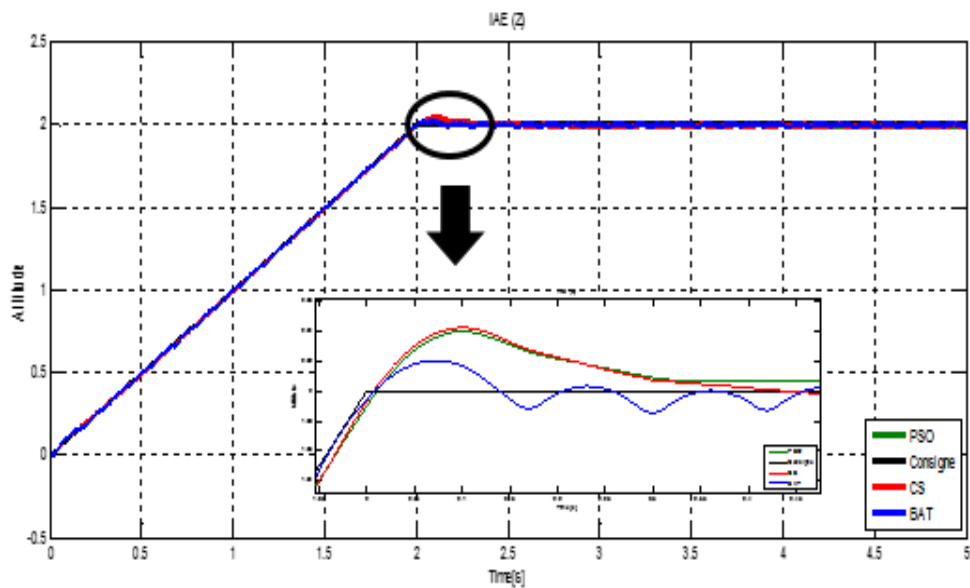
Figure IV.9 : les graphes obtenus par minimisation d'ITAE

Interprétation des résultats

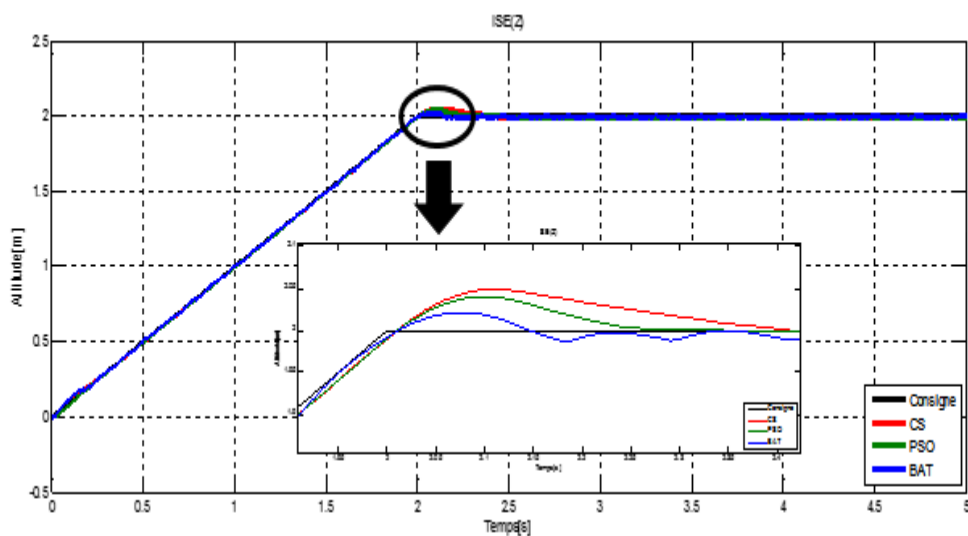
Le critère ITAE a été adopté dans ce travail afin d'améliorer la réponse en régime permanent du système. A travers les résultats obtenus et présentés dans les figures (IV.9-1, 2 et 3), les contrôleurs optimisés par le BAT ont pu améliorer les résultats en régime permanent et on voit bien que l'erreur dans ce régime est faible dont on peut dire qu'elle est quasiment nulle.

- **Les réponses optimisées en altitude**

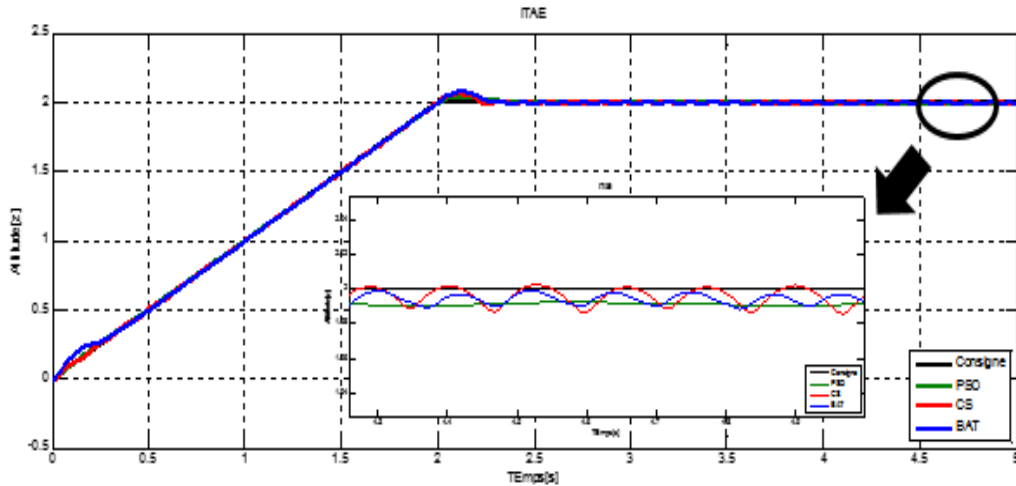
La commande ici concerne le réglage de l'altitude dont le quadrotor est commandé par les contrôleurs optimisés pour effectuer un déplacement vertical. Pour les graphes de trajectoires, les courbes en noir représentent les consignes, les courbes en bleu représentent les réponses obtenues par l'algorithme BAT, les courbes en vert représentent les réponses obtenues par PSO et les courbes en rouge représentent les courbes obtenues par CS.



1 : les réponses obtenues par minimisation d'IAE



2 : les réponses obtenues par minimisation d'ISE



3 : les angles obtenus par minimisation d'ITAE

Figure IV.10 : les graphes en altitude

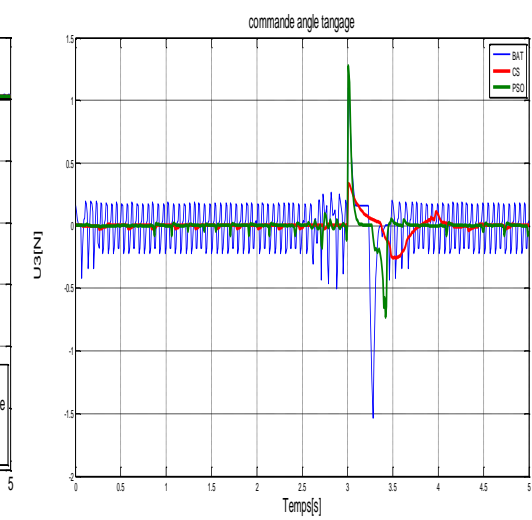
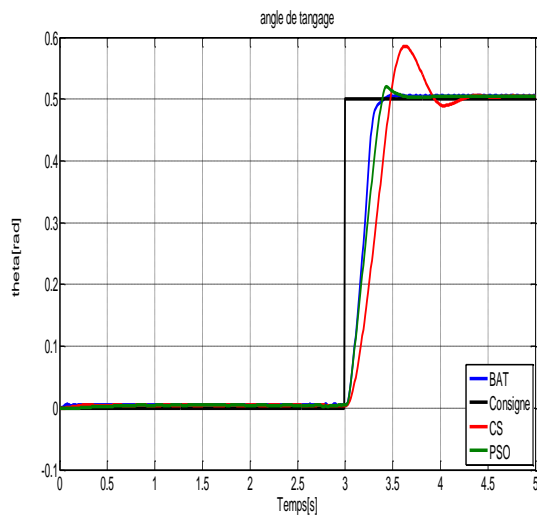
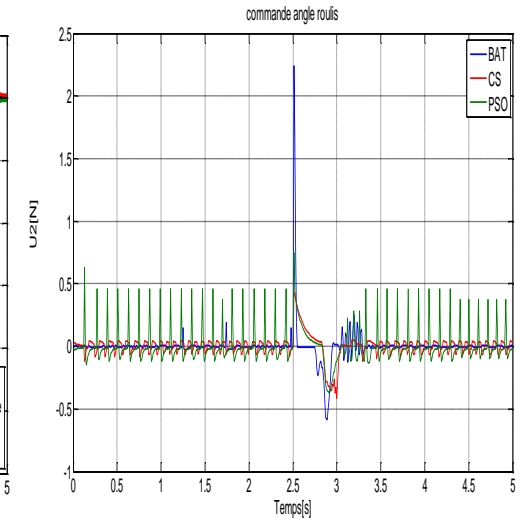
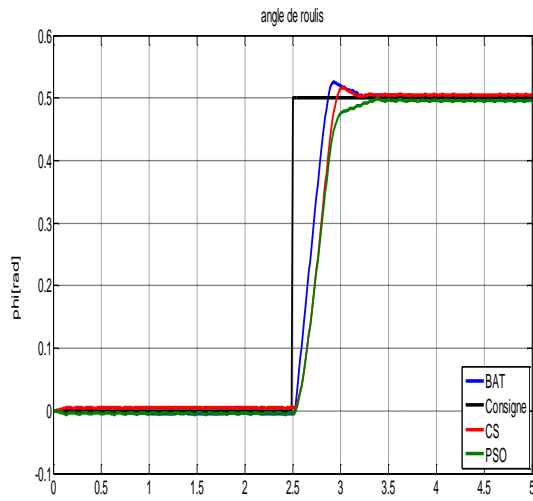
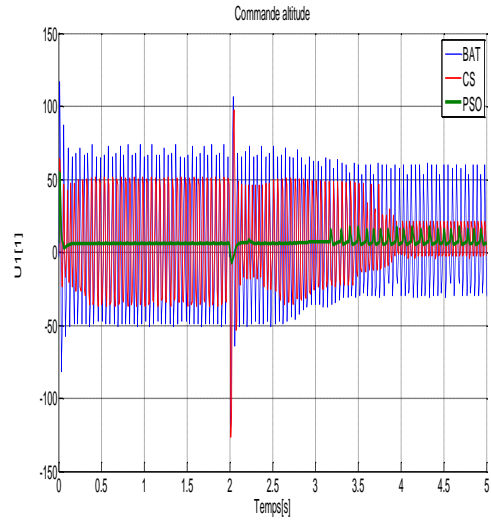
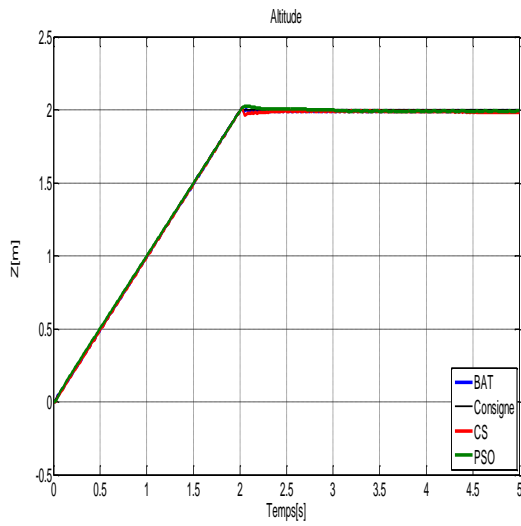
Interprétation

Les résultats de contrôle en altitude ont confirmés ceux obtenus dans le cas de l'attitude. En effet, les contrôleurs flous optimisés par le BAT basant sur les différentes fonctions fitness donnent des meilleurs résultats que les deux autres algorithmes. Les réponses obtenues par cet algorithme sont les plus proches aux consignes imposées en utilisant les trois critères. L'algorithme optimisation essaim particulaire donne des résultats acceptables et le CS reste toujours le plus défavorable.

IV.3.2 Les résultats de commande floue optimale de l'altitude et de l'attitude à la fois

Maintenant, nous avons commandé le quadrotor en altitude (déplacement le long de l'axe Z) et en attitude. Les résultats de cette section représentent les réponses du quadrotor en altitude et en attitude à la fois.

Le quadrotor démarre avec un état initial égale $z(0)=0$, $\phi(0) = 0$, $\theta(0) = 0$, $\psi(0) = 0$. Le système est commandé tout d'abord pour atteindre un altitude désiré vaut $z_d = 2 m$. avec l'angle de lacet vaut 0.5 rad. Après stabilisation en altitude à l'instant enivrant 2.5s. Par la suite, les angles désirés sont des échelons de 0.5 rad, pour le roulis qui démarre à l'instant 2.5s et l'angle de tangage démarre à l'instant 3s. Le temps de simulation est 5 secondes avec un pas d'échantillonnage de 0.01s. On représente les résultats en déplacement linéaires, en déplacement angulaire et les commandes.



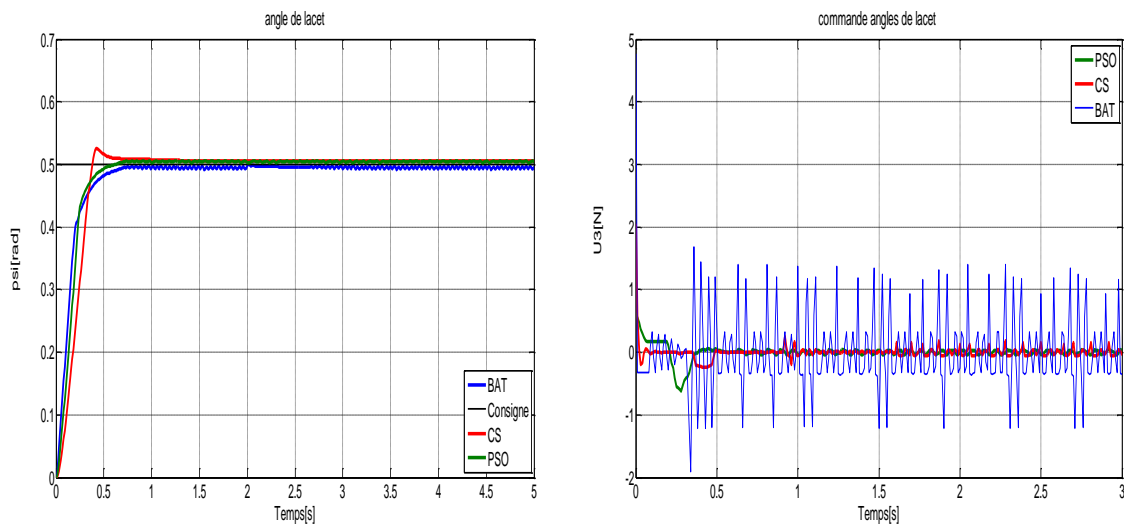


Figure IV.11 : les graphes en altitude

Interprétation

Malgré que les résultats de la commandes floues optimales obtenues par l’usage de BAT en termes de réponses que ce soit en régime transitoire ou permanent la représentent des fluctuations qu’on peut les considérées acceptables. Malgré que l’intervalle de la fréquence choisi était court ($f_{min}=0$ et $f_{max}=0.5$) ces oscillations des commandes sont dues à cet intervalle. Globalement, les commandes les plus laisses sont celles obtenues par l’usage de l’algorithme CS. Mais les réponses sont les plus défavorables. Le PSO donne des résultats intermédiaires mais avec une erreur statique globalement importante. Le tableau ci-dessous juge qualitativement les trois l’algorithme d’optimisation étudiés dans ce mémoire :

L’algorithme	Précision	Temps de calcul	Energie de commande
PSO	*	+	*
CS	-	-	+
BAT	+	+	-

Tableau IV.8 : Comparaison entre les algorithmes

+: *milleur*, **-**: *pire*, *****: *moyenne*

IV.4 Conclusion

Nous avons présenté dans ce chapitre la commande optimale d'un quadrotor en attitude et en altitude. Après avoir examiné les algorithmes d'optimisation BAT,CS et PSO par rapport à des critères (ISE, IATE et IAE), des comparaisons ont été données dont elles montre la supériorité de l'algorithme BAT par rapport au deux autres algorithmes notamment en termes de performances par rapport au temps de traitement et de convergence et en terme de minimisation des fonctions fitness.

Les commandes floues optimisées ont été appliquées tour d'abord en attitude puis en altitude on a pu constater que l'algorithme BAT a apporté de meilleurs performances (temps de montée, précision, erreur en régime permanent) par rapport au l'algorithme optimisation par essaim particulaire et au la recherche de coucou. Les signaux de commande ont montrés des fluctuations dans le cas de BAT qu'on pense acceptables pour un système avionique tel que le quadrotor.

Les travaux présentés dans ce mémoire ont conduit à l'optimisation métaheuristique des systèmes d'inférence floue pour la commande d'un drone de type quadrotor. D'une manière plus précise il s'agissait d'une optimisation des distributions des sorties des contrôleurs flous par trois algorithmes métaheuristiques tels que : l'algorithme d'optimisation par essaim particulaire (PSO), l'algorithme BAT et la recherche coucou (CS)

Nous avons présenté dans une première partie un état de l'art sur les drones et en particulier sur les quadrotor. La commande d'un système nécessite de connaître son modèle, à cet effet, nous avons présenté les démarches de la modélisation dynamique de quadrotor en détail, ce modèle est utilisé par la suite pour les tests en simulation des commandes développées.

Les résultats obtenus sont donnés dans le chapitre 4, ils montrent que la commande optimale d'un quadrotor en attitude et en altitude. Après avoir examiné les algorithmes d'optimisation BAT, CS et PSO par rapport à des critères (ISE, IATE et IAE) des comparaisons ont été données dont elles montrent la supériorité de l'algorithme BAT par rapport au deux autres notamment en termes de performances par rapport au temps de traitement et de convergence et en terme de minimisation des fonctions fitness.

Les commandes floues optimisées ont été appliquées tour d'abord en attitude puis en altitude on a pu constater que l'algorithme BAT a apporté de meilleurs performances (temps de montée, précision, erreur en régime permanent) par rapport au l'algorithme optimisation par essaim particulaire et au la recherche de coucou. Les signaux de commande dans le cas de BAT ont montrés des fluctuations qu'on pense acceptables pour un système avionique tel que le quadrotor.

Références et bibliographiques

- 1 : N. RACHEDI, “**Commande hybrid avec observation d’un UAV de type Quadrotor**”, mémoire de magistère, EMP Bordj EL-Bahri, algerie, Janvier 2011.
- 3 : Prof. Kamel ADI, “**La montée des drones : Comment construire le vôtre**”, Université de Québec en Outaouais.
- 3 : R. Lozano, P. Castillo and A. Dzul, “**Stabilization of a mini rotorcraft having four rotors**”, Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 2693 – 2698, 2004.
- 4 : S. Bouabdallah, P. Murrieri and R. Siegwart, “**Design and control of an indoor micro quadrotor**”, IEEE International conference on Robotics and Automation, New Orleans, USA, 2004.
- 5 : T. Hamel, R. Mahony, R. Lozano and J.P. Ostrowski, “**Dynamic modeling and configuration stabilization for an x4-flyer**”, in IFAC 15th World Congress on Automatic Control, Barcelona, Spain, 2002.
- 6 : H. Bouadi, M. Bouchoucha, and M. Tadjine “**Modelling and Stabilizing Control Laws Design Based on Sliding Mode for an UAV Type-Quadrotor**” Engineering Letters, London, England, Vol. 15, No. 2, pp. 15-24, 2007.
- 7: L. Derafa, T. Madani, and A. Benallegue “**Dynamic modelling and experimental identification of four rotor helicopter parameters**”, IEEE-ICIT Mumbai, India, pp. 1834-1839, 2006.
- 8: O.Castillo, P.Melin “**Type-2 Fuzzy Logic: Theory and Applications**” Springer, Studies in Fuzziness and Soft Computing, Volume 223.
- 9: H.Boubertakh “**Contribution à l’optimisation par algorithmes évolutionnaires des contrôleurs floues**”, Thèse doctorat, ENP 2009.
- 10 : S.Digabel, “**Introduction aux métaheuristiques**”, Ecole Polytechnique de Montréal.
- 11 : J. Kennedy, R.C.Eberhart, “**Swarm intelligence**” New York: Morgan Kaufmann, 2001.
- 12 : X.-S. Yang, “**A New Metaheuristic Bat-Inspired Algorithm**”, in: Nature Inspired Cooperative Strategies for Optimization (NISCO 2010) (Eds. J. R. Gonzalez et al.), Studies in Computational Intelligence, Springer Berlin, 284, Springer, 65-74 (2010).
- 13 : A. Gherboudj, “**Méthodes de résolution de problèmes difficiles académiques**”, Thèse Doctorat, Université de Constantine 2, 2013.

- 14 : X.-S. Yang, “**Nature-Inspired Metaheuristic Algorithms, Second Edition**”, Luniver Press, ISBN-13: 978-1-905986-28-6, 2010.
- 15 : Yang, X.-S., and Deb, S. (2010), “**Engineering Optimisation by Cuckoo Search**”, Int. J. Mathematical Modelling and Numerical Optimisation, Vol. 1, No. 4, 330–343 (2010).
- 16 : E. Mamdani, “**Application of fuzzy logic to approximate reasoning using linguistic systems**”, Fuzzy Sets and Systems, vol.26, 1977, pp.1182-1191.
- 17 : T.J. Procyk and E.H. Mamdani, “**A Linguistic Self-Organising Process Controller**”, Automatica, Vol. 15, pp. 15-30, 1979.
- 18 : R. Palm, “**Sliding Mode Fuzzy Control**”, Proc. of the IEEE Conf. on Fuzzy Systems (Fuzz IEEE 92), San Diego, USA, 1992, pp. 519-526
- 19 : L.Foulloy, “**Contrôle qualitative et contrôle flou: vers une méthode d’écriture des contrôleurs flous**”, Actes des 12 ièmes journées internationales sur les systèmes expertset leurs applications”, Avignon, France, 1992.
- 20 : S. Galichet, M. Dussud and L. Foulloy, “**Contrôleurs flous: Equivalences et études comparatives**”, Actes des rencontres francophones sur la logique floue et ses applications (LFA’92), Nimes, France, 1992, pp. 229-236.
- 21 : T. Takagi, M. Sugeno, “**Fuzzy identification of systems and its applications to modeling and control**”, IEEE Transactions on systems Man and cybernetics, Vol. 15, n° 1, 1985, pp. 116-132.
- 22 : H. Bersini, V. Gorrini, “**FUNNY (Fuzzy or Neural Net) Methods for Adaptive Process Control**”, in Proc. of the 1st European Congress on Fuzzy Intelligent Technologies EUFIT’93, Aachen, Germany, 1993, pp. 55-61.
- 23 : F. Herrera, M. Lozano and J.L. Verdegay, “**A learning process for fuzzy control rules using genetic algorithms**,” Fuzzy Sets and Systems, 100(1–3), 1998, pp.143–158.
- 24 : N, TALBI, “**Conception des Systèmes d’Inférence Floue par des Approches Hybr ides : Application pour la Commande et la Modélisation des Systèmes Nonlinéaires**” Thèse doctorat , Université de Constantine 1, 2014.

Résumé

L'objectif de ce projet est l'optimisation métaheuristique d'un système d'inférence flou pour la commande d'un drone de type Quadrotor, le travail commence par une recherche bibliographique sur la commande du quadrotor, en suite une modélisation de l'engin doit être donnée, la commande par logique floue classique est à proposer. Afin d'obtenir la meilleure distribution des fonctions d'appartenances sur l'univers de discours on doit utiliser les méthodes d'optimisation métaheuristiques telles que : la méthode d'essaim de particules (PSO), l'algorithme BAT et la recherche Coucou (CS). Les performances et la robustess des commandes qui seront développées sont à comparer en simulation.

ملخص

العمل المقدم في هذا المشروع يتمحور حول تحسين الأداء لنظام تحكم ضبابي من نوع T.S باستخدام خوارزميات البحث الشامل (الأدلة العليا)، لاستعماله في التحكم في طائرة من دون طيار من نوع رباعية المراوح. يبدأ المشروع ببحث مكتبي حول التحكم في الطائرات رباعية المراوح، بعد ذلك تقديم نمذجة رياضية للطائرة المستعملة، تقديم أيضا طريقة تحكم في الطائرة باستخدام المراقبة الضبابية الكلاسيكية، بعد ذلك نستعمل خوارزميات البحث الشامل (الأدلة العليا)، مثل خوارزمية الأسراب، خوارزمية الخفافيش، خوارزمية طائر الوقواق. وفي الأخير قمنا بمقارنة النتائج المتحصل عليها.