



Bureau d'Études Industriel — BEI 2025/2026

École Nationale Supérieure d'Électrotechnique, d'Électronique,
d'Informatique, d'Hydraulique et des Télécommunications (ENSEEIHT)

En partenariat avec EasyMile

Stratégies d'allocation de couple pour le système multi-moteurs de l'EZDolly

Analyse, modélisation et simulation de différentes approches pour l'optimisation
énergétique et la stabilité

Table des matières

1	État de l'art des stratégies d'allocation de couple	2
1.1	Introduction	2
1.2	Répartition fixe (ou proportionnelle)	2
1.3	Contrôle optimal ou prédictif (MPC, LQR, QP)	2
1.4	Méta-heuristiques (PSO, Algorithmes génétiques, etc.)	3
1.5	Logique floue	3
1.6	Apprentissage automatique (Neural Networks, Reinforcement Learning)	4
1.7	Contrôle robuste (SMC, ADRC...)	4
1.8	Synthèse des stratégies existantes	4
1.8.1	Évaluation quantitative des stratégies d'allocation de couple	4
2	Proposition de stratégie pour l'EZDolly	7
3	Exploration d'une approche basée sur l'apprentissage automatique	8
3.1	Motivation et contexte	8
3.2	Apprentissage supervisé	8
3.2.1	Principe	8
3.2.2	Mise en œuvre proposée	8
3.2.3	Avantages et limites	9
3.3	Apprentissage par renforcement (Reinforcement Learning)	9
3.3.1	Principe	9
3.3.2	Exemple de fonction de récompense	9
3.3.3	Méthodes de RL adaptées	9
3.3.4	Processus d'apprentissage	9
3.3.5	Avantages et limites	10
3.4	Proposition de mise en œuvre pour le BEI	10
3.5	Architecture conceptuelle de la stratégie proposée	10
3.6	Objectif expérimental	10
4	État de l'art	11
4.1	But du projet	11
4.2	Données disponibles	11
4.3	Variables du modèle	12
5	Implémentation des différentes stratégies d'allocation de couple	14
5.0.1	Répartition fixe (ou proportionnelle)	14
5.0.2	Contrôle optimal ou prédictif (MPC, LQR, QP)	14
5.0.3	Méta-heuristiques (PSO, Algorithmes génétiques)	15
5.0.4	Logique floue	15
5.0.5	Apprentissage automatique	16
5.0.6	Contrôle robuste (SMC / ADRC)	17
5.0.7	Résumé comparatif	17

Chapitre 1

État de l’art des stratégies d’allocation de couple

1.1. Introduction

Dans un véhicule électrique à entraînement distribué, comme l’EZDolly développé par EasyMile, chaque roue dispose de son propre moteur asynchrone. Cette architecture dite « overactuated » permet un contrôle indépendant du couple de chaque roue, offrant de nouveaux degrés de liberté pour optimiser à la fois la performance dynamique, la stabilité et l’efficacité énergétique du véhicule. Le but de cette étude est d’identifier et d’analyser les principales stratégies d’allocation du couple entre les moteurs, selon plusieurs critères définis dans le sujet : efficacité énergétique, stabilité, robustesse, complexité de mise en œuvre et adaptabilité aux conditions réelles.

1.2. Répartition fixe (ou proportionnelle)

Principe : Cette méthode consiste à distribuer le couple total demandé C_{total} de manière égale ou proportionnelle entre les moteurs :

$$C_i = k_i \times C_{total}, \quad \text{avec} \quad \sum k_i = 1$$

Par exemple, une répartition uniforme (25% par roue) ou fixe (60% avant / 40% arrière).

Avantages :

- Très simple à implémenter (aucun calcul en ligne);
- Robuste, car indépendante du modèle du véhicule;
- Compatible avec tous les types de véhicules multi-moteurs.

Inconvénients :

- Pas d’optimisation énergétique (rendement moteur non pris en compte);
- Aucune adaptabilité (même répartition quelle que soit la pente, la charge ou la défaillance);
- Risque d’usure inégale des moteurs ou des pneus.

Applications typiques : Utilisée comme stratégie de base ou de référence dans les simulations pour évaluer des méthodes plus avancées.

1.3. Contrôle optimal ou prédictif (MPC, LQR, QP)

Principe : Cette famille de méthodes vise à minimiser une fonction de coût, par exemple :

$$J = \sum (Pertes_nergiques + Erreur_couple + Dviation_stabilit)$$

sous contraintes (limites de couple, stabilité du véhicule, etc.). Les approches les plus utilisées sont :

- **MPC (Model Predictive Control)** : contrôle prédictif sur un horizon glissant ;
- **LQR (Linear Quadratic Regulator)** : minimise un coût quadratique pour stabiliser le système ;
- **QP (Quadratic Programming)** : résolution rapide d'un problème d'optimisation quadratique.

Avantages :

- Excellente efficacité énergétique ;
- Gestion simultanée de plusieurs objectifs (énergie + stabilité) ;
- Résultat optimal garanti selon le modèle.

Inconvénients :

- Calcul intensif, difficile à embarquer en temps réel ;
- Sensibilité au modèle et aux capteurs ;
- Mise en œuvre complexe.

Applications typiques : Présent dans les prototypes de véhicules haut de gamme et dans la recherche sur la traction intégrale optimisée.

1.4. Méta-heuristiques (PSO, Algorithmes génétiques, etc.)

Principe : Ces méthodes d'optimisation globale s'inspirent de phénomènes naturels :

- **PSO (Particle Swarm Optimization)** : les solutions se déplacent collectivement vers le meilleur résultat ;
- **GA (Genetic Algorithm)** : la solution évolue par sélection, croisement et mutation.

Elles permettent de déterminer des coefficients de répartition k_i optimaux selon plusieurs critères (énergie, stabilité, adhérence...).

Avantages :

- Recherche d'un optimum global même dans un problème non linéaire ;
- Capacité à traiter plusieurs objectifs simultanément ;
- Peu dépendantes du modèle mathématique exact.

Inconvénients :

- Calculs lourds, souvent effectués hors ligne ;
- Résultats non garantis en temps réel ;
- Sensibilité aux paramètres de recherche.

Applications typiques : Utilisées pour calibrer des stratégies (hors ligne), puis appliquer les résultats sous forme de loi simplifiée dans le véhicule.

1.5. Logique floue

Principe : La logique floue repose sur des règles linguistiques du type : « *Si pente forte et adhérence faible, alors réduire le couple arrière* ». Elle utilise plusieurs variables d'entrée (pente, glissement, vitesse, charge, etc.) et ajuste le couple transmis à chaque moteur selon des règles définies par des experts.

Avantages :

- Robuste face aux incertitudes et aux erreurs de mesure ;
- Facile à modifier et à comprendre (intuitive) ;
- Bonne adaptabilité aux scénarios variés (charge, défaillance, pente...).

Inconvénients :

- Non optimale au sens mathématique ;
- Calibration empirique (règles à ajuster par essai/erreur) ;
- Complexité croissante avec le nombre d'entrées.

Applications typiques : Utilisée seule ou combinée à une approche analytique (hybride). Elle est particulièrement adaptée aux systèmes soumis à des conditions variables, comme l'EZDolly.

1.6. Apprentissage automatique (Neural Networks, Reinforcement Learning)

Principe : Ces méthodes apprennent automatiquement la meilleure répartition du couple à partir de données :

- **Apprentissage supervisé :** le réseau apprend à imiter un contrôleur optimal existant ;
- **Apprentissage par renforcement (RL) :** l'agent apprend par essais/erreurs à minimiser une fonction de coût.

Avantages :

- Très grande adaptabilité à de nouveaux contextes ;
- Gère les systèmes fortement non linéaires ;
- Possibilité d'entraînement en simulation avant déploiement.

Inconvénients :

- Besoin de données et de calculs importants ;
- Pas de garantie formelle de stabilité ;
- Difficulté d'interprétation (« boîte noire »).

Applications typiques : Utilisé dans la recherche récente sur les véhicules autonomes et la traction intelligente (Tesla, etc.).

1.7. Contrôle robuste (SMC, ADRC...)

Principe : Ces méthodes visent à maintenir la stabilité même en cas d'incertitudes :

- **SMC (Sliding Mode Control) :** force le système à suivre une trajectoire de référence malgré les perturbations ;
- **ADRC (Active Disturbance Rejection Control) :** estime et compense activement les perturbations en temps réel.

Avantages :

- Très robuste aux variations de charge, pentes et défaillances moteur ;
- Bonne stabilité dynamique ;
- Moins dépendant d'un modèle précis.

Inconvénients :

- Optimisation énergétique limitée (priorise la stabilité) ;
- Réglage complexe ;
- Risque d'oscillations (chattering) dans le SMC.

Applications typiques : Utilisé dans les systèmes critiques (véhicules tout-terrain, robots lourds, chariots industriels).

1.8. Synthèse des stratégies existantes

1.8.1. Évaluation quantitative des stratégies d'allocation de couple

Afin de compléter la comparaison qualitative des stratégies, une évaluation chiffrée a été réalisée en attribuant une note à chaque critère et une pondération reflétant leur importance pour le système EZDolly.

Stratégie	Efficacité	Stabilité	Robustesse	Complexité	Adaptabilité
Répartition fixe	Faible	Faible	Très élevée	Très faible	Faible
Contrôle optimal (MPC/LQR)	Très élevée	Élevée	Moyenne	Très élevée	Moyenne
Méta-heuristiques (PSO/GA)	Élevée	Élevée	Moyenne	Élevée	Moyenne
Logique floue	Moyenne	Moyenne à élevée	Élevée	Moyenne	Élevée
Apprentissage automatique	Très élevée	Moyenne à élevée	Moyenne	Très élevée	Très élevée
Contrôle robuste (SMC/ADRC)	Moyenne	Très élevée	Très élevée	Élevée	Élevée

TABLE 1.1 – Comparaison des principales stratégies d'allocation de couple.

Barème de notation

Chaque critère est noté sur une échelle de 1 à 5 :

Niveau	Note
Très faible	1
Faible	2
Moyenne	3
Élevée	4
Très élevée	5

Pondération des critères

Les coefficients de pondération w_i sont choisis en fonction des priorités du projet :

- $w_{\text{eff}} = 0.30$: l'efficacité énergétique est l'objectif principal ;
- $w_{\text{stab}} = 0.25$: la stabilité du mouvement est essentielle ;
- $w_{\text{rob}} = 0.20$: la robustesse du système doit être garantie ;
- $w_{\text{compl}} = -0.15$: la complexité doit rester raisonnable (coefficient négatif) ;
- $w_{\text{adapt}} = 0.10$: l'adaptabilité constitue un critère secondaire.

La formule de calcul du score global est donc :

$$\text{Score} = 0.3E + 0.25S + 0.2R - 0.15C + 0.1A$$

Notation des stratégies

Stratégie	Efficacité	Stabilité	Robustesse	Complexité	Adaptabilité
Répartition fixe	2	2	5	1	2
Contrôle optimal (MPC/LQR)	5	4	3	5	3
Méta-heuristiques (PSO/GA)	4	4	3	4	3
Logique floue	3	4	4	3	4
Apprentissage automatique	5	4	3	5	5
Contrôle robuste (SMC/ADRC)	3	5	5	4	4

TABLE 1.2 – Notes attribuées à chaque critère pour les différentes stratégies.

Résultats pondérés

Les scores finaux obtenus à partir de la formule précédente sont résumés dans le tableau suivant :

Stratégie	Calcul du score	Score final (/5)
Répartition fixe	$0.3(2) + 0.25(2) + 0.2(5) - 0.15(1) + 0.1(2)$	2.45
Contrôle optimal (MPC/LQR)	$0.3(5) + 0.25(4) + 0.2(3) - 0.15(5) + 0.1(3)$	3.55
Méta-heuristiques (PSO/GA)	$0.3(4) + 0.25(4) + 0.2(3) - 0.15(4) + 0.1(3)$	3.25
Logique floue	$0.3(3) + 0.25(4) + 0.2(4) - 0.15(3) + 0.1(4)$	3.55
Apprentissage automatique	$0.3(5) + 0.25(4) + 0.2(3) - 0.15(5) + 0.1(5)$	3.55
Contrôle robuste (SMC/ADRC)	$0.3(3) + 0.25(5) + 0.2(5) - 0.15(4) + 0.1(4)$	3.70

TABLE 1.3 – Calcul pondéré et score global des différentes stratégies d'allocation de couple.

Classement final et interprétation

Stratégie	Type de contrôle	Score	Commentaires
1	Contrôle robuste (SMC/ADRC)	3.70	Excellent compromis stabilité-robustesse, complexité modérée.
2	Contrôle optimal (MPC/LQR)	3.55	Très performant mais difficile à embarquer.
3	Logique floue	3.55	Bon équilibre entre adaptabilité et simplicité.
4	Apprentissage automatique	3.55	Très prometteur, mais complexe et coûteux en données.
5	Méta-heuristiques (PSO/GA)	3.25	Performantes mais computationnellement lourdes.
6	Répartition fixe	2.45	Simple mais peu performante et non adaptative.

TABLE 1.4 – Classement final des stratégies selon le score pondéré.

Conclusion

L'évaluation met en évidence que le **contrôle robuste (SMC/ADRC)** constitue le meilleur compromis entre stabilité, robustesse et faisabilité. En alternative, une **approche floue ou hybride (flou + optimisation)** représente une solution intéressante pour combiner adaptabilité et performance énergétique sans alourdir la complexité de mise en œuvre. Enfin, le **MPC** peut être conservé comme référence théorique pour comparer et valider les performances des autres méthodes.

Chapitre 2

Proposition de stratégie pour l'EZDolly

Au vu des contraintes opérationnelles du véhicule, une approche **hybride** est proposée :

- **Allocation de base** : une répartition issue d'une optimisation rapide (QP ou LQR simplifié) assurant un bon rendement moteur.
- **Supervision floue** : une logique adaptative ajustant la répartition selon la charge, la pente ou les défauts.
- **Apprentissage hors ligne** : une adaptation continue des paramètres à partir des données réelles.

Cette approche permet de concilier **efficacité énergétique**, **robustesse** et **adaptabilité**, tout en restant compatible avec les contraintes de calcul embarqué.

Chapitre 3

Exploration d'une approche basée sur l'apprentissage automatique

3.1. Motivation et contexte

Le système de traction de l'EZDolly présente une architecture multi-moteurs complexe, avec des comportements non linéaires dus aux variations de charge, aux frottements, et aux conditions d'adhérence. Les méthodes classiques d'allocation de couple, telles que le *Model Predictive Control (MPC)* ou la commande optimale quadratique, nécessitent un modèle dynamique précis du véhicule, ce qui limite leur robustesse dans des environnements réels.

L'utilisation de l'**intelligence artificielle (IA)** et de l'**apprentissage automatique** constitue une alternative prometteuse. Ces méthodes permettent d'apprendre, à partir de données ou de simulations, la meilleure stratégie d'allocation de couple sans dépendre d'un modèle physique exact. Elles offrent ainsi la possibilité d'adapter dynamiquement le comportement du système à différentes conditions d'exploitation (pente, charge, adhérence, défaillance moteur).

3.2. Apprentissage supervisé

3.2.1. Principe

L'apprentissage supervisé consiste à entraîner un *réseau de neurones* à reproduire le comportement d'un contrôleur optimal préexistant (par exemple, un contrôleur MPC ou une loi d'optimisation quadratique). Le réseau apprend une fonction d'approximation reliant les entrées du système aux couples optimaux à appliquer sur chaque moteur :

$$f : [C_{total}, v, \text{charge}, \text{adhérence}] \rightarrow [C_1, C_2, C_3, C_4]$$

Ainsi, à partir du couple global demandé et de l'état du véhicule, le modèle prédit la répartition optimale du couple entre les quatre moteurs.

3.2.2. Mise en œuvre proposée

- **Phase d'apprentissage hors ligne :**
 - Génération d'une base de données à partir d'un contrôleur optimal simulé (MPC ou QP).
 - Entraînement du réseau de neurones à minimiser l'erreur de prédiction :

$$\text{Loss} = \sum_i \|\hat{C}_i - C_i^{opt}\|^2$$

- **Phase d'exploitation en ligne :**
 - Le réseau calcule en temps réel la répartition du couple, remplaçant les calculs complexes par une inférence rapide.

3.2.3. Avantages et limites

Avantages :

- Exécution rapide en temps réel (calcul instantané).
- Facile à entraîner à partir d'un simulateur.
- Permet d'approcher un comportement optimal sans modèle exact.

Limites :

- Ne s'adapte pas dynamiquement aux nouvelles conditions.
- Dépend de la qualité des données d'apprentissage.

3.3. Apprentissage par renforcement (Reinforcement Learning)

3.3.1. Principe

L'*apprentissage par renforcement* (RL) consiste à entraîner un agent à interagir avec un environnement simulé. L'agent apprend par essais et erreurs : à chaque étape, il choisit une action (distribution du couple), observe la réaction du système (vitesse, énergie consommée, stabilité) et reçoit une *récompense* positive ou négative selon la qualité de cette action.

L'objectif est de maximiser la récompense cumulée :

$$R = \sum_t \gamma^t r_t$$

où r_t représente la récompense instantanée et γ un facteur de pondération temporelle.

3.3.2. Exemple de fonction de récompense

Une fonction de récompense typique peut prendre la forme :

$$r_t = -(\alpha E_{consomme} + \beta |v - v_{ref}| + \gamma P_{instable})$$

où :

- $E_{consomme}$: énergie électrique utilisée par les moteurs (à minimiser),
- $|v - v_{ref}|$: écart entre la vitesse réelle et la consigne (à minimiser),
- $P_{instable}$: pénalité liée à une perte d'adhérence ou une instabilité (à éviter).

3.3.3. Méthodes de RL adaptées

Plusieurs variantes du RL sont envisageables :

- **DQN (Deep Q-Network)** : pour des actions discrètes (par exemple, choisir parmi quelques répartitions prédéfinies).
- **DDPG (Deep Deterministic Policy Gradient)** : pour un contrôle continu des couples.
- **PPO (Proximal Policy Optimization)** : algorithme stable et performant pour des applications de commande.

3.3.4. Processus d'apprentissage

1. Création d'un environnement de simulation Python représentant la dynamique longitudinale du véhicule.
2. Définition des variables d'état (vitesse, couple total, charge, adhérence, etc.).
3. Définition des actions (couples C_1, C_2, C_3, C_4).
4. Définition de la fonction de récompense.
5. Entraînement de l'agent jusqu'à convergence.

Une fois l'entraînement terminé, l'agent est capable d'adapter en temps réel la répartition du couple selon la situation (pente, charge, conditions de friction, etc.).

3.3.5. Avantages et limites

Avantages :

- Auto-apprentissage sans connaissance précise du modèle.
- Adaptabilité à des conditions nouvelles.
- Capacité à gérer plusieurs objectifs simultanément (énergie, stabilité, sécurité).

Inconvénients :

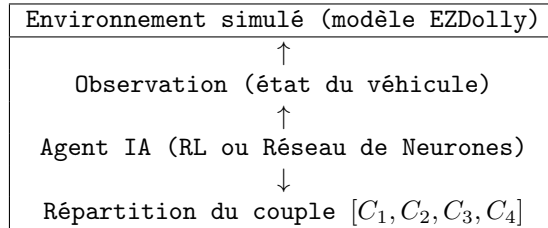
- Temps d'entraînement long.
- Absence de garanties analytiques de stabilité.
- Difficulté d'interprétation (modèle boîte noire).

3.4. Proposition de mise en œuvre pour le BEI

Une approche progressive peut être envisagée :

1. **Simulation de base** : modéliser la dynamique longitudinale du véhicule avec une répartition fixe de couple (référence).
2. **Contrôleur de référence** : implémenter une allocation optimale (QP, LQR, ou loi analytique) pour générer des données d'apprentissage.
3. **Entraînement IA** : entraîner un réseau (supervisé ou RL) à reproduire ou surpasser la performance du contrôleur optimal.
4. **Évaluation** : comparer les performances (énergie, stabilité, robustesse) sur différents scénarios : pente, variation de charge, défaillance moteur.

3.5. Architecture conceptuelle de la stratégie proposée



3.6. Objectif expérimental

L'objectif visé dans le cadre du BEI est de démontrer la **faisabilité et l'intérêt** d'une telle approche. L'agent devra être capable de réduire la consommation énergétique par rapport à une répartition fixe, tout en maintenant une stabilité satisfaisante (suivi de la vitesse et répartition équilibrée du couple).

La simulation sera implémentée en **Python**, à l'aide des bibliothèques suivantes :

- `gymnasium` ou `stable-baselines3` pour le cadre RL ;
- `tensorflow` ou `pytorch` pour les réseaux de neurones ;
- `numpy` et `matplotlib` pour le traitement et la visualisation des données.

Cette approche ouvre la voie vers une **allocation de couple auto-apprenante**, capable de s'adapter en permanence aux conditions réelles d'exploitation du véhicule, tout en conciliant efficacité énergétique, robustesse et flexibilité.

Chapitre 4

État de l’art

Un chariot électrique à quatre roues motrices indépendantes, chacune équipée d’un moteur asynchrone. Le défi principal est de répartir le couple total demandé entre les quatre moteurs afin de :

- optimiser la consommation énergétique,
- garantir la stabilité dynamique du chariot,
- assurer la sûreté de fonctionnement en cas de défaillance moteur.

Approches existantes

1. **Répartition uniforme** : simple mais peu efficace énergétiquement. (5–6, section 2.2.1–2.2.2)
2. **Heuristique à rendement** : privilégie les moteurs opérant dans leur zone de meilleur rendement. (11, section 4.1.1)
3. **Optimisation quadratique (QP)** : minimise les pertes énergétiques sous contraintes mécaniques et électriques. (6–8, section 2.2.4)
4. **Méthodes robustes** : ajoutent la gestion de défaillances moteur et la reconfiguration. (8–9, section 2.2.6)
5. **Couplage longitudinal–latéral** : prend en compte la stabilité directionnelle lors des virages. (14–15, section 4.3.1)

4.1. But du projet

Concevoir, simuler et valider une stratégie d’allocation de couple multi-moteurs pour le chariot **EZ-Dolly**, en visant une optimisation énergétique et une sûreté de fonctionnement accrue.

Objectifs détaillés

1. Minimiser la consommation énergétique à l’aide des courbes de rendement moteur.
2. Respecter les contraintes physiques (couple, puissance, charge, vitesse).
3. Garantir la stabilité et la continuité de service en cas de panne moteur.
4. Comparer plusieurs stratégies : uniforme, heuristique, optimisation (QP).
5. Valider les résultats par simulation Python avec les données réelles fournies.

4.2. Données disponibles

Fichiers expérimentaux

- Time (ms)
- Axle_Torque_Setpoint (N.m)
- Axle_Torque_Feedback (N.m)
- Speed_Feedback (m/s)

Ces données proviennent d’essais à différentes vitesses : 5, 8, 10, 13 et 15 km/h.

Fichiers caractéristiques moteur

Type : moteur asynchrone triphasé

- Puissance nominale : 8 kW
- Couple nominal : 34.3 N.m
- Vitesse nominale : 2200 tr/min
- Courant nominal : 109 A
- Rapport de réduction : 26
- Nombre de pôles : 4

Fichiers de caractéristiques mesurées :

- Speed (RPM)
- Power_5mn (W), Torque_5mn (N.m)
- Power_30mn (W), Torque_30mn (N.m)
- Power_60mn (W), Torque_60mn (N.m)
- CosPhi

Ces données permettent d'établir les limites de couple admissible et la carte de rendement $\eta(T, \omega)$.

Données fixes du véhicule

Paramètre	Symbole	Valeur (Unité)
Masse à vide	m_0	4900 kg
Charge maximale	m_{\max}	7000 kg
Vitesse maximale	v_{\max}	15 km/h
Empattement	L	4.8 m
Voie	l	2.2 m
Rayon de roue	r	0.24 m
Type de motricité	-	4 moteurs indépendants

4.3. Variables du modèle

Variables d'état du véhicule

$x(t)$: position longitudinale

$v(t)$: vitesse longitudinale

$a(t)$: accélération

Variables de commande

$T_{\text{tot}}(t)$: couple total demandé

$T_i(t)$: couple attribué à chaque moteur ($i = 1$ à 4)

$\omega_i(t)$: vitesse de rotation moteur

Variables dérivées

$$P_{\text{mec},i} = T_i \times \omega_i \quad (\text{puissance mécanique})$$

$$P_{\text{elec},i} = \frac{P_{\text{mec},i}}{\eta_i(T_i, \omega_i)} \quad (\text{puissance électrique absorbée})$$

$$P_{\text{loss},i} = P_{\text{elec},i} - P_{\text{mec},i} \quad (\text{pertes moteur})$$

$$E_{\text{tot}} = \int \sum_i P_{\text{elec},i} dt \quad (\text{énergie totale consommée})$$

Variables de contexte

- Charge utile m
- Adhérence μ
- Température ambiante ou pente (si simulées)

Chapitre 5

Implémentation des différentes stratégies d'allocation de couple

Cette section présente les principales méthodes d'allocation de couple étudiées, ainsi que leur mise en œuvre possible sous **MATLAB** et **Python**.

5.0.1. Répartition fixe (ou proportionnelle)

Principe : La méthode consiste à répartir le couple total demandé C_{total} entre les moteurs selon des coefficients fixes k_i tels que :

$$C_i = k_i \times C_{total}, \quad \text{avec} \quad \sum_i k_i = 1$$

Implémentation MATLAB :

```
C_total = 400; % Nm
k = [0.25 0.25 0.25 0.25]; % répartition uniforme
C = k * C_total; % vecteur des couples
```

Implémentation Python :

```
import numpy as np
C_total = 400 # Nm
k = np.array([0.25, 0.25, 0.25, 0.25])
C = k * C_total
```

Méthode très simple et robuste, utilisée comme référence.

5.0.2. Contrôle optimal ou prédictif (MPC, LQR, QP)

Principe : Minimiser une fonction de coût représentant les pertes énergétiques et la stabilité :

$$J = \sum (\text{Pertes énergétiques} + \text{Erreur de stabilité})$$

sous contraintes de couple, puissance, etc.

Implémentation MATLAB (MPC Toolbox) :

```
mpcobj = mpc(plant_model, Ts);
mpcobj.Weights.ManipulatedVariables = [0.1 0.1 0.1 0.1];
mpcobj.Weights.OutputVariables = [1 1 1 1];
mpcobj.MV = struct('Min',0,'Max',100);
C = mpcmove(mpcobj, x_state, y_measured, y_ref);
```

Implémentation Python (optimisation quadratique) :

```
import cvxpy as cp
import numpy as np

C = cp.Variable(4)
C_total = 400
eff = np.array([0.9, 0.85, 0.88, 0.9])

objective = cp.Minimize(cp.sum_squares(C/eff))
constraints = [cp.sum(C) == C_total, C >= 0, C <= 100]
prob = cp.Problem(objective, constraints)
prob.solve()
print(C.value)
```

Méthode précise mais computationnellement coûteuse.

5.0.3. Méta-heuristiques (PSO, Algorithmes génétiques)

Principe : Recherche globale de coefficients k_i optimaux minimisant une fonction d'énergie à l'aide de techniques évolutionnaires.

Implémentation MATLAB (Genetic Algorithm) :

```
fun = @(k) energy_consumption(k); % fonction coût
nvars = 4;
Aeq = [1 1 1 1]; beq = 1; % somme = 1
lb = zeros(1,4); ub = ones(1,4);
opts = optimoptions('ga','Display','iter');
[k_opt, fval] = ga(fun, nvars, [], [], Aeq, beq, lb, ub, [], opts);
```

Implémentation Python (PSO) :

```
from pyswarm import pso
import numpy as np

def cost(k):
    return np.sum((k*400)**2)

lb = [0,0,0,0]
ub = [1,1,1,1]
xopt, fopt = pso(cost, lb, ub)
k_opt = xopt / np.sum(xopt)
```

Approche utile pour le calibrage hors ligne.

5.0.4. Logique floue

Principe : La logique floue repose sur des règles linguistiques, par exemple :

Si pente forte et adhérence faible, alors réduire le couple arrière.

Implémentation MATLAB :

```
fis = mamfis('Name','TorqueAllocation');
fis = addInput(fis, [0 15], 'Name', 'slope');
fis = addOutput(fis, [0 1], 'Name', 'rear_factor');
fis = addRule(fis, [1 1 1 1]);
C_rear = evalfis(fis, slope_value);
```

Implémentation Python (scikit-fuzzy) :

```
import numpy as np
import skfuzzy as fuzz
from skfuzzy import control as ctrl

slope = ctrl.Antecedent(np.arange(0, 20, 1), 'slope')
rear_factor = ctrl.Consequent(np.arange(0, 1.1, 0.1), 'rear_factor')

slope['low'] = fuzz.trimf(slope.universe, [0, 0, 10])
slope['high'] = fuzz.trimf(slope.universe, [5, 15, 20])
rear_factor['reduce'] = fuzz.trimf(rear_factor.universe, [0, 0, 0.5])

rule1 = ctrl.Rule(slope['high'], rear_factor['reduce'])
system = ctrl.ControlSystem([rule1])
sim = ctrl.ControlSystemSimulation(system)
sim.input['slope'] = 12
sim.compute()
print(sim.output['rear_factor'])
```

Approche intuitive et robuste, bien adaptée aux systèmes variables.

5.0.5. Apprentissage automatique

Apprentissage supervisé

Principe : Un réseau de neurones apprend à reproduire le comportement d'un contrôleur optimal à partir de données simulées.

Implémentation MATLAB :

```
X = input_data; Y = optimal_torque;
net = feedforwardnet(10);
net = train(net, X', Y');
C_pred = net(new_input');
```

Implémentation Python (TensorFlow) :

```
import tensorflow as tf
model = tf.keras.Sequential([
    tf.keras.layers.Dense(32, activation='relu', input_shape=(4,)),
    tf.keras.layers.Dense(4)
])
model.compile(optimizer='adam', loss='mse')
model.fit(X_train, Y_train, epochs=100)
C_pred = model.predict(X_test)
```

Apprentissage par renforcement (RL)

Principe : L'agent apprend par interaction avec un environnement simulé à maximiser une récompense énergétique et de stabilité.

Implémentation MATLAB :

```
env = rlPredefinedEnv('CartPole-Continuous');
agent = rlDDPGAgent(obsInfo, actInfo, agentOpts);
train(agent, env);
```

Implémentation Python (Stable-Baselines3) :

```
from stable_baselines3 import PPO

model = PPO("MlpPolicy", env, verbose=1)
model.learn(total_timesteps=1e5)
C_pred, _ = model.predict(state)
```

Approche très prometteuse, capable d'adaptation dynamique.

5.0.6. Contrôle robuste (SMC / ADRC)

Principe : Forcer le système à suivre une trajectoire de référence malgré les perturbations :

$$u = -k \operatorname{sign}(s), \quad s = \lambda(x - x_{ref}) + (\dot{x} - \dot{x}_{ref})$$

Implémentation MATLAB :

```
lambda = 10; k = 5;
s = lambda*(x - x_ref) + (dx - dx_ref);
u = -k*sign(s);
```

Implémentation Python :

```
lambda_ = 10; k = 5
s = lambda_*(x - x_ref) + (dx - dx_ref)
u = -k * np.sign(s)
```

Méthode robuste aux incertitudes et défaillances, garantissant la stabilité dynamique.

5.0.7. Résumé comparatif

Méthode	Outil MATLAB	Outil Python	Difficulté
Répartition fixe	Script simple	Numpy	Très faible
Contrôle optimal (MPC/LQR/QP)	MPC Toolbox	cvxpy	Élevée
Méta-heuristiques	GA Toolbox	pyswarm / deap	Moyenne
Logique floue	Fuzzy Logic Designer	scikit-fuzzy	Moyenne
Apprentissage supervisé	Neural Network Toolbox	TensorFlow / PyTorch	Élevée
Apprentissage par renforcement	RL Toolbox	Stable-Baselines3	Très élevée
Contrôle robuste	Script manuel (SMC/ADRC)	Numpy	Moyenne

TABLE 5.1 – Comparatif des méthodes et outils disponibles pour MATLAB et Python.