Universidad de Valladolid

ESCUELA DE INGENIERÍAS INDUSTRIALES

UNIVERSIDAD DE VALLADOLID

ESCUELA DE INGENIERIAS INDUSTRIALES

Grado en Ingeniería Electrónica Industrial y Automática

REINFORCEMENT LEARNING FOR CONTROL

Autor:

Daniel Arévalo Gil

Responsable de Intercambio en la Uva

Santiago Cáceres Gómez

Universidad de destino

Aix Marseille

Resumen :

En este trabajo, realizamos un estudio del machine learning, enfocándonos en su variante reinforcement learning. El trabajo se compone de una primera sección que contiene definiciones de los diferentes aspectos de esta tecnología, así como su desarrollo a lo largo de los años y su estado actual . En la sección siguiente, explicamos las principales variantes de esta tecnología: aprendizaje supervisado, aprendizaje no supervisado y reinforcement learning. Para nuestro trabajo, implementaremos la tecnología de reinforcement learning para realizar el control de velocidad de un motor eléctrico asíncrono de 550w con una velocidad nominal de 1500 rpm, controlado mediante un variador de frecuencia. Para implementar esta tecnología utilizaremos la Toolbox Reinforcement Learning de Matlab. Explicaremos como hemos realizado esta implementación así como una explicación como los pasos necesarios para utilizarla adecuadamente. En la parte final del trabajo, analizaremos los datos obtenidos, así como una conclusión final sobre problemas y soluciones que surgen al implementar esta tecnología.

Palabras Clave :

Aprendizaje Automático, Aprendizaje por Refuerzo, Redes Neuronales, control, automatización.

Abstrat :

In this work, we conduct a study on machine learning, focusing on its variant reinforcement learning. The work consists of an initial section that contains definitions of the different aspects of this technology, as well as its development over the years and its current state. In the next section, we explain the main variants of this technology: supervised learning, unsupervised learning, and reinforcement learning. For our work, we will implement reinforcement learning technology to control the speed of a 550W asynchronous electric motor with a nominal speed of 1500 rpm, controlled by a frequency converter. To implement this technology, we will use Matlab's Reinforcement Learning Toolbox. We will explain how we implemented this, as well as a step-by-step explanation of how to use it properly. In the final part of the work, we will analyze the data obtained, as well as provide a final conclusion on the problems and solutions that arise when implementing this technology.

Keywords :

Machine learning, Reinforcement Learning, Neural Networks, control, automation

Valladolid, Julio 2024.

TFG REALIZADO EN PROGRAMA DE INTERCAMBIO

TÍTULO:	REINFORCEMENT LEARNING FOR CONTROL
ALUMNO:	Daniel Arévalo Gil
FECHA:	26 Junio 2024
CENTRO:	Poytech Marseille
UNIVERSIDAD:	Aix Marseille
TUTOR:	Hassan NOURA

Génie industriel et informatique

REINFORCEMENT LEARNING

FOR CONTROL

Author:

Daniel Arévalo Gil

Tutor :

Hassan NOURA

Resumen:

Dans ce travail, nous réalisons une étude de l'apprentissage automatique, en nous concentrant sur sa variante, l'apprentissage par renforcement. Le travail se compose d'une première section qui contient des définitions des différents aspects de cette technologie, ainsi que son développement au fil des ans et son état actuel. Dans la section suivante, nous expliquons les principales variantes de cette technologie : l'apprentissage supervisé, l'apprentissage non supervisé et l'apprentissage par renforcement. Pour notre travail, nous mettrons en œuvre la technologie d'apprentissage par renforcement pour contrôler la vitesse d'un moteur électrique asynchrone de 550 W avec une vitesse nominale de 1500 tr/min, contrôlé par un variateur de fréquence. Pour mettre en œuvre cette technologie, nous utiliserons la boîte à outils Reinforcement Learning de Matlab. Nous expliquerons comment nous avons réalisé cette mise en œuvre ainsi que les étapes nécessaires pour l'utiliser correctement. Dans la partie finale du travail, nous analyserons les données obtenues, ainsi qu'une conclusion finale sur les problèmes et solutions qui surgissent lors de la mise en œuvre de cette technologie.

Key words:

Machine learning, Reinforcement Learning, Neural Networks, policy, algorithms, electric Motor.

Abstract:

In this work, we conduct a study on machine learning, focusing on its variant reinforcement learning. The work consists of an initial section that contains definitions of the different aspects of this technology, as well as its development over the years and its current state. In the next section, we explain the main variants of this technology: supervised learning, unsupervised learning, and reinforcement learning. For our work, we will implement reinforcement learning technology to control the speed of a 550W asynchronous electric motor with a nominal speed of 1500 rpm, controlled by a frequency converter. To implement this technology, we will use Matlab's Reinforcement Learning Toolbox. We will explain how we implemented this, as well as a step-by-step explanation of how to use it properly. In the final part of the work, we will analyze the data obtained, as well as provide a final conclusion on the problems and solutions that arise when implementing this technology.

Key words:

Machine learning, Reinforcement Learning, Neural Networks, policy, algorithms, electric Motor.

Indice

1 Les dèfinitions initiaux

Avant de commencer l'étude des technologies telles que le machine learning, nous devons définir cette technologie ainsi que certains de ses composants principaux.

Machine learning (1)

C'est une branche de l'intelligence artificielle (IA) et de l'informatique qui se concentre sur l'utilisation de données et d'algorithmes pour permettre à l'IA d'imiter la manière dont les humains apprennent, améliorant ainsi progressivement sa précision.

Machine learning (2)

Est une forme d'intelligence artificielle (IA) qui est axée sur la création de systèmes qui apprennent, ou améliorent leurs performances, en fonction des données qu'ils traitent. L'intelligence artificielle est un terme large qui désigne des systèmes ou des machines simulant une forme d'intelligence humaine. Le machine learning et l'IA sont souvent abordés ensemble et ces termes sont parfois utilisés de manière interchangeable bien qu'ils ne renvoient pas exactement au même concept. Une distinction importante est que, même si l'intégralité du machine learning repose sur l'intelligence artificielle, cette dernière ne se limite pas au machine learning.

Intelligence artificielle

L'intelligence artificielle est un domaine de l'informatique qui se concentre sur la création de systèmes capables d'imiter des processus cognitifs humains, tels que l'apprentissage, la résolution de problèmes et la prise de décision, en utilisant des algorithmes et des données.

Reinforcement learning

"L'apprentissage par renforcement consiste à apprendre quoi faire - comment mapper des situations à des actions - afin de maximiser un signal de récompense numérique. L'apprenant n'est pas informé des actions à prendre, mais doit plutôt découvrir quelles actions rapportent le plus de récompenses en les essayant."

2 Histoire

La machine learning, bien que récemment popularisée, existe depuis des décennies. Il est difficile de désigner une date ou une personne qui l'a inventée, car elle résulte de la combinaison des travaux de différents individus ayant contribué à cette discipline avec différents inventifs ou algorithmes. Voici les événements les plus marquants.

En 1943, Walter Pitts et Warren McCullock ont présenté le premier modèle mathématique de réseaux neuronaux dans leur travail commun intitulé "Un calcul logique des idées immanentes dans l'activité nerveuse".

Donald Hebb publie son livre "L'organisation du comportement" en 1949, dans lequel il expose de nouvelles théories sur la relation entre le comportement, les réseaux neuronaux et l'activité cérébrale. Il a eu un impact significatif sur le développement de l'apprentissage automatique.

Dans les années 1950, Arthur Samuel, employé d'IBM et pionnier de l'intelligence artificielle, a développé un programme informatique capable de jouer aux dames avec les meilleurs joueurs de l'époque. Il a utilisé un algorithme qui, au lieu d'évaluer tous les mouvements possibles, effectue une élagage alpha-bêta pour trouver le mouvement optimal. De plus, il a été capable d'introduire des mécanismes pour que son programme puisse apprendre et s'améliorer continuellement, se souvenant des positions qu'il avait occupées. Son impact a été très important car, en plus de ses inventions, il a instauré le terme "machine learning". Il l'a défini comme "la capacité donnée aux ordinateurs d'apprendre sans être explicitement programmés".

En 1951, Marvin Minsky et Dean Edmonds ont inventé le premier réseau neuronal artificiel, comprenant 40 neurones interconnectés.

Cinq ans plus tard, un groupe de scientifiques de prestige de différents domaines s'est réuni pour discuter de l'intelligence artificielle. Il est largement considéré comme l'événement fondateur de l'intelligence artificielle en tant que domaine de recherche.

Alexey Ivakhnenko et Valenti Lapa, deux scientifiques soviétiques, ont développé en 1965 le premier réseau neuronal profond, qui est un réseau neuronal avec plusieurs couches. Ils ont démontré que l'utilisation de plusieurs couches plutôt qu'une seule offrait de meilleures performances.

Les scientifiques Thomas Cover et Peter Hart, professeurs à l'université Stanford, ont publié en 1967 un article présentant l'algorithme "the nearest neighbor algorithm", utilisé en classification et régression. C'était le début de la reconnaissance de motifs de base. Dans une section ultérieure, cet algorithme sera expliqué plus en détail.

En 1979, l'informaticien japonais Kunihiko Fukushima a publié un travail expliquant le néocognitron, un réseau hiérarchique composé de plusieurs couches dont le but est de localiser des motifs. Il a été l'ancêtre des modernes réseaux neuronaux convolutifs, utilisés aujourd'hui pour analyser des images.

Six ans plus tard, Terrence Sejnowski a inventé un réseau neuronal artificiel capable d'apprendre à articuler correctement 20 000 mots en une semaine. Le système d'apprentissage utilisé était similaire à celui utilisé par un bébé.

Un an plus tard, Paul Smolensky a conçu une machine pour examiner les données d'entrée et apprendre leur distribution de probabilité. Aujourd'hui, elle est utilisée, par exemple, dans des recommandations basées sur des actions antérieures.

En 1990, Robert Schapire et Yoav Freund ont publié l'article "The strength of weak learnability", dans lequel ils présentent l'algorithme Boosting, dont le but est d'améliorer la prédiction d'un modèle d'intelligence artificielle. Ils affirment qu'un ensemble d'apprentissages faibles génère un apprentissage fort. Un apprentissage faible est un classificateur qui est légèrement corrélé avec la classification réelle. En revanche, un apprentissage fort est un classificateur qui a une forte corrélation avec la classification réelle.

Il y a 25 ans, Tin Kam Ho a publié un article scientifique dans lequel il introduit l'algorithme Random Decision Forests, qui sera expliqué ultérieurement. En 1997, Tom Mitchell a proposé une définition plus complète du terme apprentissage automatique : "un programme qui apprend de l'expérience E par rapport à une tâche T et à une performance P, si sa performance dans T, mesurée en P, s'améliore avec l'expérience E.

Cette même année, Deep Blue, un ordinateur d'IBM, a réussi à battre aux échecs Garri Kasparov, champion du monde. C'était un événement historique car il a démontré que les machines étaient capables de battre l'intelligence humaine dans certains domaines.

En 2011, Google a développé l'algorithme Google Brain en intelligence artificielle. Il a eu un grand impact car il a été capable de reconnaître des chats dans des vidéos YouTube avec une précision de 75 %.

Au cours des dernières années, l'apprentissage automatique s'est fortement développé. Les principaux jalons sont les suivants. En 2014, Facebook a développé DeepFace, un système de reconnaissance faciale dans les images qui se distingue par une précision (97,35 %) similaire à celle d'un humain. Le système est un réseau neuronal composé de neuf couches et qui a été formé avec 4 millions d'images des utilisateurs de Facebook. La même année, Eugene Goostman est devenu le chatbot (programme informatique simulant une conversation humaine) le plus proche de surpasser le test de Turing. Eugene se faisait passer pour un enfant ukrainien de 13 ans. Il a réussi à convaincre 33 % des juges qu'il était humain lors d'un tournoi sur le test de Turing. En 2015, AlphaGo est devenu le premier programme à battre le champion du monde de Go, considéré comme le jeu le plus complexe et abstrait. En 2016, Face2Face a été présenté, dont la logique et les algorithmes ont inspiré les logiciels "deepfake" actuels. En 2017, les premières voitures autonomes ont été observées conduisant en ville sans chauffeur.

Il existe de nombreux autres inventions et développeurs.

3 Promoteurs du développement du machine learning

Nous devons souligner que cette technologie n'est pas nouvelle, mais au cours des dernières années, elle a subi une évolution notable, principalement en raison du développement technologique réalisé dans ces domaines:

Augmentation de la Puissance Informatique:

Le développement continu de processeurs plus rapides et l'amélioration des unités de traitement graphique (GPU) ont permis d'effectuer des calculs plus complexes de manière plus efficace.

Cela permet également de traiter un volume beaucoup plus important de données, de variables et de scénarios potentiels, en analysant des millions de situations de manière beaucoup plus rapide.

Disponibilité de Grandes Volumes de Données:

Le développement du Big Data, l'explosion des données provenant de diverses sources telles que les réseaux sociaux, les capteurs, les transactions commerciales et les appareils IoT ont fourni une grande quantité de données pour entraîner des modèles d'apprentissage machine et plus spécifiquement de reinforcement learning.

Avancées en Termes d'Algorithmes et de Techniques:

Le développement de nouveaux algorithmes et approches, comme les réseaux de neurones profonds (deep learning), a permis d'améliorer considérablement les performances des modèles dans des tâches complexes.

Investissement et Soutien Économique:

L'augmentation du potentiel et des performances de cette technologie a conduit à une augmentation significative de l'investissement de la part des gouvernements, des institutions académiques et des entreprises dans la recherche et le développement en intelligence artificielle et en apprentissage machine.

La création de nouvelles entreprises axées sur l'apprentissage machine et l'adoption de ces technologies par de grandes entreprises technologiques ont accéléré l'innovation et la mise en œuvre pratique.

Une fois connu le contexte historique et les progrès les plus significatifs dans le domaine du machine learning nous allons procéder à l'étude nous allons procéder à développer ses deux piliers fondamentaux.

Dynamic Programming

Le contrôle optimal aborde le problème de trouver une loi de commande pour un système dynamique donné afin de minimiser un critère particulier. Il existe trois piliers du contrôle optimal, comprenant le calcul des variations, le principe du maximum de Pontryagin et la programmation dynamique. Le développement de ces théories est largement attribué au travail de Lev Pontryagin et Richard Bellman dans les années 1950 et à celui de Leonhard Euler et Joseph-Louis Lagrange au XVIIIe siècle.

Trial-and-Error Learning

L'apprentissage par essais et erreurs est l'un des mécanismes fondamentaux par lesquels les humains et les animaux acquièrent des comportements intelligents. En essayant différentes réponses à la même situation, les humains et les animaux peuvent apprendre quel type de comportements sont bons et quel type de comportements sont mauvais, et ainsi ils peuvent répéter les bons comportements et éviter les mauvais comportements. Finalement, le mécanisme d'essais et d'erreurs rend les humains et les animaux capables de prédire les Rewards futures et de choisir la meilleure action.

Ce sont les principales avancées technologiques qui ont été réalisées au fil des ans et qui ont permis le progrès et le développement de cette technologie.

Aujourd'hui, de grandes avancées et de nouvelles découvertes continuent d'être faites et conduisent à un développement plus poussé de ces technologies.

4 L'avenir de Machine learning et de reinforcement learning

En machine learning, on anticipe le développement de modèles plus avancés et généralisables, capables d'apprendre à partir de diverses sources de données et de s'adapter rapidement à de nouvelles tâches grâce au méta-apprentissage. L'interprétabilité et la transparence des modèles seront cruciales, avec un accent sur la création de modèles explicables et sur le respect des réglementations éthiques.

Certains des domaines et applications qui connaîtront les plus grands progrès seront les industries et les domaines sociaux, comme la médecine personnalisée et l'automatisation industrielle, qui continueront à croître, poussés par ces technologies.

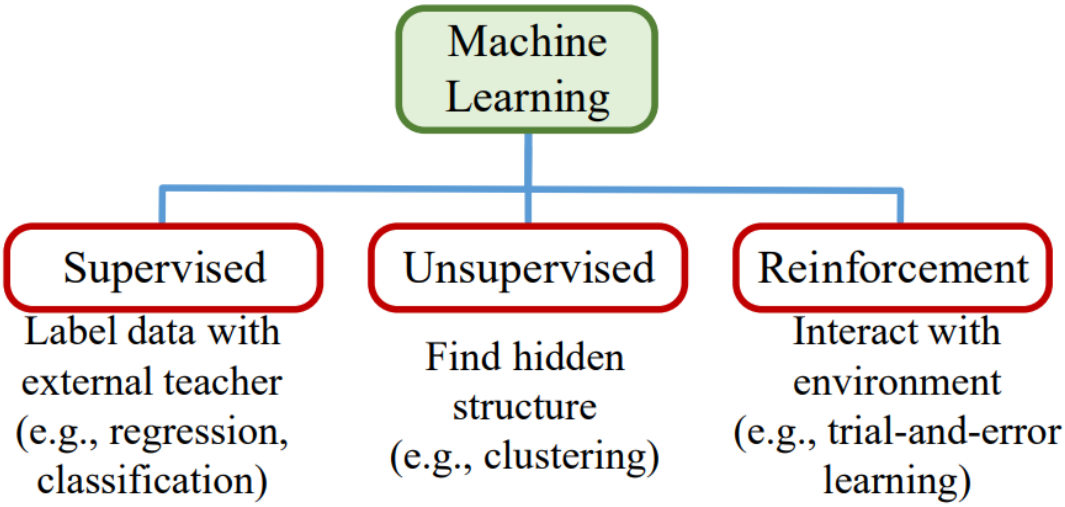
Dans le domaine du reinforcement learning, on s'attend à des progrès significatifs dans la capacité d'apprentissage dans des environnements complexes et dynamiques, avec l'utilisation de simulations avancées et de réalité augmentée. La scalabilité et la robustesse des algorithmes s'amélioreront grâce à des techniques d'apprentissage distribué et des approches hybrides combinant le reinforcement learning avec d'autres techniques d'IA.

L'intégration avec le deep learning permettra de résoudre des problèmes complexes en robotique, dans l'automobile autonome et plus encore. L'avenir du machine learning et du reinforcement learning promet de transformer de nombreuses industries et d'améliorer la qualité de vie grâce à des technologies plus intelligentes et efficaces.

5 Machine Learning

Une fois connue l'évolution de cette technologie au fil du temps et ses définitions les plus précises, nous procéderons à son étude.

Tout d'abord, nous devons souligner que le machine learning se divise principalement en trois catégories:



Unsupervised learning:

Ces algorithmes s'appuient sur un apprentissage préalable basé sur un système d'étiquettes associées à des données qui leur permettent de prendre des décisions ou de faire des prédictions. Un exemple est un détecteur de spam qui étiquette un e-mail comme spam ou non en fonction des modèles qu'il a appris à partir de l'historique des courriels (expéditeur, relation texte/images, mots-clés dans l'objet, etc.).

Supervised learning:

Ces algorithmes ne disposent pas de connaissances préalables. Ils affrontent le chaos des données avec l'objectif de trouver des motifs qui permettent de les organiser d'une manière ou d'une autre. Par exemple, dans le domaine du marketing, ils sont utilisés pour extraire des motifs à partir de données massives provenant des réseaux sociaux et créer des campagnes publicitaires hautement segmentées.

Reinforcement learning:

Son objectif est qu'un algorithme apprenne à partir de sa propre expérience. C'est-à-dire, qu'il soit capable de prendre la meilleure décision face à différentes situations selon un processus d'essais et d'erreurs dans lequel les décisions correctes sontrewards. Actuellement, il est utilisé pour permettre la reconnaissance faciale, réaliser des diagnostics médicaux ou classifier des séquences d'ADN.

6 Citations Reinforcement Learning

Nous nous concentrerons sur l'étude du "Reinforcement Learning (RL)", en commençant par quelques définitions bibliographiques:

Shengbo Eben Li (2024). Reinforcement Learning for Sequential Decision and Optimal Control:

> *“Reinforcement learning (RL) is a biologically inspired learning approach that focuses on finding optimal decision or control strategies for dynamic environments.”*

Sutton, R. S., & Barto, Andrew G. (1998). Reinforcement Learning: An Introduction. MIT Press.

> *“L'apprentissage par renforcement est un domaine de l'apprentissage automatique qui concerne la façon dont les agents doivent prendre des actions dans un environnement afin de maximiser une notion de Reward cumulative.”*

Kaelbling, Leslie Pack , Littman, M. L., & Moore, Andrew W. Moore: (1996). "Reinforcement learning: A survey." Journal of Artificial Intelligence Research.

> *"Reinforcement learning is the problem of learning to behave through direct interaction with a dynamic environment."*

Sergi Monroy, https://www.apd.es/que-es-reinforcement-learning/: (2023)

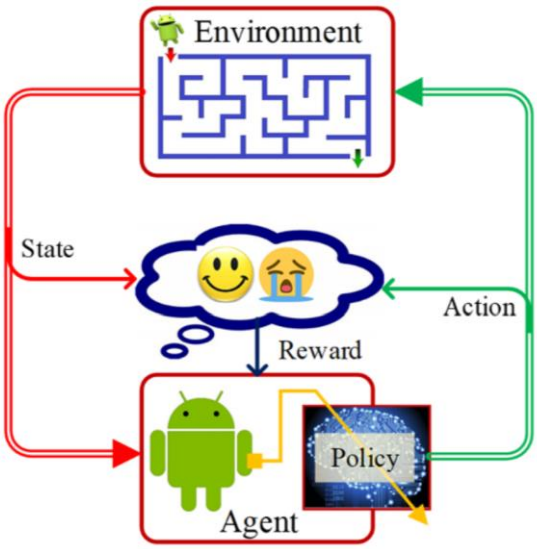
> *"L'apprentissage par renforcement est un algorithme d'apprentissage automatique qui résout les problèmes par essais et erreurs. Sur la base de données, il entraîne différents scénarios à prendre une série de décisions. En fonction des actions entreprises, il reçoit des récompenses ou des pénalités. En fonction de ces réponses, il cherche à maximiser la récompense. "*

Silver, David (2015). "Lecture 1: Introduction to Reinforcement Learning." UCL Course on Reinforcement Learning.

> *“Reinforcement learning is a form of learning in which an agent learns to behave in an environment by taking actions and observing the outcomes of those actions.."*

7 Reinforcement Learning

En général, cinq domaines différents doivent être abordés avec l'apprentissage par renforcement.



Environement:

L'environnement est tout ce qui existe à l'extérieur de l'agent. C'est là où l'agent envoie des actions, et c'est ce qui génère des *rewards* et des observations. Le type d'environnement, qu'il s'agisse d'une simulation ou d'une configuration physique, est également un aspect crucial à considérer.

L'environnement peut être un environnement physique réel ou une simulation, nous devons choisir entre ces deux options. Chacun d'eux a des aspects positifs et des aspects négatifs.

Environnement réel:

-Précision : Rien ne représente l'environnement de manière plus complète que l'environnement réel.

-Simplicité : Il n'est pas nécessaire de passer du temps à créer et valider un modèle.

-Nécessité : Il peut être nécessaire de s'entraîner avec l'environnement réel s'il change constamment ou s'il est difficile à modéliser avec précision.

Environnement simulated:

Vitesse : Les simulations peuvent s'exécuter plus rapidement que le temps réel ou être parallélisées, accélérant ainsi un processus d'apprentissage lent.

Conditions simulées : Il est plus facile de modéliser des situations qui seraient difficiles à tester.

Sécurité : Il n'y a aucun risque d'endommager le matériel.

Normalement, nous développerons les algorithmes et les politiques dans un environnement simulé, et lorsque nous aurons atteint un bon niveau de développement, nous les implémenterons dans l'environnement réel.

Reward:

La *reward* est une fonction essentielle qui produit un nombre scalaire représentant la "reward" d'un agent se trouvant dans un état particulier et prenant une action spécifique. Elle fournit une indication à l'agent sur la qualité de ses actions et influence son comportement dans l'environnement. En d'autres termes, la *reward* agit comme un signal d'encouragement ou de désapprobation pour l'agent, l'aidant à apprendre quelles actions sont les plus bénéfiques dans différentes situations.

Un signal de *reward* est une fonction qui mappe le triplet de l'état actuel, de l'action actuelle et de l'état suivant à un nombre réel, défini comme suit:

$$\text{Reward}=R'(\text{state}, \text{action}, \text{state}+1)$$

L'idée est de savoir si l'agent doit exploiter l'environnement en choisissant les actions qui rapportent le plus de récompenses qu'il connaît déjà, ou s'il doit explorer des parties de l'environnement qui sont encore inconnues ?

Pour répondre à cette question, nous devrons analyser la différence entre exploration et exploitation.

Exploration vs. Exploitation:

Si l'agent exploite toujours ce qu'il pense être la meilleure action à un moment donné, il pourrait ne jamais recevoir d'informations supplémentaires sur les états qui existent au-delà d'une action à faible *reward*.

Cette exploitation pure peut augmenter le temps nécessaire pour trouver la policy optimale ou peut entraîner la convergence de l'algorithme d'apprentissage vers une policy sous-optimale, car des sections entières de l'espace des états peuvent ne jamais être explorées.

L'exploration pure n'est pas une méthode efficace pour apprendre, car l'agent passera probablement du temps à couvrir une plus grande partie de l'espace des états. Bien que cela soit bénéfique pour trouver une solution globale, une exploration excessive peut ralentir le taux d'apprentissage à tel point qu'aucune solution suffisante n'est trouvée dans un laps de temps d'apprentissage raisonnable. Par conséquent, les meilleurs algorithmes d'apprentissage trouvent un équilibre entre l'exploration et l'exploitation de l'environnement.

Policy:

La policy est la fonction cruciale qui cartographie les observations aux actions, et l'algorithme d'apprentissage est la méthode d'optimisation utilisée pour trouver la policy optimale. Il est essentiel de considérer comment vous souhaitez structurer les paramètres et la logique qui composent la partie de prise de décision de l'agent, car cela influencera directement son comportement et sa performance dans l'environnement donné.

Representing a Policy with a Table.

Pour représenter la policy, nous utilisons un tableau de nombres où une entrée agit comme une adresse de recherche et la sortie est le nombre correspondant dans le table. Un type de fonction basée sur le table est la table Q, qui mappe les états et les actions à une valeur.

Training:

L'entraînement en apprentissage par renforcement implique le perfectionnement de la policy de l'agent vers l'optimale. Pendant cette étape, les connaissances acquises sont consolidées et des ajustements fins sont apportés à la policy et aux paramètres de l'algorithme pour améliorer les performances. Une fois cette phase terminée, l'agent est prêt à être évalué et testé dans des environnements du monde réel ou simulés plus difficiles.

8 Implementacion de reinforcement learning en Matlab

Pour implémenter la technologie d'apprentissage par renforcement dans un système contrôlé, nous utiliserons MATLAB avec la boîte à outils Reinforcement Learning Toolbox.

À cette fin, nous utiliserons comme exemple son implantation dans le processus de contrôle du moteur electric.

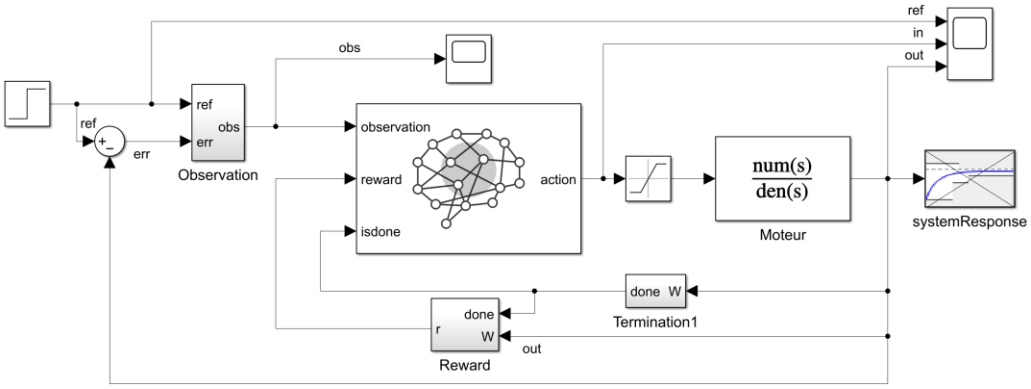
Phases et procédures

Pour implémenter cette technologie, nous devrons réaliser les étapes suivantes:

- Modélisation de notre système réel (environnement).
- Modélisation de notre fonction de récompense. Cette partie pourra être réalisée dans Simulink, ou par une fonction de code ou une combinaison des deux.
- Création de l'agent d'apprentissage, en définissant ses critiques, l'acteur et l'agent lui-même.
- Définition des caractéristiques de notre agent.
- Entraînement de l'agent.
- Simulation du modèle.
- Implémentation dans un modèle réel.

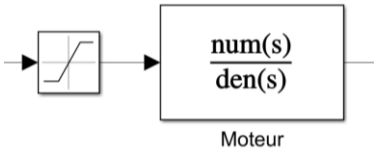
Modélisation de notre système réel (environnement).

Pour générer l'environnement, nous devrons créer un environnement dans Simulink.

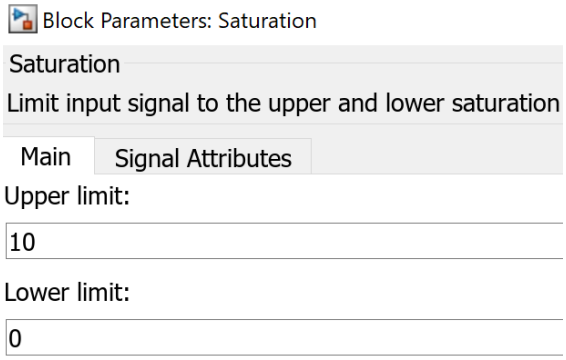


Dans cet environnement, nous devrons implémenter les blocs ou structures suivants :

Modèle réel, dans notre cas nous l'implémenterons par une fonction de transfert.



Dans ce cas, nous ajouterons un bloc de Saturation pour plus de sécurité, car notre moteur n'accepte que des consignes entre [0 et 10].

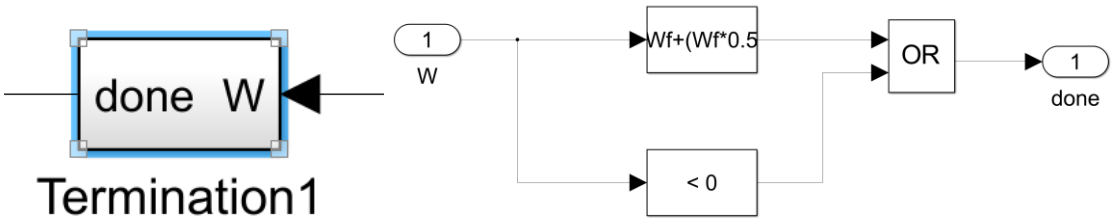


Blocs liés à la récompense de notre système :

Bloc logique :

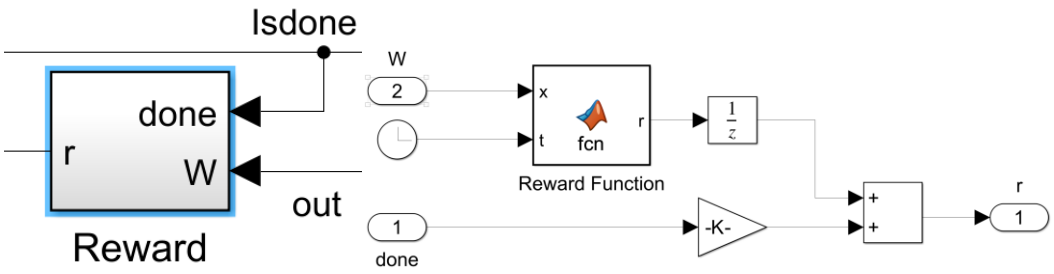
Dans ce bloc, nous définirons les paramètres selon lesquels nous devrons arrêter l'entraînement de l'agent. Dans notre cas, ces conditions seront: une valeur de consigne excessivement supérieure à celle désirée et une valeur de vitesse inférieure à zéro, car une telle vitesse n'est pas possible dans notre système.

Cette valeur logique sera connectée à la valeur d'entrée de « isdone » de l'agent.



Bloc récompense:

Un des blocs les plus importants de le reinforcement learning



Pour le générer, nous prendrons en compte si "isdone" a été activé, ce qui indiquera que notre système a dépassé les limites que nous avions définies précédemment. Cette condition pénalisera plus ou moins notre système en fonction de la valeur que nous attribuons au gain K, qui sera ajoutée à la récompense générée par la fonction décrite ci-dessous.

Ce bloc définira la valeur de la récompense de notre système.

Nous écrirons une fonction en MATLAB qui dépendra de l'objectif de notre entraînement. Dans notre cas, nous voulons modifier la consigne de vitesse d'un moteur électrique comprise [0, 10].

Pour cela, nous allons générer une fonction rewardFunctionRLsystem(x, t) en MATLAB. Dans cette fonction, nous pourrons définir les propriétés que nous voulons que la réponse de notre système ait.

Analyse de la fonction rewardFunctionRLsystem(x, t) :

Nous définissons la fonction reward = rewardFunctionRLsystem(x, t). Cette fonction accepte deux arguments d'entrée : x, la valeur de sortie du modèle, et t, le temps de simulation, et renvoie une valeur scalaire reward, Cette valeur sera celle qui sera implémentée dans la fonction Simulink. .

```
function reward = rewardFunctionRLsystem(x,t)
% REWARDFUNCTION génère des récompenses à partir des spécifications des blocs Simulink.
% x : Input of RLmodel, vitesse du moteur
% t : Temps de simulation (s)
```

Dans cette section, plusieurs variables sont définies qui semblent être des paramètres liés à la simulation et à l'analyse de la réponse du système.

```
%% Simulation and sample times
St=0; %Step time (seconds)
Wo=0; %Initial value
Wf=2; %Final value

Rs=2; %Rise time (seconds)
R=80; % Rise
Stt=5; %Settling time (seconds):
S=5; %Settling%
Ov=10; % Overshoot%
Un=5;% Undershoot%:
Tf = 30;
Ts = 0.1;
```

Ici sont définies les spécifications du système RL (Reinforcement Learning), en utilisant les valeurs définies précédemment pour calculer les limites et caractéristiques telles que le temps de passage, les temps de montée, l'installation, le dépassement et la subordination.

```
%% Spécifications de RLsystem/SystemResponse
Block1_InitialValue = Wo;
Block1_FinalValue = Wf;
Block1_StepTime = St;
Block1_StepRange = Block1_FinalValue - Block1_InitialValue;
Block1_MinRise = Block1_InitialValue + Block1_StepRange * R/100;
Block1_MaxSettling = Block1_InitialValue + Block1_StepRange * (1+S/100);
Block1_MinSettling = Block1_InitialValue + Block1_StepRange * (1-S/100);
Block1_MaxOvershoot = Block1_InitialValue + Block1_StepRange * (1+Ov/100);
Block1_MinUndershoot = Block1_InitialValue - Block1_StepRange * Un/100;
```

Dans ce bloc, les valeurs de Block1_xmax et Block1_xmin sont calculées en utilisant une interpolation linéaire pour obtenir les valeurs maximales et minimales des limites supérieure et inférieure définies précédemment, en fonction du temps de simulation t. Les valeurs maximales et minimales de ces variables seront les mêmes que celles de notre système [0, 10].

```
% Syntonisation des paramètres pour calculer la récompense
if t >= Block1_StepTime
    if Block1_InitialValue <= Block1_FinalValue
        Block1_UpperBoundTimes = [0,Stt; Stt,max(Stt+1,t+1)];
        Block1_UpperBoundAmplitudes = [Block1_MaxOvershoot,Block1_MaxOvershoot;
        Block1_MaxSettling,Block1_MaxSettling];
        Block1_LowerBoundTimes = [0,Rs; Rs,Stt; Stt,max(Stt+1,t+1)];
        Block1_LowerBoundAmplitudes = [Block1_MinUndershoot,Block1_MinUndershoot;
        Block1_MinRise,Block1_MinRise; Block1_MinSettling,Block1_MinSettling];
    else
        Block1_UpperBoundTimes = [0,Rs; Rs,Stt; Stt,max(Stt+1,t+1)];
        Block1_UpperBoundAmplitudes = [Block1_MinUndershoot,Block1_MinUndershoot;
            Block1_MinRise,Block1_MinRise; Block1_MinSettling,Block1_MinSettling];
        Block1_LowerBoundTimes = [0,Stt; Stt,max(Stt+1,t+1)];
        Block1_LowerBoundAmplitudes = [Block1_MaxOvershoot,Block1_MaxOvershoot;
        lock1_MaxSettling,Block1_MaxSettling];
    end

    Block1_xmax = zeros(1,size(Block1_UpperBoundTimes,1));
    for idx = 1:numel(Block1_xmax)
        tseg = Block1_UpperBoundTimes(idx,:);
        xseg = Block1_UpperBoundAmplitudes(idx,:);
        Block1_xmax(idx) = interp1(tseg,xseg,t,'linear',NaN);
    end
    if all(isnan(Block1_xmax))
        Block1_xmax = 10;
    else
        Block1_xmax = max(Block1_xmax,[],'omitnan');
    end
```

```
        Block1_xmin = zeros(1,size(Block1_LowerBoundTimes,1));
        for idx = 1:numel(Block1_xmin)
            tseg = Block1_LowerBoundTimes(idx,:);
            xseg = Block1_LowerBoundAmplitudes(idx,:);
            Block1_xmin(idx) = interp1(tseg,xseg,t,'linear',NaN);
        end
        if all(isnan(Block1_xmin))
            Block1_xmin = 0;
        else
            Block1_xmin = max(Block1_xmin,[],'omitnan');
        end
    else
        Block1_xmin = 0;
        Block1_xmax = 10;
    end
```

Ensuite, une variable Weigt est définie. Cette fonction servira à pénaliser plus ou moins les écarts du modèle par rapport aux paramètres souhaités.

Enfin, la pénalisation Penalty est calculée. Dans cette fonction, nous pouvons la configurer avec une pénalisation 'step' ou une pénalisation 'quadratic'.

Dans le cas de la pénalisation 'step', toutes les déviations sont pénalisées de la même manière, qu'elles soient proches ou éloignées de l'objectif.

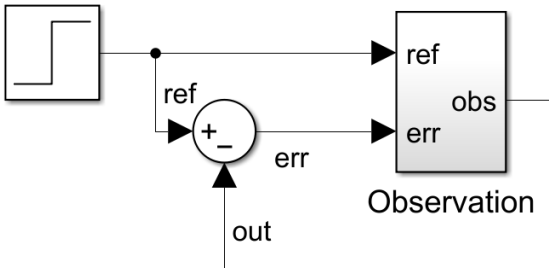
La configuration 'quadratic' pénalise davantage les valeurs les plus éloignées de l'objectif, de manière exponentielle.

Pour notre travail, nous choisirons la deuxième configuration 'quadratic'.

```
%% Poids de la fonction de pénalité (spécifiez une valeur non négative)
Weight = 3;
%Quadratic pénalise les écarts de manière exponentielle
%Step Pénalise de la même manière tous les écarts.
Penalty = sum(exteriorPenalty(x,Block1_xmin,Block1_xmax,'quadratic'));
%% Calculer la récompense
reward = -Weight * Penalty;
end
```

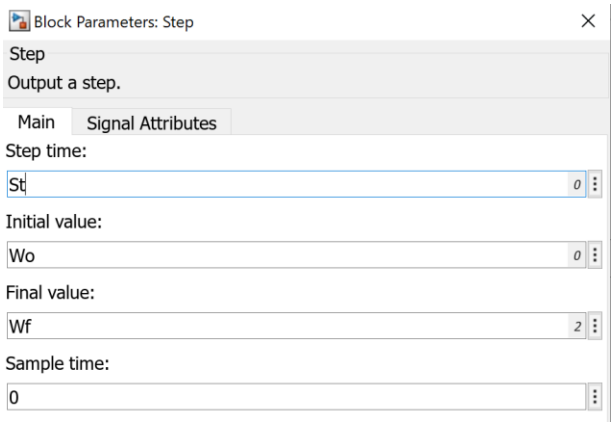
Observations:

Pour générer les observations de notre modèle, nous allons créer la structure suivante.

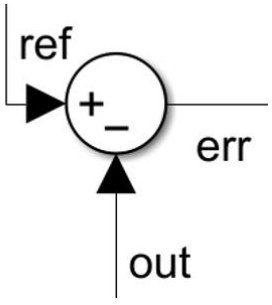


Cette structure sera composée des éléments suivants:

La consigne que nous devons atteindre, représentée dans le bloc step. Dans notre cas, nous ferons un saut de zéro à deux à l'instant zéro. Nous définirons ces paramètres dans notre code MATLAB à l'aide des variables St pour définir l'instant du step, Wo pour définir la valeur inférieure du step, et Wf pour définir la valeur finale du step.

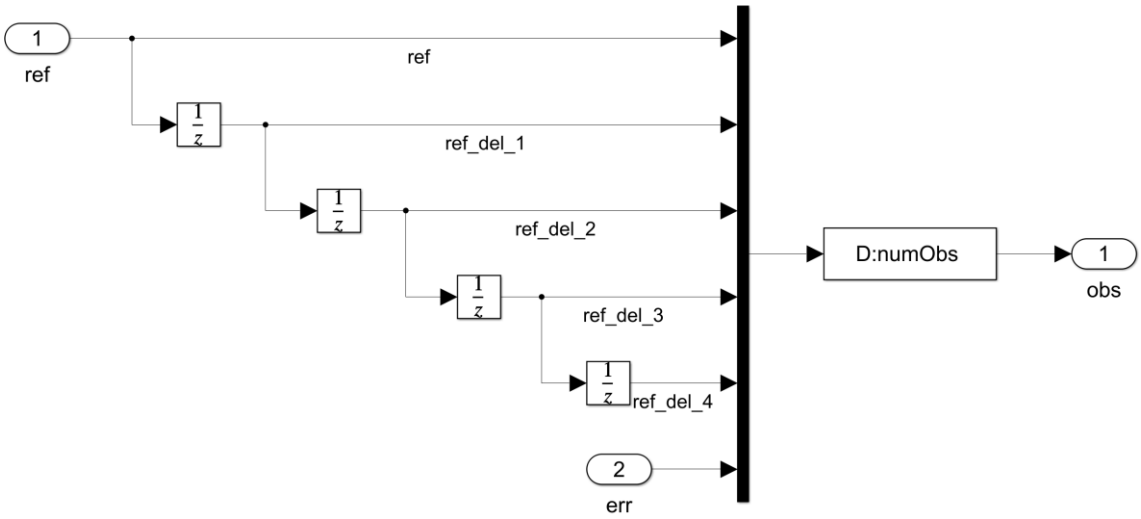


Nous aurons également une erreur de sortie de notre système, c'est-à-dire la différence entre la consigne et la valeur de sortie de notre modèle.

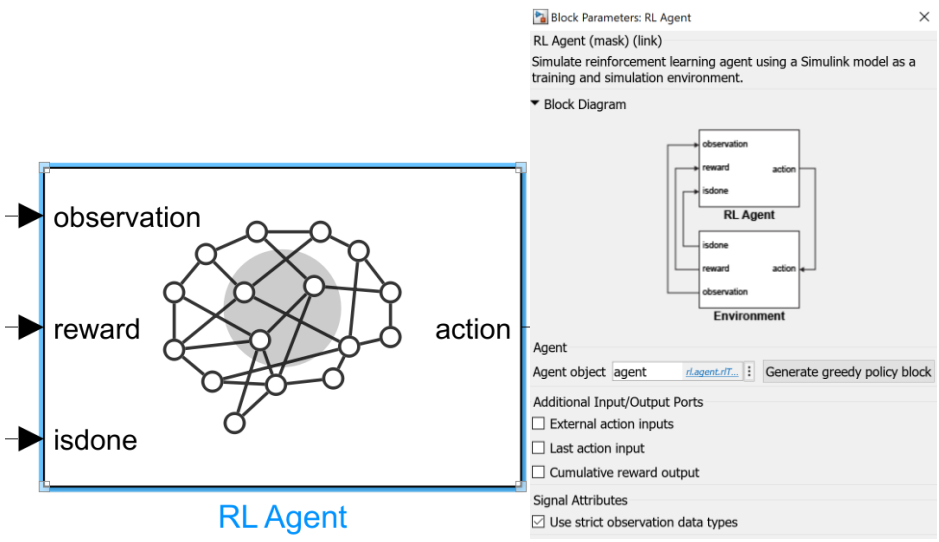


Bloc observateur, pour notre modèle, nous devons définir le nombre d'observations que l'agent reçoit. Dans notre cas, il y en aura six, dont cinq sont définies par la référence. Ces valeurs seront la référence sans délai et des mesures prises avec un délai de 0,1 seconde entre chacune d'elles.

La dernière entrée de l'observateur sera l'erreur, que nous avons déjà calculée précédemment. Cette information sera recueillie dans un vecteur et connectée à l'entrée "Observation" de notre RL agent.



Le bloc RL Agent sera chargé de réaliser l'entraînement de notre moteur. Dans ce bloc, nous devrons sélectionner l'agent à utiliser, que nous aurons préalablement entraîné à l'aide de la boîte à outils Reinforcement Learning Designer.



Deuxième étape, création et définition des éléments dans Reinforcement Learning

Nous créerons ces éléments à l'aide du code MATLAB suivant et de l'outil Reinforcement Learning Designer.

Nous commencerons par définir et expliquer les caractéristiques de notre modèle réel.

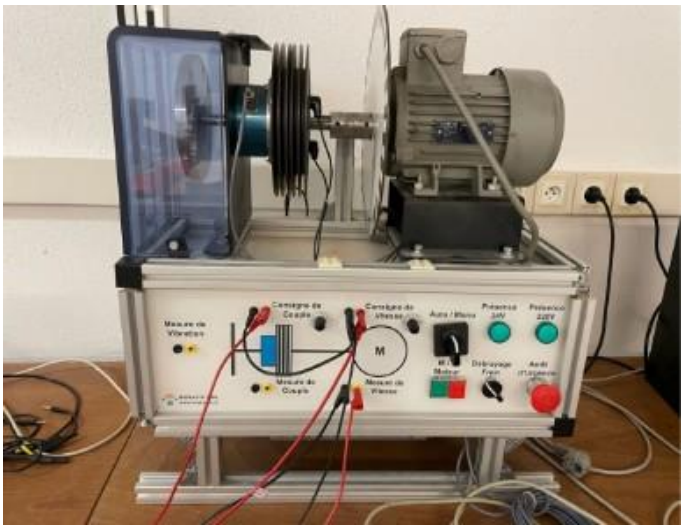
Le système est composé des éléments suivants:

- Un moteur asynchrone d'une puissance de 550W, avec une vitesse nominale de 1500 tr/min à 5 Nm de couple
- Un variateur de fréquence responsable du pilotage du moteur
- Une dynamo tachymétrique utilisée pour mesurer la vitesse de rotation du moteur
- Un couple mètre employé pour mesurer le couple

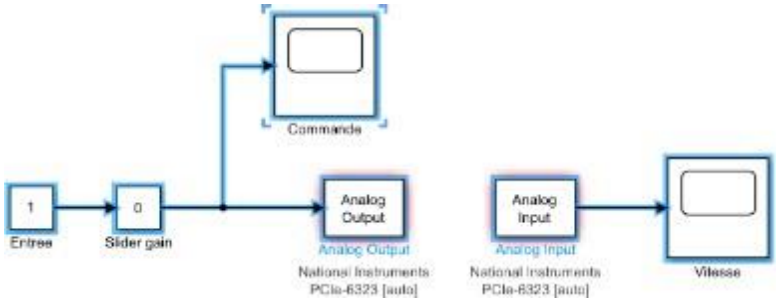
Les alimentations, les interfaces de conversion de signal et autres équipements électriques sont intégrés dans une armoire de commande centralisée.

L'utilisateur dispose de la possibilité de commander le moteur et le frein à poudre soit manuellement, soit automatiquement. À cet effet, sur la face avant de l'armoire de commande, une clé à deux positions permet la sélection du mode de commande.

En mode manuelle, l'utilisateur peut contrôler le moteur et le frein à poudre à l'aide des potentiomètres de commande également situés sur la face avant de l'armoire. En mode automatique, la commande du moteur peut être effectuée à partir de l'ordinateur équipé d'une carte d'acquisition. Dans les deux cas, les niveaux de tension des entrées et sorties varient entre 0 et 10V.



Pour cette étude, la carte électronique employée est la National Instruments PCIe-6323, et l'acquisition des données ainsi que le contrôle du système sont réalisés à l'aide du logiciel Matlab/Simulink.

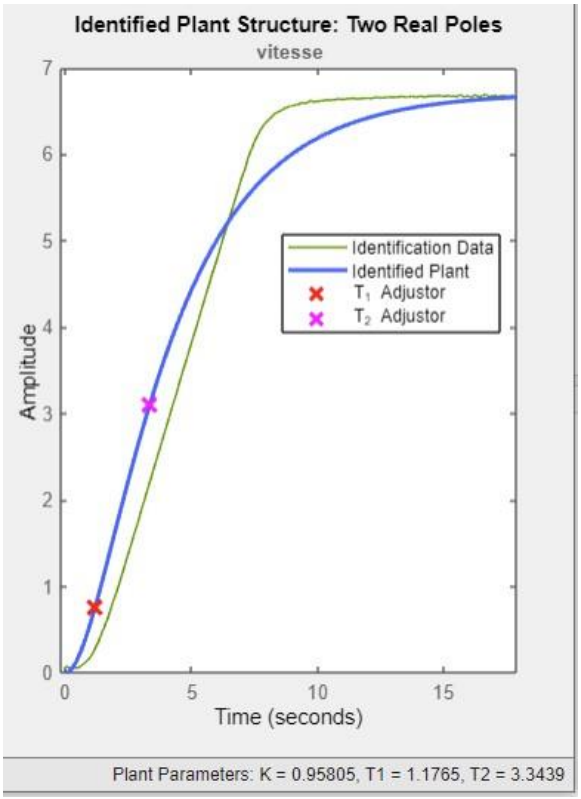


Le module "Analog Output" assure la transmission des commandes au moteur depuis l'ordinateur. De manière similaire, le module "Analog Input" est dédié à la capture des données provenant du capteur de vitesse de rotation.

Ensuite, nous obtiendrons la fonction de transfert de ce modèle afin de pouvoir l'implémenter dans Simulink pour réaliser les simulations.

Modélisation du système en boucle ouverte

Pour construire un modèle du système, avec la tension de commande 'U' en tant qu'entrée et la vitesse de rotation du moteur 'V' en tant que sortie, nous avons identifié ses paramètres à partir d'une réponse indicielle. Pour ce faire, nous avons utilisé l'outil graphique "PID Turner" en considérant un système avec deux pôles.



L'identification a conduit à la fonction de transfert suivante:

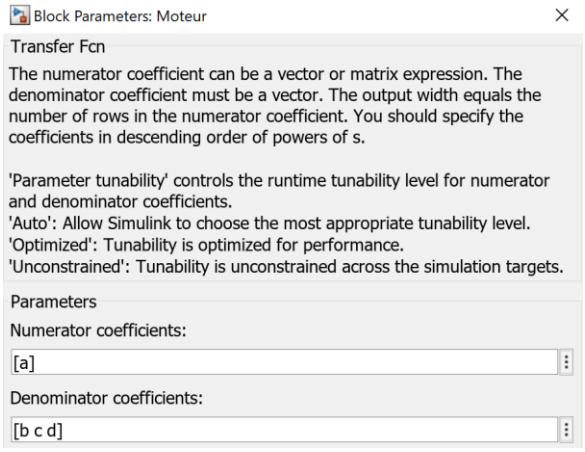
$$G(s) = \frac{V(s)}{U(s)} = \frac{K}{(1+T_1 s)(1+T_2 s)} = \frac{0{,}95805}{(1+1{,}1765s)(1+3{,}3439s)}$$

En développant la fonction de transfert:

$$G(s) = \frac{V(s)}{U(s)} = \frac{0{,}95805}{(1+1{,}1765s)(1+3{,}3439s)} = \frac{0{,}95805}{3{,}3439s^2 + 4{,}5204s + 1}$$

Nous implémenterons cette fonction de transfert dans MATLAB à l'aide du code suivant et du bloc moteur préalablement implémenté dans Simulink.

```
%% RLSYSTEM MOTEUR ELECTRIQUE
%% Représentation d'un moteur électrique par une fonction de transfert
a=0.96;
b=3.3439;
c=4.5204;
d=1;
```



De cette manière, nous pourrons simuler notre modèle réel pour implémenter le Reinforcement Learning.

Définition des paramètres:

Nous définissons les valeurs de notre simulation, notre step dans la consigne, ainsi que les valeurs que nous voulons que la réponse de notre moteur ait. Ces paramètres sont les mêmes que ceux que nous avons implémentés dans la fonction rewardFunctionRLsystem(x, t).

```
%% Paramètres de simulation
St=0; %Step time (seconds)
Wo=0; %Initial value
Wf=2; %Final value

Rs=2; %Rise time (seconds)
R=80; % Rise
Stt=5;%Settling time (seconds):
S=5;  %Settling%
Ov=10;% Overshoot%
Un=5; % Undershoot%:

Tf = 30; %simulation time
Ts = 0.1;%Sample Time
```

Nous démarrons Simulink.

Nous ouvrons notre modèle Simulink ainsi que quelques-uns de ses blocs caractéristiques, ainsi que la fonction rewardFunctionRLsystem.

```
%% OPEN System
open_system('RLmodel')
blk = 'RLmodel/systemResponse';
open_system(blk)
type rewardFunctionRLsystem.m;
open_system('RLmodel/Reward/Reward Function')
```

Polytic Observations

Pour notre modèle, nous définirons six observations, qui correspondent aux observations du modèle Simulink, et une action, qui sera la sortie de notre agent, c'est-à-dire le signal de vitesse qui arrivera au moteur. Cette action sera comprise [0, 10], les consignes de vitesses maximale et minimale de notre moteur.

```
%% Polytic Observations
numObs = 6;
numAct = 1;
oinfo = rlNumericSpec([numObs 1]);
ainfo = rlNumericSpec([numAct 1],LowerLimit=0,UpperLimit=10);
```

Environnement

L'environnement sera celui que nous avons modélisé dans Simulink. Pour le générer, nous devons fournir les informations suivantes : le modèle (c'est-à-dire le code avec toutes les informations de notre modèle), notre modèle Simulink, le nombre d'observations et le nombre d'actions que nous avons définis précédemment.

```
%% Environement
env = rlSimulinkEnv('RLmodel','RLmodel/RL Agent',oinfo,ainfo);
```

Critics

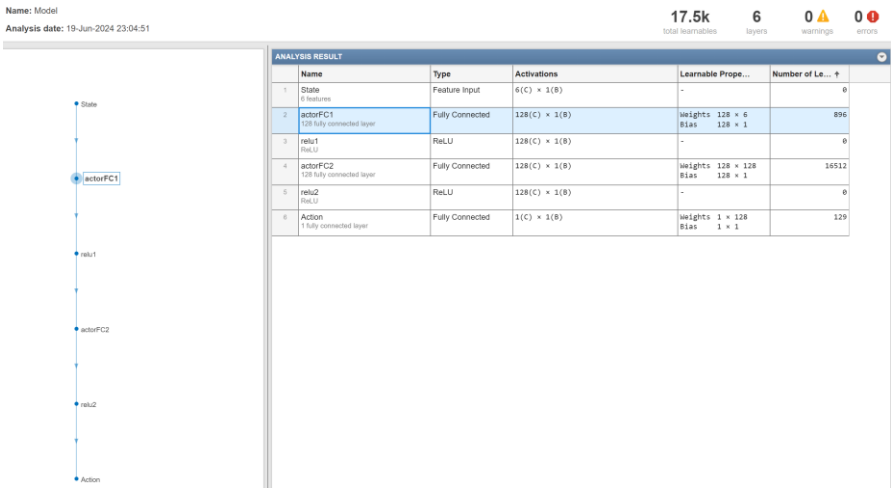
Pour notre modèle, nous allons générer deux critiques identiques, car nous avons choisi un agent TD3.

Nous définissons l'architecture des réseaux neuronaux pour les deux critiques de l'agent TD3, y compris les couches d'entrée, les couches entièrement connectées et les fonctions d'activation.

```
%% Critic representations.
% Paramètres critic
cnet = [
    featureInputLayer(numObs,'Normalization','none','Name', 'State')
    fullyConnectedLayer(128, 'Name', 'fc1')
    concatenationLayer(1,2,'Name','concat')
    reluLayer('Name','relu1')
    fullyConnectedLayer(128, 'Name', 'fc3')
    reluLayer('Name','relu2')
    fullyConnectedLayer(1, 'Name', 'CriticOutput')];
    actionPath = [featureInputLayer(numAct,'Normalization','none', 'Name', 'Action')
    fullyConnectedLayer(8, 'Name', 'fc2')];
    criticNetwork = layerGraph(cnet);
    criticNetwork = addLayers(criticNetwork, actionPath);
    criticNetwork = connectLayers(criticNetwork,'fc2','concat/in2');
    criticOptions = rlRepresentationOptions('LearnRate',1e-3,'GradientThreshold',1);

% Critic_1
critic1 = rlQValueRepresentation(criticNetwork,oinfo,ainfo,...
    'Observation',{'State'},'Action',{'Action'},criticOptions);
% Critic_2
critic2 = rlQValueRepresentation(criticNetwork,oinfo,ainfo,...
    'Observation',{'State'},'Action',{'Action'},criticOptions);
```

Dans notre cas, puisque nous allons implémenter un agent TD3, nous allons créer deux critiques identiques. Nous expliquerons plus tard la raison de ce choix.

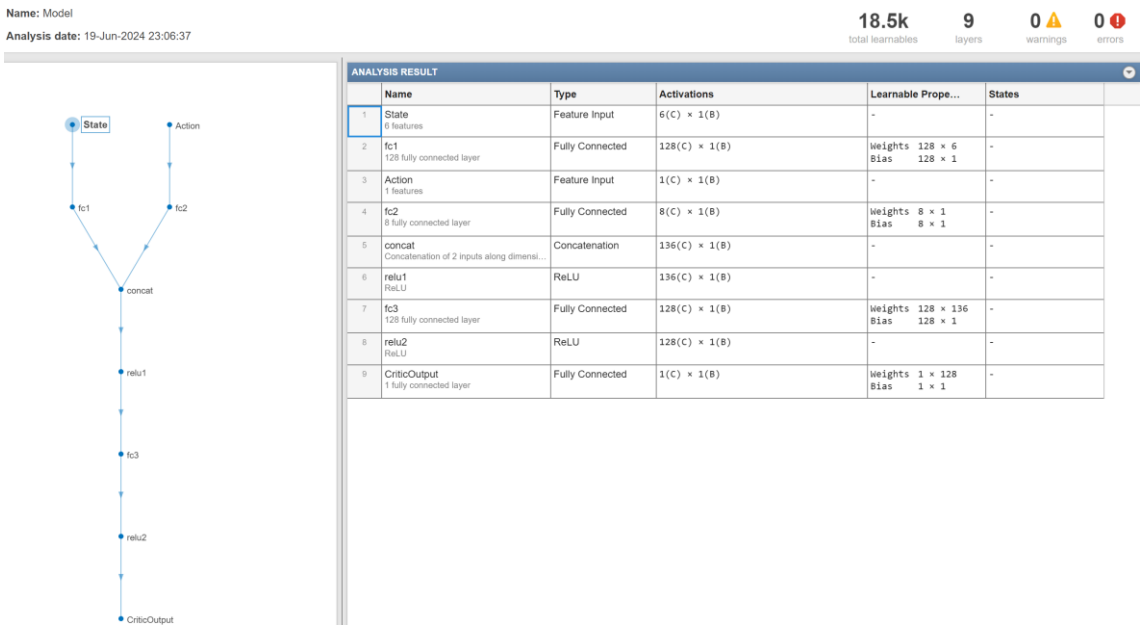
Ci-dessous est présenté le diagramme du critique, où sont reflétés les paramètres que nous avons définis à l'aide du code MATLAB.



Nous définirons une graine RNG pour générer des données aléatoires que nous utiliserons dans notre modèle. Nous créerons également l'acteur (réseau neuronal) pour l'adapter et le concevoir selon un agent TD3, en utilisant les mêmes paramètres que ceux choisis pour les critiques.

```
%% Reinforcement Learning Agent
%random speed
rng(100)
%% Actor
actorNetwork = [featureInputLayer(numObs,'Normalization','none','Name','State')
    fullyConnectedLayer(128, 'Name','actorFC1')
    reluLayer('Name','relu1')
    fullyConnectedLayer(128, 'Name','actorFC2')
    reluLayer('Name','relu2')
    fullyConnectedLayer(numAct,'Name','Action')
    ];
actorOptions = rlRepresentationOptions('LearnRate',1e-3,'GradientThreshold',1);
actor = rlDeterministicActorRepresentation(actorNetwork,oinfo,ainfo,...
    'Observation',{'State'},'Action',{'Action'},actorOptions);
```

Voici le diagramme de l'acteur, où sont reflétés les paramètres que nous avons définis à l'aide du code MATLAB.



ANALYSIS RESULT

	Name	Type	Activations	Learnable Prope...	States
1	State 6 features	Feature Input	6(C) × 1(B)	-	-
2	fc1 128 fully connected layer	Fully Connected	128(C) × 1(B)	Weights 128 × 6 Bias 128 × 1	-
3	Action 1 features	Feature Input	1(C) × 1(B)	-	-
4	fc2 8 fully connected layer	Fully Connected	8(C) × 1(B)	Weights 8 × 1 Bias 8 × 1	-
5	concat Concatenation of 2 inputs along dimensi...	Concatenation	136(C) × 1(B)	-	-
6	relu1 ReLU	ReLU	136(C) × 1(B)	-	-
7	fc3 128 fully connected layer	Fully Connected	128(C) × 1(B)	Weights 128 × 136 Bias 128 × 1	-
8	relu2 ReLU	ReLU	128(C) × 1(B)	-	-
9	CriticOutput 1 fully connected layer	Fully Connected	1(C) × 1(B)	Weights 1 × 128 Bias 1 × 1	-

Agent TD3.

Création de l'agent, où nous définirons ses caractéristiques principales. Plus tard, lors de l'entraînement de l'agent à l'aide de l'outil Reinforcement Learning Designer, nous pourrons modifier les options de l'agent (agentOpts) et leurs implications sur l'entraînement final.

```
%% Agent TD3A
agentOpts = rlTD3AgentOptions("SampleTime", Ts, ...
    "DiscountFactor", 0.99, ...
    "ExperienceBufferLength", 1e6, ...
    "MiniBatchSize", 256);

agentOpts.ExplorationModel.StandardDeviation = 1.0;
agentOpts.ExplorationModel.StandardDeviationDecayRate = 1e-4;
agentOpts.ExplorationModel.StandardDeviationMin = 0.1;

agent = rlTD3Agent(actor,[critic1,critic2],agentOpts);
```

L'agent TD3 est un algorithme de gradient de politique déterministe profond (TD3) avec double retard. C'est une méthode d'apprentissage par renforcement hors politique, en ligne et sans modèle. Un agent TD3 est un agent d'apprentissage par renforcement acteur-critique qui cherche une politique optimale pour maximiser la récompense accumulée attendue à long terme.

Nous choisirons cet agent dans notre modèle car il est optimal pour notre modèle pour les aspects suivants :

- Gestion des actions continues : Il peut apprendre des politiques qui génèrent des actions continues et douces. Dans notre cas, cette action est le contrôle de la vitesse du moteur par sa fréquence d'entrée.
- Stabilisation de l'entraînement : L'agent TD3 introduit un retard dans la mise à jour de la politique, par rapport au critique, ce qui aide à stabiliser l'entraînement en évitant de grandes fluctuations dans la politique apprise.
- Robustesse : L'agent TD3 travaille avec deux critiques identiques, ce qui améliore la robustesse et la stabilité de l'apprentissage, augmentant ainsi la performance générale de notre système.
- Implémentation en temps réel : Cela nous permet de nous adapter aux changements possibles dans l'environnement ou dans le moteur.

Pour ces caractéristiques, nous choisirons l'agent TD3 pour notre modèle.

Entraînement de l'agent

Enfin, nous effectuerons l'une des phases les plus importantes en Reinforcement Learning : l'entraînement de l'agent.

Cet entraînement peut être réalisé soit en utilisant le code MATLAB suivant, soit en utilisant l'outil Reinforcement Learning Designer, qui nous permettra d'analyser plus en détail les données obtenues.

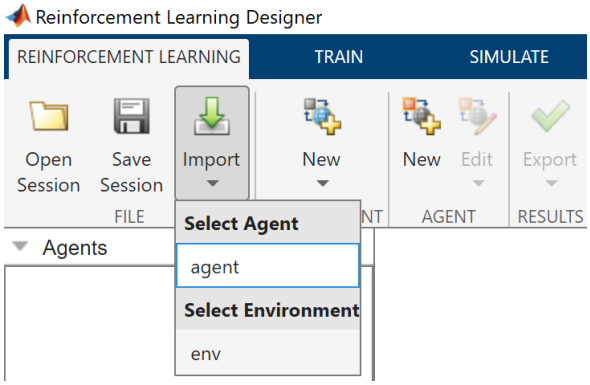
```
%% Train the Agent
save("initialAgent.mat","agent")

trainOpts = rlTrainingOptions(...
    'MaxEpisodes', 500, ...                     % Nombre maximal d'épisodes
    'MaxStepsPerEpisode', ceil(Tf/Ts), ...      % Nombre maximal de pas par épisode (ajuster selon les besoins)
    'StopTrainingCriteria', 'AverageReward', ... % Criterio de parada basado en la recompensa promedio
    'StopTrainingValue', -5, ...                % Critère d'arrêt basé sur la récompense moyenne
    'SaveAgentCriteria', 'EpisodeReward', ...   % Critère pour sauvegarder l'agent
    'SaveAgentValue', -5, ...                   % Sauvegarder l'agent lorsque la récompense de l'épisode est meilleure que -5
    'SaveAgentDirectory', 'bestAgents', ...     % Répertoire pour sauvegarder les meilleurs agents
    'Plots', 'training-progress');              % Afficher la progression de l'entraînement

% Sauvegarder l'agent entraîné
save('finalTrainedAgent.mat', 'agent');

%Fonction pour choisir d'entraîner l'agent (true)
% ou de charger un agent déjà entraîné (false)

doTraining = false;
if doTraining
    trainingStats=train(agent,env,trainOpts);
else
    load('Agent70.mat')
end
```

Entraînement de l'agent avec Reinforcement Learning Designer.

Nous expliquerons ensuite comment réaliser l'entraînement de l'agent avec Reinforcement Learning Designer. Pour ce faire, nous devons suivre les étapes suivantes:

Simuler notre système dans MATLAB pour générer tous les éléments de notre modèle et collecter toutes les informations associées dans l'environnement de travail (Workspace)

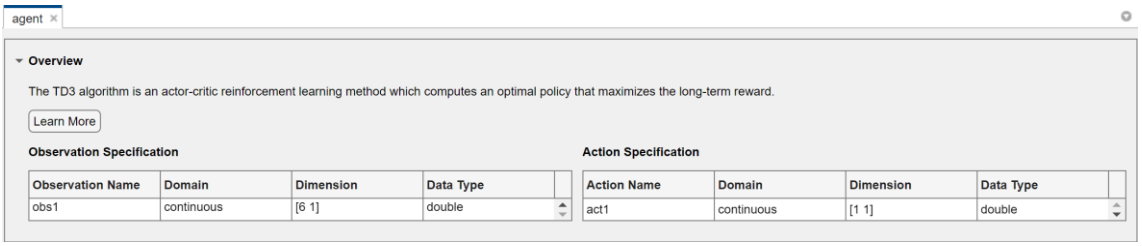
Une fois cette étape réalisée, nous ouvrirons l'application Reinforcement Learning Designer, où nous importerons notre agent et notre environnement.



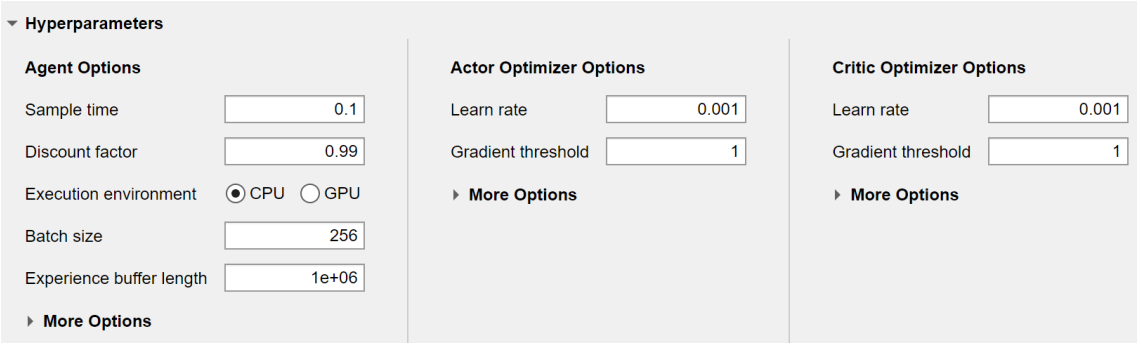
Configuration de l'agent pour l'entraînement.

Dans la section Aperçu (Overview), nous pourrons voir les spécifications d'observation (Observation Specification) et les spécifications d'action (Action Specification).

Ces spécifications sont celles que nous avons déjà définies précédemment dans le code MATLAB.



Ensuite, nous procéderons à la paramétrisation et à l'explication des principaux hyperparamètres.



Agent Options

- Sample time :Le temps d'échantillonnage est l'intervalle de temps entre deux échantillons successifs dans le systeme. Un temps d'échantillonnage court signifie que les données sont collectées à des intervalles de temps très rapprochés, ce qui permet une capture plus précise et détaillée des variations rapides dans le système.
- Discount factor appliqué aux récompenses futures lors de la formation, spécifié sous la forme d'un scalaire positif inférieur ou égal à 1.
- Expérience buffer lenght : Taille du tampon d'expérience, spécifiée sous la forme d'un entier positif. Pendant la formation, l'agent calcule les mises à jour à l'aide d'un mini-lot d'expériences échantillonnées de manière aléatoire dans la mémoire tampon.

Actor Optimizer Options :

• Taux d'apprentissage utilisé pour entraîner l'approximateur de fonction de l'acteur ou du critique, spécifié comme un scalaire positif. Si le taux d'apprentissage est trop bas, alors l'entraînement prend beaucoup de temps. Si le taux d'apprentissage est trop élevé, alors l'entraînement peut atteindre un résultat sous-optimal ou diverger.

• Valeur seuil du gradient utilisée pour entraîner l'approximateur de fonction de l'acteur ou du critique, spécifiée comme Inf ou un scalaire positif. Si le gradient dépasse cette valeur, le gradient est coupé comme spécifié par l'option GradientThresholdMethod. Couper le gradient limite la mesure dans laquelle les paramètres du réseau peuvent changer lors d'une itération d'entraînement.

Critic Optimizer options

• Taux d'apprentissage utilisé pour entraîner l'approximateur de fonction de l'acteur ou du critique, spécifié comme un scalaire positif. Si le taux d'apprentissage est trop bas, l'entraînement prend beaucoup de temps. Si le taux d'apprentissage est trop élevé, l'entraînement peut atteindre un résultat sous-optimal ou diverger.

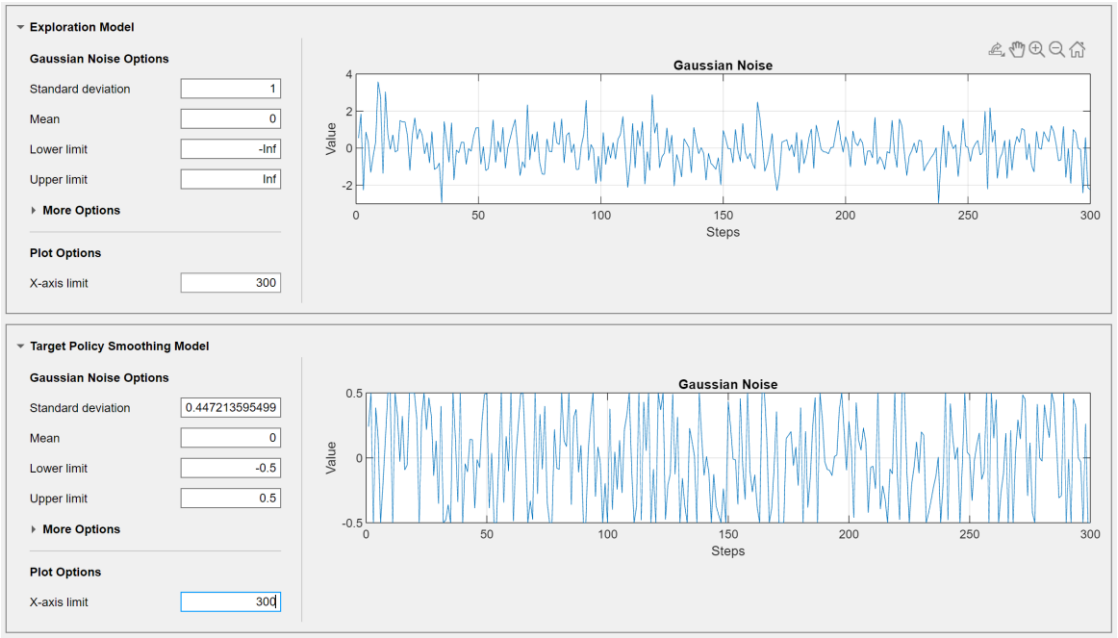
• Valeur seuil du gradient utilisée pour entraîner l'approximateur de fonction de l'acteur ou du critique, spécifiée comme Inf ou un scalaire positif. Si le gradient dépasse cette valeur, il est limité comme spécifié par l'option GradientThresholdMethod. La limitation du gradient réduit la mesure dans laquelle les paramètres du réseau peuvent changer lors d'une itération d'entraînement.

Exploration Model / Taget policy Smoothing Model

Le modèle d'exploration en TD3 implique l'ajout de bruit gaussien à nos actions pour encourager l'exploration dans l'espace des actions. Ce bruit diminue avec le temps pour passer d'une exploration large à une exploration plus précise. Le modèle de lissage de la politique cible en TD3 adoucit les actions prédites par l'acteur en ajoutant du bruit à la fonction Q-cible, ce qui réduit la variance et améliore la stabilité de notre entraînement. Ces approches sont cruciales pour équilibrer l'exploration et l'exploitation dans l'apprentissage par renforcement avec TD3.

- *Déviation standard:* détermine l'ampleur du bruit ajouté aux actions au début de l'entraînement. Une valeur plus élevée permet une exploration plus large au début, ce qui est utile pour découvrir une variété de stratégies potentielles. À mesure que l'agent apprend, cette valeur peut être ajustée pour réduire l'amplitude du bruit et permettre une convergence plus précise vers la politique optimale.

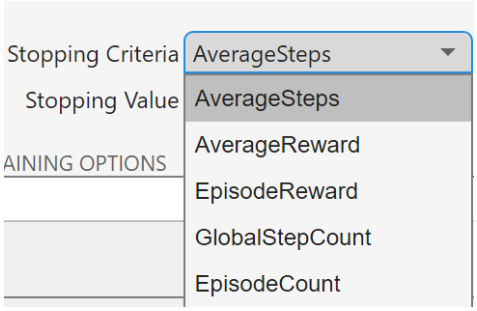
- *Lower limit (limite inférieur), upper limit (limite supérieur) et mean* (valeur moyenne) définissent les caractéristiques du bruit.



Train

Le prochain pas consiste à commencer l'entraînement. Pour cela, nous devrons choisir le système que nous utiliserons, à la fois l'environnement et l'agent, ainsi que le nombre maximum de tentatives que nous effectuerons, ainsi que le nombre maximum d'étapes par entraînement.

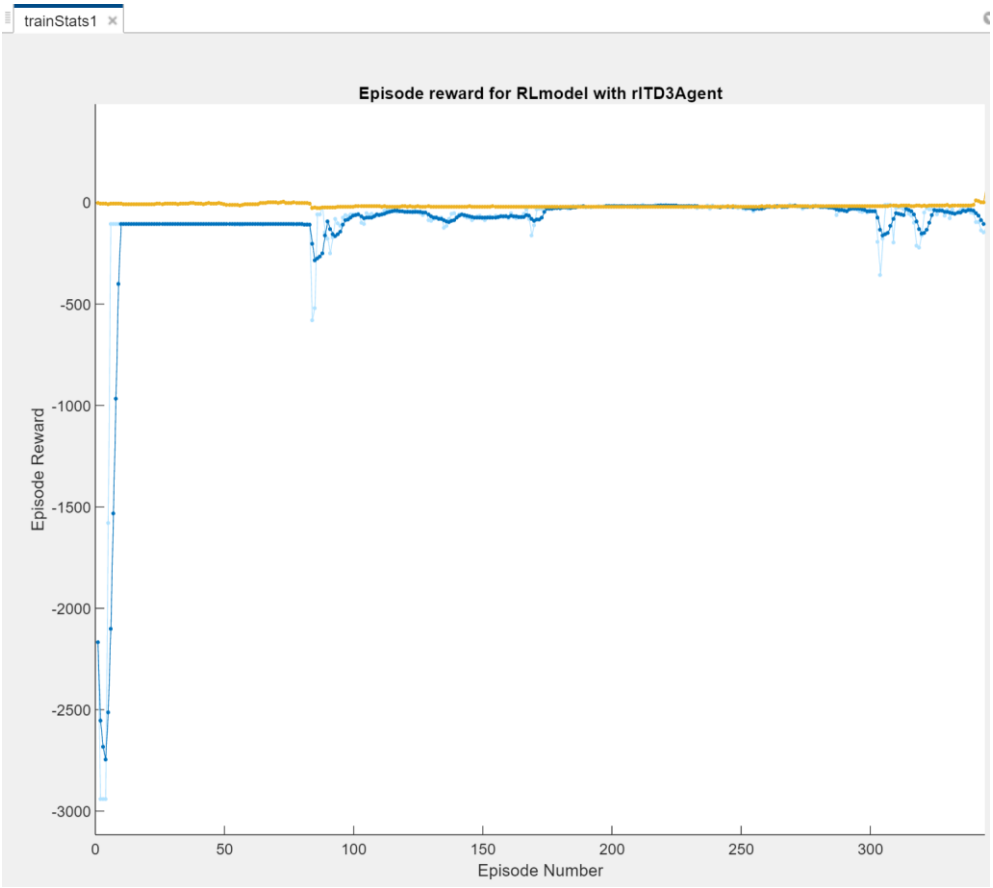
Nous devrons également définir le critère d'arrêt. Nous choisirons parmi les options disponibles, dans notre cas, nous opterons pour AverageReward avec une valeur de 0. Cela signifie que nous arrêterons notre entraînement lorsque la récompense moyenne obtenue pendant l'entraînement sera égale à zéro. De cette manière, nous nous assurerons d'obtenir un agent adapté à notre simulation.



Nous avons réalisé l'entraînement avec l'environnement et l'agent configurés.

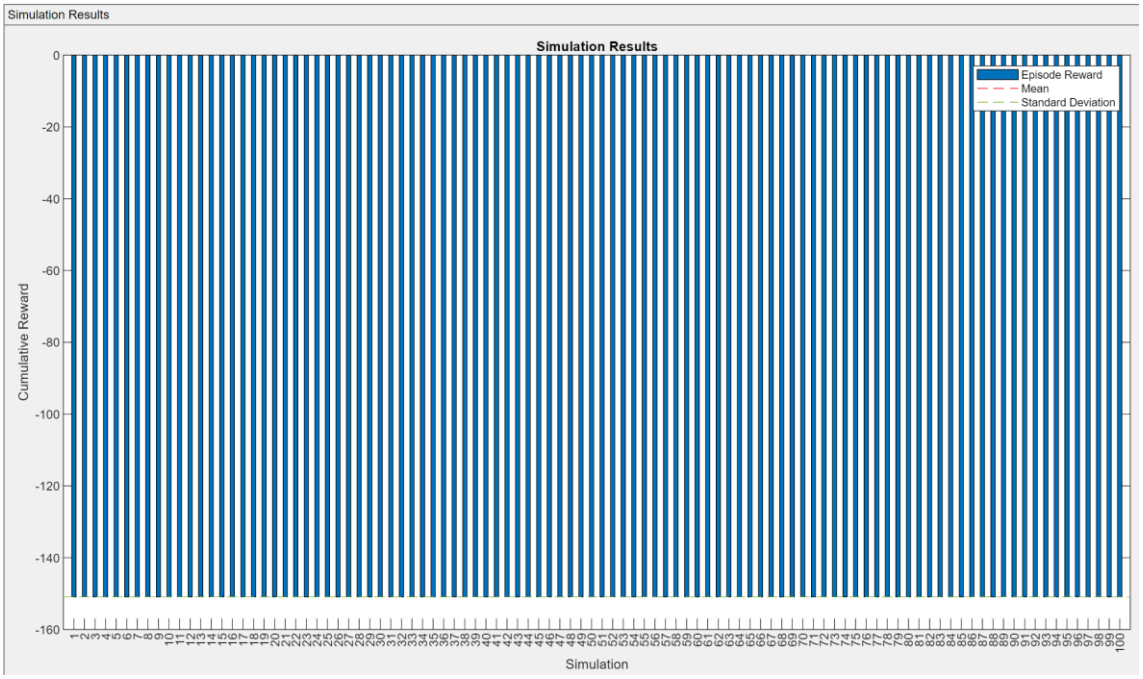
Nous pouvons voir l'évolution de notre entraînement dans ce graphique, où nous observons les éléments suivants :

- **Bleu clair, episode reward:** le rendement immédiat de l'agent pour un épisode spécifique. Une augmentation de la récompense par épisode au fil du temps indique que l'agent apprend à prendre des décisions meilleures et plus efficaces qui maximisent les récompenses.
- **Bleu foncé, average reward:** cette métrique atténue les fluctuations individuelles des récompenses par épisode et fournit une vision plus claire de la tendance générale du rendement de l'agent. Elle est utile pour évaluer les progrès à long terme de l'apprentissage de l'agent.
- **Jaune, episode Q0** : cette valeur reflète les attentes de l'agent concernant les récompenses futures basées sur sa politique actuelle. Une augmentation de la valeur Q au fil du temps suggère que l'agent apprend à mieux prédire les récompenses futures et à prendre des décisions qui maximisent ces récompenses.



Simulation

Une fois que nous avons entraîné l'agent, nous pouvons réaliser une simulation dans laquelle nous pourrons voir la valeur des récompenses obtenues, leur valeur moyenne ainsi que leur tendance moyenne.



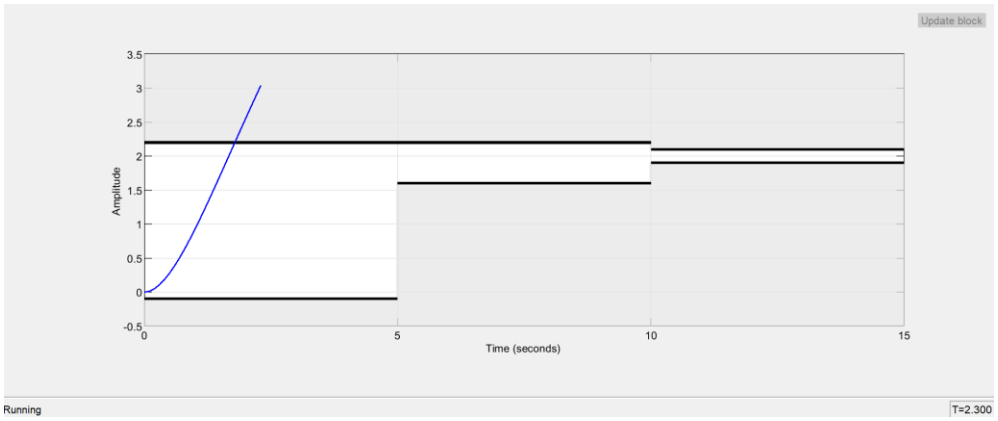
Dans ce cas, nous pouvons constater que l'objectif proposé n'est pas atteint car les valeurs des récompenses sont négatives. Il faudra donc modifier les configurations de notre agent et améliorer son entraînement.

Exemples d'entraînements

Nous présenterons ci-dessous certains des entraînements réalisés par l'agent.

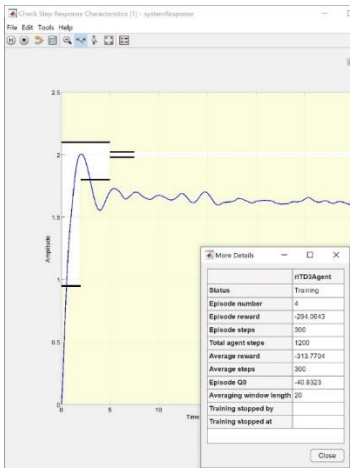
Dans l'entraînement montré ci-dessous, la condition 'isdone' a été activée car la valeur atteinte par la sortie est supérieure à celle que nous avons déterminée dans le bloc logique qui régule la sortie 'isdone'.

Par conséquent, l'entraînement s'arrêtera, les données de ce qui s'est passé seront collectées et un nouvel entraînement commencera.



Dans l'entraînement que nous observons ci-dessous, nous constatons que le bloc logique n'a pas été activé et donc le signal 'isdone' n'a pas été activé, permettant ainsi de terminer l'entraînement complet.

Nous observons que la vitesse du moteur s'écarte des conditions que nous avons déterminées. Par conséquent, nous constatons que cet épisode de l'entraînement a eu une récompense de moins trois cents.



9 Conclusions de l'entraînement

Une fois l'implémentation de l'apprentissage par renforcement pour contrôler un moteur asynchrone terminée, nous avons tiré les conclusions suivantes:

Pour contrôler la vitesse d'un moteur électrique de ces caractéristiques avec une commande variable de 0 à 10V, il n'est pas possible d'implémenter efficacement un contrôle optimal à l'aide de l'apprentissage par renforcement.

Problèmes :

- **Manque d'informations sur le modèle réel** :

  Pour effectuer les simulations, comme nous l'avons constaté en analysant le modèle du moteur électrique à l'aide d'une fonction de transfert, nous obtenons une réponse très similaire mais inadéquate, car nous manquons d'informations sur le moteur réel, ce qui est important pour pouvoir mettre en œuvre un modèle d'apprentissage par renforcemen.

- **Capacité de calcul:**

  Pour réaliser un entraînement correct et approprié de l'agent, des dispositifs avec une capacité de calcul élevée sont nécessaires. Cela permettrait de définir des paramètres tels que le Sample Time de manière plus stricte.

- **Temps de simulation:**

  En raison de la forte demande de calcul nécessaire pour l'entraînement de l'agent, les temps d'entraînement sont très longs. Tout changement apporté à notre modèle d'apprentissage par renforcement ou aux hyperparamètres de l'agent nécessitera de redémarrer la simulation et de recommencer l'entraînement depuis le début.

- **Changements dans l'environnement ou dans la consigne de vitesse:** Les moteurs électriques industriels peuvent fonctionner à différentes vitesses nominales, et leur charge varie également en fonction de l'environnement. Pour pouvoir gérer ces changements, il serait nécessaire d'implémenter un agent différent pour chaque situation, ce qui prolongerait la durée de l'entraînement.

Solutions :

- **Réaliser une modélisation plus précise du moteur électrique réel :** Utilisant des outils de modélisation tels que Power Systems Control Toolbox dans Simulink. Pour implémenter cette solution, nous aurons besoin de plus d'informations, notamment les spécifications du moteur, les paramètres électriques, les caractéristiques de la charge du moteur, ainsi que son contrôle et les variateurs de fréquence (VFD).

- **Dispositifs avec une plus grande capacité:**

  Disposer d'équipements ayant une plus grande capacité de calcul et d'analyse des données. Pouvoir gérer davantage et mieux les simulations que nous effectuons pour résoudre notre problème.

- **Disposer de plusieurs dispositifs:**

  Capables de réaliser plusieurs entraînements simultanément, ou des dispositifs avec une plus grande capacité de calcul et d'analyse de données.

- **<u>Connaître les caractéristiques de le moteur</u>**

  Pour connaître les caractéristiques exactes de notre moteur, il est important de connaître la tension et l'intensité avec lesquelles le moteur est alimenté.

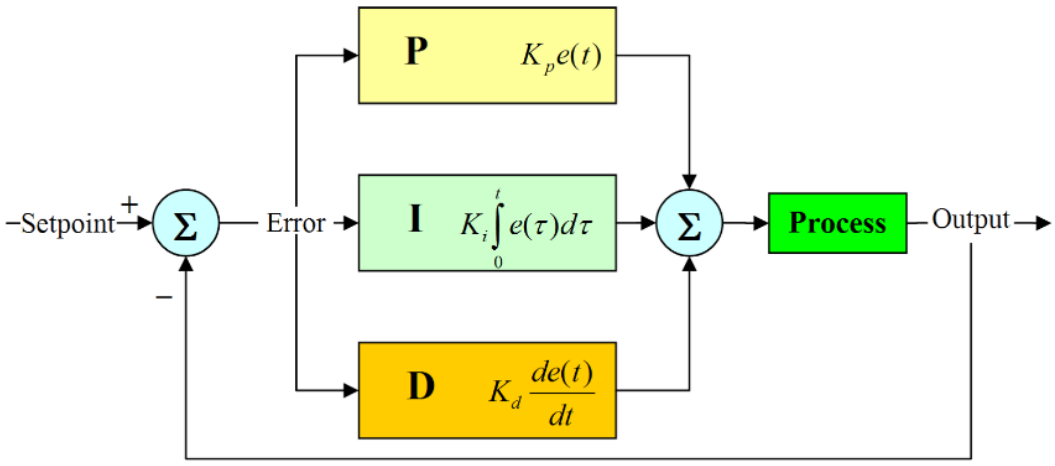
  Le type de commande qui permet de faire varier la vitesse du moteur. Tous les paramètres de résistance interne du moteur, le type de bobinage et d'autres données.

  Il serait également important, une fois ces données connues, de savoir, par le biais d'expériences, si le fonctionnement de notre moteur de laboratoire est correct ou si, par défaut, il ne fonctionne pas correctement et a perdu de sa performance au fil des ans ou a subi un certain type de modification qui n'a pas été enregistré.

10 Possibles options alternatives de contrôle pour un moteur électrique asynchrone :

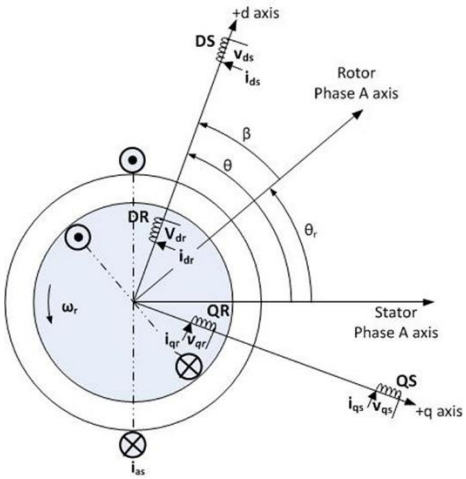
Contrôle de vitesse par PID (Proportionnel-Intégral-Dérivé) :

Le contrôle PID est une méthode classique qui ajuste la sortie du système en fonction de l'erreur actuelle, de l'erreur accumulée dans le temps et du taux de variation de l'erreur. Dans le contexte du contrôle de vitesse d'un moteur électrique, un régulateur PID peut ajuster le signal de commande pour maintenir la vitesse souhaitée en ajustant les paramètres proportionnels, intégraux et dérivés selon les besoins.



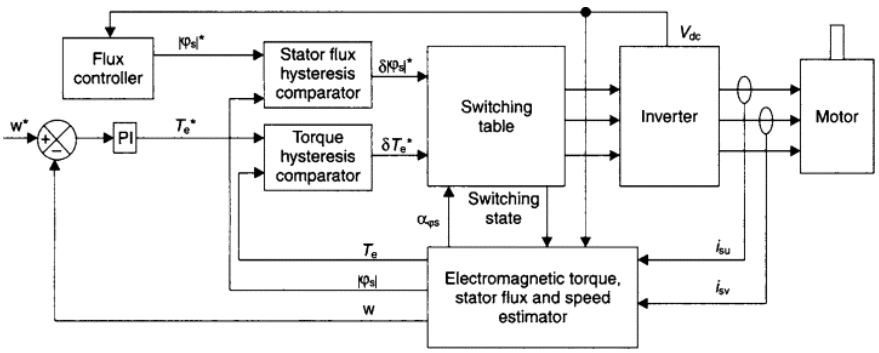
Contrôle vectoriel (FOC ou Field-Oriented Control) :

Le FOC est une technique avancée qui permet un contrôle précis de la vitesse et du couple d'un moteur électrique en orientant le champ magnétique du rotor par rapport au champ magnétique statorique. En utilisant des transformations mathématiques telles que la transformation de Clarke et la transformation de Park, le FOC décompose les courants de phase du moteur en composantes directes et quadrature, permettant un contrôle indépendant du flux magnétique et du couple.



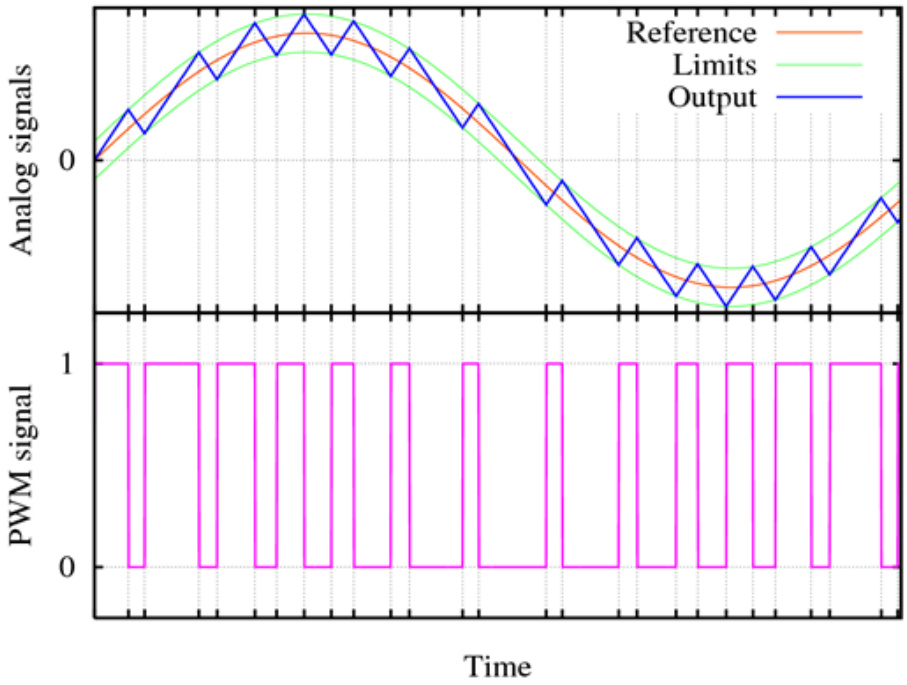
Contrôle direct de couple (DTC ou Direct Torque Control) :

Le DTC est une technique de contrôle qui se concentre directement sur le contrôle du couple et de la vitesse d'un moteur électrique sans nécessiter de transformations de coordonnées. Au lieu de contrôler directement le flux magnétique comme dans le FOC, le DTC ajuste la tension et la fréquence appliquées au moteur pour atteindre le couple désiré. Il est connu pour sa réponse rapide et robuste dans les applications nécessitant des changements rapides de charge et de vitesse.



Contrôle par modulation de largeur d'impulsion (PWM) :

Le PWM est une technique utilisée pour contrôler la vitesse des moteurs électriques en variant la largeur des impulsions d'un signal de tension appliqué au moteur. En ajustant le rapport cyclique des impulsions, la puissance moyenne fournie au moteur, et donc sa vitesse, est contrôlée. MATLAB offre des outils pour simuler et concevoir des systèmes de contrôle PWM, permettant d'ajuster des paramètres tels que la fréquence de commutation et le rapport cyclique pour optimiser les performances du moteur.



11 Conclusión

En conclusión, ce rapport a mis en lumière les résultats obtenus dans le domaine de la commande par machine learning, plus précisément en apprentissage par renforcement, tant dans sa compréhension théorique que dans son implémentation pratique pour le contrôle d'un moteur électrique monophasé asynchrone.

Nous avons également identifié les problèmes que cette technologie peut engendrer et leurs solutions afin d'obtenir les meilleures performances possibles.
Ce travail m'a également permis de connaître les ressources nécessaires pour pouvoir mener à bien un projet dans lequel on applique l'apprentissage automatique, et plus particulièrement l'apprentissage par renforcement pour le contrôle.

Il est très important de réaliser une étude préliminaire du travail à effectuer, ainsi que des outils disponibles dans le laboratoire pour pouvoir étudier si oui ou non nous pouvons utiliser cette technologie, ou si non il serait préférable d'utiliser d'autres techniques de contrôle car elles seront plus efficaces et moins chères.

Ces analyses approfondies fournissent une base solide pour comprendre et mettre en œuvre efficacement différentes stratégies de Reinforcement Learning Toolbox.

Bibliographie

- Bishop, C. M. (2006). Pattern recognition and machine learning. Springer.
- Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep learning. MIT Press.
- Kober, J., Bagnell, J. A., & Peters, J. (2013). Reinforcement learning in robotics: A survey. The International Journal of Robotics Research, 32(11), 1238-1274. https://doi.org/10.1177/0278364913495721
- MathWorks. (2023). Reinforcement Learning Toolbox™ User's Guide. Retrieved from https://www.mathworks.com/help/reinforcement-learning/
- MathWorks. (2023). Reinforcement Learning Toolbox™ - Release Notes. Retrieved from https://www.mathworks.com/help/reinforcement-learning/release-notes.html
- MathWorks. (2023). Train reinforcement learning agents. In MATLAB & Simulink documentation. Retrieved from https://www.mathworks.com/help/reinforcement-learning/train-agents.html
- Murphy, K. P. (2012). Machine learning: A probabilistic perspective. MIT Press.
- Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., ... & Hassabis, D. (2016). Mastering the game of Go with deep neural networks and tree search. Nature, 529(7587), 484-489. https://doi.org/10.1038/nature16961
- Sutton, R. S., & Barto, A. G. (2018). Reinforcement learning: An introduction (2nd ed.). MIT Press.

- Larousse. (n.d.). Dictionnaire français. Retrieved from ttps://www.larousse.fr/dictionnaires/francais.
- Ginguay, M. (s.d.). Diccionario Técnico: Español-Francés / Français-Espagnol. Acribia.
- Alcaraz Varó, E. (s.d.). Diccionario Técnico e Industrial Español-Francés / Français-Espagnol. Ariel.
- Ramírez, L. (s.d.). Diccionario Técnico de Ingeniería y Manufactura Español-Francés. Limusa.
- Office Québécois de la Langue Française. (s.d.). Grand Dictionnaire Terminologique (GDT).