Rapport final projet Rocket Lander

IDOUAR Yacine

Encadrant : SIGAUD Olivier

STERLEY Gilbert Labady

14 mai 2022

Table des matières

1 L'apprentissage pas renforcement **2**

2 Environnement Rocket Lander **4**
2.1 Environnement Gym 4
2.2 Présentation de l'environnement Rocket Lander 5
2.2.1 Introduction 5
2.2.2 Les variables d'apprentissage 5

3 Apprentissage **8**
3.1 Introduction 8
3.2 Stable-Baselines3 8
3.3 Choix des algorithmes 9
3.4 Début de l'apprentissage 9
3.5 Optimisation des hyperparamètres 10
3.6 Vignette 13
3.7 Conclusion 13

4 Analyse de la politique **14**
4.1 La mécanique de la fusée 14
4.2 La fonction de récompense 15
4.3 Observations expérimentales 15

5 Conclusion **24**

6 Références **25**

Chapitre 1

L'apprentissage pas renforcement

L'apprentissage par renforcement est l'une des trois branches principales du machine learning aux côtés de l'apprentissage supervisé et du non-supervisé.

Représentation :

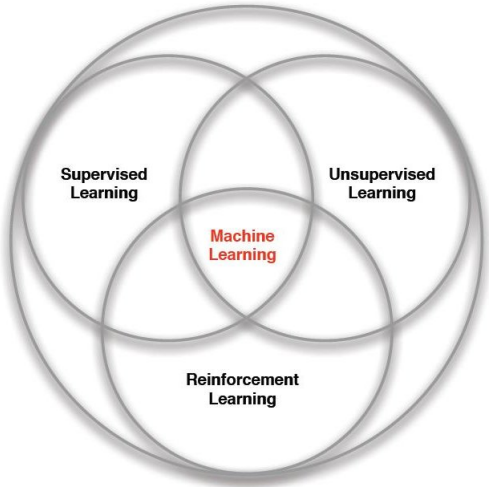

FIGURE 1.1 – Les branches du machine learning

Présentation : l'apprentissage par renforcement consiste à entraîner un agent dans un certain environnement afin de trouver une politique optimale qui permet à ce dernier d'atteindre un objectif fixé, et ce en maximisant la récompense reçue pour chaque action prise.

Pour bien comprendre notre projet, il faut se familiariser avec le vocabulaire de l'apprentissage par renforcement, nous vous présentons alors quelques mots clés importants :

- **Environnement :** L'environnement est l'un des éléments principaux de l'apprentissage par renforcement, il représente le monde dans lequel évolue notre agent. Une bonne compréhension de ce dernier permet de définir une meilleure approche pour l'apprentissage.
- **État :** L'état est la représentation des variables de l'environnement à chaque étape de l'apprentissage.
- **Agent :** L'agent est le l'élément sur lequel porte l'apprentissage.
- **Action :** Les actions sont les décisions prises par l'agent afin d'évoluer dans son environnement et atteindre son objectif.
- **Politique :** La politique désigne la fonction qui relie un état à une action correspondante $\pi(s) = a$.
- **Récompense :** La récompense est l'évaluation de l'action prise pour un état.

Schéma récapitulatif :

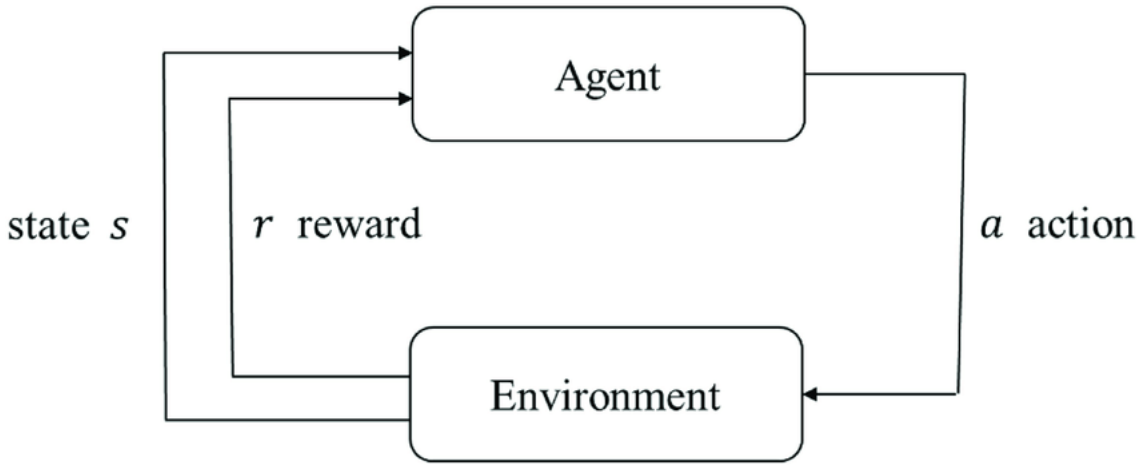

FIGURE 1.2 – Interaction agent-environnement

Chapitre 2

Environnement Rocket Lander

2.1 Environnement Gym

Comme précisé dans la première partie, l'environnement est un élément central dans un problème d'apprentissage par renforcement. La bonne définition de ce dernier permet d'avoir des apprentissages pertinents. OpenAI Gym est une bibliothèque python qui permet de standardiser la création d'environnement personnalisé de RL. Elle implémente toutes les variables présentées au premier chapitre et permet d'avoir un suivi précis de ces dernières à chaque étape d'apprentissage.

```
Action Taken    [-0.7947902  -0.00329512  0.5604856 ]
Observation     [ 1.35146243  1.79460442  0.1288576  -0.05004346 -0.05004346 -1.27120465
  0.05156126 -0.03989081 -0.51553193 -1.71810623]
Reward Gained   -0.0051874151094266395
Info            {}

Action Taken    [-0.9683982  -0.5538641   0.11725986]
Observation     [ 1.34933253  1.78275255  0.09957284 -0.05004346 -0.05004346 -1.27120465
  0.03080139 -0.03989147 -0.52395283 -1.71806342]
Reward Gained   -0.003531798452199434
Info            {}

Action Taken    [-0.9448901  -0.25661004 -0.26830107]
Observation     [ 1.34720303  1.77087851  0.07028893 -0.05004346 -0.05004346 -1.27120465
  0.01054548 -0.03989455 -0.5323741  -1.7180192 ]
Reward Gained   -0.0035357187249955993
Info            {}
```

FIGURE 2.1 – Exemple d'un run sur l'environnement Rocket Lander

Elle offre aussi une interface graphique aux environnements, cette dernière est très importante lors d'un apprentissage, car elle représente le premier outil de visualisation afin d'observer l'amélioration de la politique de l'agent.

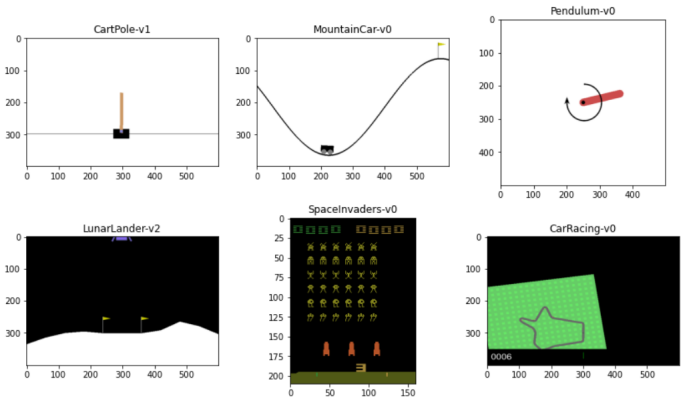

FIGURE 2.2 – Interface graphique de quelques environnements gym

2.2 Présentation de l'environnement Rocket Lander

2.2.1 Introduction

Rocket Lander est un environnement gym continu qui a pour but de faire descendre une fusée verticalement sur une plateforme à la manière d'une falcon9 de SpaceX, et ce en appliquant des méthodes d'apprentissage par renforcement.

FIGURE 2.3 – Descente Falcon9 de space X

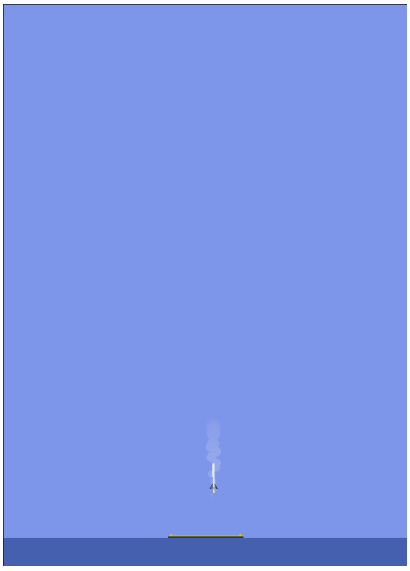

FIGURE 2.4 – Fenêtre de L'environnement Rocket Lander

2.2.2 Les variables d'apprentissage

Pour bien choisir les méthodes d'apprentissage par renforcement à appliquer sur l'environnement, une bonne connaissance de ce dernier est très importante. Nous allons traiter dans cette section les paramètres de Rocket Lander.

Les variables d'état

Les variable d'états représentent la vision qu'a notre agent de l'environnement, dans notre cas l'agent est le contrôleur de la fusée et l'état est défini par :

- La position de la fusée (x,y).
- L'angle de la fusée par rapport à la verticale Θ.
- La vitesse de descente de la fusée, Vélocité_linéaire(x,y) (Translation).
- La vitesse angulaire de la fusée, Vélocité_angulaire (Rotation).
- La valeur de l'angle du cardan moteur Φ.
- La poussée du moteur principale (Manette à gaz).
- Le contact des pieds de la fusée avec la plateforme (Valeur booléenne).

Ces variables sont mises dans un vecteur nommé "vecteur d'état" en apportant quelques modifications sur leurs valeurs :

$$S = \begin{bmatrix} 2 * pos_x \\ 2 * pos_y \\ Angle(\Theta) \\ Pied\_1\_platefrome\_contact \\ Pied\_2\_platefrome\_contact \\ 2 * (Manette\_a\_gaz - 0.5) \\ \frac{Angle\_cardan(\Phi)}{Valeur\_sueil\_angle\_cardan} \\ Velocit\_lineraire[x] \\ Velocit\_lineraire[y] \\ Velocit\_angulaire \end{bmatrix}$$

La valeur de seuil de l'angle du cardan est une constante = 0.4.

Les actions

Les actions représentent les décisions prises par l'agent, en l'occurrence le contrôleur de la fusée en fonction de l'état dans lequel il se trouve. Le contrôleur peut agir sur 3 paramètres qui sont :

- L'angle du cardan moteur Φ : qui travaille à déplacer la fusée horizontalement (sur l'axe x).
- La direction de la force latérale (Force_dir) : La fusée possède deux moteurs latéraux qui génèrent des impulsions qui ont pour but de rééquilibrer l'angle de cette dernière par rapport à l'axe vertical (y).
- La valeur de la manette à gaz : gère la puissance du moteur principal de la fusée qui a pour but de ralentir la chute.
- La fusée peut ne pas faire d'action.

Remarque : Les valeurs des actions sont toutes continues et se trouvent dans l'intervalle [-1,1].

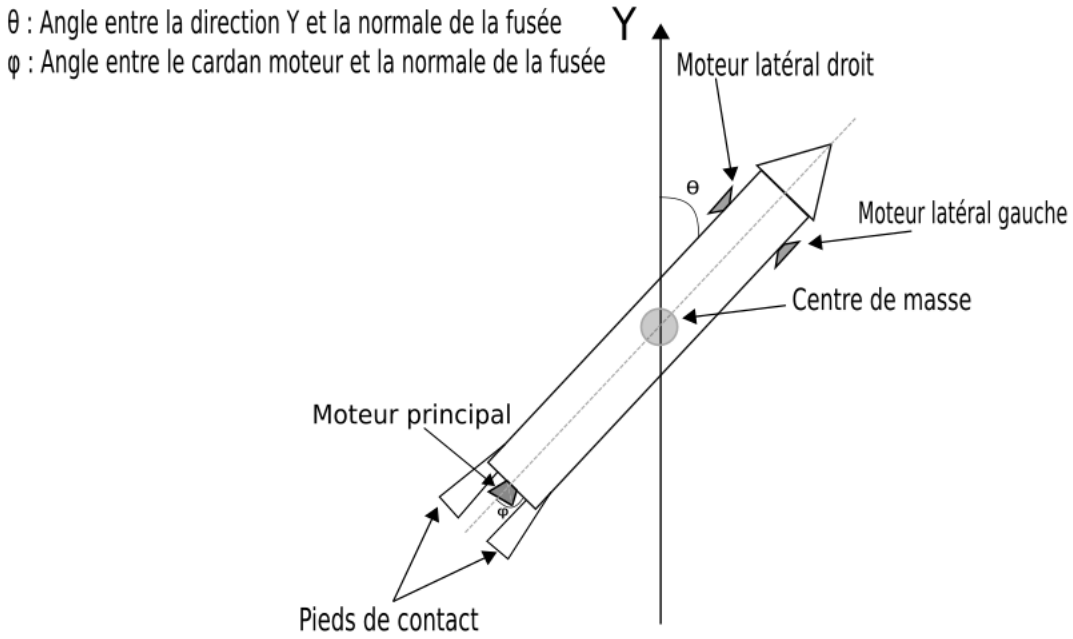

FIGURE 2.5 – Schema de la fusee en vol

Les récompenses

La fonction de récompense évalue la politique adoptée par l'agent et essaye de la diriger vers un comportement optimal. S'intéresser à cette dernière permet de comprendre l'objectif de l'apprentissage et comment il est dirigé afin d'atteindre cet objectif.
Pour l'environnement Rocket Lander l'objectif est clair, faire descendre la fusée à la verticale au-dessus de la plateforme. Nous allons alors analyser sa fonction de récompense.

Les variables utilisées :

$$Module\_position = norm(3 * x, y) \tag{2.1}$$

Pour la récompense on donne un poids supérieur à la position en x.

$$Vitesse = norm(Velocite\_lineaire[x], Velocite\_lineaire[y]). \tag{2.2}$$

Le module de la vitesse de translation de la fusée.

$$Cout\_carburant = \frac{0.1 * (\epsilon * puissance + |Force\_dir|)}{Constante} \tag{2.3}$$

La variable "puissance" décrit la puissance du moteur principal de la fusée, elle dépend de la valeur de la manette à gaz.
La variable "Force_dir" indique lequel des deux moteurs latéraux a été sollicité.

$$Atterrissage = Pied\_1\_platefrome\_contact \;\; ET \;\; Pied\_2\_platefrome\_contact \;\; ET \;\; vitesse < 0.1 \tag{2.4}$$

Une valeur booléenne indiquant si la fusée a bien atterri.

Initialisation :

récompense = -Cout_Carburant

Nous remarquons que le cout_carburant est compté négativement dans la récompense.

Définition du shaping : $shaping = -0.5 * (distance + vitesse + |angle|^2)$
Le shaping oriente la valeur de la récompense en fonction des variables pertinentes qui vont la mener vers le résultat final souhaité.
Le shaping pour notre environnement dépend de 3 variables en particulier :

- Module_position.
- Vitesse.
- Angle Θ.

Mise à jour de la récompense : récompense += shaping - prev_shaping
prev_shaping est la valeur du shaping à l'état précédent.

Si l'atterrissage est réussi : récompense = 1.

Chapitre 3

Apprentissage

3.1 Introduction

Nous connaissons maintenant notre environnement, ses états, les actions de l'agent et la fonction de récompense. Nous pouvons alors commencer l'apprentissage dans le but de faire atterrir notre fusée.

3.2 Stable-Baselines3

Stable-baselines3 est une bibliothèque python qui implémente un ensemble de fonctions (algorithmes) d'apprentissage par renforcement, ces dernières sont compatible avec les environnements openAI Gym présentés plus haut. L'outil permet de lancer des apprentissages de manière simplifiée en utilisant le terminal.

```
python train.py --algo algo_name --env env_id
```

FIGURE 3.1 – Exemple lancement d'un apprentissage

SB3 permet aussi de modifier les hyperparamètres (nous traiterons ce sujet un peu plus loin) afin d'améliorer les performances de l'algorithme, en plus d'un ensemble de fonctions annexes qui permettent de suivre l'avancement de l'apprentissage, par exemple la possibilité de dessiner des graphiques afin de visualiser des paramètres importants, tel que la récompense moyenne sur un ensemble de tentatives et la possibilité de sauvegarder la politique avec le meilleur résultat.

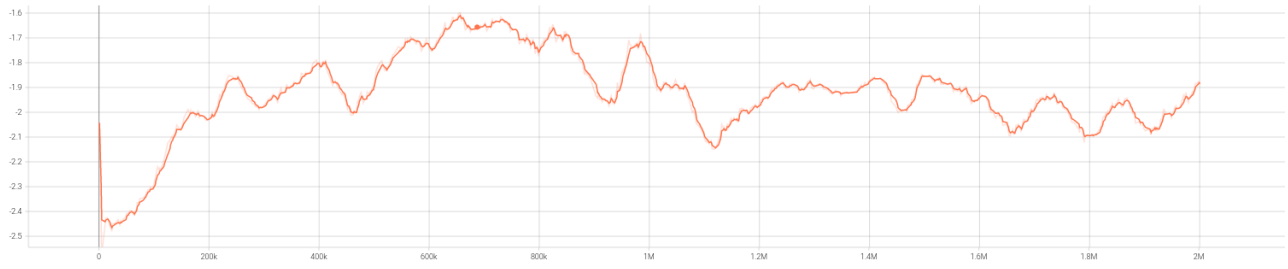

FIGURE 3.2 – Récompense moyenne en fonction du nombre de tentatives

3.3 Choix des algorithmes

Comme dit plus haut, SB3 offre un ensemble d'algorithmes d'apprentissage tous implémentant différents principes mathématiques assez complexes que nous n'allons pas traiter dans notre projet. Alors au moment du choix pour l'environnement Rocket Lander, nous avons dû travailler avec nos connaissances limitées de ces principes. L'idée que nous avons suivi était de chercher des environnements bien documentés qui ressemblent à Rocket Lander, nous avons alors trouvé Lunar Lander Continuous qui a pour but de faire descendre un atterrisseur lunaire et qui ressemble en tout point à Rocket Lander, l'algorithme utilisé pour ce problème était Proximal Policy Optimization (PPO). En plus de ça notre encadrant nous a aussi conseiller un autre algorithme Truncated Quantile Critics (TQC).

3.4 Début de l'apprentissage

Les algorithmes d'apprentissage par renforcement repose sur un principe d'exploration-exploitation. Au début l'algorithme choisit des valeurs aléatoires pour son réseau de neurone, puis il effectue un ensemble de réajustement dans une zone restreinte en modifiant les valeurs avec un facteur ϵ très petit (exploitation)dans le but d'améliorer la politique. Ensuite, de manière périodique, il change complètement les valeurs (exploration) du réseau et il compare les résultats de la nouvelle zone explorer avec l'ancienne, si la politique est meilleure dans la nouvelle zone alors le programme exploite cette dernière, sinon il retourne vers l'ancienne.
La première approche de l'apprentissage consiste à laisser l'algorithme tourner en augmentant le nombre de tentatives en espérant avoir des résultats prometteurs. Nos deux algorithmes ont fourni des résultats prometteurs lors des premiers apprentissage où l'on peut observer un comportement haussier de la récompense en fonction du nombre de tentative.
Après une semaine de tentatives, nous vous présentons les deux meilleures graphiques obtenus pour les deux algorithmes.

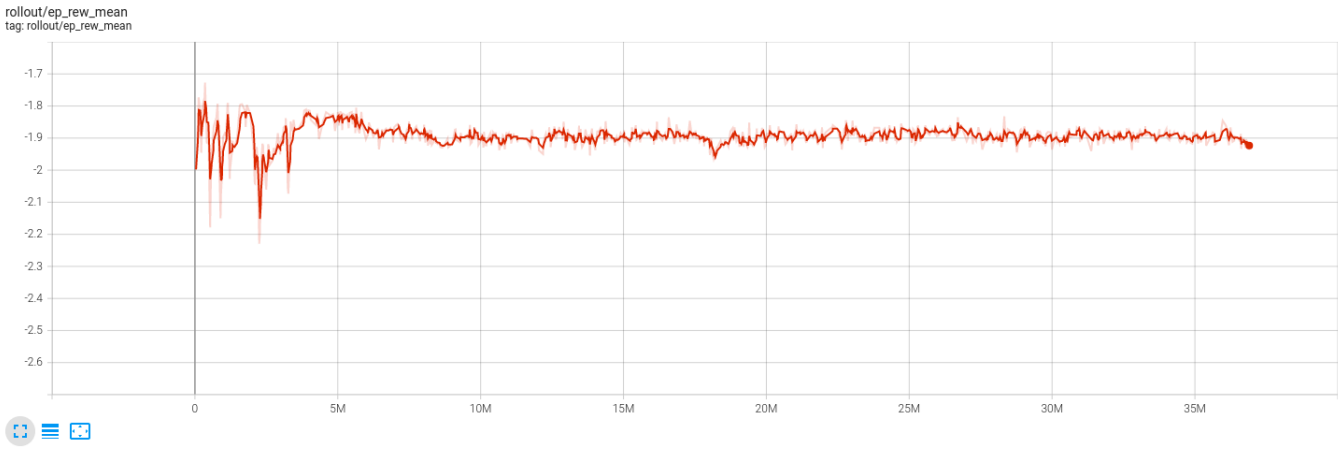

FIGURE 3.3 – Récompense moyenne en fonction du nombre de tentatives pour PPO avec une meilleure récompense à -1.74.

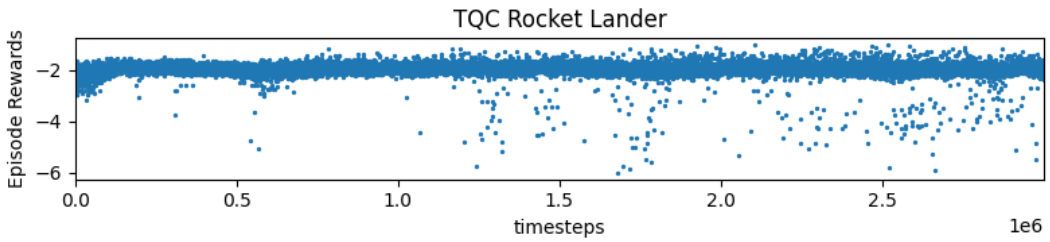

FIGURE 3.4 – Récompense moyenne en fonction du nombre de tentatives pour TQC avec une meilleure récompense à -1.82.

Remarques :
— On remarque très vite qu'à partir d'un certain nombre d'itérations la récompense reste sur un niveau fixe et n'augmente plus, ce qui montre les limites de cette approche.
— Le résultat moyen de PPO est meilleur que celui de TQC sur la meilleure tentative et en valeur moyenne, nous avons alors décidé de continuer avec PPO.

3.5 Optimisation des hyperparamètres

Les hyperparamètres sont des paramètres réglables qu'on utilise pour ajuster les algorithmes d'apprentissage par renforcement. Ils peuvent porter sur les coefficients des fonctions mathématiques, ou la structure du réseau de neurone.
L'optimisation des hyperparamètre est une étape importante dans un apprentissage car elle permet d'améliorer les résultats obtenus. On retrouve essentiellement deux méthodes afin de faire l'optimisation :
— Une modification manuelle : c'est une méthode purement statistique car il n'existe aucun moyen mathématique pour s'orienter dans les choix. Elle consiste à modifier les paramètres puis lancer des apprentissages et vérifier si la politique se comporte mieux. L'ensemble des possibilités est infinie ce qui rend cette méthode très fastidieuse.
— Utiliser des outils d'optimisation : il existe des outils spécialisés dans l'optimisation des hyperparamètres, c'est l'automatisation de la première méthode avec quelques fonctions mathématiques qui permettent une utilisation intelligente des statistiques obtenues afin de converger plus rapidement vers les valeurs optimales des paramètres.

Optuna : est un outil d'optimisation des hyperparamètres supporté par stables-Baseline3, qu'on peut directement lancer sur le terminal.

```
python3 train.py --algo PPO --env RocketLander-v0 -optimize
```

FIGURE 3.5 – Exemple lancement d'une optimisation des hyperparamètres en utilisant Optuna

Résultats de l'optimisation en utilisant Optuna :

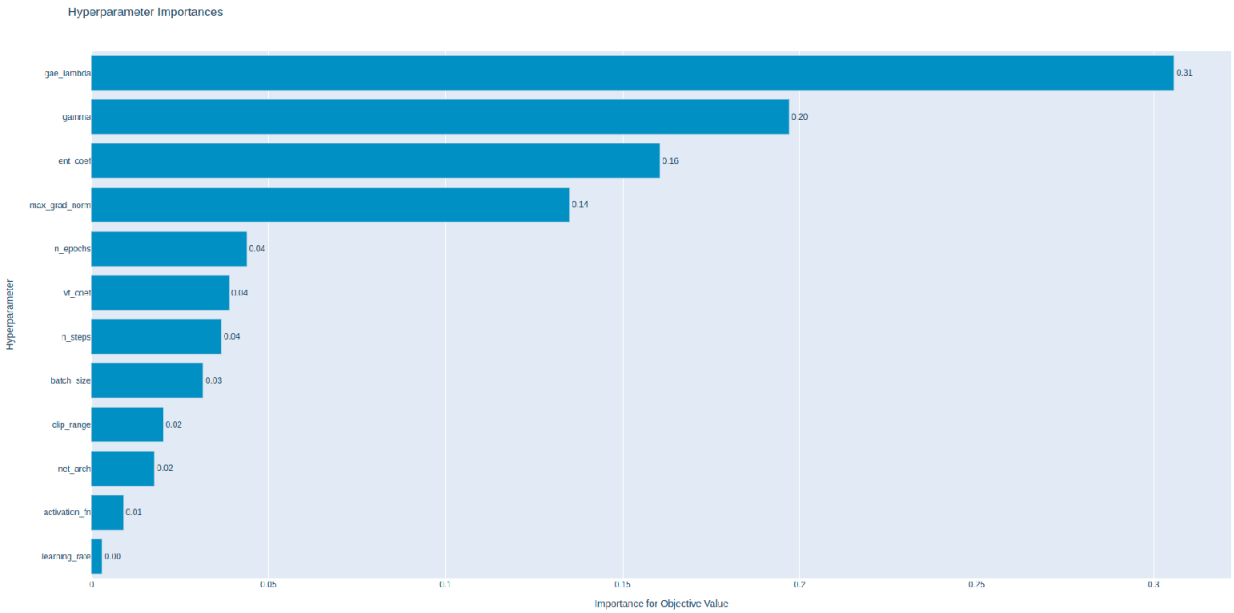

FIGURE 3.6 – Importance des différents hyperparamètres

Explication : La figure montre les hyperparamètres qui ont le plus d'influence sur le changement des résultats de l'apprentissage.

Rewards	batch_size	n_steps	gamma	learning_rate	ent_coef	clip_range	n_epochs	gae_lambda	max_grad_norm	vf_coef	net_arch
-1.688068	128	1024	0.999	0.00010464624	3.57998661e-06	0.3	10	0.98	2	0.1032071476	medium
-1.68919499	128	1024	0.999	0.00013378589	6.78225033e-05	0.3	10	0.98	0.5	0.2385517878	medium
-1.7327342	128	1024	0.999	0.00016843134	0.000418948	0.3	20	0.98	0.7	0.3300335467	medium
-1.7375526	256	1024	0.999	1.08236817e-05	0.00035598516	0.3	20	1.0	0.3	0.1310345850	small
-1.73755260	256	1024	0.999	1.08236817e-05	0.00035598516	0.3	20	1.0	0.3	0.1310345850	small
-1.7447218	256	1024	0.999	0.00039575442	1.11357232e-06	0.3	10	0.98	0.3	0.50190573518	small
-1.75108	16	512	0.9999	2.0162429e-05	2.58850822e-06	0.3	20	1.0	0.7	0.59474644719	medium
-1.766511	256	1024	0.999	5.86606611e-05	8.04513874e-05	0.3	20	1.0	0.3	0.411239053	small
-1.794434	512	512	0.999	8.85830056e-05	2.00078151e-06	0.3	20	1.0	0.7	0.7074654450	medium
-1.7964576	64	512	0.9999	1.15892693e-05	2.25114677e-06	0.3	20	1.0	0.7	0.2873724832	medium
-1.8088089999	32	512	0.999	1.49230219e-05	1.55977450e-06	0.3	20	1.0	0.7	0.07448117293	medium
-1.809951	32	16	0.999	0.000150437	0.003552594	0.1	20	0.9	0.7	0.1330886028	medium
-1.8229174000	32	512	0.999	1.18726088e-05	0.00441662863	0.3	20	1.0	0.7	0.05363714746	medium
-1.8317352	16	256	0.999	1.22454598e-05	0.00311285394	0.2	10	0.98	2	0.422029	small
-1.8467956	32	1024	0.999	0.00014870140	0.0002182740	0.3	5	1.0	0.8	0.10500857823	small
-1.8507311999	256	1024	0.999	0.0001650331	0.00844519387	0.4	20	1.0	0.8	0.0577901757	small
-1.869024	256	32	0.999	3.46689965e-05	3.73820857e-05	0.3	10	0.98	0.6	0.0786391003	medium
-1.891295	32	2048	0.999	0.00010817654	0.0231528103	0.1	20	0.98	0.6	0.07914377862	medium
-1.89375199	16	64	0.999	0.00016316237	0.0160447398	0.2	10	0.98	1	0.3963441586	small

FIGURE 3.7 – Tableau résumant les valeurs des hyperparamètres et la reward obtenu

Nous avons repris les hyperparamètres de la première ligne du tableau (meilleure récompense), et nous avons augmenté le nombre de tentatives (500K dans le cas de l'optimisation), nous avons alors réussi à hausser la valeur de la récompense.

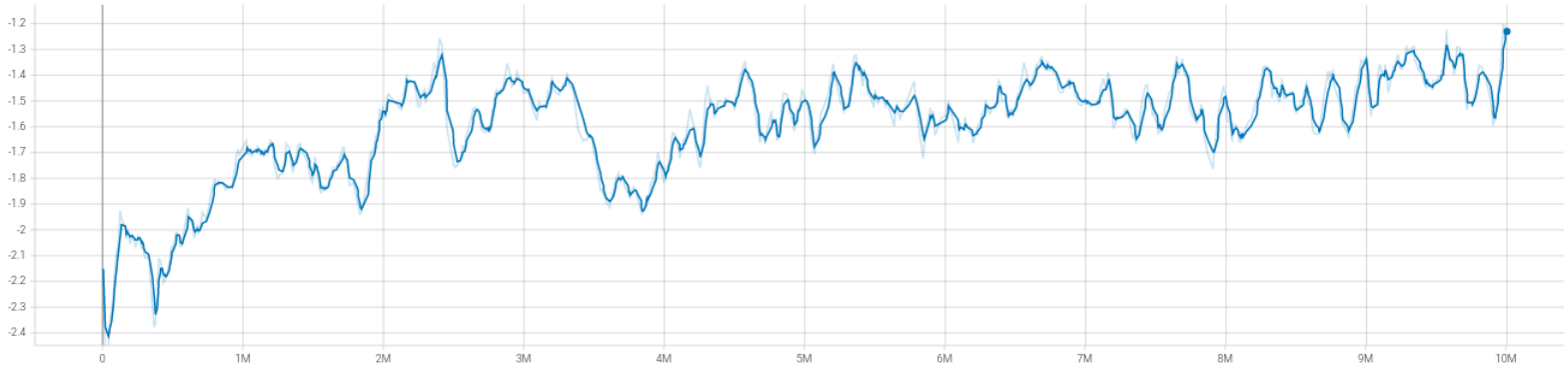

FIGURE 3.8 – Valeur de la récompense en fonction du nombre de tentatives (10M)

La nouvelle valeur de la récompense est meilleure à hauteur de -1.23.

Une autre méthode pour essayer d'améliorer le comportement de la politique est de lancer un apprentissage sur une politique déjà en place. Tous les apprentissages que nous avons mené démarrait toujours de zéro avec un réseau de neurones dont les poids sont aléatoires, donc si nous essayons de commencer à partir d'un réseau avec des valeurs en place, nous pouvons améliorer la politique.

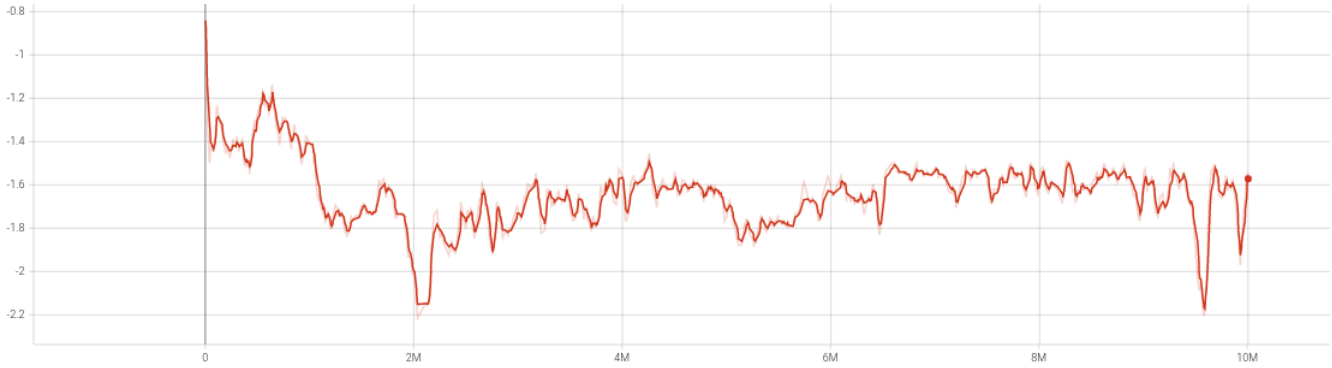

FIGURE 3.9 – Valeur de la récompense en fonction du nombre de tentatives (10M), apprentissage lancé sur la précédente meilleure politique (-1.23)

La nouvelle valeur de la récompense est meilleure à hauteur de -0.84.

Nous remarquons par contre que la politique décroche directement après et passe sur des valeurs inférieures, et toutes les autres tentatives même lancé à partir de la politique à -0.84 n'ont pas pu faire mieux.

3.6 Vignette

Vignette est un outil développé par des étudiants de Sorbonne Université qui permet de visualiser différentes politiques tirées aléatoirement (en apportant des modifications au réseau de neurones) à partir d'une politique fournie. L'intérêt premier de cet outils est d'avoir une vision en 2D/3D de l'évolution d'une valeur autour de la politique optimal (savoir si on est sur un maximum ou minimum).
Notre but est d'améliorer la politique, alors nous avons utilisé cet outil pour visualiser le paysage de la récompense en fournissant la meilleure politique trouvée précédemment (récompense = -0.84), et nous avons modifié le code afin de sauvegarder les nouvelles politiques qui seront meilleures.

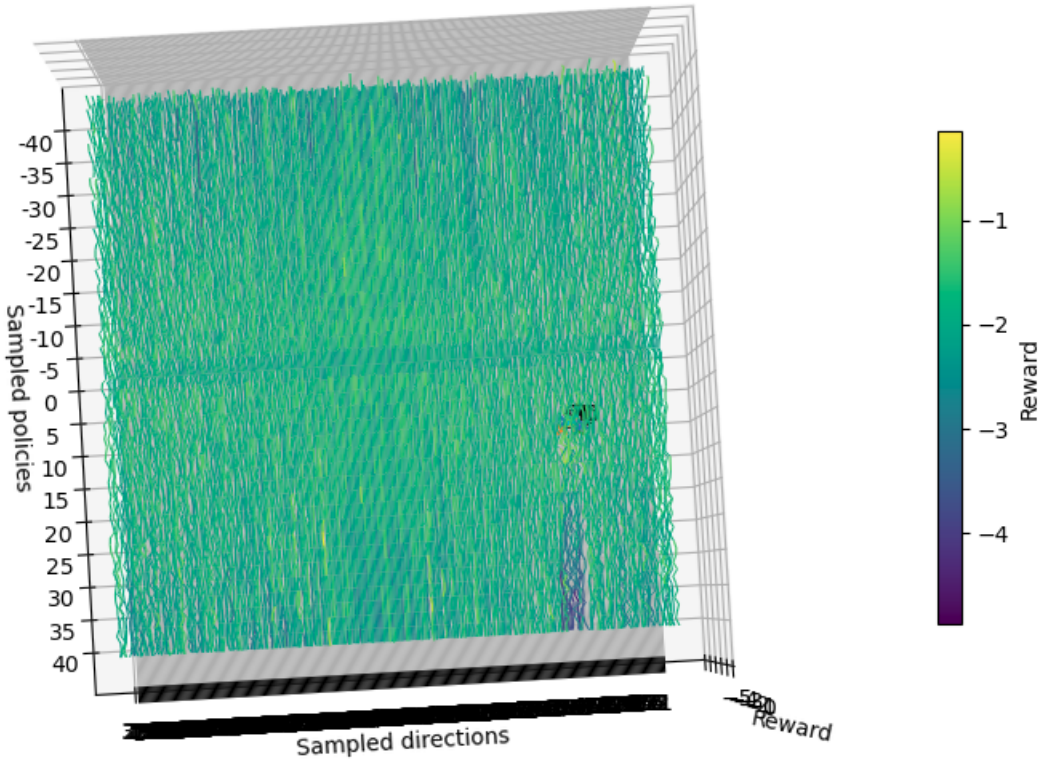

FIGURE 3.10 – Résultat d'application de l'outil Vignette sur la meilleure politique (-0.84)

Nous pouvons voir que le paysage est assez homogène, avec une récompense moyenne entre -2 et -1. Par contre, nous remarquons une valeur en jaune au niveau de la politique numéro 10 sur la droite qui est supérieur à -1, cette dernière à une récompense de -0.67 et c'est la nouvelle politique que nous avons enregistré qui est supérieure à l'ancienne.

3.7 Conclusion

Après avoir appliqué tout ce que nous avons expliqué dans ce chapitre, nous avons réussi à avoir une meilleure politique avec une récompense de -0.67 qui réussit à faire atterrir la fusée correctement mais qui est irrégulière (atterrissage réussi à hauteur de 66%).

Chapitre 4

Analyse de la politique

Le but de la dernière partie est de visualiser comment se comporte la politique et la comparer à une politique optimale théorique qu'on essayera d'esquisser en analysant la mécanique de la fusée et la fonction de récompense de l'environnement.

4.1 La mécanique de la fusée

On commence par un bilan des forces appliquées sur la fusée :

— Le poids de la fusée : Appliqué sur l'axe y, c'est la force qui fait descendre la fusée.

$$\overrightarrow{P} = -m * g \tag{4.1}$$

m : masse de la fusée.
g : la constante de gravitation terrestre.

— La force du moteur principal : cette force a deux composantes, une suivant l'axe y qui sert à ralentir la chute de la fusée, l'autre suivant l'axe x sert à déplacer la fusée horizontalement afin de la ramener au dessus de la plateforme.

$$\overrightarrow{F\_mot_x} = -\sin(\theta + \phi) * puissance * CST1 \tag{4.2}$$

$$\overrightarrow{F\_mot_y} = \cos(\theta + \phi) * puissance * CST1 \tag{4.3}$$

θ : Angle entre la fusée et l'axe vertical (y).
ϕ : Angle entre le cardan moteur et la normale de la fusée (revoir figure 7).
puissance : variable qui détermine la puissance du moteur, dépend de la valeur de la manette à gaz.

— La force des moteurs latéraux : cette force a aussi deux composantes, et a pour but de toujours maintenir la fusée à la verticale.

$$\overrightarrow{F\_lat_x} = -\cos(\theta) * Force\_dir * CST2 \tag{4.4}$$

$$\overrightarrow{F\_lat_y} = \sin(\theta) * Force\_dir * CST2 \tag{4.5}$$

Force_dir : est une variable qui prend deux valeurs (-1 et 1), et détermine lequel des deux moteurs latéraux que la fusée sollicite.

Résumé des forces : on résume les forces sur chacun des axes.

$$Sur\ x\ : -\cos(\theta) * Force\_dir * CST2 - \sin(\theta + \phi) * puissance * CST1 \tag{4.6}$$

$$Sur\ y\ : -m * g + \sin(\theta) * Force\_dir * CST2 + \cos(\theta + \phi) * puissance * CST1 \tag{4.7}$$

4.2 La fonction de récompense

On reprend la fonction de récompense présentée au chapitre 2, et ce qui nous intéresse le plus est la variable Cout_Carburat, et la variable de shaping.

- Cout_Carburat : est une variable qui dépend de la puissance, et qui est considérée comme une pénalité car ajoutée à la récompense avec un signe négatif, la politique va alors diminuer au maximum la consommation de carburant afin de réduire la pénalité.
- Shaping : est une variable qui oriente la politique en fonction des variables d'états afin d'atteindre l'objectif voulu.

$$shaping = -0.5 * (distance + vitesse + |angle|^2) \tag{4.8}$$

Nous remarquons aussi que la variable shaping est négative, donc considérée comme une pénalité.

La politique essaye alors de minimiser la valeur de cette variable et pour cela :

- La variable distance = |3*x,y|, doit être la plus petite possible, donc diminuer la valeur des coordonnées en x et y.
- La variable vitesse doit aussi être minimiser, donc la fusée doit ralentir de plus en plus lors de sa descente.
- La valeur de l'angle Θ doit être maintenue sur une valeur nulle, donc la fusée se tient à la verticale.

4.3 Observations expérimentales

Nous allons nous intéresser dans cette dernière partie à la visualisation de deux fonctions importantes de l'apprentissage (la politique, le critic). La difficulté de cette étape réside dans le fait que l'environnement possède 10 variables d'états, et pour faire des visualisations pertinentes il faut pouvoir prendre en compte tous les paramètres, donc avoir des graphiques en 11 dimensions (variables d'états + valeur observée). L'idée que nous a alors conseiller notre encadrant repose sur une sélection des paramètres importants (2) pour chaque valeur que nous voulons observer et le forçage des autres paramètres à des valeurs aléatoires neutres afin d'avoir le minimum d'influence possible sur le graphique en sortie.

La politique

Comme expliqué au début du rapport, la politique est la fonction qui associe à un état une fonction : $\pi(s) = a$.
Nous allons alors observer expérimentalement les actions que mène notre politique.
Pour rappel, l'environnement Rocket Lander a 3 actions : la manette à gaz, l'inclinaison de l'angle du moteur principal (ϕ) et le choix du moteur latéral à allumer.

La manette à gaz :
La manette à gaz gère la puissance du moteur principal, si elle est up(1) alors le moteur est allumé, si down (-1) le moteur est éteint. Nous souhaiterons alors observer cette action en fonction de la position spatial de la fusée (x,y).

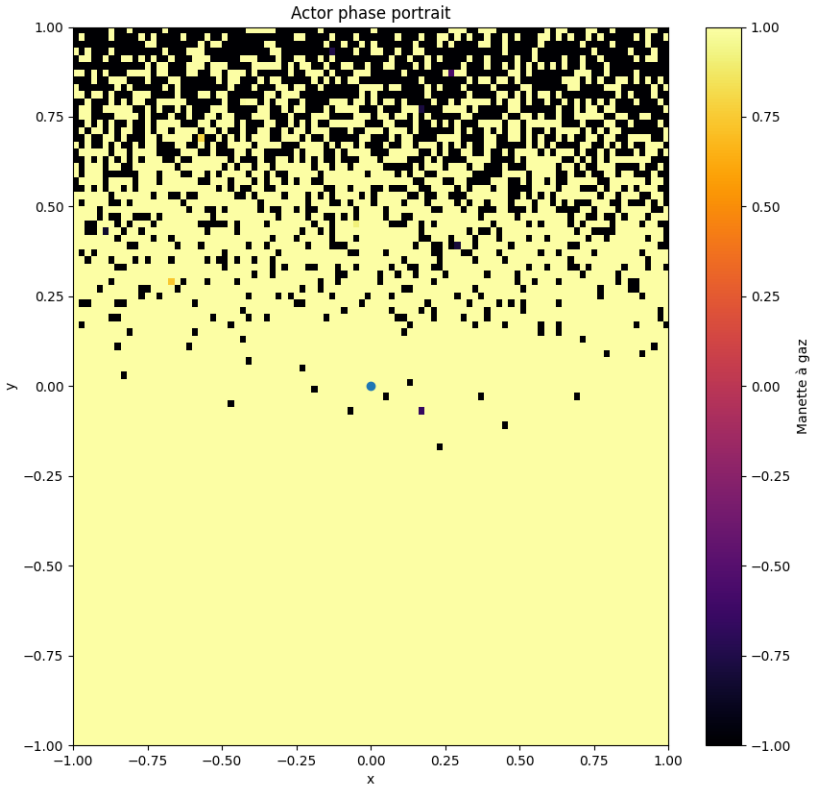

FIGURE 4.1 – Valeur de la manette à gaz en fonction de la position (x,y)
Les valeurs de x et y sont redimensionnées pour être entre -1 et 1 car l'environnement Rocket Lander les utilise avec ces valeurs lors des calculs.
Le repère est orienté dans le sens croissant pour les deux variables

Analyse : Nous remarquons que le moteur est allumé de manière continue à partir d'une certaine hauteur afin de réajuster la position en x et de ralentir la descente de la fusée pour éviter le crash. Le moteur n'est pas utilisé dès le début à cause de la contrainte de carburant.

L'angle du cardan moteur :
Cette action sert principalement à réajuster la position en x (horizontale) de la fusée, elle a aussi deux valeurs opposée (1) pour une inclinaison gauche, et (-1) pour une inclinaison droite. Pour la visualisation le paramètre important est la position en x, pour avoir le graphique il faut un second paramètre et on choisira y.

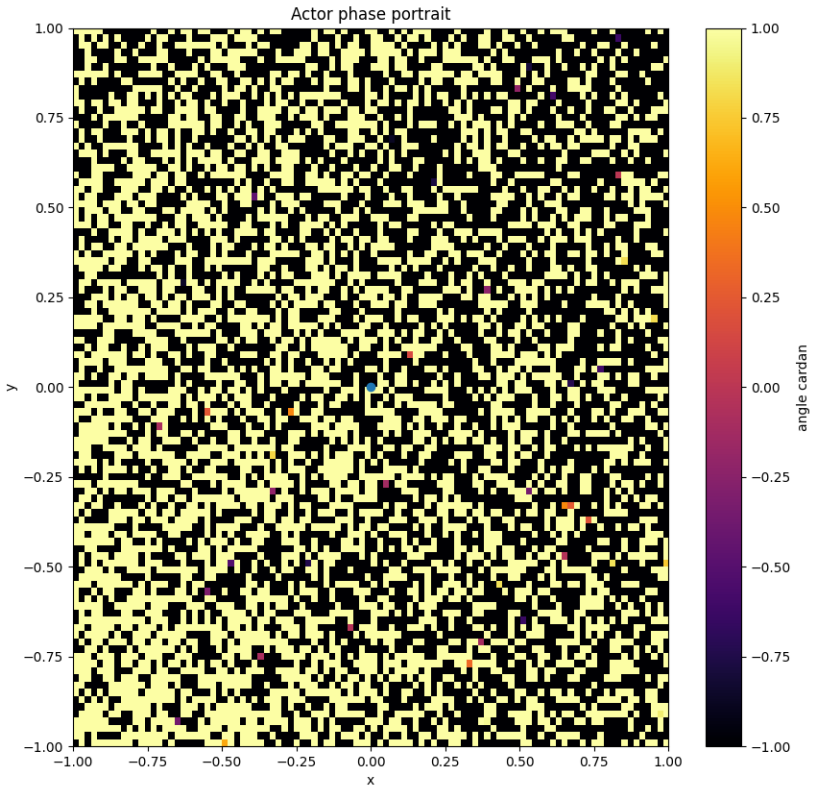

FIGURE 4.2 – Valeur de l'angle du cardan en fonction de la position (x,y)
Les valeurs de x et y sont redimensionnées pour être entre -1 et 1 car l'environnement Rocket Lander les utilise avec ces valeurs lors des calculs.
Le repère est orienté dans le sens croissant pour les deux variables

Analyse : Nous remarquons que la politique est brouillon au niveau de la gestion de l'angle du cardan, même si nous apercevons une plus grosse concentration de noir sur des valeurs de x à droite, la gestion des valeurs en x à gauche est moins bonne. La politique a des difficultés à déplacer la fusée de la gauche vers la droite.

La Direction de la force latérale :
Cette action est la seule à être continue entre -1 et 1, elle sert à déterminer quel moteur latéral à mettre en marche.

$$\begin{bmatrix} if\ action > 0.5\ then\ force\_dir = 1 \\ if\ action < -0.5\ then\ force\_dir = -1 \\ sinon\ la\ fusee\ n'allume\ aucun\ moteur \end{bmatrix}$$

L'objectif de cette action est d'équilibrer l'angle d'inclinaison de la fusée et d'essayer de le maintenir autour de 0, c'est alors naturellement que nous choisissons une observation de l'action en fonction de l'angle et la position en y.

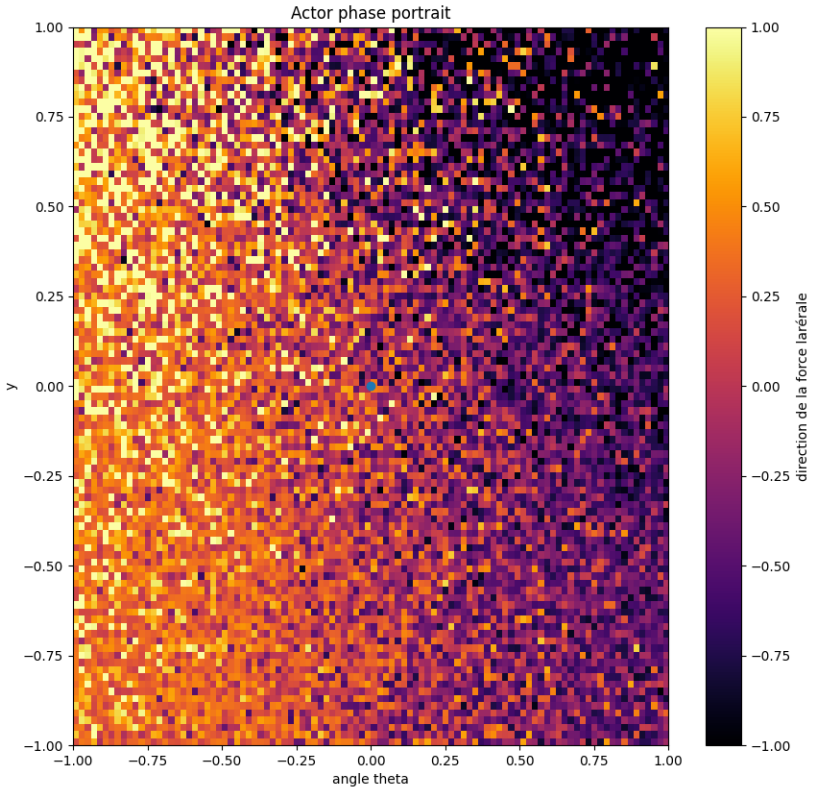

FIGURE 4.3 – Valeur de la direction de la force latérale en fonction de l'angle θ et la position en y (θ, y)
Les valeurs de θ et y sont redimensionnées pour être entre -1 et 1 car l'environnement Rocket Lander les utilise avec ces valeurs lors des calculs.
Le repère est orienté dans le sens croissant pour les deux variables

Analyse : Nous remarquons que la valeur de l'action est inférieur à -0.5 quand l'angle θ est trop incliné vers la droite => force_dir = -1, donc la politique allume le moteur droit de la fusée pour équilibrer, la politique est symétrique pour l'autre côté. Pour le centre les valeurs sont comprises entre -0.5 et 0.5, donc aucun moteur n'est allumé.

Résumé : La politique que nous avons testé (récompense = -0.67) gère de manière cohérente les actions pour le choix du moteur latéral et la puissance du moteur principal, par contre pour l'inclinaison de ce dernier la politique reste assez brouillon ce qui explique le taux de réussite de 66%. On peut alors déduire qu'on est encore loin de la politique optimale.

Le critic

Le critic est un réseau de neurones utilisé spécialement pour calculer la fonction de valeur (Value function), cette dernière nous permet de savoir à quel point le fait d'être dans un état donné permet à l'agent d'atteindre son objectif.
Exemple sur un problème simple, un agent qui veut partir d'un point A à un point B.

FIGURE 4.4 – Exemple d'une value function sur un problème simple

Explication : Nous avons un agent qui veut partir d'un point A (en rouge) vers un point B (vert clair). Nous visualisons la value function du problème.
Nous remarquons alors que le point de départ est pénalisant (-1), ce qui va pousser l'agent à se déplacer, et plus il se rapproche de son point d'arrivé plus la valeur augmente ce qui pousse l'agent à se déplacer dans la bonne direction.

L'environnement Rocket Lander étant plus complexe, nous devons comme pour la politique faire des choix sur les variables d'environnement à observer. Nous avons alors décidé d'observer la value function en fonction des variables d'états utilisées pour le shaping de la récompense, car par définition le shaping pousse l'agent vers des états qui sont proches de l'objectif, et donc reste corrélé indirectement à la value function.

La position (x,y) :

Le shaping de la récompense oriente l'agent vers des valeurs x et y de plus en plus petite, ce qui théoriquement pousse la fusée à se poser mais à tendre vers le côté gauche de la plateforme (x petit).

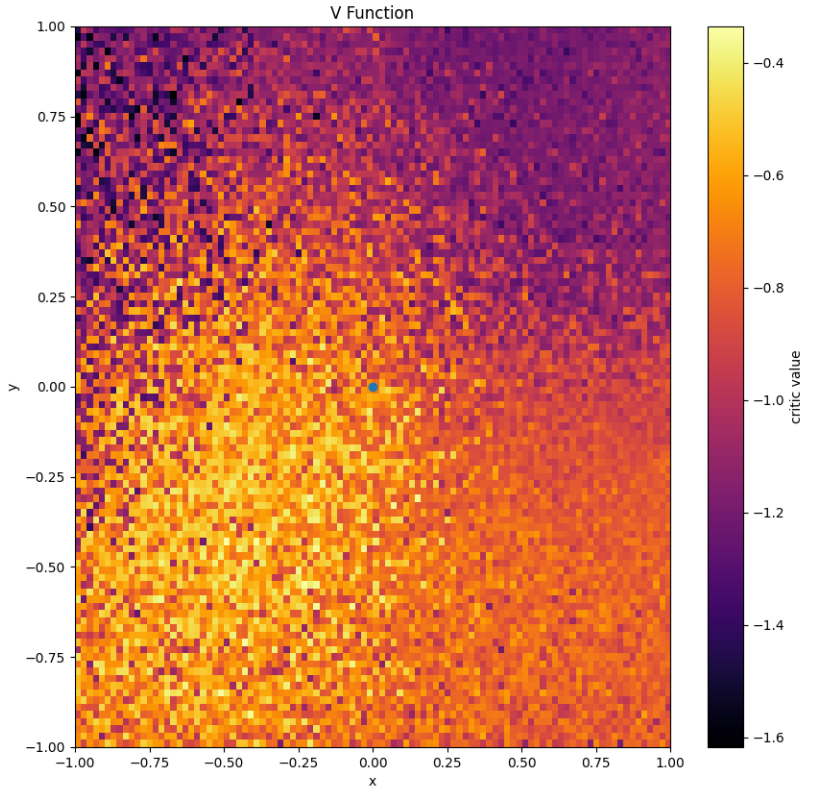

FIGURE 4.5 – Visualisation de la value function en fonction des variables x,y
Les valeurs de x et y sont redimensionnées pour être entre -1 et 1 car l'environnement Rocket Lander les utilise avec ces valeurs lors des calculs.
Le repère est orienté dans le sens croissant pour les deux variables

Analyse : Nous remarquons que la value function correspond exactement à ce qui a été expliquer plus haut, la fusée tend à descendre et se décaler vers la gauche, mais ce comportement n'est pas optimal car la plateforme d'atterrissage se situe au milieu (x entre -0.23 et 0.23). Ce problème peut expliquer la mauvaise gestion de l'angle du moteur, car quand la fusée est excentrée sur la gauche, la politique ne fait rien pour la remettre au milieu. Par hypothèse ce comportement est engendré par le shaping car il pousse les deux coordonnées vers des valeurs basses.

L'angle d'inclinaison θ :

La politique est censée maintenir l'angle à une valeur autour de zéro tout le long de la descente de la fusée. Nous allons alors visualiser la value function en fonction de l'angle et la position en y.

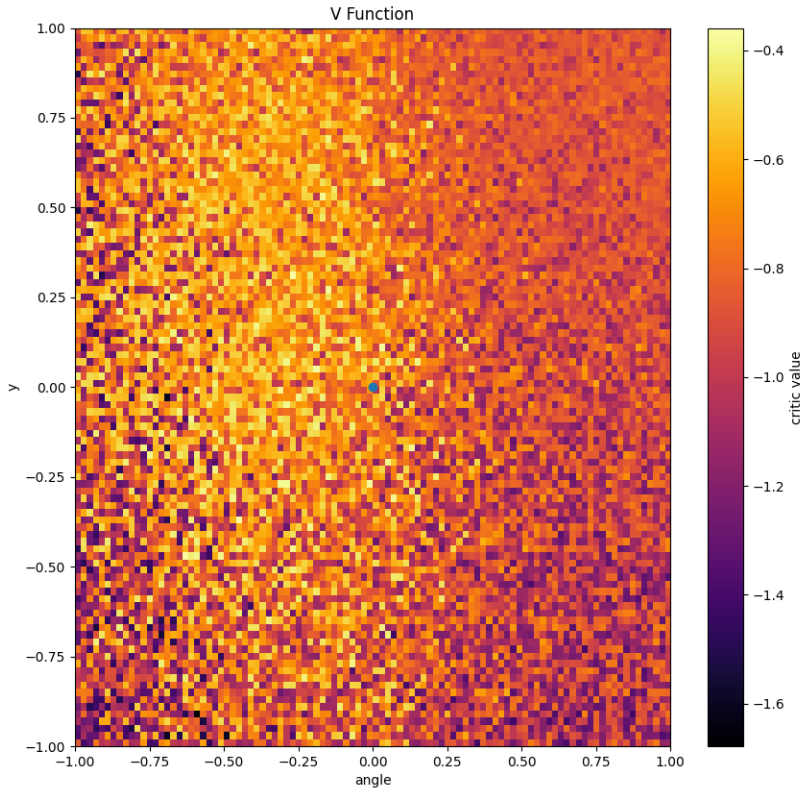

FIGURE 4.6 – Visualisation de la value function en fonction des variables θ, y
Les valeurs de θ et y sont redimensionnées pour être entre -1 et 1 car l'environnement Rocket Lander les utilise avec ces valeurs lors des calculs.
Le repère est orienté dans le sens croissant pour les deux variables

Analyse : Nous observons que la value function montre un schéma ressemblant à la théorie, mais la valeur de l'angle n'est pas symétrique autour de zéro donc l'angle n'est pas parfaitement centré.

La vitesse suivant l'axe vertical V(y) :

La vitesse de la fusée est censé diminuer en descendant afin d'éviter le crash et d'arriver à vitesse nulle sur la plateforme.

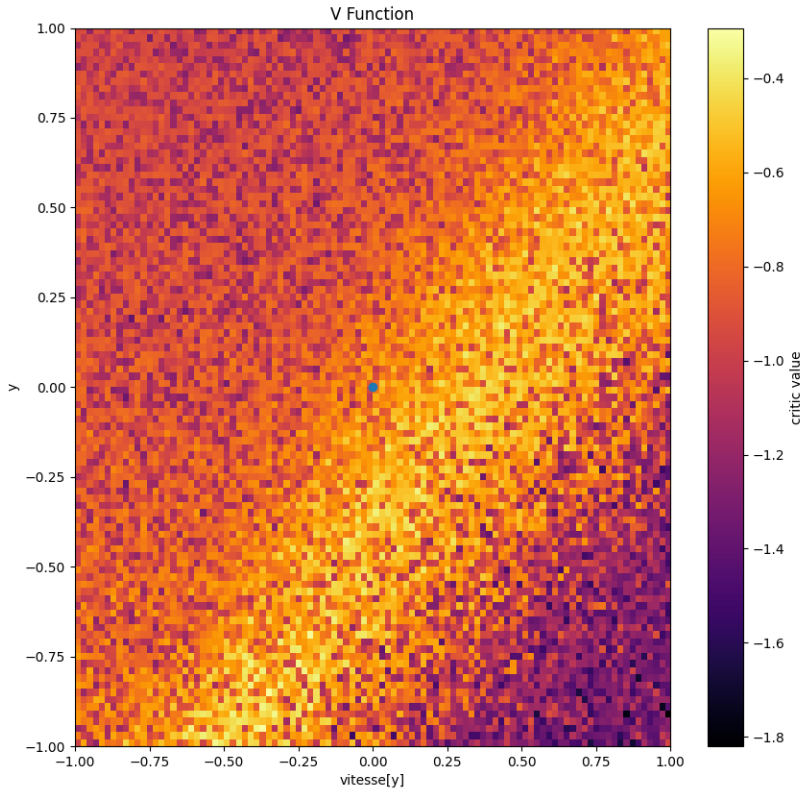

FIGURE 4.7 – Visualisation de la value function en fonction des variables V(y), y
Les valeurs de V(y) et y sont redimensionnées pour être entre -1 et 1 car l'environnement Rocket Lander les utilise avec ces valeurs lors des calculs.
Le repère est orienté dans le sens croissant pour les deux variables

Analyse : Nous observons que la value function dessine une droite avec des valeurs optimale de vitesse en fonction de la position que doit respecter la fusée lors de sa descente, et la tendance de cette droite est de baisser la vitesse en se rapprochant du sol la valeur -1 de la vitesse dans le graphe étant la vitesse nulle.

Conclusion des observations

Nous remarquons que la plupart des observations montre que la politique se rapproche du comportement optimal mais ne l'atteint pas encore, cela est dû à plusieurs problématiques comme le fait que la politique n'est peut-être pas encore assez entraînée(Plus de tentatives), ou le fait que la définition des fonctions de l'environnement n'est pas bonne (fonction de shaping qui pousse la fusée vers la gauche).

Chapitre 5

Conclusion

Le projet nous a permis de découvrir l'apprentissage par renforcement et les différents outils utilisés pour définir et résoudre des problèmes. Nous avons aussi appris à gérer un projet assez important en autonomie, en améliorant notre organisation et notre travail d'équipe et en développant notre capacité à prendre en main de nouveaux outils très rapidement afin de pouvoir avancer. Notre capacité à rédiger des documents s'est aussi améliorée grâce à l'exigence de notre encadrant qui ne laisse aucune erreur passée. Nous avons aussi découvert énormément de lacunes qu'il faudra corriger si nous voulons intégrer le domaine de la recherche, mais l'expérience tiré de ce projet positive.

Chapitre 6

Références

1. R. S. Sutton et A. G. Barto, Reinforcement learning : an introduction. Cambridge (Mass.), Etats-Unis d'Amérique : The MIT Press, 2018
2. David Silver UCL Course on RL https://www.davidsilver.uk/teaching/
3. Olivier Sigaud youtube Channel https://www.youtube.com/c/OlivierSigaud
4. Rocket Lander environement on github https://github.com/sdsubhajitdas/Rocket_Lander_Gym
5. Stable-Baselines3 python library website https://stable-baselines3.readthedocs.io/en/master/
6. Olivier Sigaud Stable-Baselines3 library fork on github https://github.com/osigaud/stable-baselines3
7. OpenAI Gym python library website https://www.gymlibrary.ml/
8. Box2D C++ library website https://box2d.org/
9. Vignette project on github https://github.com/sohio92/Reinforcement-learning-visualization
10. Medium website RL articles https://medium.com/tag/reinforcement-learning