Stabilisation d'un pendule inversé à l'aide d'un apprentissage par renforcement

Culture Sciences de l'Ingénieur

Édité le 13/06/2022

école normale supérieure paris—saclay

Guénolé CHÉROT – Maël GALLOIS

Cette ressource est issue d'une publication du numéro 108 de La Revue 3EI d'avril 2022 et fait partie du « Dossier Intelligence Artificielle » [7] sur Culture Sciences de l'Ingénieur.
Guénolé Chérot est doctorant en deuxième année à l'ENS Rennes, rattaché au Laboratoire SATIE et Maël Gallois est étudiant de deuxième année à l'ENS Rennes.

Cette ressource propose d'illustrer, à travers une séance de travaux pratiques, les principes de base de l'apprentissage par renforcement.

1 – Déroulement de la séance de travaux pratiques

Nous commencerons tout d'abord par rappeler les concepts de base de l'apprentissage par renforcement. Cette partie théorique présentera les notions d'action, d'observation, d'état, de récompense et de politique. Puis nous montrerons comment formaliser le redressement et la stabilisation d'un pendule simple sous la forme d'un problème d'apprentissage par renforcement. Enfin, une séance de travaux pratiques mêlant simulations et expérimentations sur un système réel permettra d'illustrer la pertinence de la méthode proposée. Nous discuterons en particulier de l'importance du modèle utilisé pour modéliser l'environnement. L'activité pratique permettra également de mettre en évidence la différence entre phases d'apprentissage et d'inférence.

La phase d'expérimentation se base sur Control'X et Matlab - fichiers disponibles sur Gitlab[1]. Cependant, la partie simulation peut également se faire sur python via google colab[2] - dirigée principalement vers l'utilisation de la librairie Gym.

Cette ressource est à destination d'enseignants ou de chercheurs désirant se familiariser avec les concepts de l'apprentissage par renforcement. Il ne nécessite pas de connaissances préalables. Le lecteur curieux pourra approfondir grâce à l'excellent ouvrage de Sutton & Barto [1]. Les activités pratiques peuvent être effectuées par des élèves de lycée ou de classe préparatoire même si l'apprentissage par renforcement n'est pas explicitement au programme. Les points du programme de classe préparatoire abordés sont listés dans le tableau à la fin de cette ressource.

2 – Contextualisation

Handle (figure 1) est un robot à deux roues développé par l'entreprise Boston Dynamics. Dans sa dernière version, il est stabilisé grâce à des asservissements complexes. Nous étudierons ici la faisabilité de remplacer tout ou partie de ces algorithmes par une intelligence artificielle.

Pour cette activité pratique, nous étudierons uniquement la stabilisation du robot sur l'axe avant/arrière. Le problème est donc plan. Le robot Handle n'étant pas disponible dans notre laboratoire, nous illustrerons la démarche à l'aide d'un système plus simple : un pendule inversé

[1] https://gitlab.com/Guenole.cherot/controlx_rl
[2] https://colab.research.google.com/drive/1AZh_FSSImncpOuy_vdGbewK6OyqksAme?usp=sharing

monté sur un chariot mobile. Nous utiliserons le système Control'X dont une illustration est donnée figure 2.

Figure 1 : Photo du robot Handle de Boston Dynamics

Figure 2 : Control'X + Pendule

3 – Formalisme général

3.1 – Théorie

L'apprentissage par renforcement est la science de la **prise de décisions séquentielles**. Un agent autonome (robot, joueur...) cherche à **maximiser la somme des récompenses** qu'il obtient en interagissant avec un environnement. À chaque itération, l'agent reçoit une **observation** et une **récompense** (scalaire) de la part de l'environnement. Il doit ensuite choisir une **action** qui lui permet de maximiser la somme des récompenses perçues. L'interaction entre un agent et son environnement est décrite figure 3.

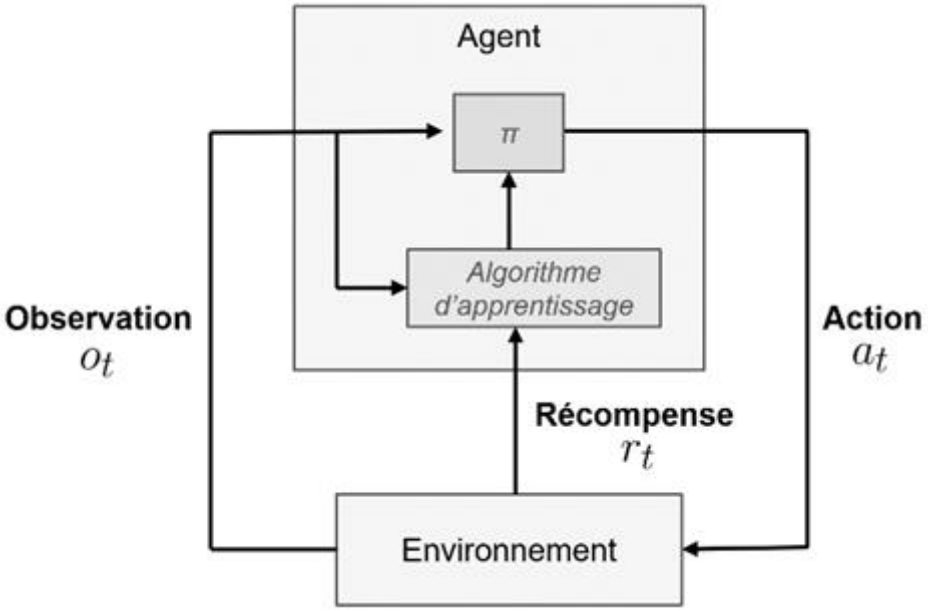


Figure 3 : Interaction entre l'agent et l'environnement

Pour prendre ses décisions, l'agent a besoin d'une stratégie de commande. Dans le formalisme de l'apprentissage par renforcement, on la nomme **politique**, noté $\boldsymbol{\pi}$. La politique est une fonction qui à une observation O associe une action $\pi(o_t) = a_t$. On pourrait aussi utiliser les termes loi de contrôle. N'importe quelle fonction paramétrable peut être utilisée pour représenter la politique, mais en pratique un **réseau de neurones artificiels** est souvent utilisé.

Pour améliorer sa politique, c'est-à-dire la façon dont il prend ses décisions, l'agent a besoin d'un **algorithme d'apprentissage**. Il existe une très grande diversité d'algorithmes [2]. Nous ne les étudierons pas ici car ils sont souvent complexes et nécessitent un grand nombre de connaissances théoriques.

Le problème mathématique s'exprime donc de la manière suivante :

$$\pi^* = \underset{\pi}{\operatorname{argmax}} \sum_{t}^{T} r_\pi(t)$$

Où T représente le temps de fin d'un épisode. C'est-à-dire le moment auquel l'expérience s'interrompt soit parce que le temps est écoulé soit parce qu'il n'y a plus d'action faisable. Si l'expérience n'a pas de fin alors $T = \infty$.

Cette formulation très générale permet de traiter une grande variété de problèmes. Nous l'appliquerons dans la suite à la stabilisation d'un pendule inverse.

3.2 – Importance de l'observation

L'agent ne peut observer l'environnement que par le biais des observations. Dans une approche système, les observations peuvent être vues comme des sorties de capteurs. Pour que l'agent converge vers la politique optimale (celle qui maximise la somme des récompenses), il est nécessaire de lui donner les bonnes observations. C'est-à-dire, celles qui permettent de décrire le fonctionnement de l'environnement de manière assez précise. Par exemple, il impossible de contrôler un pendule si on ne connait pas sa position angulaire.

Nous verrons plus en détail l'importance de l'observation dans la suite de l'article.

3.3 – Parallèle avec le formalisme habituel du contrôle-commande

Le lecteur familier avec l'automatique aura surement fait le rapprochement entre la figure 3 et les systèmes asservis. Habituellement, un système est dit asservi s'il répond aux trois conditions suivantes : Un opérateur impose une **consigne** au système. Des **capteurs** mesurent la valeur de la grandeur asservie. Une forme d'intelligence cherche à **annuler l'écart entre la consigne et la grandeur mesurée**. Cette section a donc pour but de comparer trois types d'asservissements (illustrée figure 6, annexe, page 12) :

- Système en boucle fermée « classique »
- Retour d'état - Placement de pôles
- Apprentissage par renforcement

3.3.1 – Système en boucle ouverte

Ce type de structure est illustré à gauche de la figure 6. C'est le contexte classique des systèmes asservis. Un opérateur applique une **consigne** au système. Des **capteurs** mesurent les grandeurs d'intérêt. La régulation est effectuée par le couple {soustracteur + correcteur}. **L'intelligence est placée dans le correcteur**. Si le but est d'asservir un pendule en position, alors il faut mesurer

l'écart entre la position souhaitée et la position réelle. Le correcteur va ensuite calculer la commande telle que l'erreur soit nulle. C'est adapté pour l'asservissement de grandeurs scalaires, pour des systèmes linéaires et invariants.

3.3.2 – Retour d'état - Placement de pôles

Ce type de commande est illustré au centre de la figure 6. L'idée générale est de changer les propriétés du système en changeant l'endroit de ses pôles. Il peut alors être plus rapide, ou voir un de ses équilibres instables devenir stable.

L'opérateur **n'applique pas directement une consigne** sur le système. Il a plutôt un cahier des charges qui spécifie des exigences de performances. Des **capteurs** mesurent l'état du système. L'état est souvent un vecteur de dimension supérieure à 1, par exemple dans le cas du pendule ce pourrait être la position et la vitesse du pendule. **L'intelligence est placée dans le calcul du vecteur K**, qui réalise le retour d'état. Dans le cas du pendule, une fois le vecteur K calculé, le pendule est stable en position haute mais on ne peut pas changer la position d'équilibre dynamiquement. Il faudrait recalculer K pour obtenir un nouveau comportement.

3.3.3 – Apprentissage par renforcement

Ce type de commande est illustré à droite de la figure 6. Comme nous allons le voir, le fonctionnement est très proche de celui d'un contrôle par retour d'état. Une comparaison plus complète est faite dans une série d'articles Matlab [3].

L'opérateur **n'applique pas directement une consigne** sur le système. Il doit spécifier une fonction récompense dont la maximisation fait émerger le comportement souhaité. Certains problèmes s'y prêtent mieux que d'autre. Pour les échecs par exemple, il est très facile de donner l'objectif sous forme de récompense : $+1$ si l'agent gagne une partie, -1 s'il perd, 0 si égalité. En revanche, il est difficile de formaliser ce même problème à l'aide d'une consigne.

Des **capteurs** permettent d'observer l'état du système. On nomme observation, l'ensemble des données issues des capteurs.

L'intelligence est placée dans la politique et sa mise à jour. C'est grâce à la politique que l'agent décide des actions à appliquer en fonction de l'observation reçue : $\pi(o_t) = a_t$. Comme pour le contrôle par retour d'état, nous pouvons considérer que le système est asservi car l'agent n'est pas en boucle ouverte, il sait -au moins en partie- ce qu'il se passe dans l'environnement.

4 – Formalisation du redressement et de la stabilisation du pendule inversé

Nous allons maintenant formaliser le redressement et la stabilisation d'un pendule inversé sous la forme d'un problème d'apprentissage par renforcement.

Le schéma cinématique du pendule et du chariot mobile est donné figure 4.

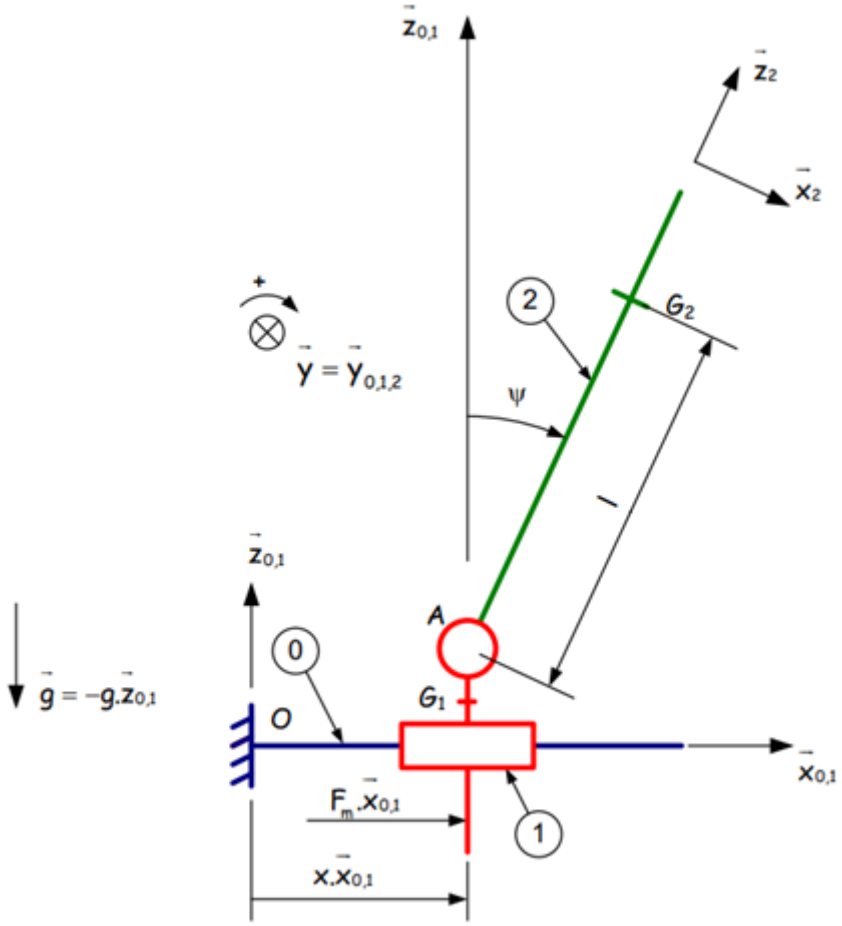


Figure 4 : Schéma cinématique de l'ensemble {chariot + pendule}

4.1 - Action

L'agent interagit avec le Control'X via la tension appliquée sur la machine à courant continu. Cette tension est comprise entre **+/- 10V**. L'espace d'action est donc : $\mathcal{A} = [-10, 10]$. C'est la seule manière pour l'agent d'interagir avec le Control'X.

4.2 - Récompense

La récompense joue un rôle central dans l'apprentissage par renforcement. Pour rappel, l'agent ne reçoit pas de consigne explicite. Son comportement émerge de la maximisation de la somme des récompenses qu'il reçoit.

La récompense est un signal **scalaire** qui doit être conçu par l'ingénieur ou le chercheur. Pour concevoir un bon signal récompense, il faut avoir l'idée suivante en tête :

*La récompense doit **récompenser le but** et **non la manière d'atteindre ce but** [4]*

Par exemple, pour entrainer l'agent à jouer aux échecs, il faut le récompenser quand il gagne une partie et le pénaliser quand il perd (**But** = gagner des parties). Il peut être tentant de l'inciter à occuper le centre, ou à échanger des pions contre des pièces de plus grandes valeurs (**stratégies** pour gagner), mais cela pourrait conduire à des comportements contre-productifs comme occuper le centre mais perdre systématiquement.

Pour ce TP, nous cherchons à stabiliser le pendule en position haute, nous étudierons deux types de fonctions récompenses selon si nous souhaitons relever et stabiliser le pendule ou uniquement le stabiliser.

4.2.1 - Initialisation du pendule en position haute

Commençons par la version simple du problème : stabiliser le pendule lorsqu'il est déjà en position haute. Dans ce cadre, la simulation débute avec le pendule en position haute, $\psi = 0$, et termine quand l'angle du pendule dépasse 15°.

Question :

Proposer une fonction récompense à maximiser. La récompense peut (ou pas) dépendre de l'état du système.

Correction :

Une récompense constante peut fonctionner. L'agent reçoit $r = 1$ à chaque fois qu'il choisit une action, quel que soit l'action ou l'état du système.

Pour maximiser la somme des récompenses sur un épisode (une simulation) l'agent doit faire durer la simulation le plus longtemps possible, donc trouver la stratégie permettant de garder l'angle du pendule entre +/- 15°. Cela revient à stabiliser le pendule en position haute.

4.2.2 - Initialisation du pendule en position basse

Prenons maintenant le problème complet : **relever** puis **stabiliser** le pendule. Chaque épisode commence avec le pendule en position basse. Cette configuration est beaucoup plus pratique si on réalise tout l'apprentissage sur le système réel (sans modèle) car redresser le pendule au début de chaque épisode est fastidieux. L'épisode termine quand le chariot arrive en fin de course ($|x| > x_{max}$) ou que la simulation dépasse 4 secondes ($t > T_{max}$).

Question :

Proposer une fonction récompense à maximiser. La récompense peut (ou pas) dépendre de l'état du système.

Correction :

Il est tout d'abord intéressant de noter que la fonction récompense proposée à la question précédente ne fonctionnerait pas. Rester sur place conduirait à la maximisation de la récompense. L'agent n'est pas incité à stabiliser le pendule en position haute.

Une solution est de donner une récompense proportionnelle à l'énergie potentielle du pendule. Elle vaut $\boldsymbol{E_c = \frac{1}{2}\, m\, l \cos(\psi)}$. Il est souvent judicieux de normaliser la récompense (le formalisme utilisé pour améliorer la politique utilise des lois de probabilités centrées réduites). On peut choisir :

$$\boldsymbol{r = (\cos(\psi) + 1)/2.}$$

La récompense est donc comprise entre 0 et 1. De manière générale, l'apprentissage est assez robuste aux choix de la fonction récompense. Dans le cas considéré ici, toute fonction paire, concave, et ayant un maximum en $\boldsymbol{\theta = 0}$° fonctionne en théorie. Nous invitons le lecteur intéressé à regarder la publication [5]. Cependant, le choix de la récompense peut grandement influencer la vitesse d'apprentissage. Par exemple, choisir la fonction $\boldsymbol{r = \cos(\psi)}$ n'est pas judicieux car en position initiale ($\boldsymbol{\psi = \pi}$), la récompense vaut $\boldsymbol{r = -1}$. Si l'agent cherche à relever le pendule sans

sortir de la course ($|x| < x_{□□□}$) alors il va dans un premier temps accumuler beaucoup de récompenses négatives. Pour maximiser la somme des récompenses, il peut être tenté de chercher à terminer la simulation le plus rapidement possible en sortant des limites. Il y a donc un fort risque qu'il reste bloqué dans une stratégie sous-optimale, sans jamais se rendre compte qu'il est possible d'obtenir des récompenses positives en relevant le pendule.

Il est possible de modifier la fonction récompense pour pénaliser les changements de consigne trop brusques : $\boldsymbol{r = (\cos(\psi) + 1)/2 + r_{pénalité}}$. En effet les à-coups peuvent endommager les machines, la courroie ou toutes autres parties mécaniques du système. Cette pénalisation peut prendre la forme suivante : $\boldsymbol{r_{pénalité} = -\left(\frac{a_t - a_{t-1}}{2 * a_{max}}\right)^2}$, avec $\boldsymbol{a}$ la consigne appliquée sur la machine.

4.3 - Algorithme d'apprentissage

Nous utiliserons l'algorithme « Soft Actor Critic » [6]. Il n'est pas nécessaire de comprendre le fonctionnement de cet algorithme pour la suite.

4.4 - Observation

La position du chariot est repérée par la variable x, sa vitesse par $\dot{x}$. La position angulaire du pendule est donnée par ψ, sa vitesse angulaire par $\dot{\psi}$. L'espace d'observation est définie par la position linéaire et angulaire du pendule et sa vitesse linéaire et angulaire. L'observation prend donc la forme suivante :

$$o = (x, \dot{x}, \psi, \dot{\psi})$$

Si l'agent est pénalisé en cas d'écarts de consigne trop grands ($r = \cos(\psi) + \boldsymbol{r_{pénalité}}$), alors il faut ajouter la dernière action à l'observation. Sinon, l'agent ne pourra pas être en mesure de comprendre pourquoi il est pénalisé[3] :

$$o = (x, \dot{x}, \psi, \dot{\psi}, a)$$

Cette formulation pose cependant un léger problème. La position du pendule est définie modulo 2π. Ainsi, lorsque le pendule est en position basse, ψ passe de $+\pi$ à $-\pi$ alors que ces états sont très proches. Cela rend l'apprentissage plus long car l'agent doit *apprendre* que ces états sont très proches physiquement. Afin de pallier ce problème, nous remplaçons la variable d'état ψ par le couple $(\sin\psi, \cos\psi)$. Finalement l'espace d'observation est :

$$o = (x, \dot{x}, \sin(\psi), \cos(\psi), \dot{\psi}, a) \in \boldsymbol{O}$$

5 – Préparer l'apprentissage

5.1 - Se mettre à la place de l'agent

Dans cette partie, nous vous proposons de prendre un instant la place de l'agent.

1. Ouvrir le fichier *jeux.exe* (figure 7).
2. Les informations auxquelles a accès l'agent sont affichées en bas à gauche de l'écran. Une version graphique de ces informations est affichée en haut à gauche. La récompense

[3] Pour rappel, l'agent prend des décisions uniquement en fonction de la dernière observation qu'il a reçu. S'il ne sait pas la dernière action qu'il a prise, il ne peut pas en choisir une proche pour éviter d'être pénalisé.

obtenue au dernier pas de temps ainsi que la récompense cumulée au cours d'un épisode sont affichées en bas à droite de l'écran.

3. Il est possible d'interagir avec le système avec le panneau de droite. A chaque pas de temps, il faut choisir une action, ici la tension appliquée sur la machine à courant continu.

Questions :

1. Cliquez sur le bouton « Mission 1 ».

2. Votre but et celui de l'agent est **maximiser la somme des récompenses**. Jouez deux ou trois épisodes en essayant d'obtenir le meilleur score possible.

3. Comment avez-vous procédé pour maximiser votre score ?

4. Maximiser la récompense sur un épisode a-t-il fait émerger un comportement particulier du pendule ?

5. Après avoir cliqué sur le bouton « Mission 2 », répondre à nouveau aux questions 2. 3. et 4.

6. A votre avis, comment sont obtenues les observations sur le système réel ?

Correction :

Question 3

Le but de cette partie est d'illustrer la **prise de décision séquentielle** ainsi que le rôle de l'**observation** et de l'**action**. Les élèves doivent être capables de formuler la politique comme une **correspondance entre** une **observation** et une **action**.

Si les élèves ne font pas directement le lien, leur montrer un état (image + vecteur) du pendule hors contexte. Ils devraient être capables de donner l'action à effectuer pour maximiser la récompense. Si c'est le cas, ils ont appris le lien entre une observation et une action : c'est ce que doit faire la politique. La politique n'est qu'une fonction qui prend en entrée un vecteur de dimension 6 et qui sort un scalaire.

Question 4

Il faut s'assurer que les élèves aient bien compris le but de l'agent : **maximiser la récompense sur un épisode**. La récompense de la « Mission 1 » est proportionnelle à l'énergie potentielle de pesanteur du pendule. Maximiser cette récompense a pour effet de maintenir le pendule en position haute. Le fait que le pendule soit en équilibre en position haute est une **conséquence** de la maximisation de la récompense. La « Mission 2 » récompense la vitesse du pendule. Maximiser la récompense revient à faire le plus de tours possibles dans le temps imparti.

5.2 – Nécessité d'un modèle

Nous avons montré dans les sections précédentes comment formaliser le redressement et la stabilisation du pendule. Il ne reste qu'à entraîner l'agent. Trois possibilités s'offrent à nous :

1. Faire l'**apprentissage** puis l'**inférence** sur le système réel.
2. Faire l'**apprentissage** avec un **modèle** puis l'**inférence** sur le système réel.
3. Faire l'**apprentissage** sur un modèle simple, puis améliorer les performances en continuant d'apprendre sur le système réel. Et enfin faire l'**inférence** sur le système réel.

Questions :

1. Définir les termes **apprentissage** et **inférence**.

2. Que se passerait-il si l'agent commençait par la phase d'inférence ?

3. Quelle est l'intérêt de chacune des approches par rapport aux autres ?

Correction :

1. Un agent est **capable d'apprendre**, si la somme des récompenses qu'il perçoit augmente avec le nombre d'expériences. Pour l'apprentissage par renforcement, la base de données **se construit au fur et à mesure** grâce à l'interaction entre l'agent et l'environnement.

La phase d'**inférence** est la phase durant laquelle l'agent utilise la politique pour choisir les actions à effectuer. Les paramètres de la politique sont alors fixés. Le comportement de l'agent n'évolue plus.

2. Si l'agent commençait par l'inférence sans s'entrainer alors il ne serait pas capable de bien remplir la tâche pour laquelle il est utilisé. Dans le cas de la stabilisation du pendule, le chariot bougerait de façon quasi aléatoire sans parvenir à maintenir le pendule en position haute.

3. La première approche – souvent nommée 'hardware in the loop' - a l'avantage de ne pas nécessiter de modèle de l'environnement. Elle n'est cependant que rarement privilégiée. D'une part, l'agent a souvent besoin de milliers d'interactions avant d'avoir des performances satisfaisantes. C'est problématique pour les applications de conduite autonome par exemple. Il n'est pas possible de laisser un agent appliquer une politique aléatoire au milieu du trafic. D'autre part, l'interaction avec le système réel est souvent beaucoup plus lente qu'avec une simulation. L'apprentissage est donc plus rapide en simulation.

La seconde approche nécessite un modèle de l'environnement. Cela suppose être capable de créer ce modèle sur la base d'une compréhension des phénomènes physiques ou de mesures effectuées sur l'environnement. De plus, un modèle est toujours imparfait, il faut donc s'assurer que les hypothèses réalisées lors de son élaboration sont raisonnables (linéarisation ou pas, prise en compte des frottements, ...). Si le modèle et l'environnement se comportent différemment, il est possible que la politique apprise sur le modèle ne fonctionne pas sur l'environnement.

La troisième approche est à privilégier car elle combine les avantages des deux précédentes. Dans un premier temps, l'agent apprend rapidement une politique convenable sur un modèle imparfait. Dans un deuxième temps, elle s'améliore en interagissant directement avec l'environnement. Comme l'agent continue d'apprendre sur l'environnement, les erreurs de modélisations sont progressivement effacées à mesure que de nouvelles expériences sont collectées.

Pour la version 2021b et les précédentes, Matlab n'offre pas la possibilité d'apprendre en interagissant avec l'environnement. Nous appliquerons donc la deuxième méthode.

6 – Application sur le Control'X

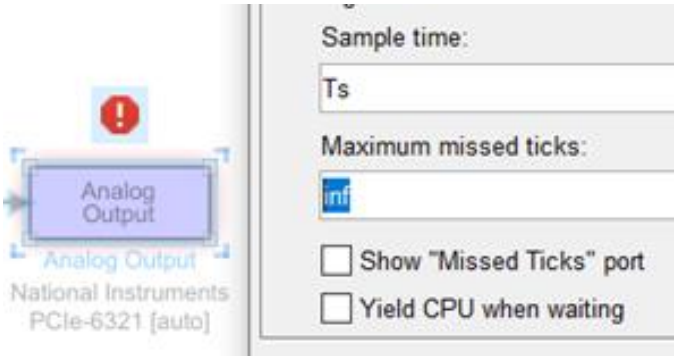
Il ne reste qu'à appliquer les résultats des parties précédentes. L'objectif est de réaliser l'apprentissage sur un modèle du Control'X puis de déployer la politique pour vérifier que l'agent est bien capable de relever et stabiliser le pendule.

Nous considérons que la modélisation du Control'X est déjà effectuée. Si ce n'est pas le cas, se référer à la documentation fournie avec le système.

6.1 – Installation

Les fichiers Matlab sont disponibles sur Gitlab[4] - version 2021b et plus. Bugs possibles :

- Les blocs faisant l'interface entre le PC et le Control'X peuvent ne pas fonctionner correctement : régler « maximum missed ticks : inf »



- Pour tout autre problème lié à des librairies, se reporter à la documentions du Control'X.

6.2 – Entraînement

ATTENTION

1. Le temps de réponse des boutons est parfois long (Ne pas cliquer plusieurs fois)
2. Il arrive que Simulink se bloque lors de l'apprentissage (il se lance mais rien ne se passe). Dans ce cas redémarrer Matlab (*Crtl+Alt+Suppr* puis *Fin de tâche*).

1. Ouvrir le projet « Control'X » dans Matlab
2. Ouvrir le fichier Simulink « model_simulink.slx », il doit correspondre avec la figure 8.
3. Initialiser les constantes du modèle avec le bouton ***Position haute***. Le pendule commencera chaque simulation avec $\psi = 0$.
4. Tester les performances de l'agent par renforcement non entrainé en cliquant sur ***Inférer sur le modèle***. Une fenêtre illustre le comportement simulé du pendule contrôlé par l'agent.
5. Choisir un type de récompense : Pour la stabilisation, les trois types de récompenses fonctionnent.
6. Choisir un type de pénalisation : pénaliser les variations de consigne trop brusques permet un contrôle plus en douceur mais l'apprentissage est plus lent.
7. Lancer l'apprentissage en cliquant sur ***Entrainer***. La fenêtre « Reinforcement Learning Episode Manager » apparait. La courbe bleue représente la récompense cumulée à chaque épisode. Celle orange représente la récompense que l'agent espère obtenir.
8. Une fois l'entrainement terminé, ***Inférer sur le modèle***. L'agent doit être capable de maintenir le pendule en position haute sur la simulation.
9. Vérifier que le Control'X est branché et qu'aucun objet n'obstrue le passage du chariot puis ***Inférer sur le système***. Le pendule doit être stabilisé sur le Control'X.

Remarque : Dans la version actuelle, seule la stabilisation du pendule fonctionne. L'agent ne parvient pas à apprendre la phase de redressement. Une illustration du redressement est disponible (uniquement en simulation) sur python et Google Colab[5].

[4] https://gitlab.com/Guenole.cherot/controlx_rl
[5] https://colab.research.google.com/drive/1AZh_FSSImncpOuy_vdGbewK6OyqksAme?usp=sharing

7 – Conclusion

Pour conclure, réaliser une courte présentation de 10 minutes en vous aidant des questions suivantes :

Questions :

1. Dans quel contexte est-il possible d'utiliser l'apprentissage par renforcement ?
2. À quoi sert la récompense ?
3. Qu'est-ce que la politique ?
4. Le signal récompense est-il toujours utile lors de la phase d'inférence ?
5. Quels sont les intérêts de l'apprentissage par renforcement par rapport à l'approche classique de contrôle-commande ?
6. Quels sont les inconvénients ?

Correction :

1. Prise de décision séquentielle

2. La récompense est l'équivalent de la commande. L'agent cherche à la maximiser. Le comportement global de l'agent est émergeant : les stratégies gagnantes aux échecs sont inconnues, cependant il est facile de récompenser un agent qui gagne une partie.

3. La politique est la stratégie de contrôle de l'agent. C'est une correspondance entre l'observation de l'environnement et l'action à prendre. C'est ce que l'on cherche à apprendre durant l'entraînement.

4. Le signal récompense ne sert que lors de la phase d'apprentissage. Lors de la phase d'inférence, l'agent applique la politique qu'il a appris sans la modifier.

5. L'apprentissage par renforcement permet de traiter des problèmes qu'il serait difficile de formaliser autrement (ex : jeu d'échec). Il permet de se passer de modèle de l'environnement : il est possible d'apprendre directement sans connaissance à priori.

6. Les inconvénients de l'apprentissage par renforcement sont les suivants : Il faut pouvoir interagir avec le système et collecter des données. L'apprentissage est souvent long et parfois instable. Les algorithmes d'apprentissages sont complexes et pour l'instant hors de portée du grand public.

8 – Annexe

	Compétences générales	*Compétences développées*	*Connaissances associées*
Analyser	A3 - Analyser l'organisation fonctionnelle et structurelle	**Analyser** les principes d'intelligence artificielle	- Phases d'apprentissage et d'inférence. - Réseaux de neurones (couches d'entrée, cachées et de sortie, neurones, biais, poids et fonction d'activation).
		Identifier la structure d'un système asservi.	- Grandeurs d'entrée et de sortie. - Capteurs
Modéliser	B2 – Proposer un modèle de connaissance et de comportement	**Choisir** un modèle adapté aux performances à prévoir ou à évaluer	Domaine de validité.
	B3 – Valider un modèle	**Vérifier** la cohérence du modèle choisi en confrontant les résultats analytiques et/ou numériques aux résultats expérimentaux.	Critères de performances
Résoudre	C3 – Mettre en œuvre une démarche de résolution numérique	**Mener** une simulation numérique	Influence des paramètres du modèle sur les performances.
Expérimenter	D1 – Mettre en œuvre un système	**Mettre en œuvre** un système en suivant un protocole	

Figure 5 : Compétences et connaissances du référentiel MPSI/PCSI/PC/PSI/MP

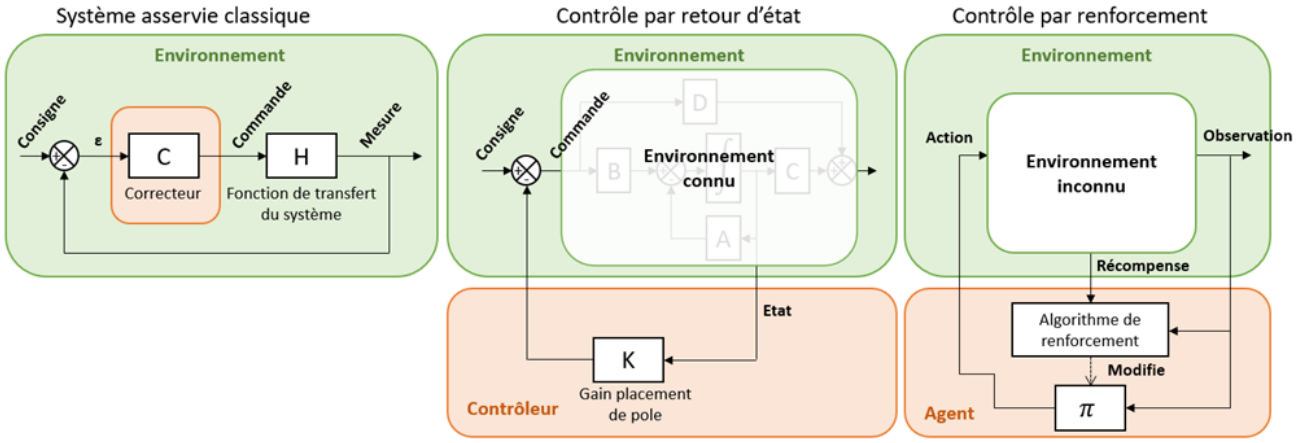


Figure 6 : Comparaison d'un contrôle par retour d'état et à l'aide d'un apprentissage par renforcement

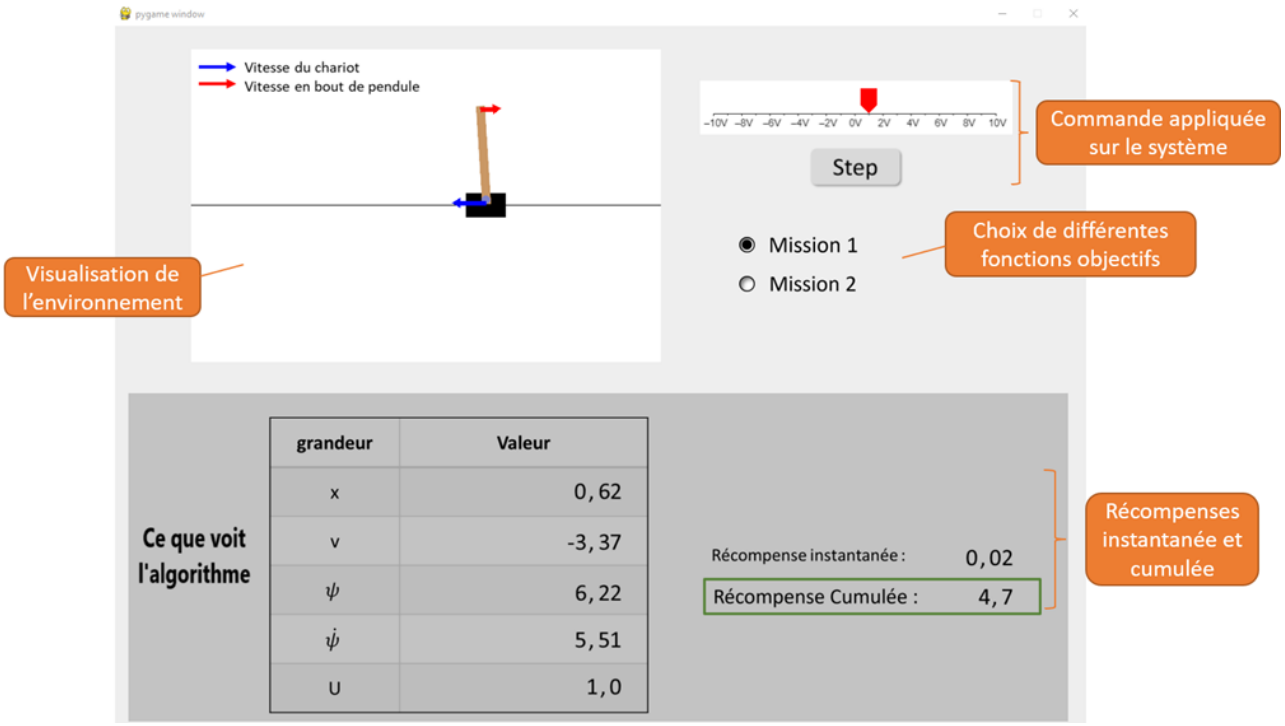


Figure 7 : Se mettre à la place de l'agent par renforcement

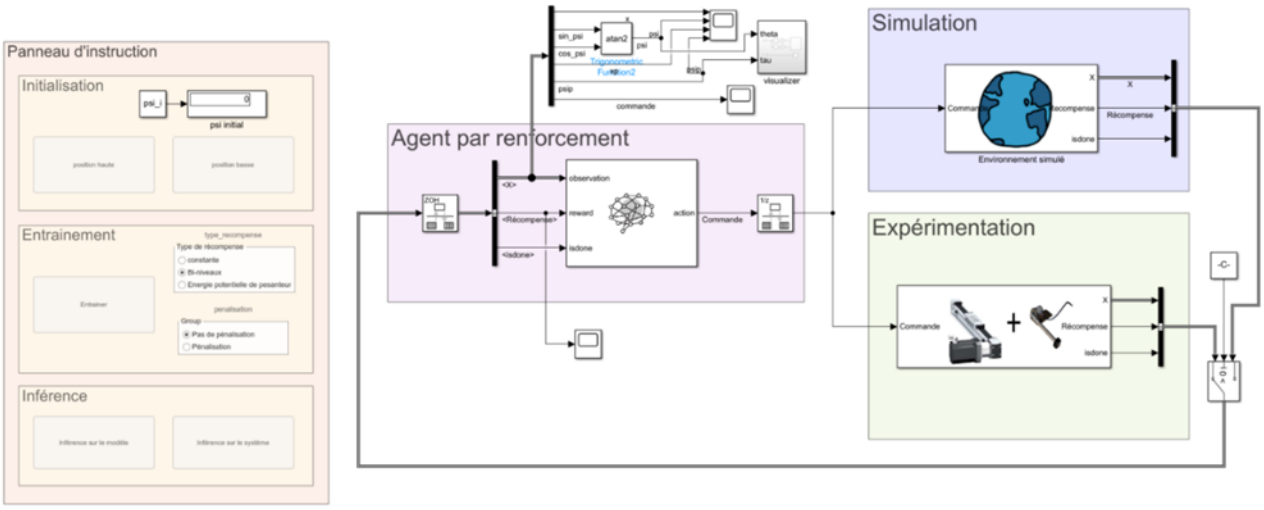


Figure 8 : Fichier Simulink

Références :

[1]: S. Sutton, A. Barto, « Reinforcement learning: An introduction », MIT press, 2018.

[2]: S. Levine, A. Kumar, G. Tucker, et al. « Offline reinforcement learning: Tutorial, review, and perspectives on open problems », arXiv preprint arXiv:2005.01643, 2020.

[3]: Matlab, « Reinforcement Learning avec MATLAB et Simulink », https://fr.mathworks.com/campaigns/offers/reinforcement-learning-with-matlab-ebook.html

[4]: D. Silver, « RL Course by David Silver », Youtube, 2015

[5]: D. Silver, S. Singh, D. Precup, et al , « Reward is enough », Artificial Intelligence, 2021, vol. 299, p. 103535.

[6]: T. Haarnoja, A. Zhou, P. Abbeel, et al. « Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor », International conference on machine learning. PMLR, 2018. p. 1861-1870.

[7]: Dossier Intelligence Artificielle, juillet 2022, https://eduscol.education.fr/sti/si-ens-paris-saclay/ressources_pedagogiques/dossier-intelligence-artificielle