Contrôle garanti par réseaux de neurones pour des robots mobiles

Pôle Systèmes Cyber-Physiques

Tarek OMRAN Ali RAMLAOUI

1er juin 2022

Encadré par Adnane SAOUD

Mise en contexte

Système dynamique

$$F : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$$
$$(x(k), u(k)) \mapsto F(x(k), u(k)) = x(k+1)$$

muni d'un contrôleur prenant des décisions sur l'entrée en fonction de l'état $x(k)$

Mise en contexte

Système dynamique

$$F : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$$
$$(x(k), u(k)) \mapsto F(x(k), u(k)) = x(k+1)$$

muni d'un contrôleur prenant des décisions sur l'entrée en fonction de l'état $x(k)$

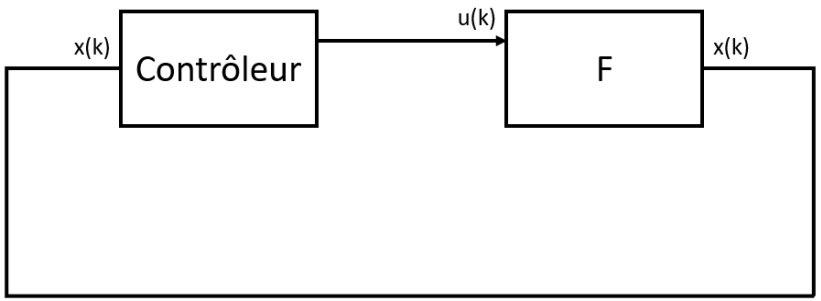


Schéma bloc du système

Mise en contexte

Système dynamique

$$F : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$$
$$(x(k), u(k)) \mapsto F(x(k), u(k)) = x(k+1)$$

muni d'un contrôleur prenant des décisions sur l'entrée en fonction de l'état $x(k)$

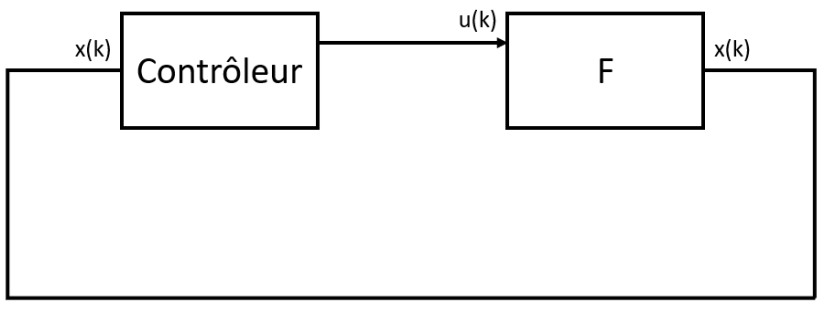


Schéma bloc du système

- Contrôleur : réseau de neurones
- Analyse d'atteignabilité par intervalles
- Images d'intervalles par la fonction F

Notion d'intervalle

Définition (Intervalle de dimension n)

Considérons l'espace $\mathbb{R}^n$, avec $n \in \mathbb{N}^*$. Alors un intervalle de dimension n est un ensemble pouvant s'écrire $\left[\underline{a_1}, \overline{a_1}\right] \times \ldots \times \left[\underline{a_n}, \overline{a_n}\right]$, où $(\underline{a_i} \leq \overline{a_i}) \in \mathbb{R}^2$, $\forall i \in [\![1, n]\!]$. On notera l'intervalle $[a]$.

Notion d'intervalle

Définition (Intervalle de dimension n)

Considérons l'espace $\mathbb{R}^n$, avec $n \in \mathbb{N}^*$. Alors un intervalle de dimension n est un ensemble pouvant s'écrire $\left[\underline{a_1}, \overline{a_1}\right] \times \ldots \times \left[\underline{a_n}, \overline{a_n}\right]$, où $(\underline{a_i} \leq \overline{a_i}) \in \mathbb{R}^2$, $\forall i \in [\![1, n]\!]$. On notera l'intervalle $[a]$.

Definition (Sur-approximation par un intervalle)

Pour tout ensemble $\mathcal{H} \subset \mathrm{R}^n$, on appelle $\mathcal{I}_{\mathcal{H}}$, le plus petit intervalle de $\mathbb{R}^n$ tel que $\mathcal{H} \subset \mathcal{I}_{\mathcal{H}}$.

Notion d'intervalle

Définition (Intervalle de dimension n)

Considérons l'espace $\mathbb{R}^n$, avec $n \in \mathbb{N}^*$. Alors un intervalle de dimension n est un ensemble pouvant s'écrire $\left[\underline{a_1}, \overline{a_1}\right] \times \ldots \times \left[\underline{a_n}, \overline{a_n}\right]$, où $(\underline{a_i} \leq \overline{a_i}) \in \mathbb{R}^2$, $\forall i \in [\![1, n]\!]$. On notera l'intervalle $[a]$.

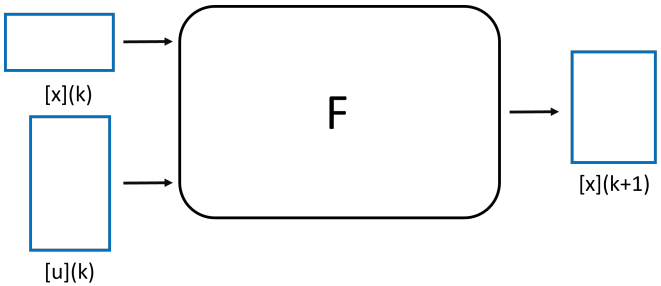
Definition (Sur-approximation par un intervalle)

Pour tout ensemble $\mathcal{H} \subset \mathrm{R}^n$, on appelle $\mathcal{I}_{\mathcal{H}}$, le plus petit intervalle de $\mathbb{R}^n$ tel que $\mathcal{H} \subset \mathcal{I}_{\mathcal{H}}$.

Definition (Longueur d'un intervalle)

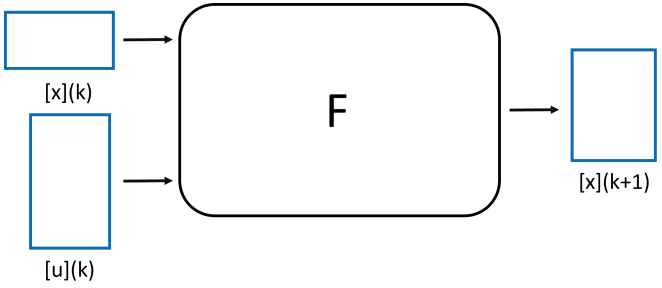
$$\rho(\mathcal{I}) = \max_{i \in [\![1,n]\!]} (\underline{a_i} - \overline{a_i})$$

Atteignabilité

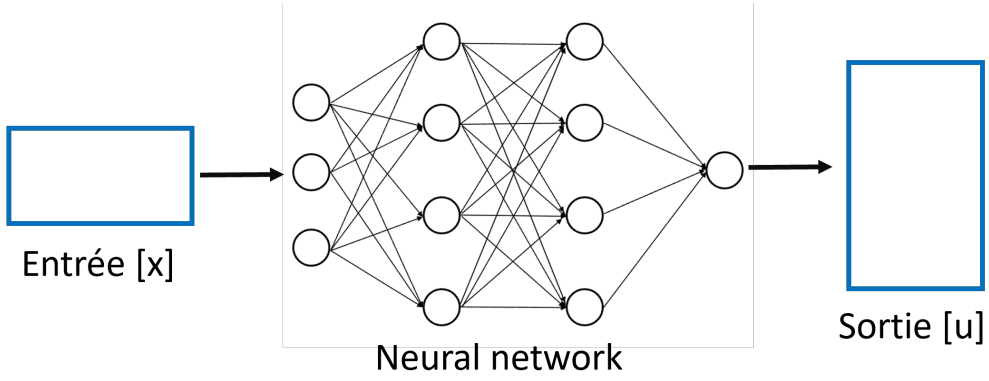


Atteignabilité Système

Atteignabilité

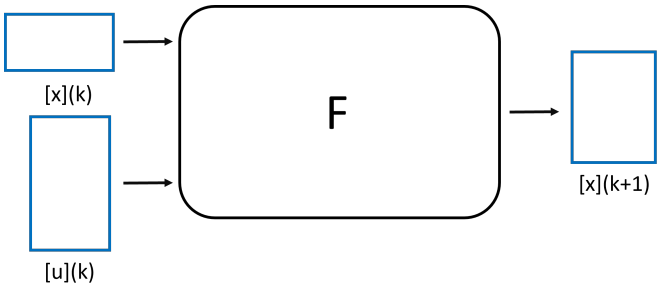


Atteignabilité Système

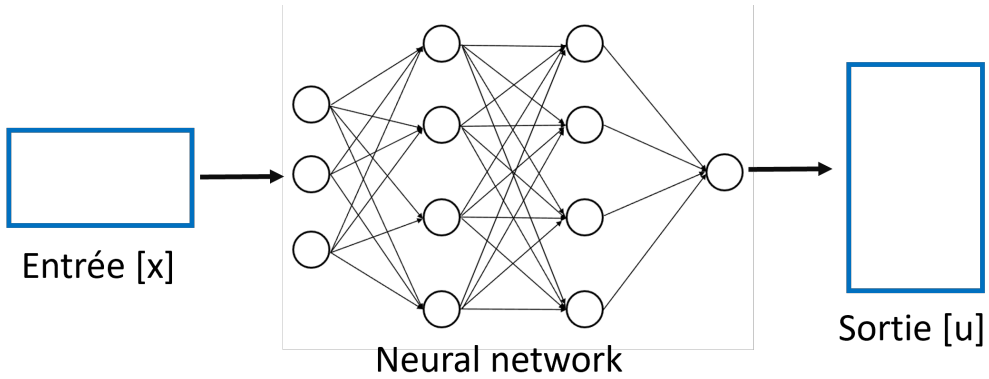


Atteignabilité contrôleur

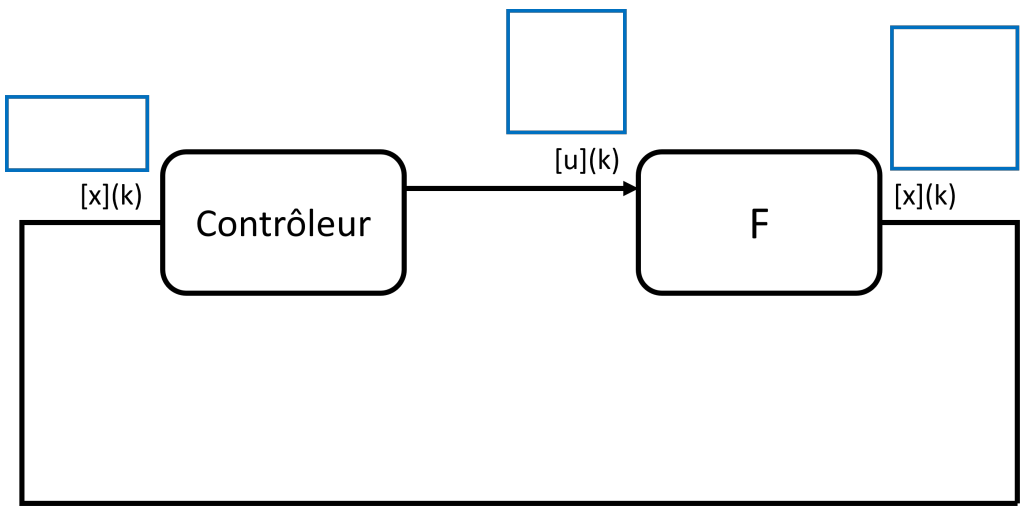
Atteignabilité



Atteignabilité Système



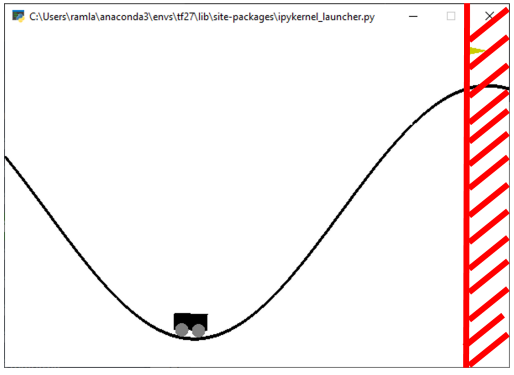
Atteignabilité contrôleur



Atteignabilité boucle fermée

Contrôle par Reinforcement Learning

Systèmes considérés - Mountain car

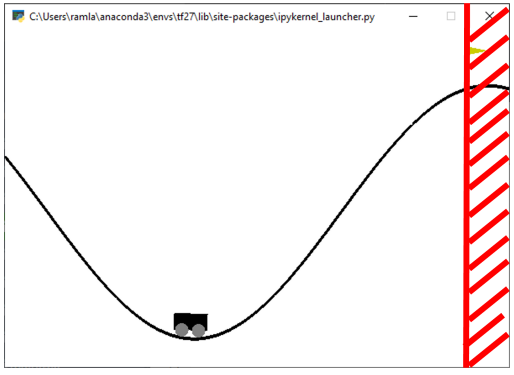


Mountain car

$$F : \begin{cases} x(k+1) & = x(k) + v(k+1) \\ v(k+1) & = u(k)P - 0.0025\cos(3x(k)) \end{cases}$$

où P, constante, $u \in [-1, 1]$, $x \in [-1.2, 0.6]$, position horizontale, et vitesse, $v \in [-0.07, 0.07]$

Systèmes considérés - Mountain car



Mountain car

$$F : \begin{cases} x(k+1) & = x(k) + v(k+1) \\ v(k+1) & = u(k)P - 0.0025\cos(3x(k)) \end{cases}$$

où P, constante, $u \in [-1, 1]$, $x \in [-1.2, 0.6]$, position horizontale, et vitesse, $v \in [-0.07, 0.07]$

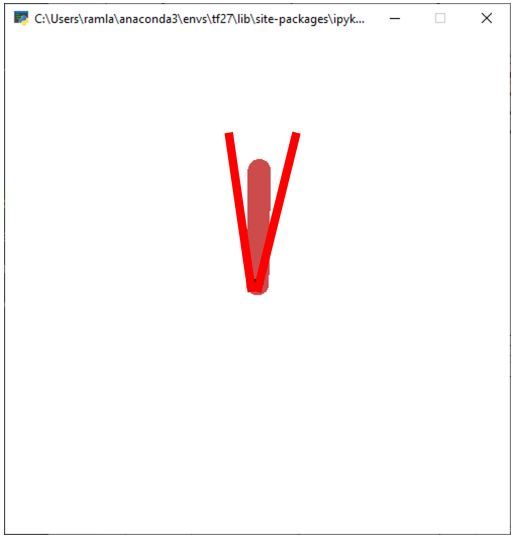
Reward function : Critiquer l'action du contrôleur

$$q(k) = \begin{cases} 100 & \text{si } x(k) \geq 0.6 \\ q(k-1) - 0.1u(k)^2 & \text{sinon} \end{cases}$$

Objectif d'atteignabilité

Exemple classique en Reinforcement Learning

Systèmes considérés - Pendule



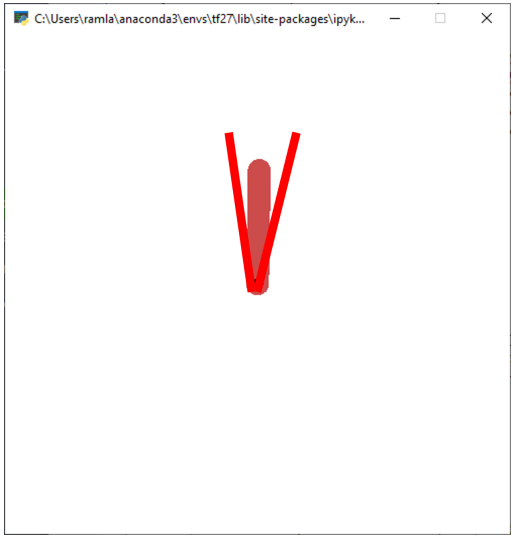
Pendule

$$F : \begin{cases} \theta(k+1) & = \theta(k) + \delta_t \dot{\theta}(k) \\ \dot{\theta}(k+1) & = \dot{\theta}(k) + \frac{3g\delta_t}{2l} \sin(\theta(k)) + \frac{3}{ml^2} u(k)\delta_t \end{cases}$$

où δ_t, g, l, m, constantes, $u(k) \in [-2, 2]$ et

$$X(k) = \begin{bmatrix} \theta \\ \dot{\theta} \end{bmatrix}, \; X(k) \in [0, 2\pi] \times [-8, 8]$$

Systèmes considérés - Pendule



Pendule

$$F : \begin{cases} \theta(k+1) & = \theta(k) + \delta_t \dot{\theta}(k) \\ \dot{\theta}(k+1) & = \dot{\theta}(k) + \frac{3g\delta_t}{2l} \sin(\theta(k)) + \frac{3}{ml^2} u(k)\delta_t \end{cases}$$

où δ_t, g, l, m, constantes, $u(k) \in [-2, 2]$ et

$$X(k) = \begin{bmatrix} \theta \\ \dot{\theta} \end{bmatrix}, \; X(k) \in [0, 2\pi] \times [-8, 8]$$

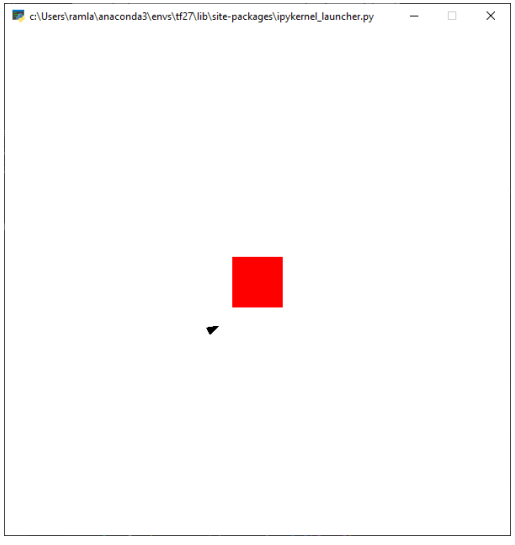
Reward function : Critiquer l'action du contrôleur

$$q(k+1) = m(\theta(k))^2 + 0.01\dot{\theta}(k)^2 + 0.001(u(k))^2, \text{ où } \theta(k) \in [-\pi, \pi]$$

où m est la mesure principale de l'angle

Objectif de stabilité

Systèmes considérés - Dubins car

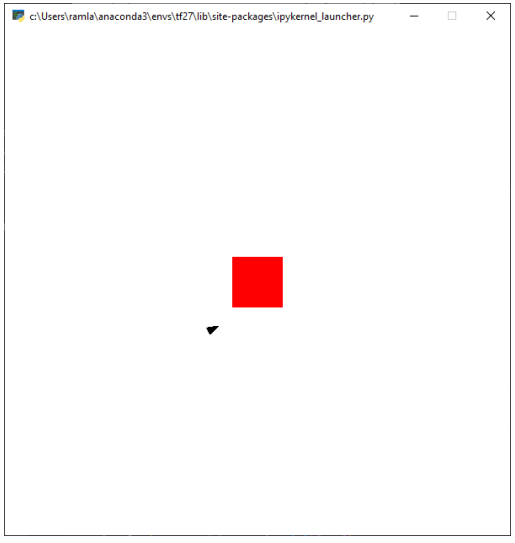


Dubins car

$$F : \begin{cases} x(k+1) &= x(k) + u_1(k)\cos(\theta(k)) \\ y(k+1) &= y(k) + u_1(k)\sin(\theta(k)) \\ \theta(k+1) &= \theta(k) + u_2(k) \end{cases}$$

où $u_1 \in [-0.05, 0.05]$, $u_2 \in [-0.2, 0.2]$, $x, y \in [-2, 2]$, $\theta \in [-\pi, \pi]$

Systèmes considérés - Dubins car



Dubins car

$$F : \begin{cases} x(k+1) &= x(k) + u_1(k)\cos(\theta(k)) \\ y(k+1) &= y(k) + u_1(k)\sin(\theta(k)) \\ \theta(k+1) &= \theta(k) + u_2(k) \end{cases}$$

où $u_1 \in [-0.05, 0.05]$, $u_2 \in [-0.2, 0.2]$, $x, y \in [-2, 2]$, $\theta \in [-\pi, \pi]$

Cost function

$$C(k) = d_{goal} + a\Delta\theta + b||u(k)||^2$$

Objectif de stabilité

Principe du Deep Reinforcement Learning

- Objectif : maximiser la fonction récompense sur l'ensemble des décisions prises
- Toute itération est représentée par un état s_t, une action a_t, une récompense r_t et un nouvel état s_{t+1}
- Critique d'une action : Récompense à l'instant t + "récompense potentiel" à partir du nouvel état (valeur Q) $q_t = r_t + \gamma q_{t+1}, 0 < \gamma < 1$

Principe du Deep Reinforcement Learning

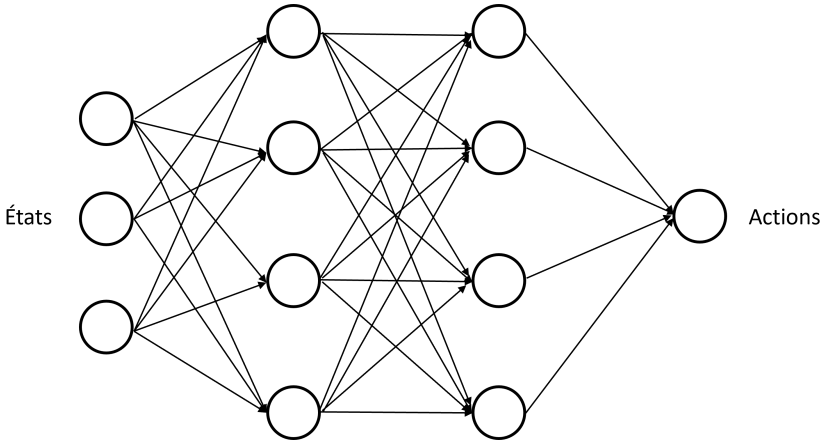
- Objectif : maximiser la fonction récompense sur l'ensemble des décisions prises
- Toute itération est représentée par un état s_t, une action a_t, une récompense r_t et un nouvel état s_{t+1}
- Critique d'une action : Récompense à l'instant t + "récompense potentiel" à partir du nouvel état (valeur Q) $q_t = r_t + \gamma q_{t+1}, 0 < \gamma < 1$

Il faut introduire un décalage entre le réseau qui renvoie q_t et q_{t+1} pour maintenir la stabilité numérique.

- Introduire des réseaux "target" qui marquent ce décalage (copie décalée des réseaux principaux)
- Implémentation sur TensorFlow et Gym (OpenAI) + parallélisation de la boucle d'entraînement

Deep Deterministic Policy Gradient (DDPG)

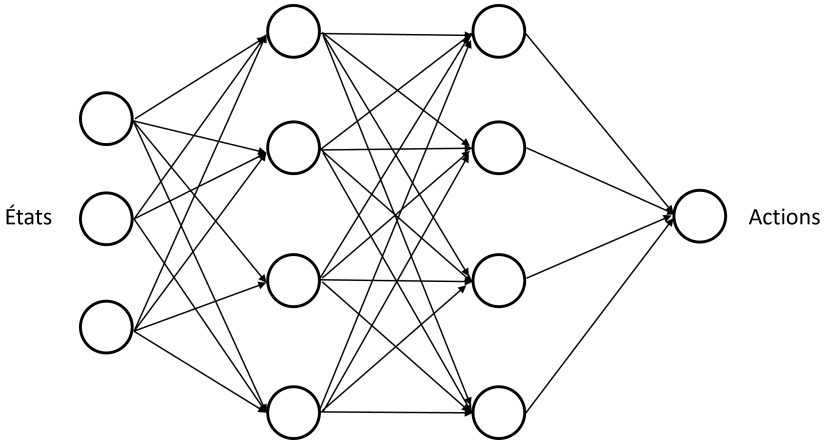
Réseau acteur



- C'est le contrôleur
- Couches 1 et 2 : 16 neurones, ReLU
- Couche 3 : 1 neurone, tanh
- Sortie ramenée à l'échelle des actions

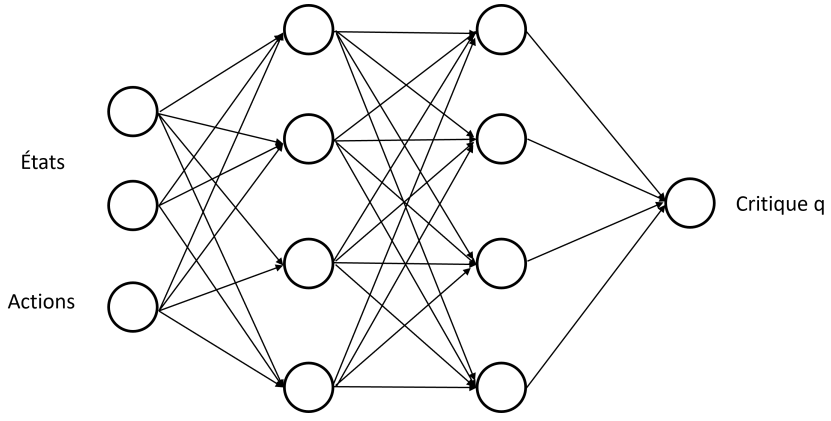
Deep Deterministic Policy Gradient (DDPG)

Réseau acteur



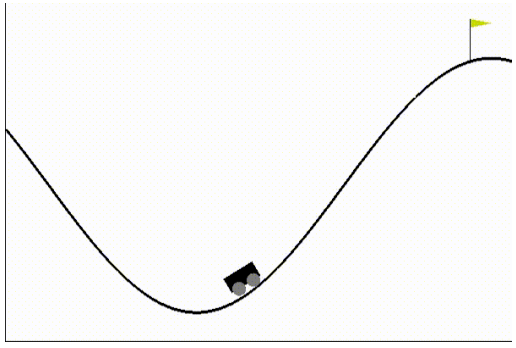
- C'est le contrôleur
- Couches 1 et 2 : 16 neurones, ReLU
- Couche 3 : 1 neurone, tanh
- Sortie ramenée à l'échelle des actions

Réseau critique

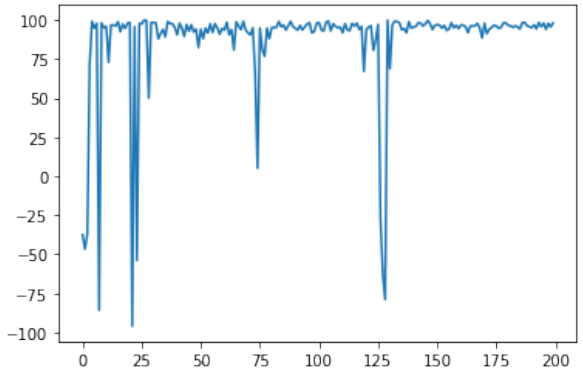


- Concaténation des états et des actions
- Couches 1 et 2 : 16 neurones, ReLU
- Couche 3 : 1 neurone

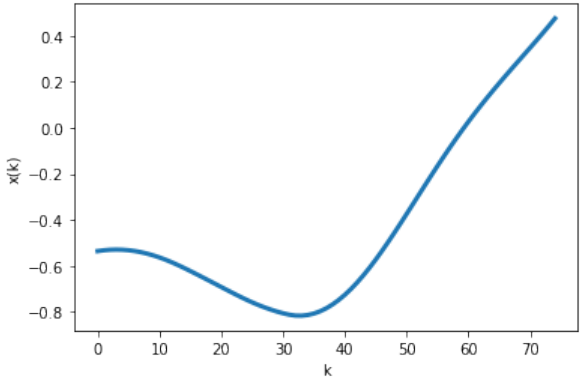
Mountain Car - Contrôleur DDPG



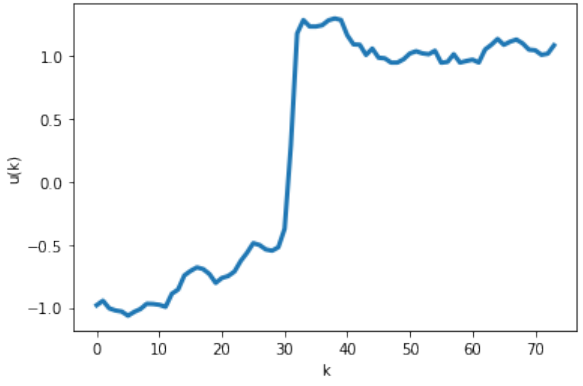
Mountain car



Reward sur 200 épisodes

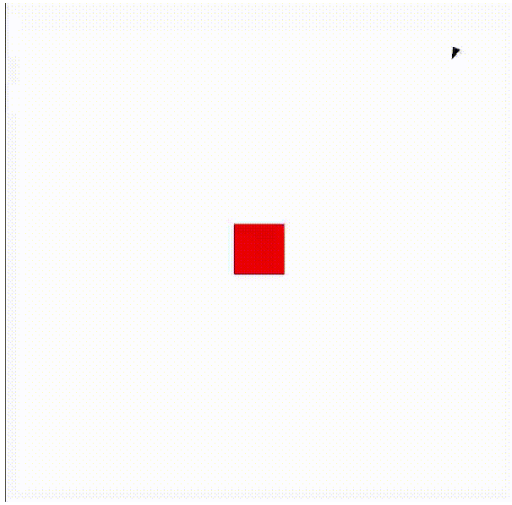


Position de la voiture



Actions du contrôleur

Dubins Car - Contrôleur DDPG



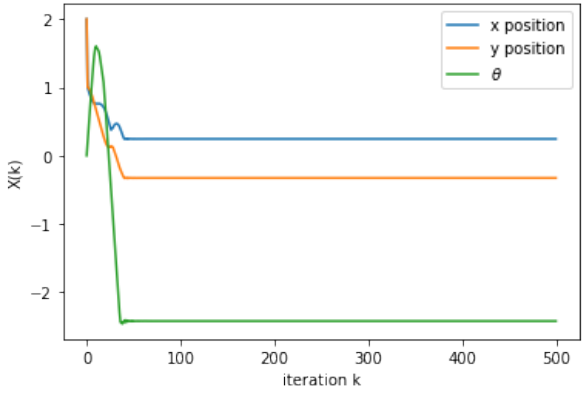
Dubins Car

GridSearch :

- 200 expériences à 40 itérations
- Paramètres aléatoires
- Choix des plus prometteurs

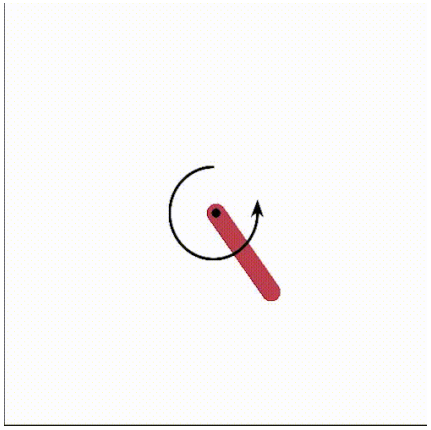
Paramètres à calibrer :

- Learning rates (actor et critic)
- Bruit d'exploration
- Fonction reward (paramètres a et b) (Difficile)

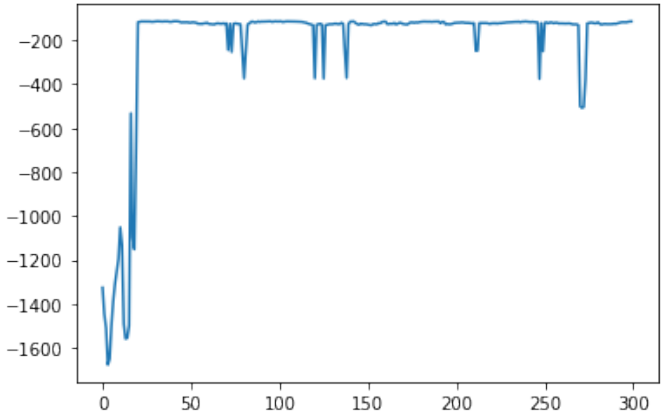


Etats sur un épisode

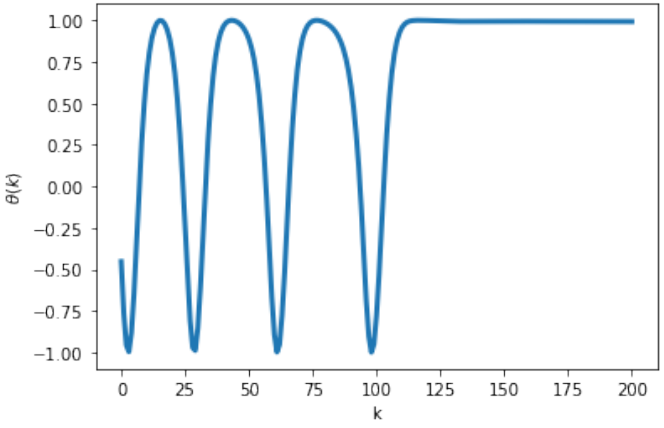
Pendule - Contrôleur DDPG



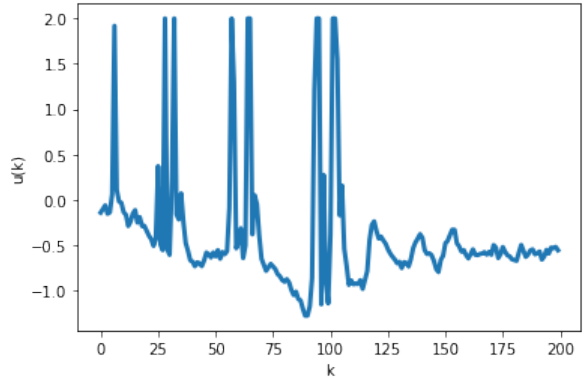
Pendule



Reward sur 200 épisodes



Position du pendule



Actions du contrôleur

Contrôleur DDPG - Régularisation continue

- Problème : Les actions du contrôleur présentent des discontinuités en fonction des états
- Analyse d'atteignabilité par intervalles : Pour éviter une propagation des incertitudes

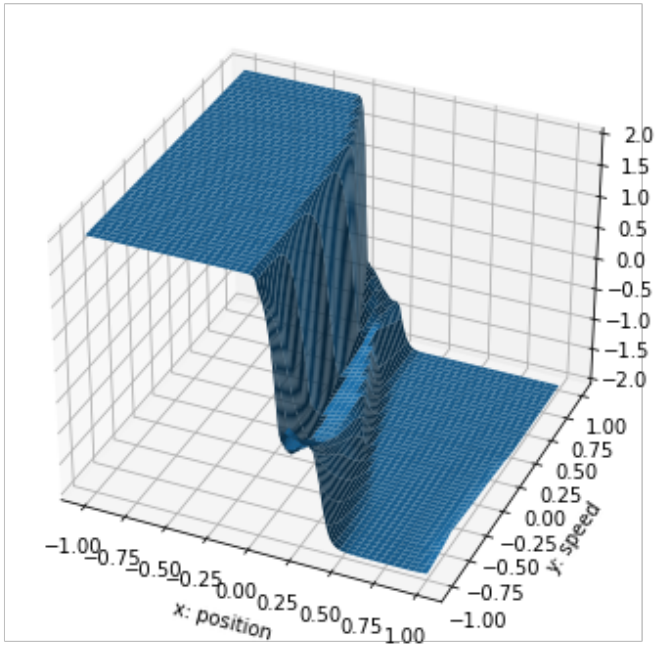
Terme de régularisation rajouté à la fonction de coût de l'acteur

$$\mathcal{R}_s = \lambda_s \mathop{\mathbb{E}}_{s \in \mathcal{B}} \max_{s' \in \mathbb{B}(s,\epsilon)} ||\mu(s) - \mu(s')||_2^2$$

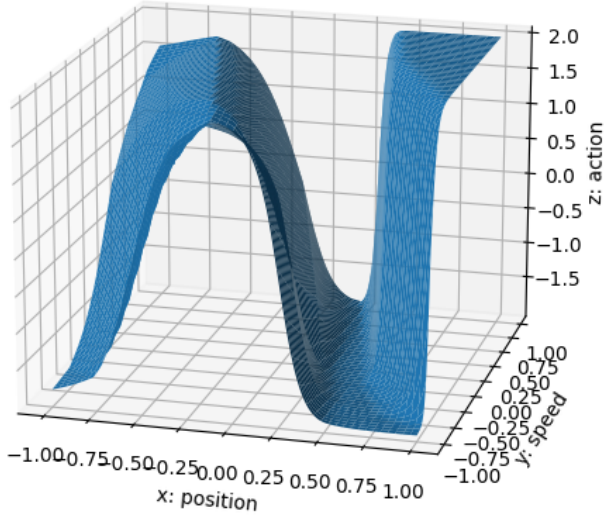
λ_s, intensité de la pénalisation
ϵ, taille de l'espace d'application de la régularisation

- Calcul du gradient : D_s points aléatoires situés autour de s pendant l'entraînement pour simuler la boule

Contrôleur DDPG - Régularisation continue

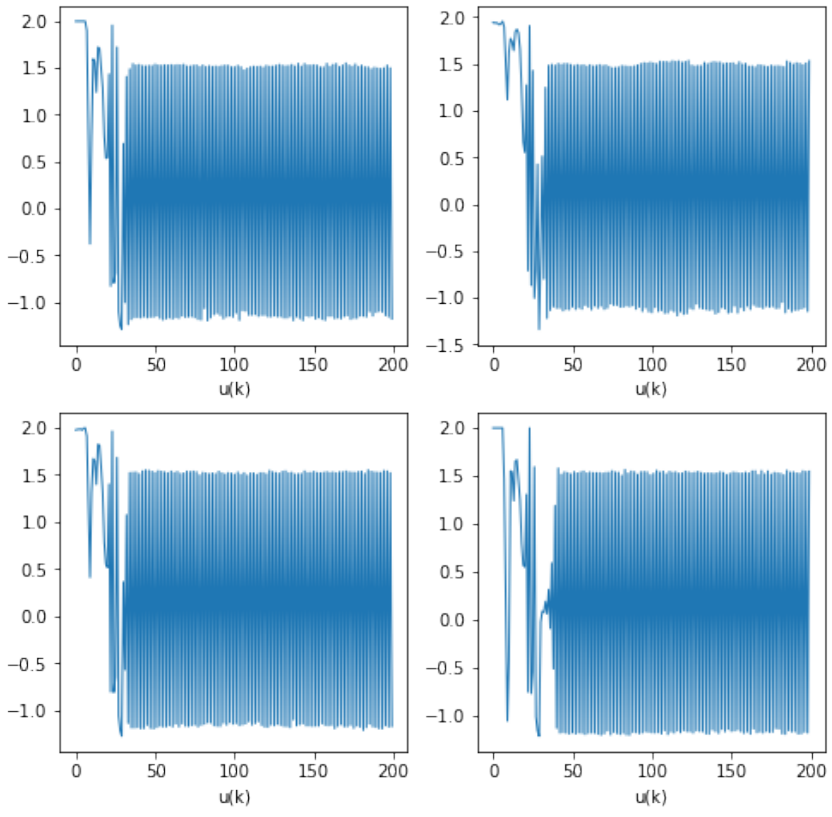


Pendule - Sans régularisation

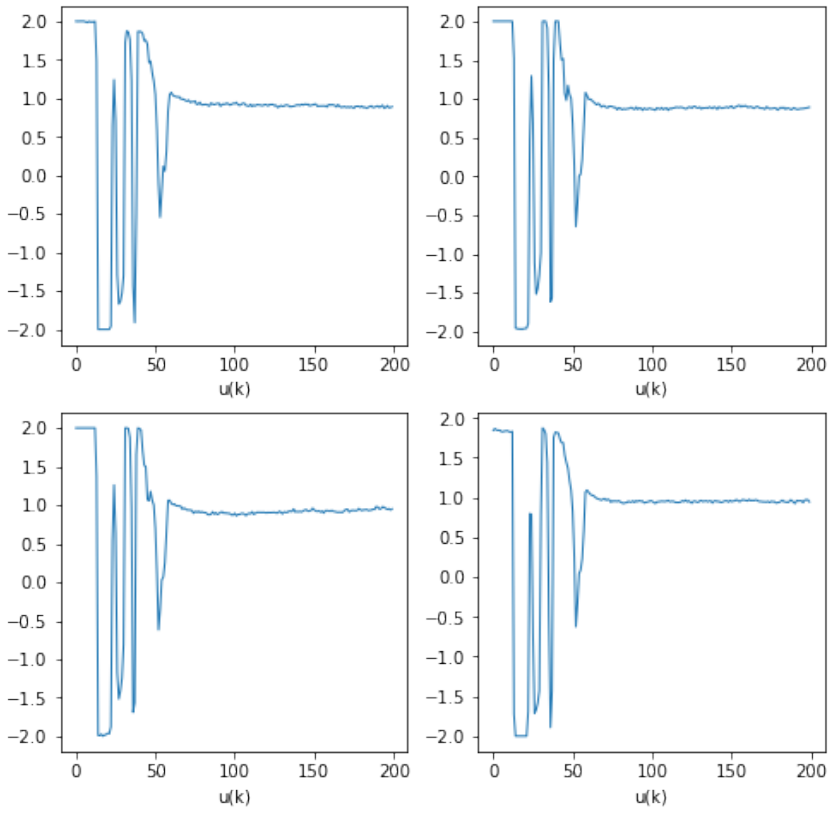


Pendule - Avec régularisation

Contrôleur DDPG - Régularisation continue



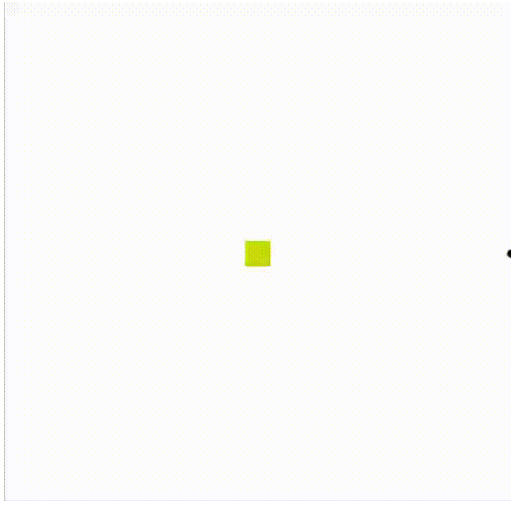
Épisodes joués - Sans régularisation



Épisodes joués - Avec régularisation

Contrôle MPC

Systèmes considérés - double intégrateur



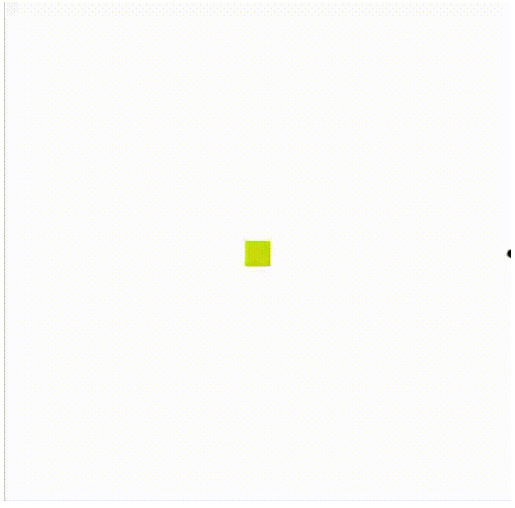
Double intégrateur

$$F : \begin{cases} x(k+1) &= \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} x(k) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u(k) \\ y(k) &= \begin{bmatrix} 1 & 0 \end{bmatrix} x(k) \end{cases}$$

où $u(k) \in [-1, 1]$ et

$$X(k) = \begin{bmatrix} x1 \\ x2 \end{bmatrix}, X(k) \in [-\infty, +\infty] \times [-\infty, +\infty]$$

Systèmes considérés - double intégrateur



Double intégrateur

$$F : \begin{cases} x(k+1) &= \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} x(k) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u(k) \\ y(k) &= \begin{bmatrix} 1 & 0 \end{bmatrix} x(k) \end{cases}$$

où $u(k) \in [-1, 1]$ et

$$X(k) = \begin{bmatrix} x1 \\ x2 \end{bmatrix}, X(k) \in [-\infty, +\infty] \times [-\infty, +\infty]$$

Cost function (à minimiser)

$$\sum_{t=0}^{\infty} x^{'}(t) Q x(t) + R u^2(t)$$

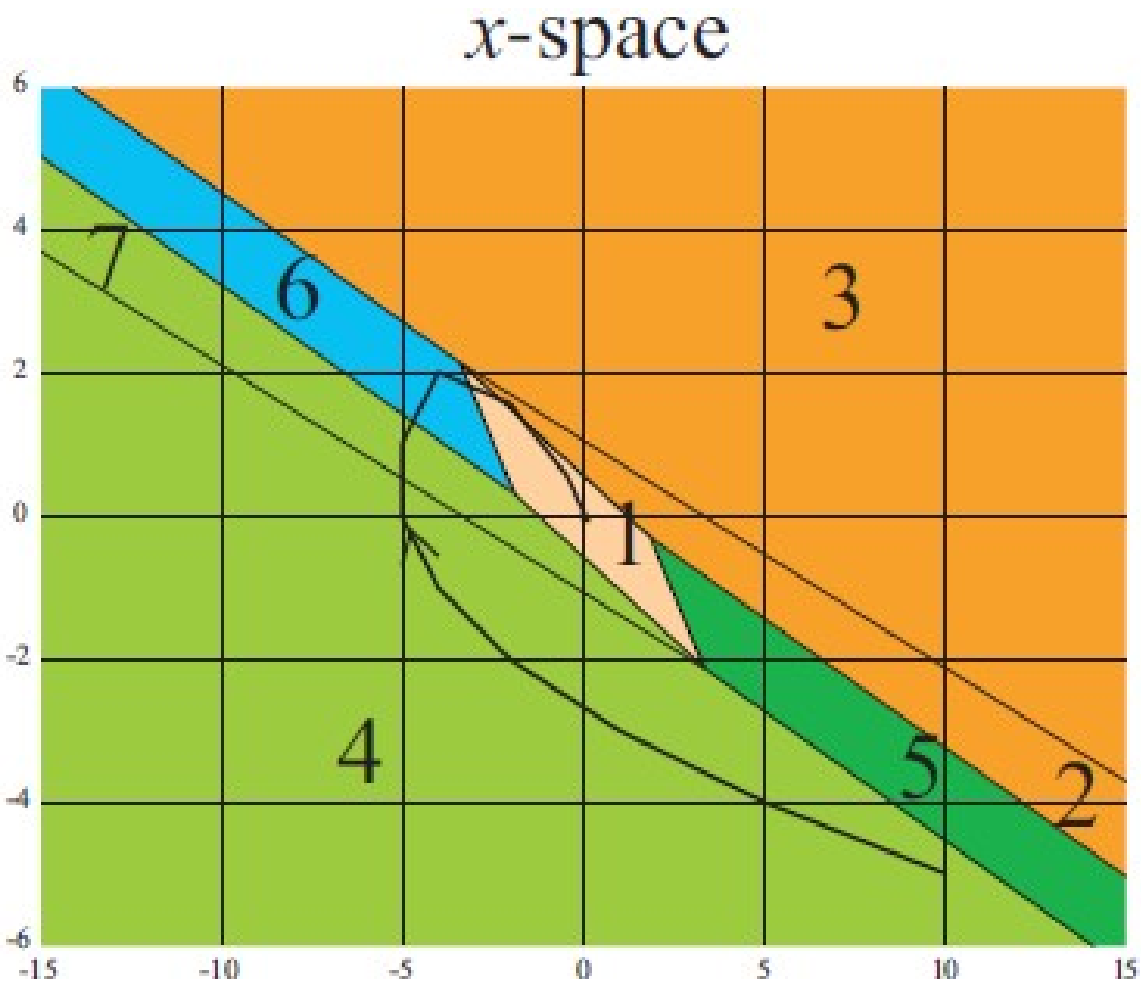
Objectif de stabilité

Contrôleur MPC (explicit)

- Problème : Stockage d'informations dans le cas où il y a plusieurs régions
- Solution : Représentation de ce MPC par des réseaux de neurones

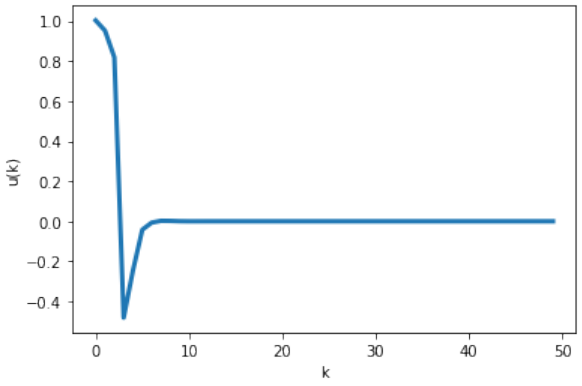
- On utilise un simple modèle de réseaux de neurones composés de deux couches intermédiaires
- Chaque couche est composée de 16 neurones, avec des fonctions ReLU

Contrôleur MPC (explicit)

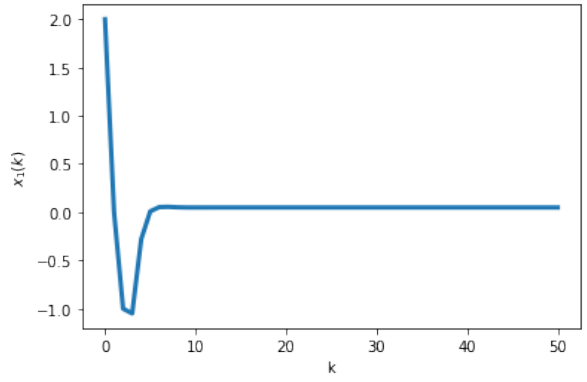


Partition polyédrique de l'espace d'état et trajectoires MPC en boucle fermée

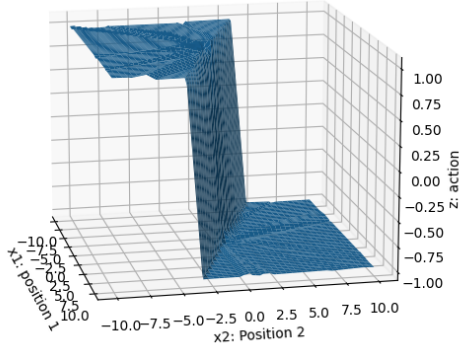
Contrôleur MPC (explicit)



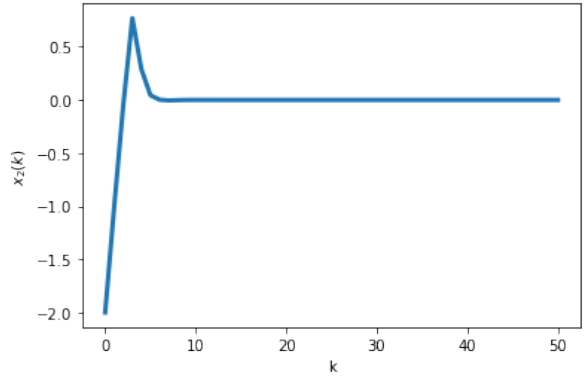
Actions du contrôleur



1ère composante de l'état X



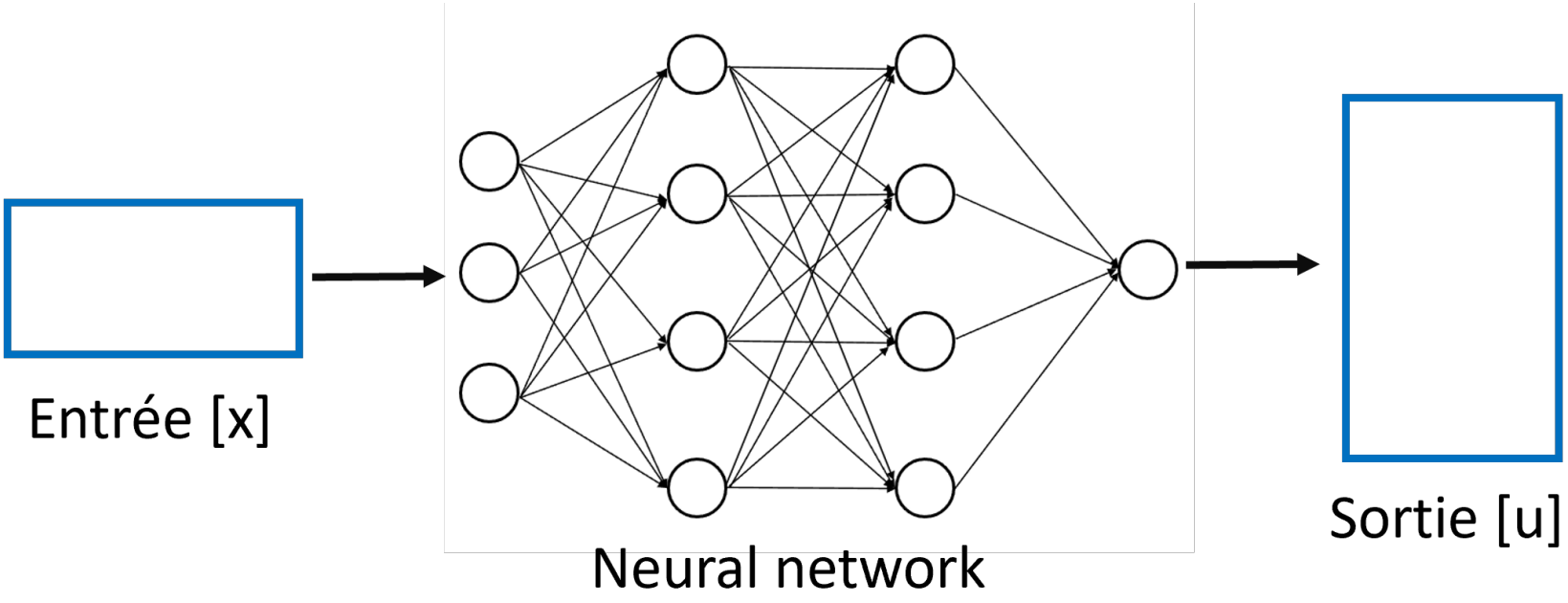
Actions du contrôleur



2ème composante de l'état X

Atteignabilité du réseau de neurones

- Fonctions d'activation croissantes
- Réseau de neurone Φ à couches denses et L couches intermédiaires
- Paramètres $(n_0, n_1, \ldots, n_L) \in \mathbb{N}^{L+1}$, $W^l \in \mathbb{R}^{n_l \times n_{l-1}}$, $b^l \in \mathbb{R}^{n_l}$, $\sigma_l : \mathbb{R}^{n_l} \to \mathbb{R}^{n_l}$
- Objectif : Image d'un intervalle $[x]$ par le réseau de neurones ?



Atteignabilité du réseau de neurones

- Fonctions d'activation croissantes
- Réseau de neurone Φ à couches denses et L couches intermédiaires
- Paramètres $(n_0, n_1, \ldots, n_L) \in \mathbb{N}^{L+1}$, $W^l \in \mathbb{R}^{n_l \times n_{l-1}}$, $b^l \in \mathbb{R}^{n_l}$, $\sigma_l : \mathbb{R}^{n_l} \to \mathbb{R}^{n_l}$
- Objectif : Image d'un intervalle $[x]$ par le réseau de neurones ?

$$\underline{x}^l = \sigma_l(\sum_{j=1}^{n_{l-1}} \underline{p}_{i,j} + b_i^l)$$

$$\overline{x}^l = \sigma_l(\sum_{j=1}^{n_{l-1}} \overline{p}_{i,j} + b_i^l)$$

avec

$$\underline{p}_{i,j} = \begin{cases} w_{i,j}^l \underline{x}_j \text{ si } w_{i,j}^l \geq 0 \\ w_{i,j}^l \overline{x}_j \text{ si } w_{i,j}^l < 0 \end{cases}$$

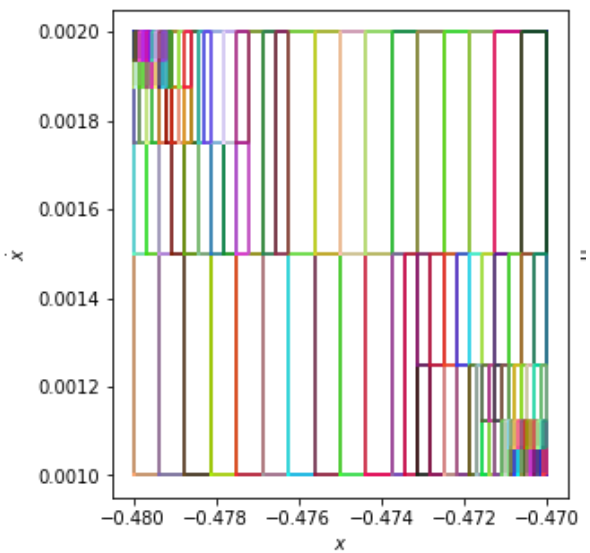
$$\overline{p}_{i,j} = \begin{cases} w_{i,j}^l \overline{x}_j \text{ si } w_{i,j}^l \geq 0 \\ w_{i,j}^l \underline{x}_j \text{ si } w_{i,j}^l < 0 \end{cases}$$

Principe de l'algorithme d'atteignabilité

- Sur-approximation de l'image réelle en utilisant les formules
- L'erreur de sur-approximation diminue avec la taille des intervalles considérés
- Idée : Partitionner l'intervalle de départ en sous-intervalles permettant de garantir une erreur inférieure à δ fixé

Principe de l'algorithme d'atteignabilité

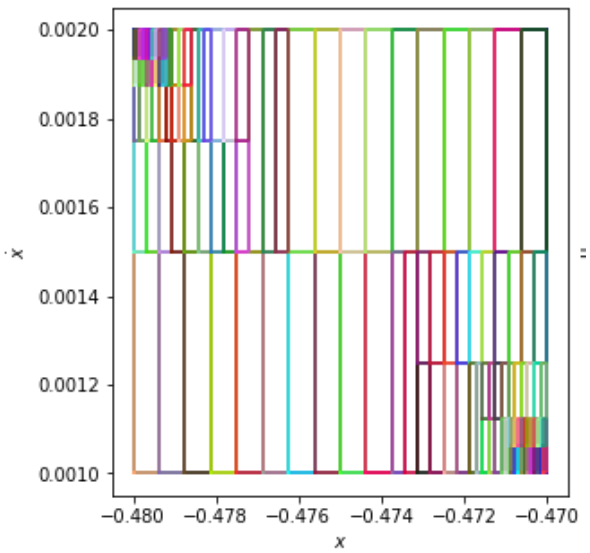
- Sur-approximation de l'image réelle en utilisant les formules
- L'erreur de sur-approximation diminue avec la taille des intervalles considérés
- Idée : Partitionner l'intervalle de départ en sous-intervalles permettant de garantir une erreur inférieure à δ fixé



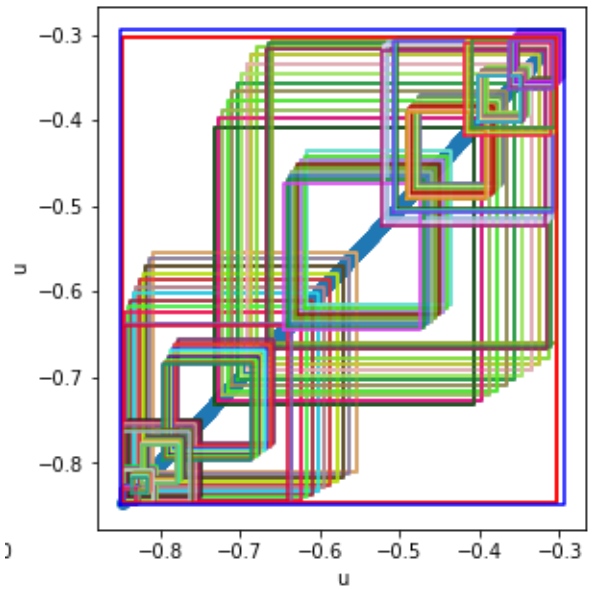
Intervalle de départ avec $\epsilon = 0.0001$

Principe de l'algorithme d'atteignabilité

- Sur-approximation de l'image réelle en utilisant les formules
- L'erreur de sur-approximation diminue avec la taille des intervalles considérés
- Idée : Partitionner l'intervalle de départ en sous-intervalles permettant de garantir une erreur inférieure à δ fixé

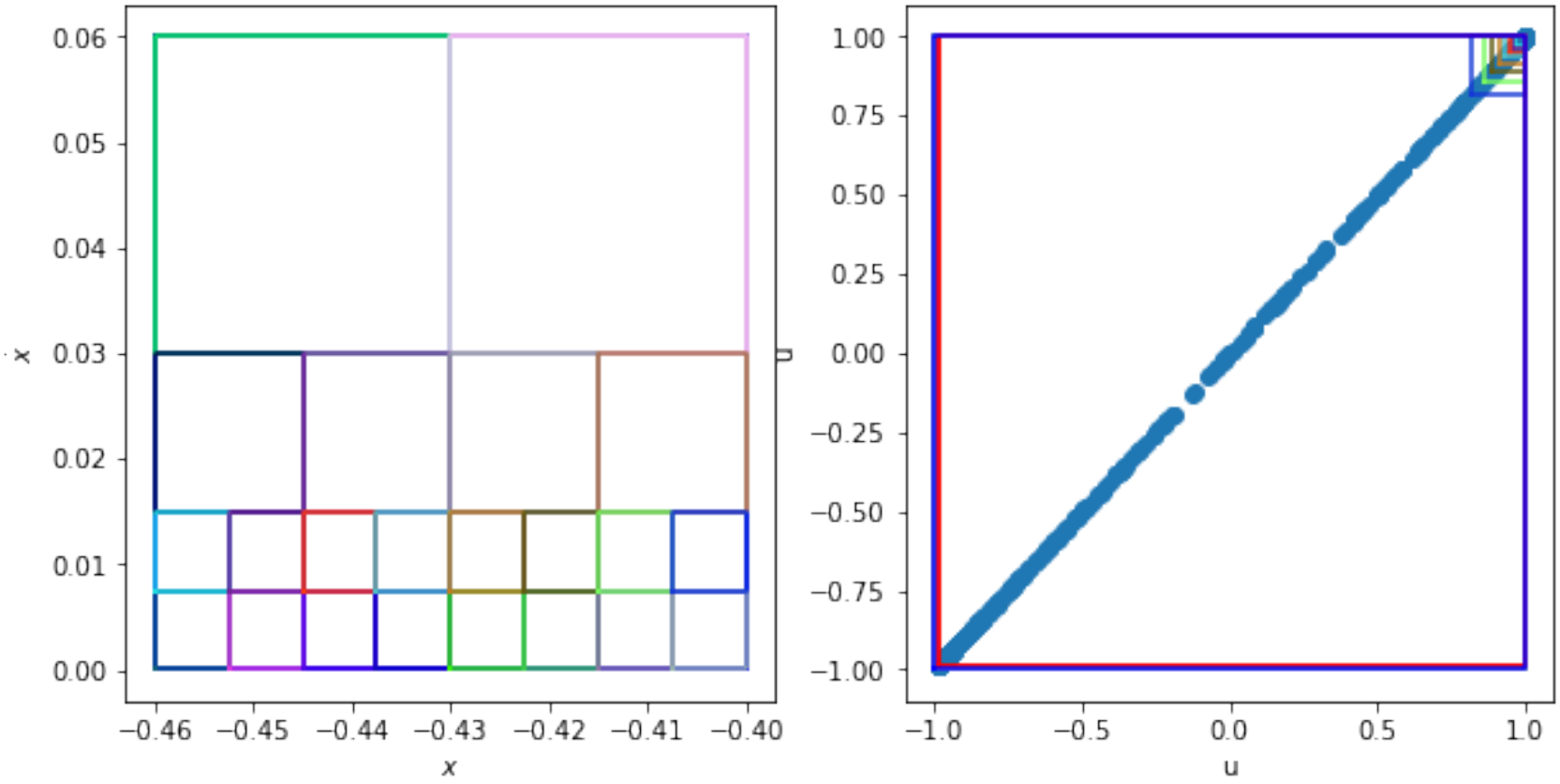


Intervalle de départ avec $\epsilon = 0.0001$



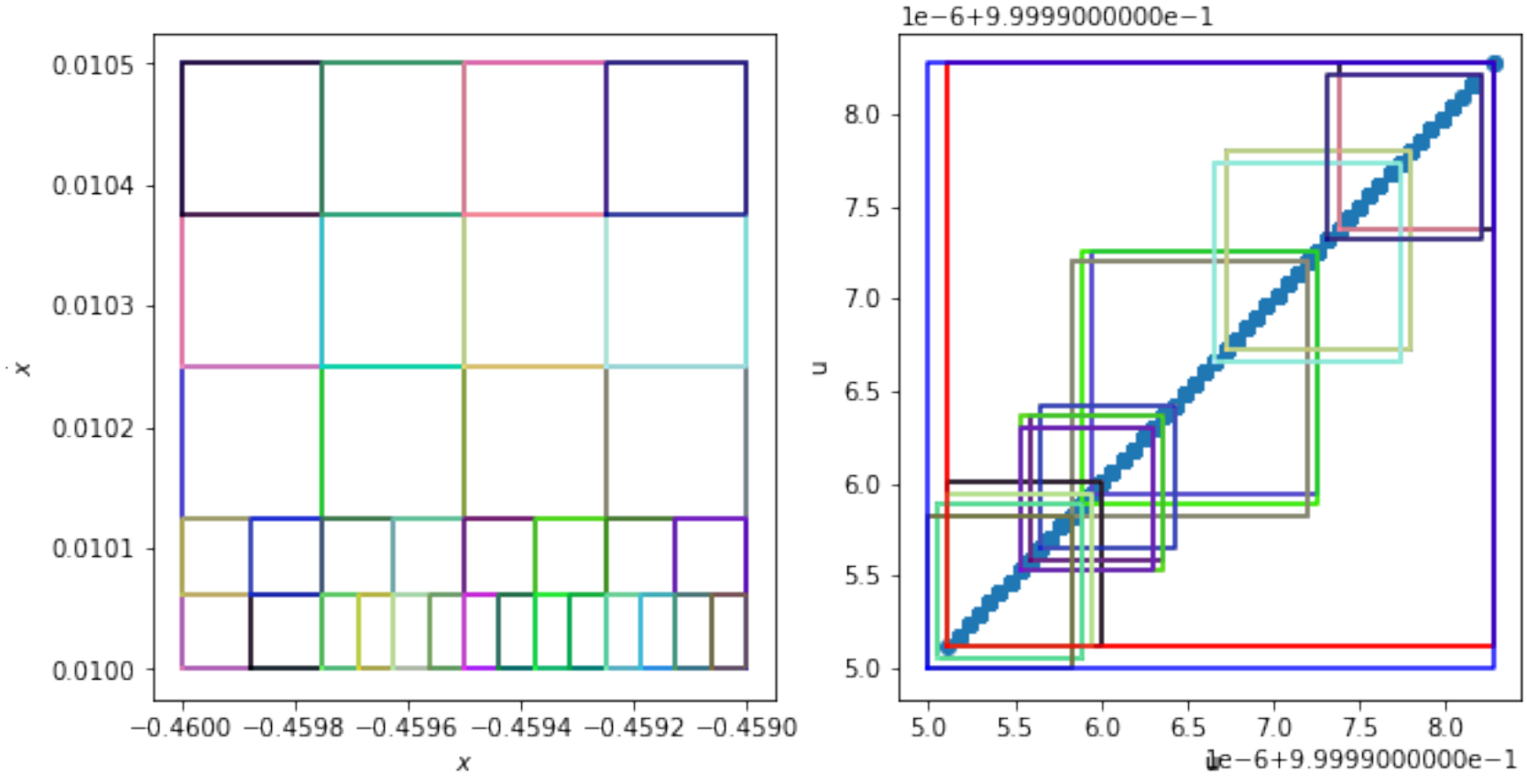
Contrôle du découpage

Résultats obtenus



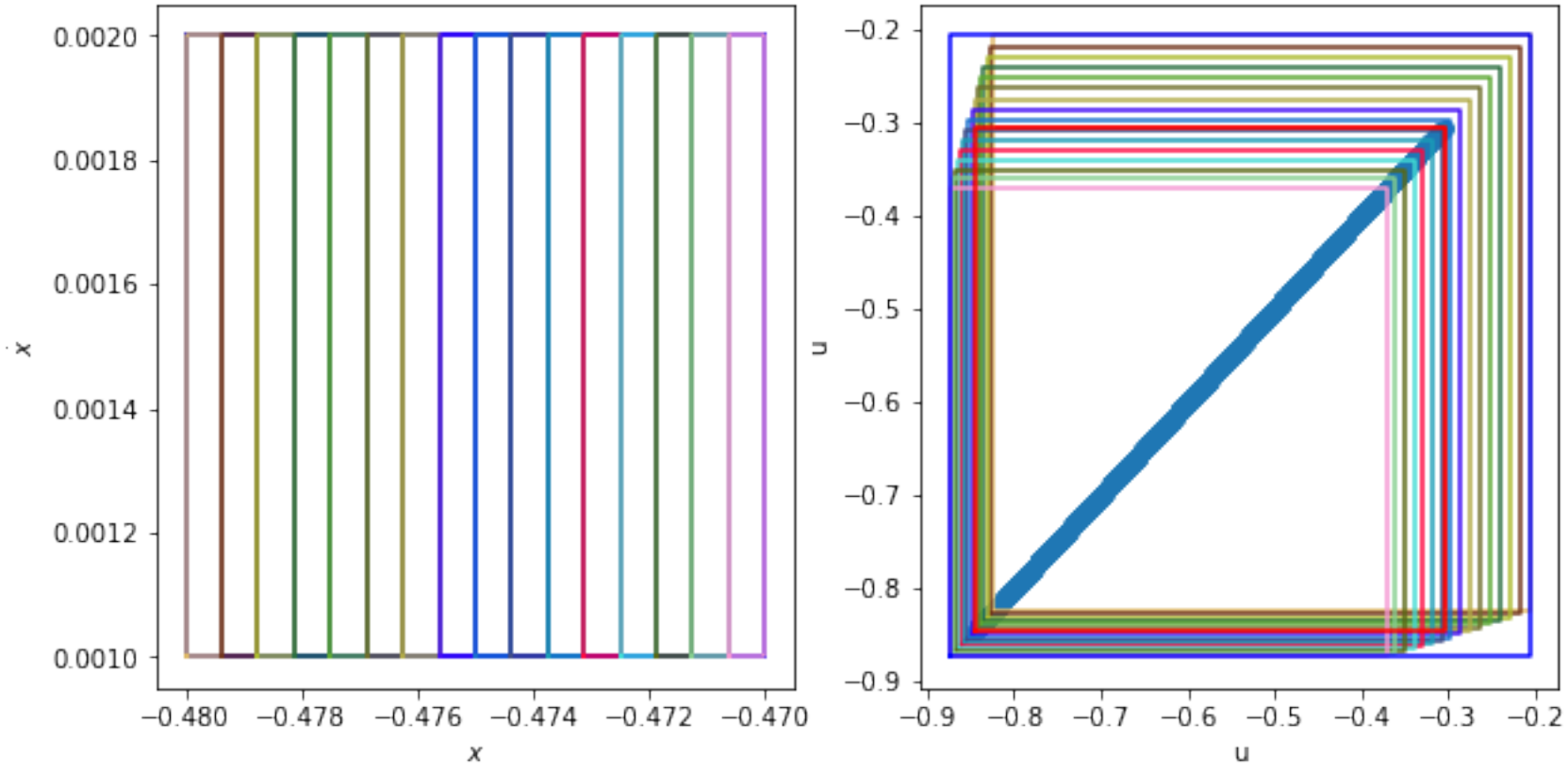
Mountain car - Intervalle large

Résultats obtenus



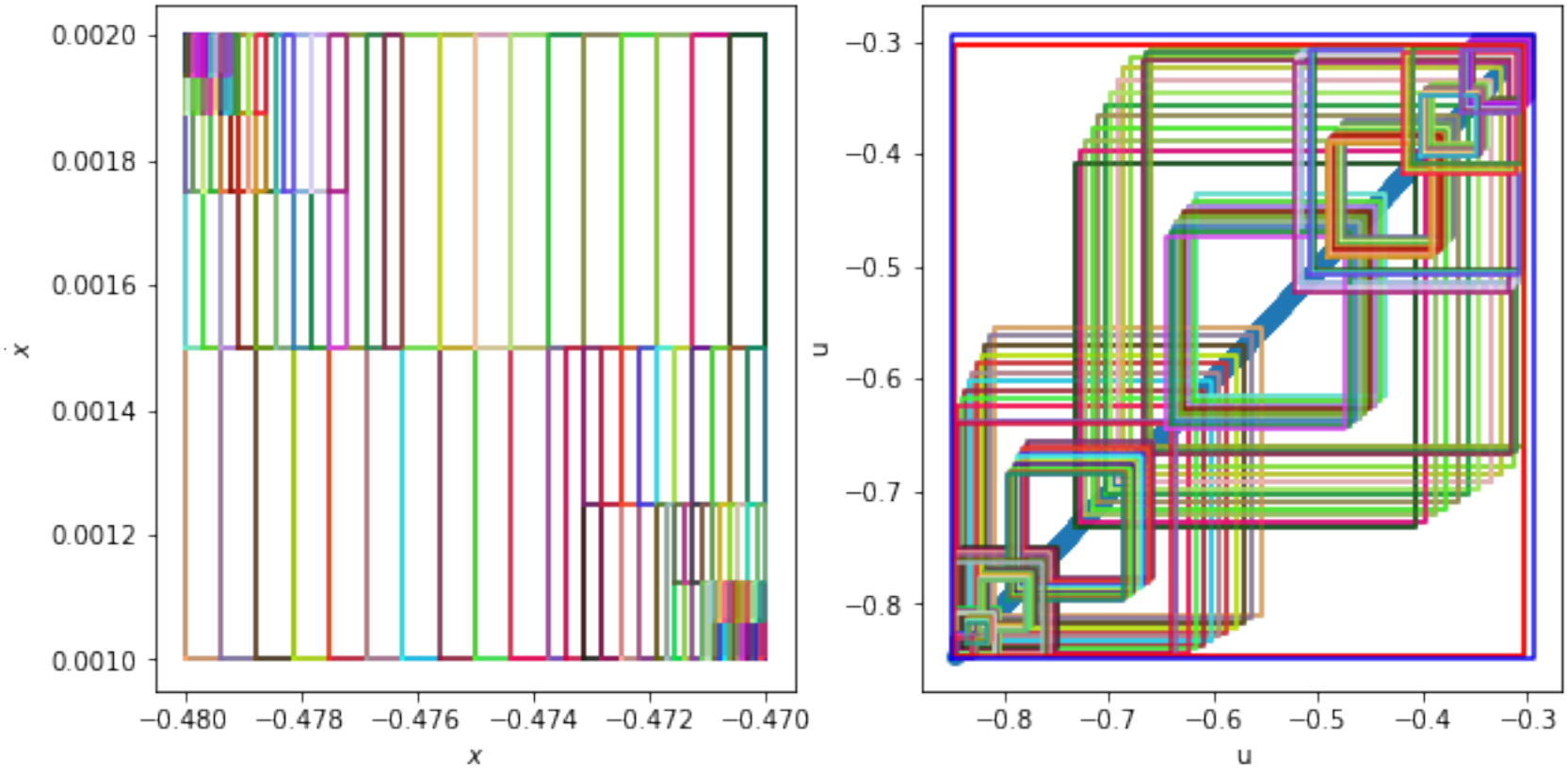
Mountain car - Petit Intervalle

Résultats obtenus



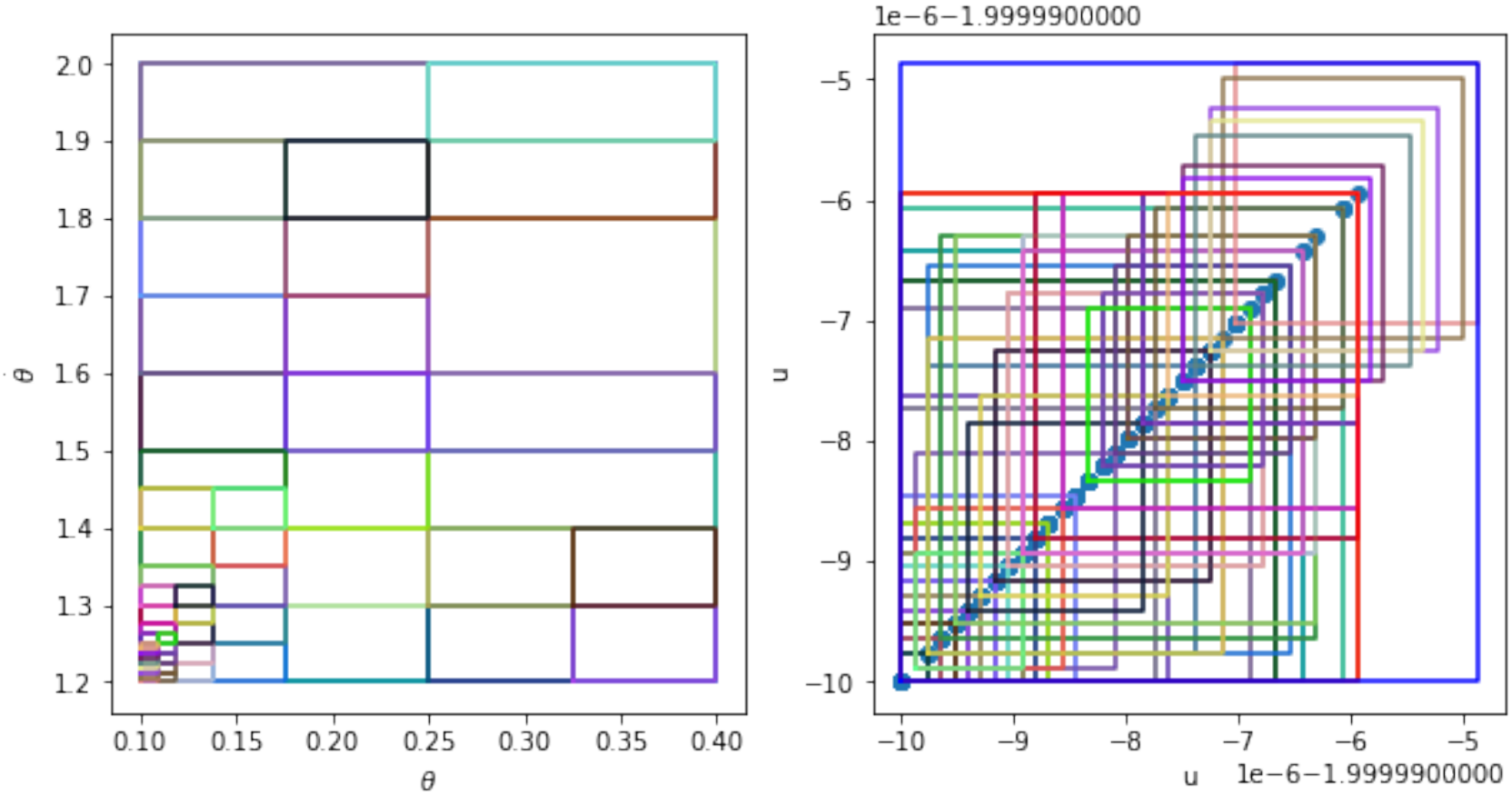
Mountain car - $\epsilon = 0.001$

Résultats obtenus



Mountain car - $\epsilon = 0.0001$

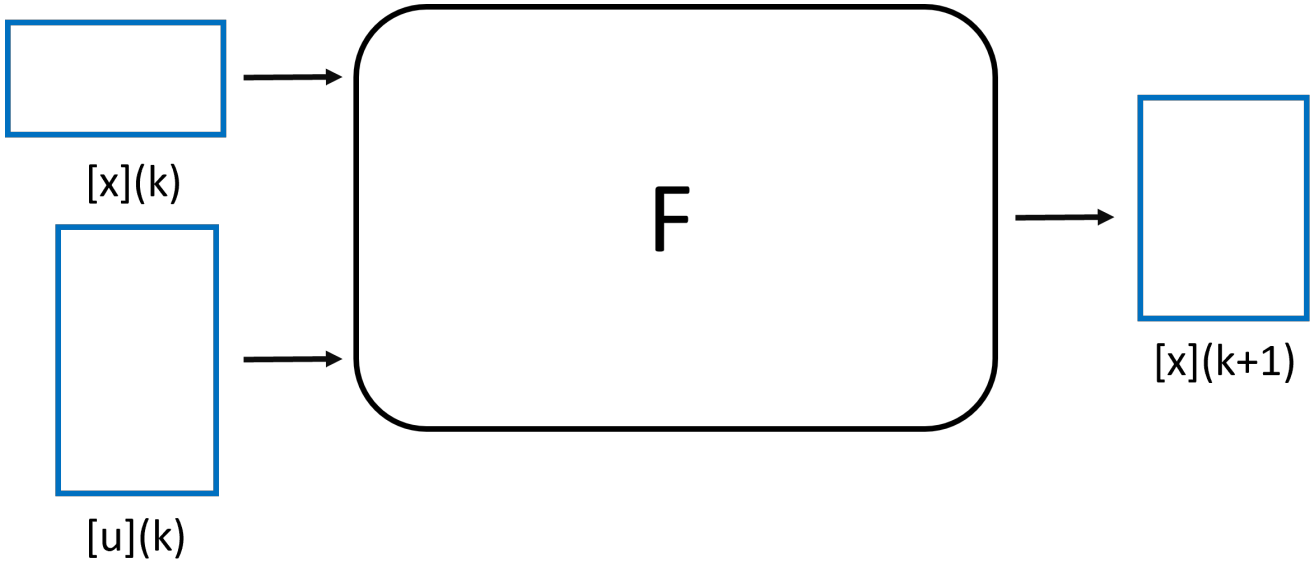
Résultats obtenus



Atteignabilité du contrôleur du pendule

Atteignabilité du système dynamique

- Faire de l'arithmétique d'intervalles
- Remplacer les opérations de la dynamique F par des opérations sur des intervalles



Atteignabilité du système dynamique

- Faire de l'arithmétique d'intervalles
- Remplacer les opérations de la dynamique F par des opérations sur des intervalles

Exemples

$$[\underline{x}, \overline{x}]^{-1} = \begin{cases} [-\infty, \infty] & \text{si } \underline{x} < 0 \text{ et } \overline{x} > 0 \\ \left[-\infty, \frac{1}{\underline{x}}\right] & \text{si } \overline{x} = 0 \\ \left[\frac{1}{\overline{x}}, \infty\right] & \text{si } \underline{x} = 0 \\ \left[\frac{1}{\overline{x}}, \frac{1}{\underline{x}}\right] & \text{si } \underline{x}\overline{x} > 0 \\ \emptyset & \text{si } \underline{x} = \overline{x} = 0 \end{cases}$$

Atteignabilité du système dynamique

- Faire de l'arithmétique d'intervalles
- Remplacer les opérations de la dynamique F par des opérations sur des intervalles

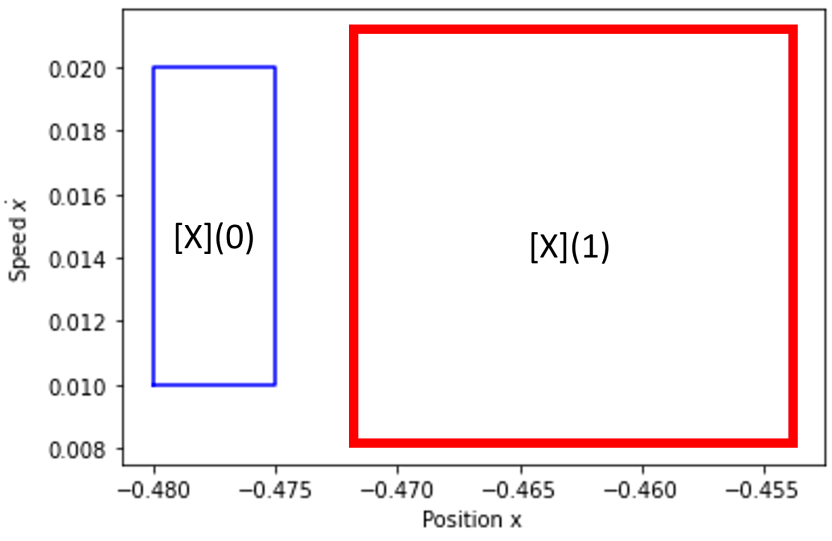
Exemples

$$[\underline{x}, \overline{x}]^{-1} = \begin{cases} [-\infty, \infty] & \text{si } \underline{x} < 0 \text{ et } \overline{x} > 0 \\ \left[-\infty, \frac{1}{\underline{x}}\right] & \text{si } \overline{x} = 0 \\ \left[\frac{1}{\overline{x}}, \infty\right] & \text{si } \underline{x} = 0 \\ \left[\frac{1}{\overline{x}}, \frac{1}{\underline{x}}\right] & \text{si } \underline{x}\overline{x} > 0 \\ \emptyset & \text{si } \underline{x} = \overline{x} = 0 \end{cases}$$

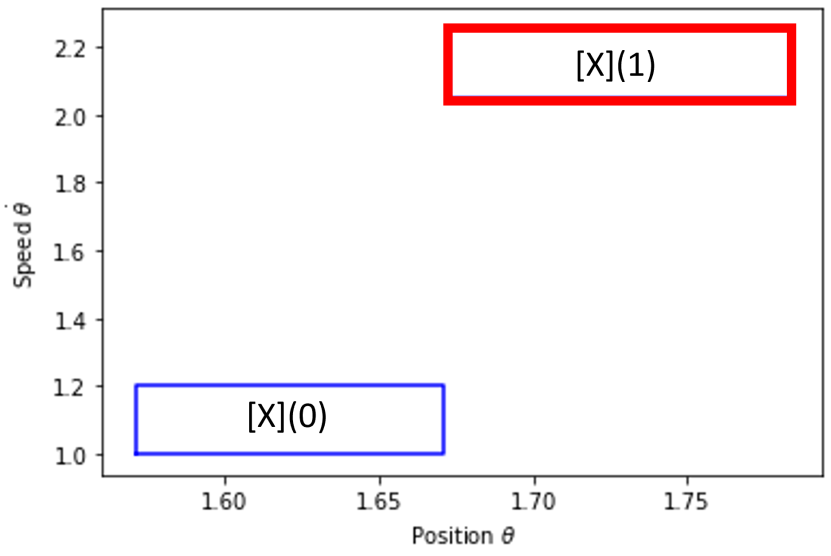
$$\cos([\underline{x}, \overline{x}]) = [\underline{a}, \overline{a}]\text{, où } \underline{a} = \begin{cases} -1 & \text{si } \exists k \in \mathbb{Z} | (2k\pi + \pi) \in [\underline{x}, \overline{x}] \\ \min(\cos(\underline{a}), \cos(\overline{a})) & \text{sinon} \end{cases}$$

$$\overline{a} = \begin{cases} 1 & \text{si } \exists k \in \mathbb{Z} | (2k\pi) \in [\underline{x}, \overline{x}] \\ \max(\cos(\underline{a}), \cos(\overline{a})) & \text{sinon} \end{cases}$$

Résultats obtenus



Mountain car sur une itération

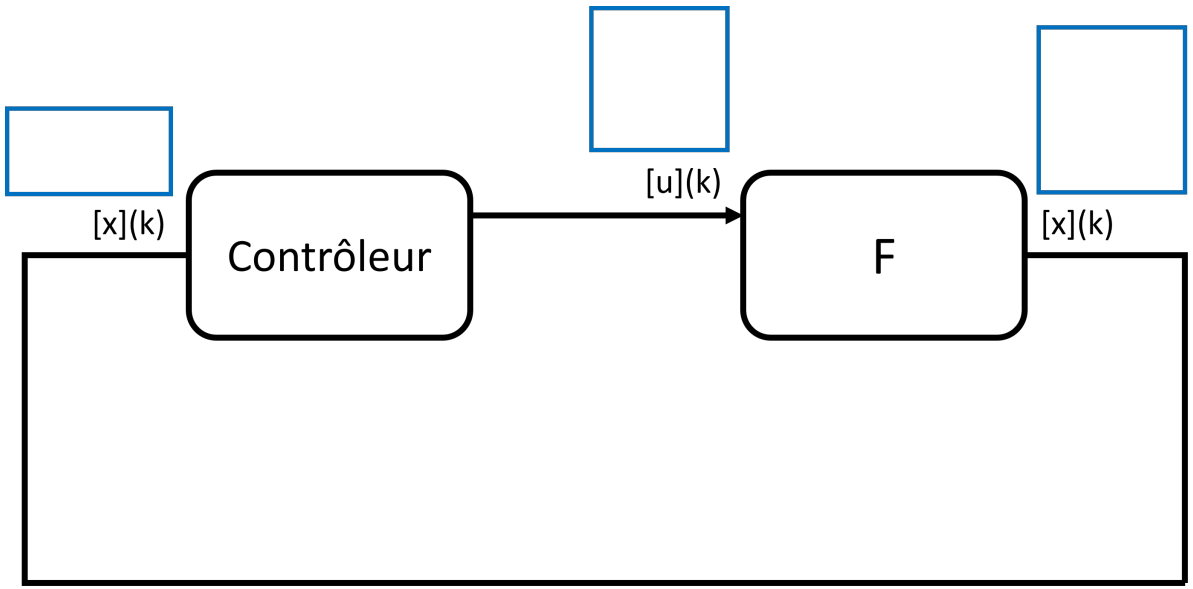


Pendule sur une itération

Contrôle garanti

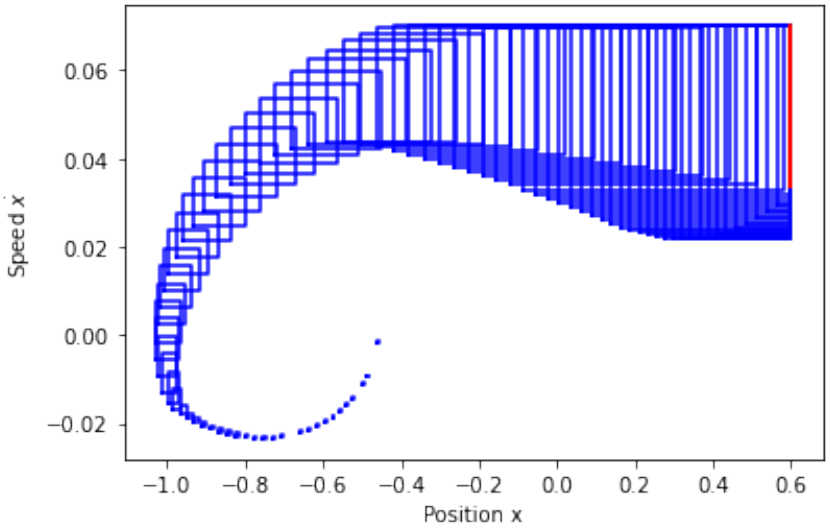
- Objectif : Montrer que le système vérifie une spécification S

- Itérer les algorithmes précédents pour estimer les ensembles atteignables au bout de k itérations et simuler le système en boucle fermée

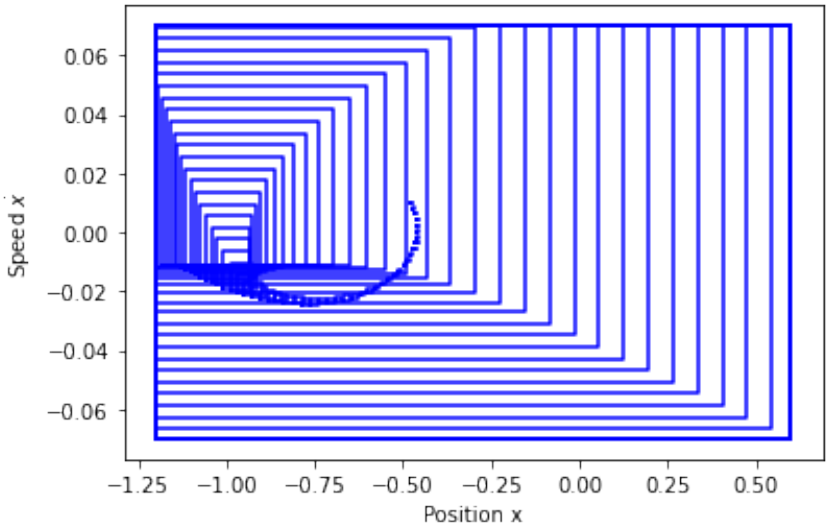


Contrôle garanti

Mountain Car



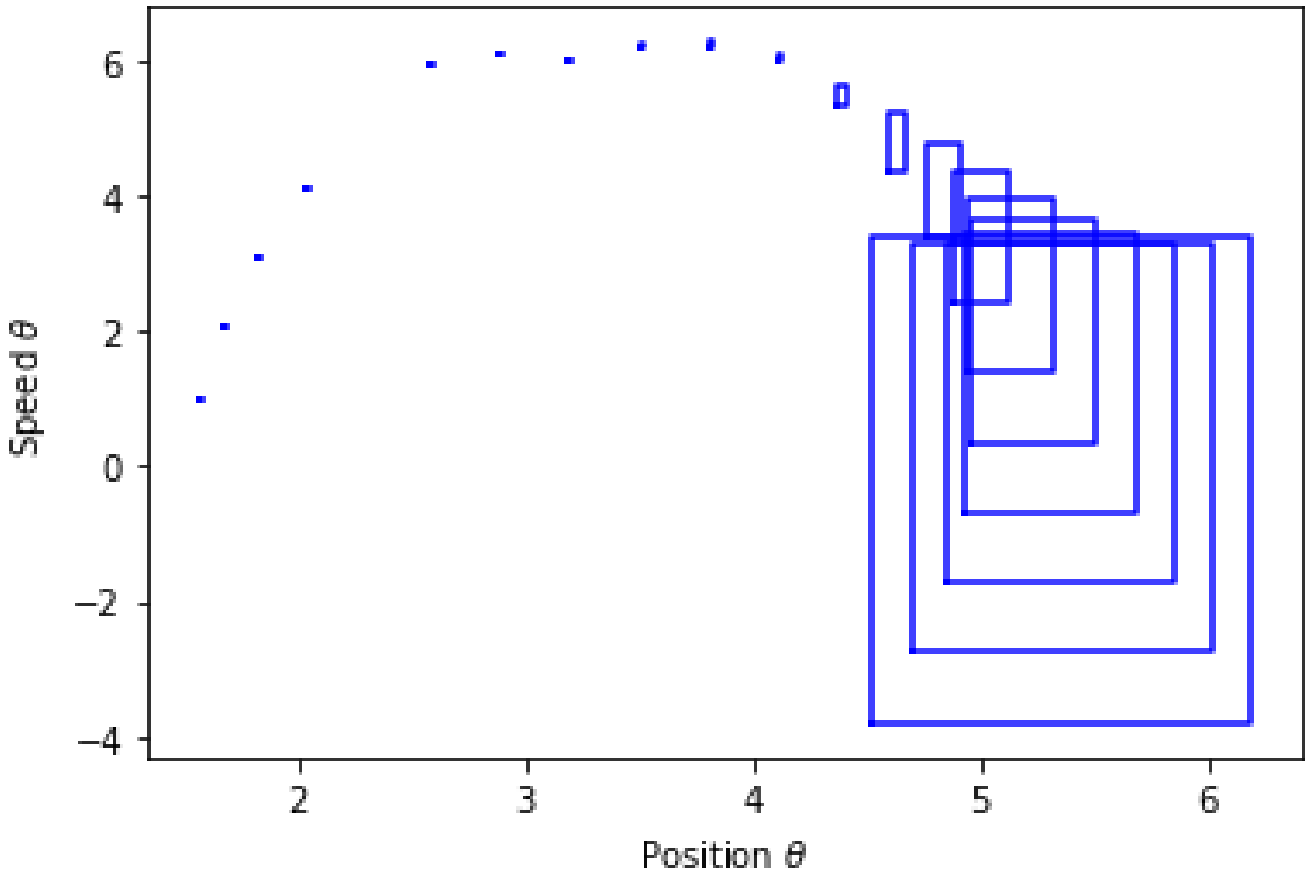
$x(0) \in [-0.48, -0.4795], \dot{x}(0) \in [0.01, 0.0101]$



$x(0) \in [-0.48, -0.475], \dot{x}(0) \in [0.01, 0.0101]$

Contrôle garanti

Pendule



Intervalle très petit

Principe de l'algorithme précis

Considérons le système $x(k+1) = x(k) + u(k)$

- Motivation : $x(k) \in [-1, 0] \rightarrow u(k) = +1$ et $x(k) \in]0, 1] \rightarrow u(k) = -1$
- Algorithme naïf d'atteignabilité : $x(k+1)\,[-2, 2]$, or on sait que $x(k+1)\,[-1, 1]$
- Idée : Séparer les intervalles en fonction des actions qui sont renvoyées par l'atteignabilité du réseau de neurones

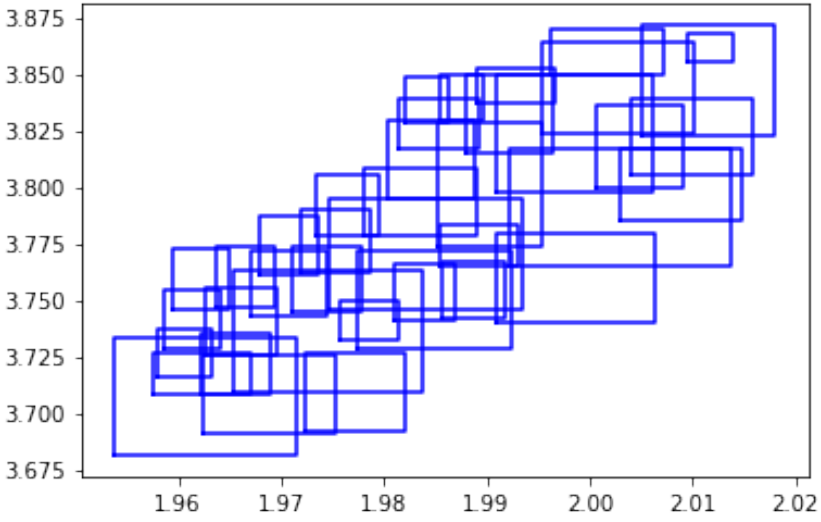
ϵ : tolérance sur les actions et dépendant du problème

Pour $x(k) \in [\underline{x}, \overline{x}] = [x]$, si $\rho([\Phi]([x])) > \epsilon$, alors on divise

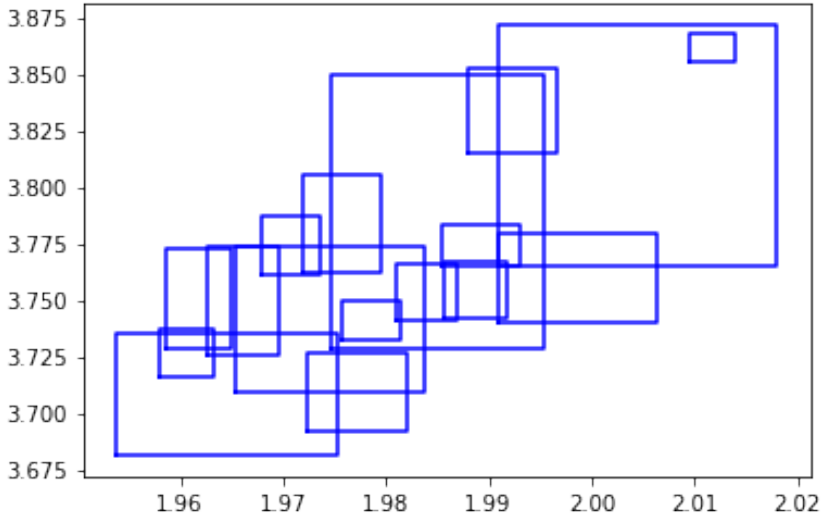
$$[x] \rightarrow \{[x_1], [x_2]\}$$

Critère IoU (Jaccard Index)

- Problème : Explosion du nombre de divisions
- Régularisation continue : permet de réduire les divisions
- Recoller des intervalles : IoU (Intersection over Union) et threshold



Avec d'appliquer le critère

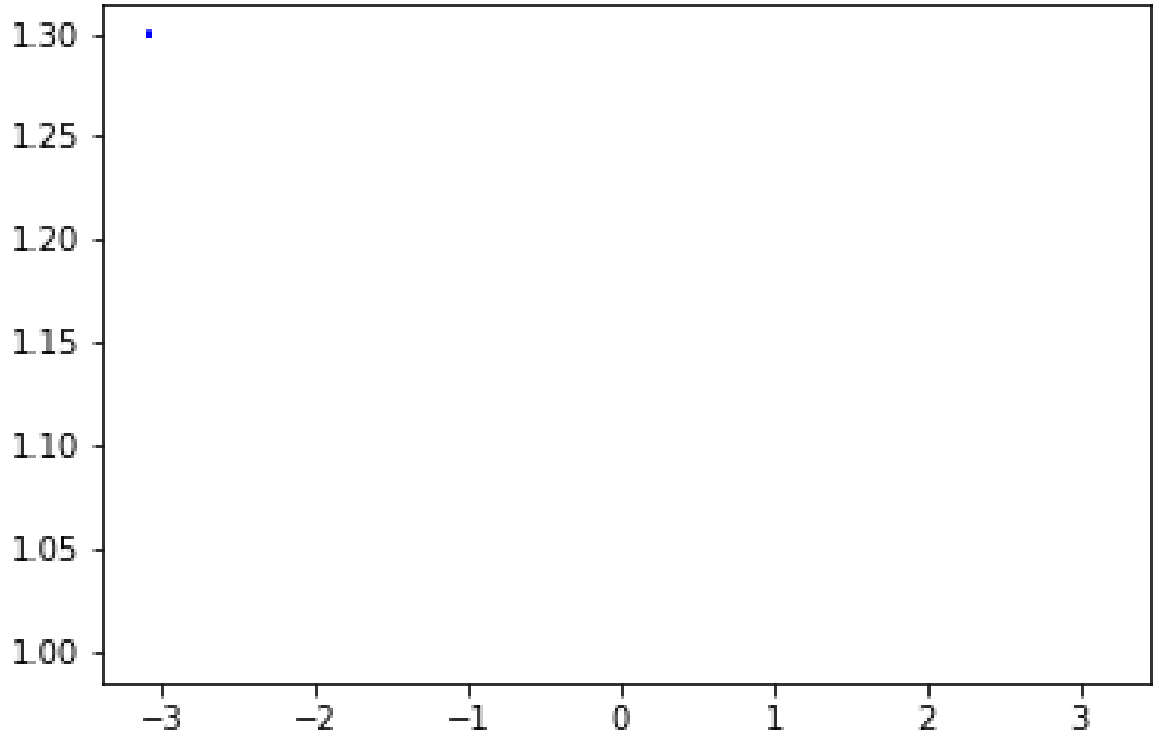


Critère IoU (threshold = 0.15)

Approche inspirée des méthodes Computer Vision en dimension n

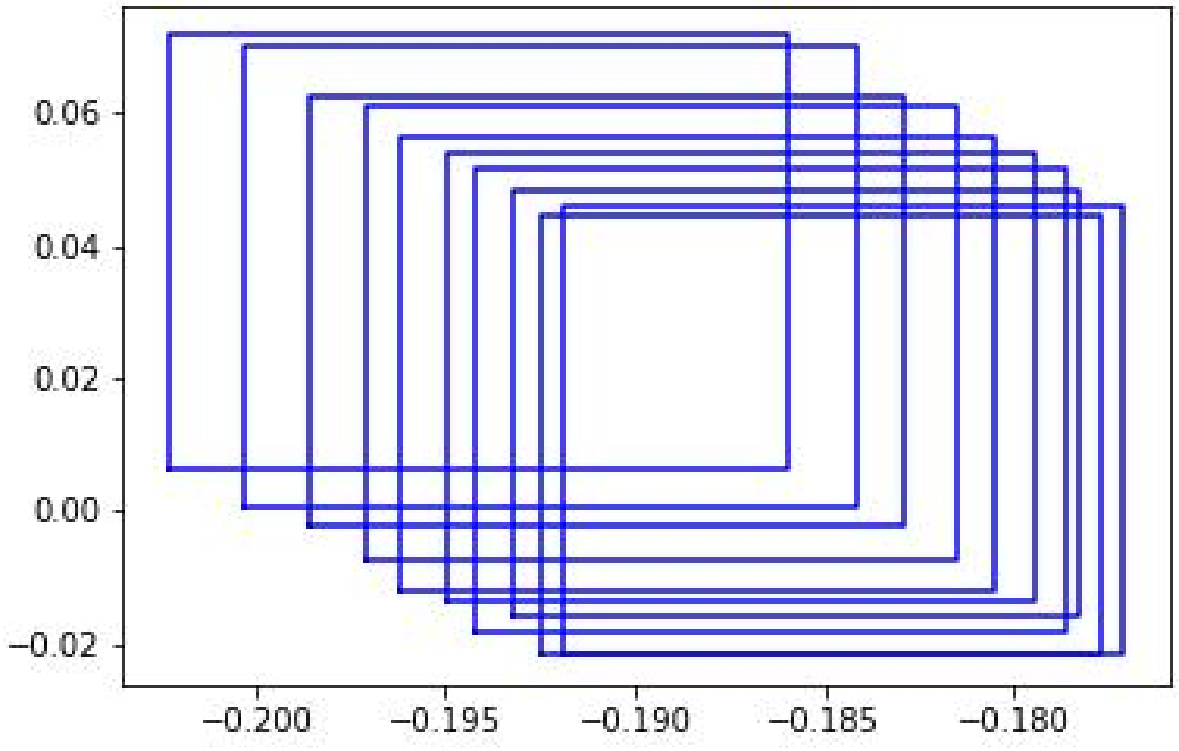
Preuve de stabilité

Stabilité du pendule



Itérations 1 à 80 - $\theta(0) \in \left[\pi, \pi + 10^{-4}\right], \dot{\theta}(0) \in \left[1, 1 + 10^{-4}\right]$ $(\epsilon = 0.2)$

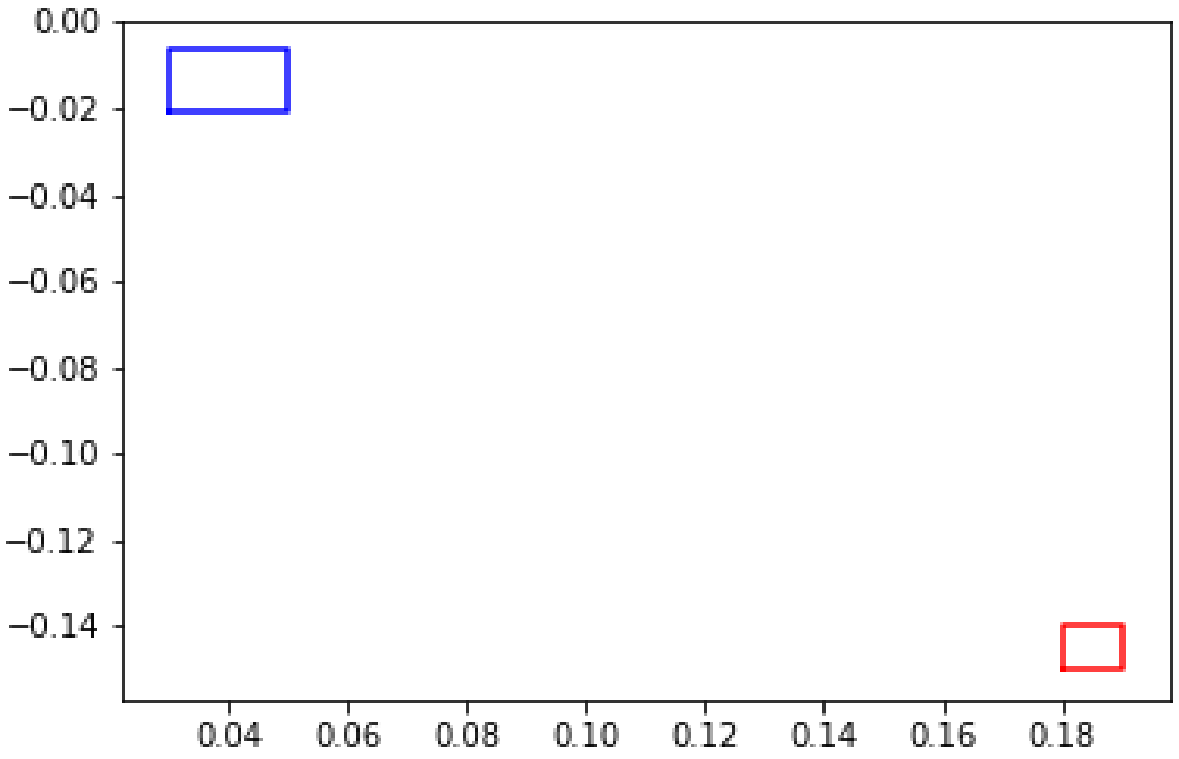
Stabilité du pendule



Itérations 70 à 80 - $\theta(0) \in \left[\pi, \pi + 10^{-4}\right], \dot{\theta}(0) \in \left[1, 1 + 10^{-4}\right]$ $(\epsilon = 0.2)$

Preuve de stabilité

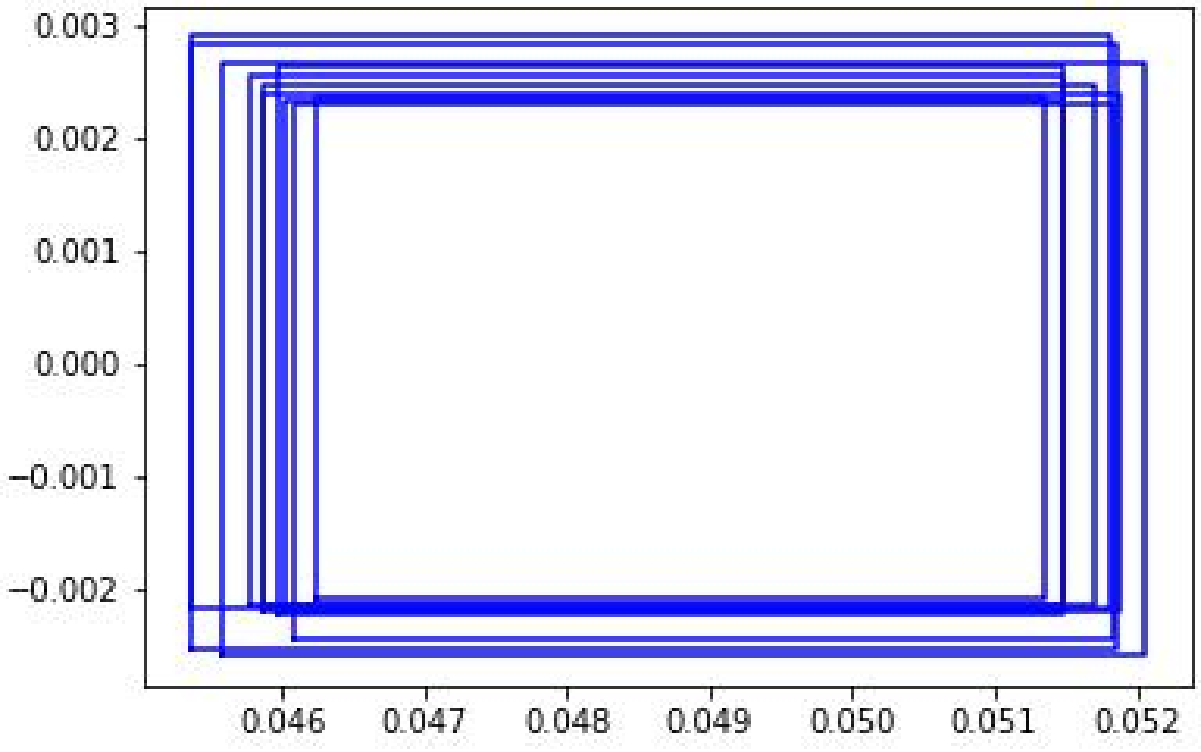
Stabilité du double intégrateur



Itérations 1 à 30 - $x_1(0) \in [0.18, 0.19]\,, x_2(0) \in [-0.15, -0.14]$ ($\epsilon = 0.001$)

Preuve de stabilité

Stabilité du double intégrateur



Itérations 20 à 30 - $x_1(0) \in [0.18, 0.19]\,, x_2(0) \in [-0.15, -0.14]$ ($\epsilon = 0.001$)

Stabilité par analyse de Lyapunov

Système : $x(k+1) = Ax(k) + Bu(k) = Ax(k) + B\Phi(x(k))$

Pour que le système soit stable (converge vers l'état nul lorsque $t \to \infty$), il suffit qu'il existe $P \in \mathbb{R}^{n \times n}$ telle que la fonction V définie par :

$$V : X \to \mathbb{R}$$
$$x \mapsto x^T P x$$

où X est l'ensemble des états atteignables est positif et décroissant.

Stabilité par analyse de Lyapunov

Système : $x(k+1) = Ax(k) + Bu(k) = Ax(k) + B\Phi(x(k))$

Soit $[x] \subset X$, $[\gamma] = [\Phi]([x])$. Stabilité locale de la région $[x]$?

$$\begin{aligned} V(x(k+1)) - V(x(k)) =& x^T(A^TPA - P)x + \Phi(x)^TB^TPAx \\ &+ x^TA^TPB\Phi(x) + \Phi(x)^TB^TPB\Phi(x) \end{aligned}$$

On peut majorer les termes $\Phi(x)$ par les bornes de $[\gamma]$ en prenant en compte le signe des termes des matrices ci-dessus

Stabilité par analyse de Lyapunov

Système : $x(k+1) = Ax(k) + Bu(k) = Ax(k) + B\Phi(x(k))$

En posant $Y^T = \begin{bmatrix} x^T & 1 \end{bmatrix}$, la condition devient :

$$\forall x \in [x], Y \begin{bmatrix} A^T PA - P & A^T PBf(\xi, [\gamma]) \\ f(\xi, [\gamma])^T B^T PA & \sum_{i=1}^{n} \sum_{j=1}^{n} g(\mu_{i,j}, [\gamma]) \end{bmatrix} Y^T < 0$$

où ξ, μ sont respectivement les matrices $x^T A^P B$ et $B^T PB$, et f, g sont des fonctions donnant les bornes appropriées de $[\gamma]$

Conclusion

- Entraînement de contrôleurs garantis sur divers problème : DDPG et MPC Explicit
- Régularisation continue pour améliorer l'entraînement
- Proposition d'un algorithme permettant d'analyser le système en boucle fermé
- Preuves de stabilité et d'atteignabilité des contrôleurs
- Toolbox de garantie de systèmes contrôlés par réseaux de neurones représentés sur TensorFlow (guaranteed-nn-control)

Références


Timothy P. Lillicrap and al.
Continuous control with deep reinforcement learning, 2015.


Weiming Xiang and al.
Reachable set estimation for neural network control systems : A simulation-guided approach, 2020.


P.J. Meyer, A. Devonport, and M. Arcak.
Interval Reachability Analysis : Bounding Trajectories of Uncertain Systems with Boxes for Control and Verification.
2021.


Greg Brockman and al.
Openai gym, 2016.


Thierry Lecomte, Thierry Servat, and Guilhem Pouzancre.
Formal methods in safety-critical railway systems.
08 2007.


Anthony Corso and Mykel J. Kochenderfer.
Interpretable safety validation for autonomous vehicles.
CoRR, abs/2004.06805, 2020.


Qianli Shen and al.
Deep reinforcement learning with smooth policy.
CoRR, abs/2003.09534, 2020.


Alberto Bemporad.
Hybrid toolbox for matlab - user's guide.
2003.


Adnane Saoud.
Stability analysis using lmis for systems with neural network controllers.