Rapport projet robotique :
Apprentissage automatique sur un bras RR

Étudiants : Tom MARINI, John Sebastian ROJAS RODRIGUEZ

1. Présentation du sujet

Pour ce projet de robotique, nous avons choisi de faire de l'apprentissage supervisé sur un bras robotique plan doté de deux liaisons pivot. Le but est d'apprendre au bras robot à se déplacer pour que son effecteur atteigne une position cible. Nous n'avons effectué que des simulations géométriques du bras, c'est-à-dire que les masses des constituants du robot n'entrent pas en jeu dans son déplacement, il ne s'agit d'un robot simulé dynamiquement.
L'objectif est d'effectuer une prédiction par un modèle de neurones multi-couches. À partir de la position finale de l'effecteur souhaitée, le modèle de neurones permet alors de donner la angles des joints du robot. Il est important de noter que la trajectoire du robot n'est pas calculée. En fait, le modèle permet simplement d'apprendre la loi géométrique inverse à partir de d'exemples de configurations du bras robot. Ces configurations ont été générées pour former la base d'apprentissage.
C'est déjà un premier pas qui nous permet de mettre en œuvre de l'apprentissage automatique supervisé de A à Z dans un projet de robotique allant de la génération de données jusqu'aux tests de prédiction du modèle sur certaines trajectoires tout en passant par l'entraînement du modèle.

Les différentes ressources associées à ce projet sont :

- un repository GitHub : https://github.com/jhsrojasro/Foundation-of-Robotics
- des vidéos stockées sur google drive : https://drive.google.com/drive/folders/1AX_UXiuuFurcJPzWwhfjpi9Zclqq2cbM?usp=sharing

2. Méthode

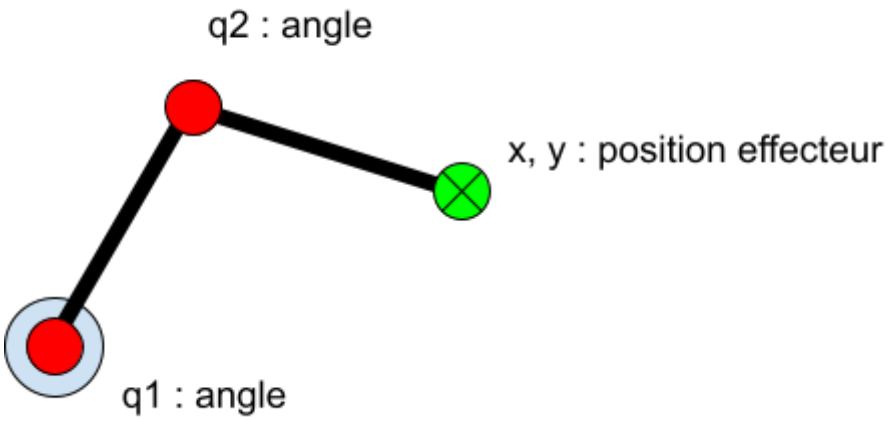


Figure 1 : Schéma du bras robot RR plan

Nous considérons un robot plan à deux bras pivotant comme repéré sur la figure 1. Il est doté d'un effecteur et le premier bras est fixé à la scène. Le bras robot utilisé pour la simulation est le Franka sur Coppelia Sim. Les joints utilisés sont les numéros 2 et 4 dans l'arborescence du robot. Les autres joints sont bloqués.

Tout d'abord, voici les paramètres du bras robot. Le robot dispose de deux paramètres d'entrée, les angles q1 et q2 comme représenté sur la figure 1, et d'un paramètre de sortie qui est la position de l'effecteur dans le plan.

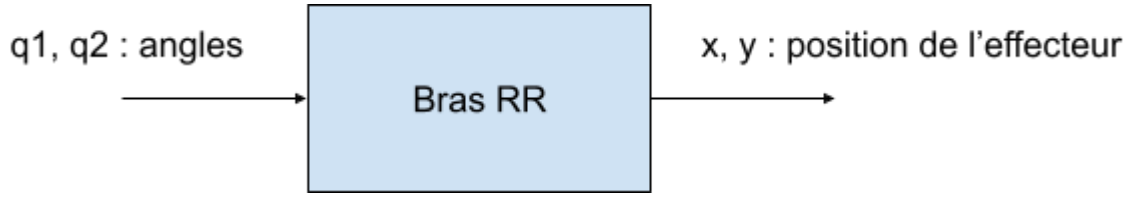


Figure 2 : Schéma des entrées - sorties du robot RR

Le but est de pouvoir donner une consigne de position de l'effecteur permettant d'envoyer les bonnes commandes en angles q1 et q2 au bras RR. Voici donc le schéma de la commande :

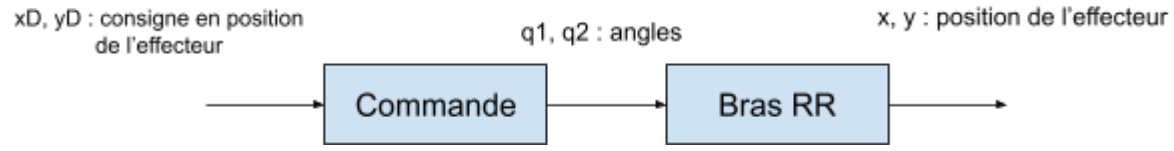


Figure 3 : Commande du bras RR

Dans notre cas, le modèle apprend donc la commande du modèle géométrique inverse donnant les angles à partir de la position de l'effecteur. Ce modèle est d'ailleurs connu est tout à fait implémentable car le calcul analytique est faisable.

Principe de l'apprentissage supervisé

Le principe est d'apprendre la commande du robot à partir de configuration d'exemples fournis au modèle d'apprentissage, autrement dit : il faut généraliser les données d'apprentissage pour que le modèle puisse prédire des configurations qui n'ont pas été présentées lors de l'apprentissage. Pour cela, on utilise un réseau de neurones dont les poids sont à trouver de manière à réaliser les meilleures prédictions sur des nouvelles données.

Base d'apprentissage

La base d'apprentissage du modèle est composée d'associations entrée - sortie. A chaque position de l'effecteur est associée une valeur de l'angle q1 et une valeur de l'angle q2 qui correspondent à une configuration géométrique valide. Voici la démarche utilisée pour construire cette base :

- Génération de deux angles q1 et q2 aléatoires. Il faut beaucoup de données de manière à couvrir tout l'espace des possibles dans les plages de valeurs choisies pour q1 et q2.
- Réglage des angles q1 et q2 des joints 1 et 2 du robot.
- Mesure de la position de la position y de l'effecteur.
- Ajout du couple entrée (q1, q2) - sortie (y)

L'implémentation est réalisée dans le fichier *createDataset.py*

La figure 4 montre finalement l'ensemble des positions de l'effecteur ajoutées à la base de données pour lesquelles les valeurs angulaires sont répertoriées.

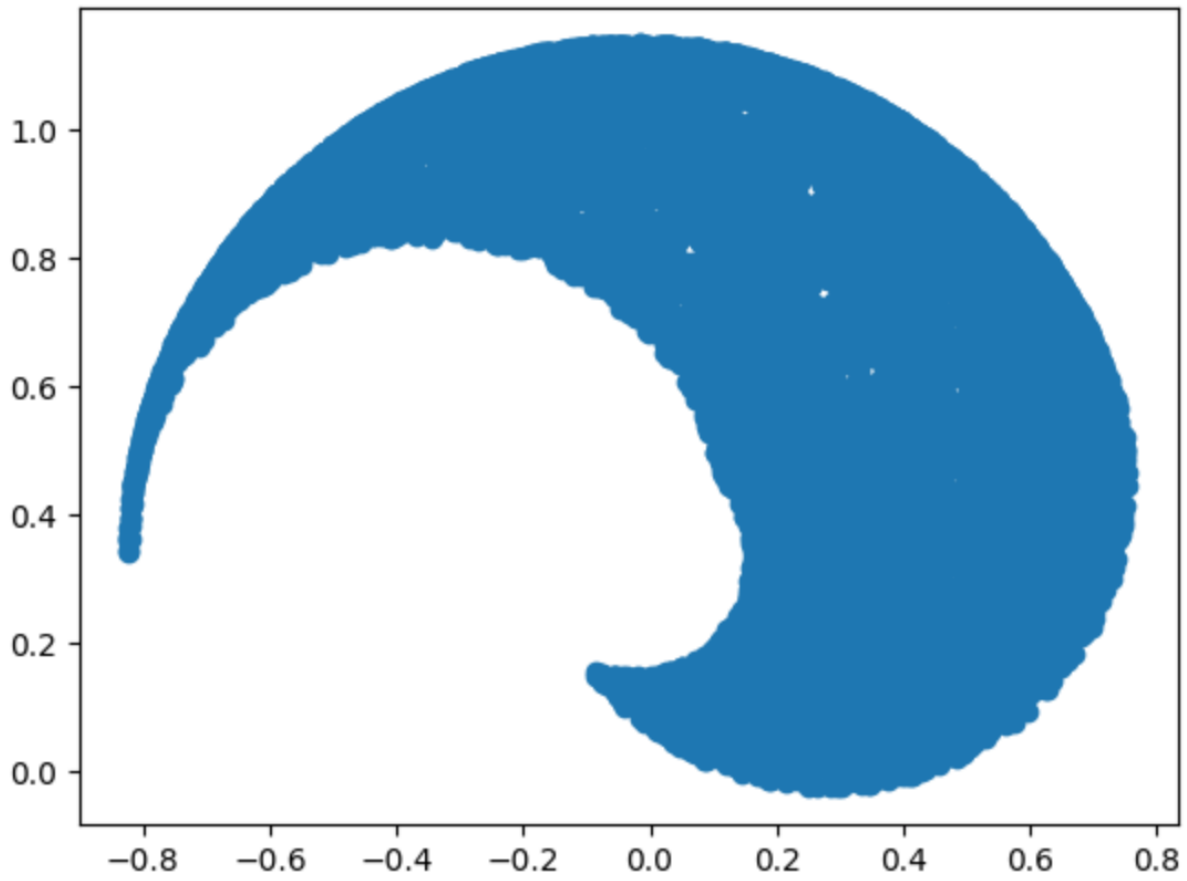


Figure 4 : Points de la base d'apprentissage

La restriction de l'espace accessible est due aux contraintes angulaires ajoutées pour la construction de la base d'apprentissage. La zone accessible est globalement assez remplie et homogène.

3. Résultats

Réseau de neurones

Trois modèles de réseaux de neurones ont été envisagés. Ils ont en commun le nombre de couches et les couches d'entrée - sortie. Les modèles contiennent 5 couches, dont 4 couches sont cachées et 1 couche d'entrée sert à spécifier les données d'entrée du réseau, mais le nombre de neurones par couche varie.

Les paramètres d'apprentissage sont les suivants :

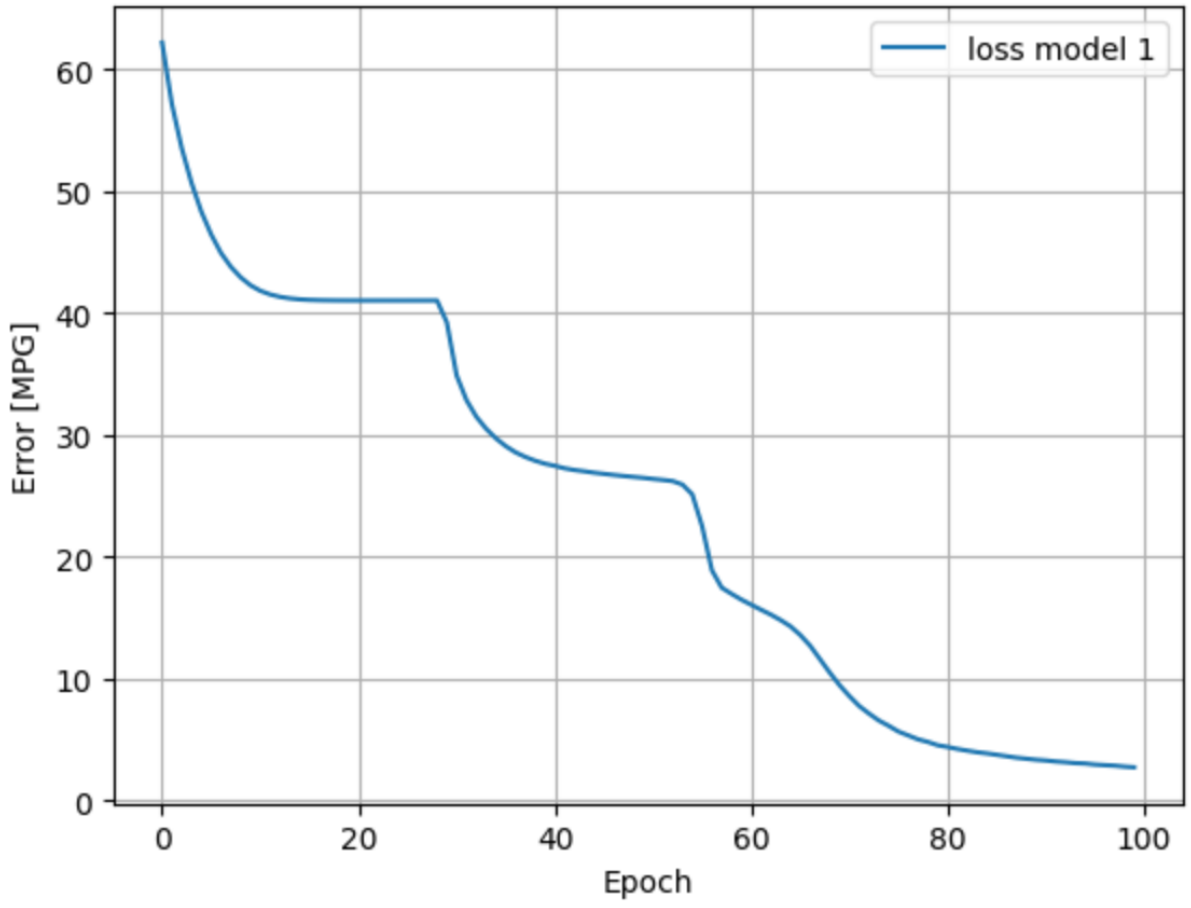
- activation = 'sigmoid' : Fonction d'activation des neurones. La couche de sortie n'est pas activée
- learning_rate = 0.001 : Pas d'apprentissage
- loss = 'mean_absolute_error' : Erreur absolue moyenne

- epochs = 100 : Nombre de sessions d'apprentissage réalisées ou nombre de rétro-propagations.

Modèle 1

Couche	entrée (x,y)	1	2	3	sortie (q1,q2)
nombre de neurones	-	64	64	64	2
activation	non	sigmoïde	sigmoïde	sigmoïde	non

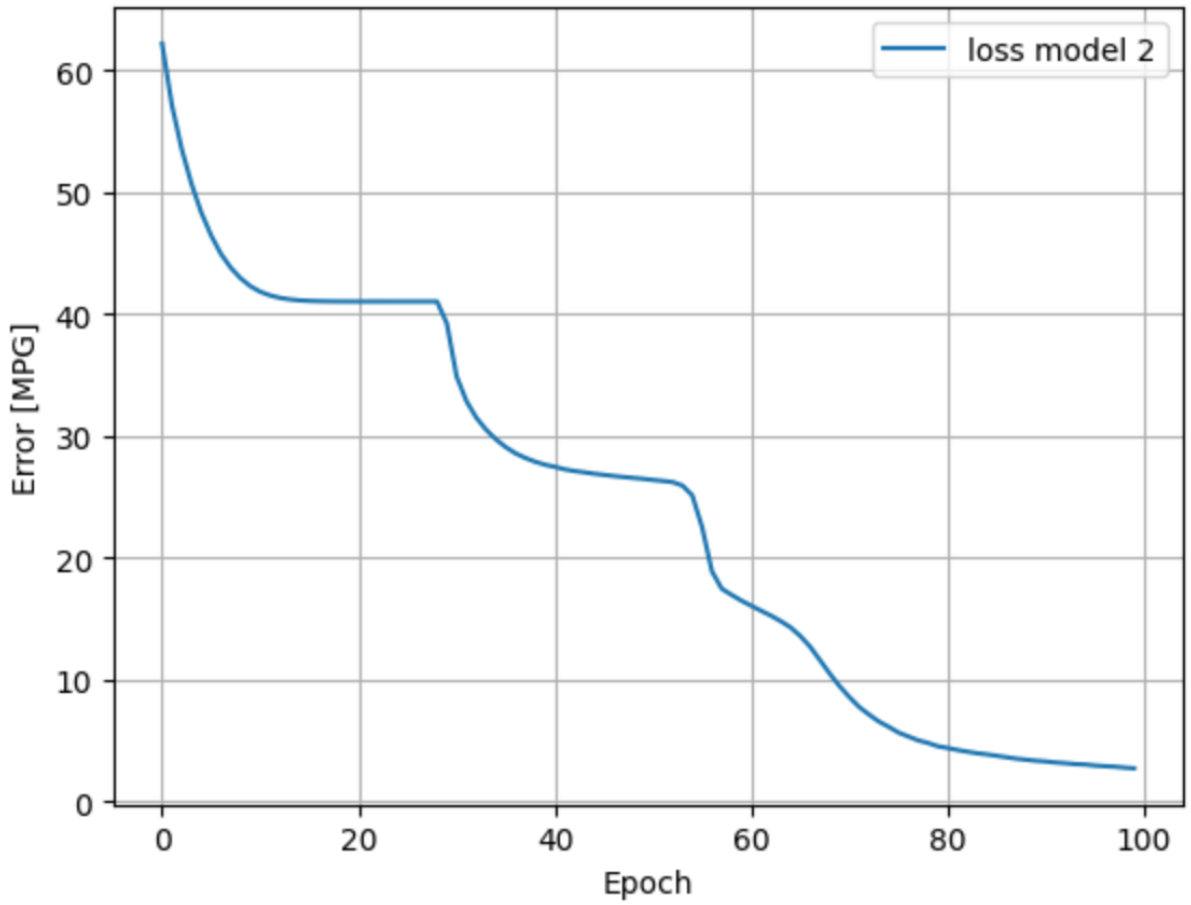
Courbe d'apprentissage



Modèle 2

Couche	entrée (x,y)	1	2	3	sortie (q1,q2)
nombre de neurones	-	256	128	64	2
activation	non	sigmoïde	sigmoïde	sigmoïde	non

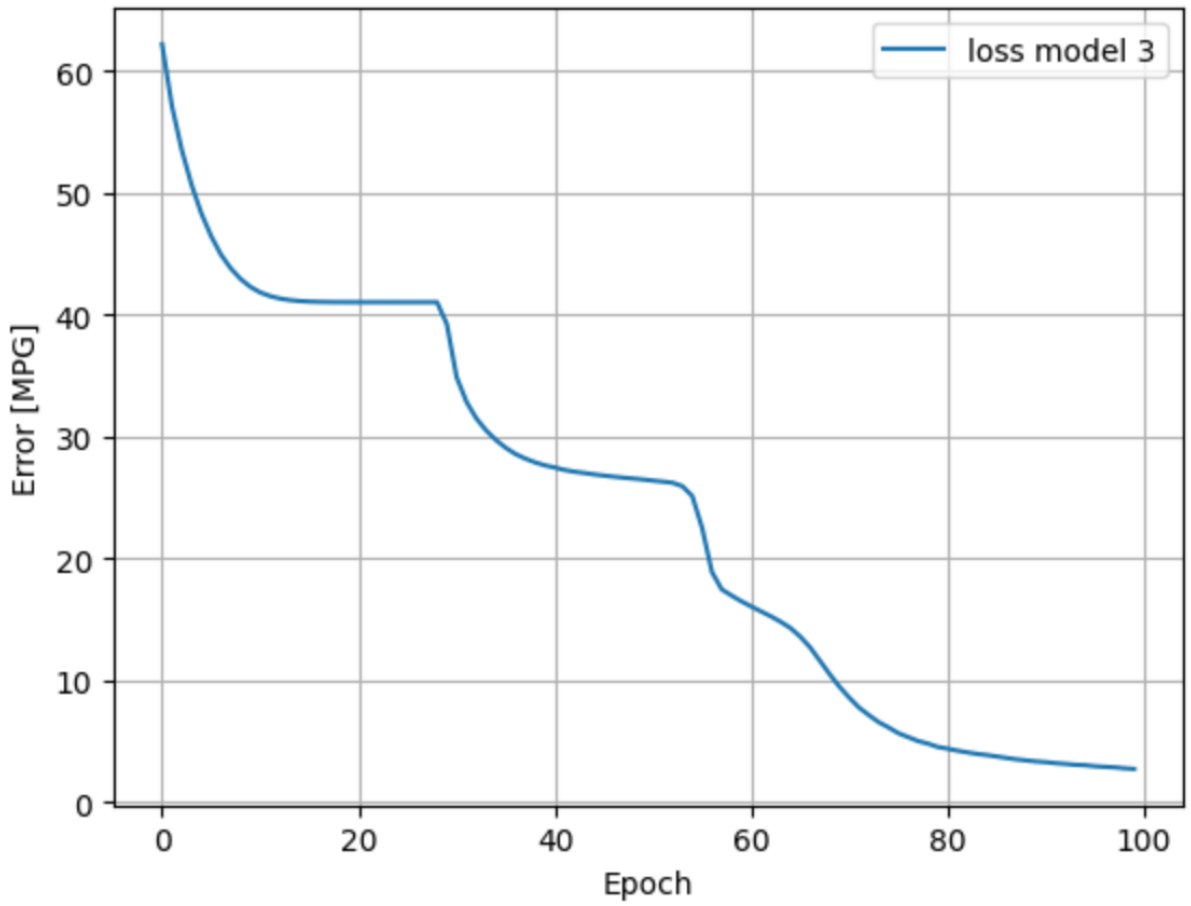
Courbe d'apprentissage



Modèle 3

Couche	entrée (x,y)	1	2	3	sortie (q1,q2)
nombre de neurones	-	256	256	256	2
activation	non	sigmoïde	sigmoïde	sigmoïde	non

Courbe d'apprentissage



Analyse des résultats

Les courbes d'apprentissages sont satisfaisantes puisque l'erreur sur la base de test (30% des données totales) est de moins de 5% : respectivement 2.9%, 2.0% et 2.7% pour les modèles 1, 2 et 3. Les trois modèles sont satisfaisants.
Le nombre d'époques durant lequel l'apprentissage a été réalisé semble optimal puisqu'on ne pourra pas gagner franchement plus sur la courbe d'apprentissage et qu'en même temps c'est le nombre d'époques à partir duquel le modèle commence à fournir des résultats vraiment satisfaisants.
Le temps nécessaire à l'entraînement est très rapide. Les durées de calculs sont respectivement de 28s, 33s et 64s.

Outils

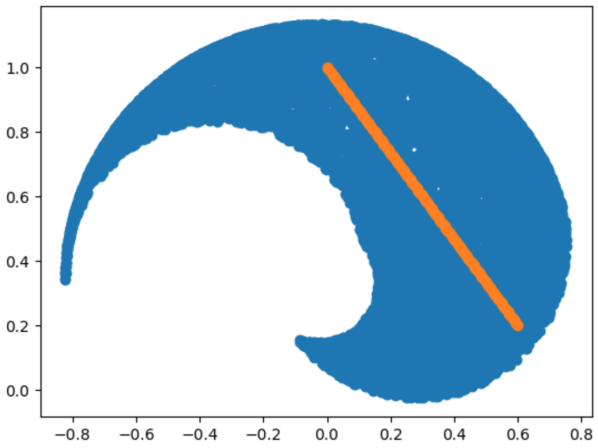
Pour effectuer l'apprentissage, nous avons utilisé **Google colab**. Cela permet d'avoir accès à des ressources de calcul gratuitement et d'avoir un environnement prêt à l'emploi puisque l'environnement est déjà configuré avec les bibliothèques courantes d'apprentissage telles

que *tensorflow* et *keras*. De plus cela permet de collaborer très facilement en temps réel avec d'autres utilisateurs en travaillant sur le même carnet jupyter

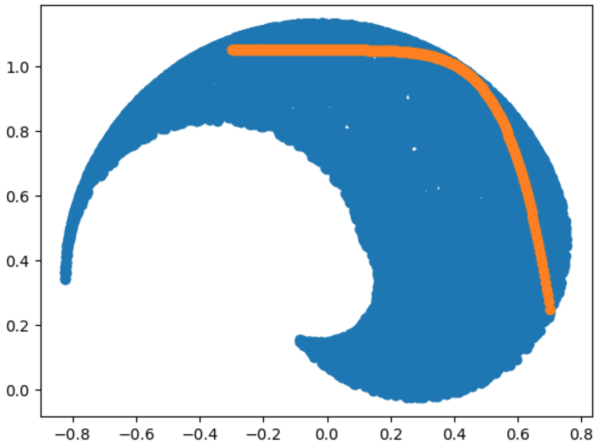
4. Tests et simulations

Maintenant que le modèle a appris la commande géométrique inverse, nous pouvons la tester sur plusieurs destinations de l'effecteur. Une manière assez parlante d'effectuer des tests est de faire suivre une trajectoire à l'effecteur. Le modèle n'a pas appris à suivre une trajectoire mais il est possible d'en construire une et de prédire les valeurs des angles du bras RR pour chaque point de cette trajectoire, ce qui permet effectivement de la suivre.
Nous avons créé trois types de trajectoires :

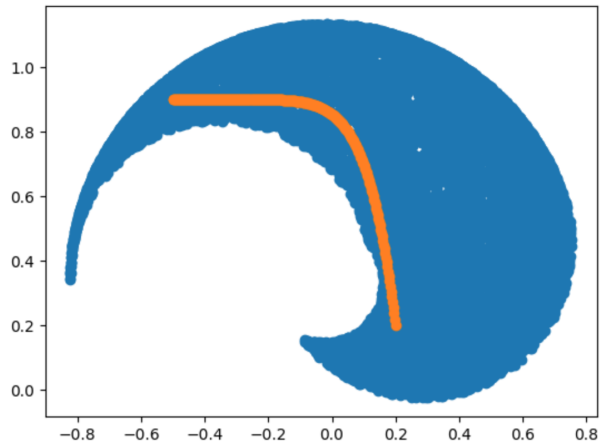
- linéaire



- parabolique



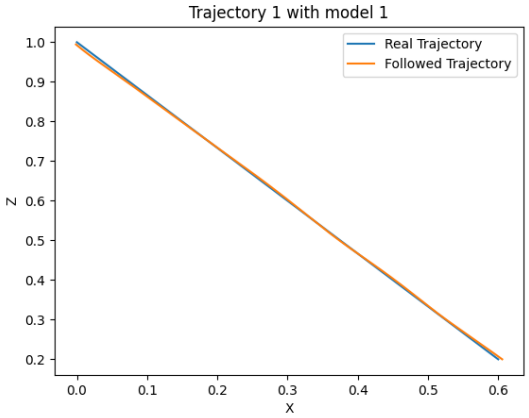
- radicale

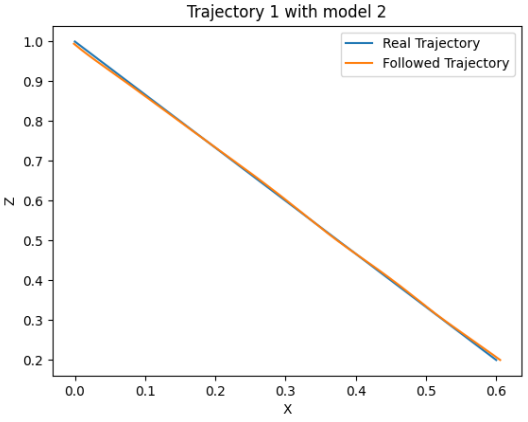


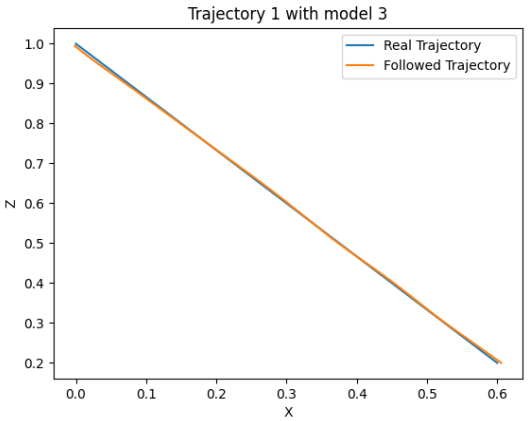
L'idée est de parcourir la zone d'apprentissage (en bleu) pour trouver des données nouvelles entre les données d'apprentissage et voir si le modèle est capable de généraliser la commande.

Voici les résultats du suivi du robot sur CoppeliaSim pour chaque trajectoire et chaque modèle. Les commandes angulaires préenregistrées issues de la prédiction du modèle ont été fournies au robot. Cela correspond au fichier *controler_Franka_MGI.py*

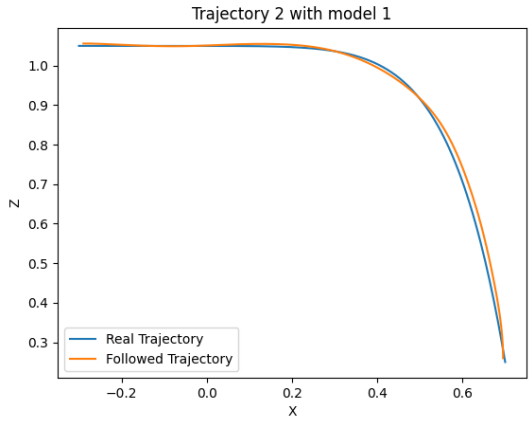
Trajectoire 1 : linéaire

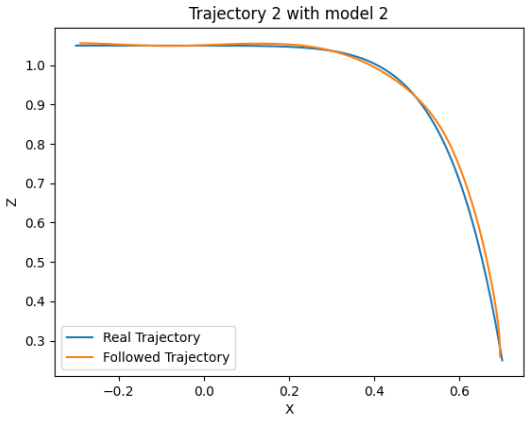


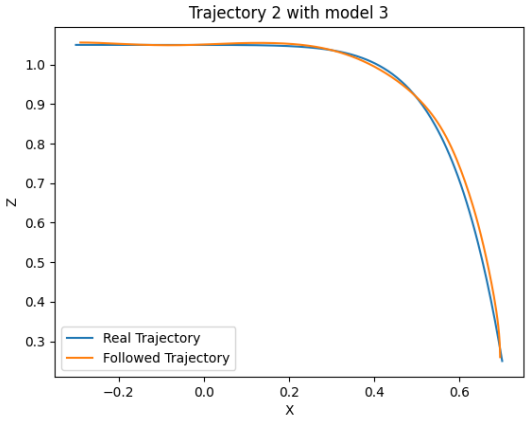




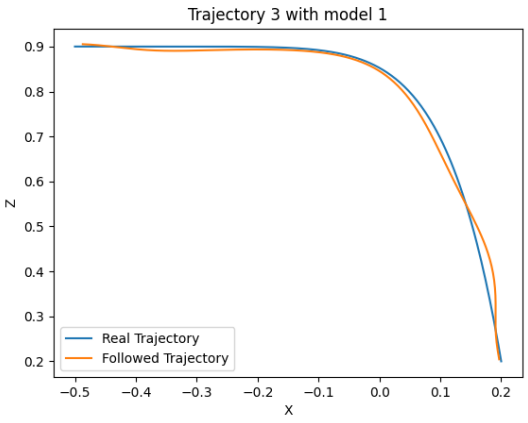
Trajectoire 2 : parabolique

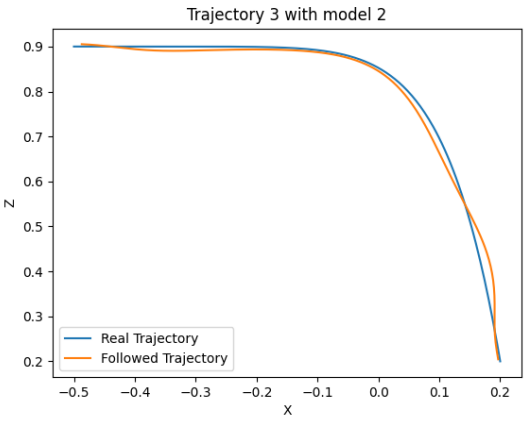


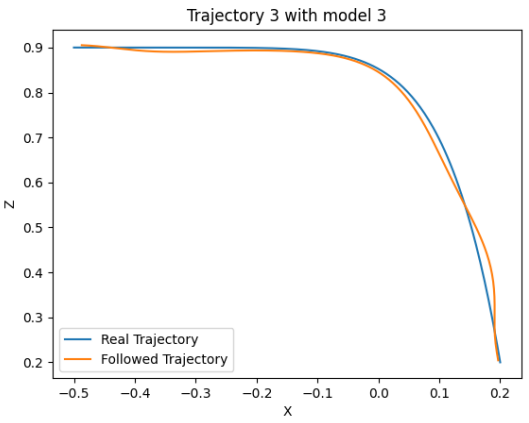




Trajectoire 3 : radicale







Analyse des résultats et limites

Le suivi des trajectoires est assez satisfaisant pour les trois modèles et pour les trois trajectoires. On retrouve les mêmes défauts pour les trois modèles tels que les dépassements similaires. Il n'y a donc pas tant de différence que ça entre les modèles de réseaux de neurones réalisés.
On observe tout de même des résultats intéressants pour les trajectoires. La trajectoire 1 est très bien suivie, cela signifie que les points de la trajectoire 1 sont bien prédit par le modèle, ce qui est corroboré par le fait que la trajectoire 1 a été prise en plein milieu de la zone accessible de la base d'apprentissage.

Cas limites

Pour les trajectoires 2 et 3, on observe des écarts importants. C'est sûrement dû au fait que les trajectoires s'approchent des limites de la zone d'apprentissage. Nous n'avons pas fait de test sur un point en dehors de la zone d'apprentissage, ce qui est dommage.

Des vidéos du robot en train de suivre les trajectoires sont disponibles dans le google drive :
https://drive.google.com/drive/folders/1AX_UXiuuFurcJPzWwhfjpi9Zclqq2cbM?usp=sharing

5. Conclusion

La commande inverse a bien été apprise en par un réseau de neurones multi-couches. Le bras robot est alors en mesure de placer son effecteur à l'endroit souhaité et peut donc suivre une trajectoire définie au préalable. Une avancée sur ce projet serait de réaliser l'apprentissage de la trajectoire comme expliqué en cours. De cette manière, le robot serait pleinement autonome et n'aurait pas besoin qu'on lui fournisse la trajectoire.

Pour un bras RR, il est aussi courant d'avoir simplement un système asservi dans lequel la commande effectue une correction PID (Proportionnelle, Intégrale, Dérivée) par exemple. C'est d'ailleurs ce que CoppeliaSim utilise en mode dynamique et qui peut être exploité par l'utilisateur du logiciel avec la fonction *setJointTargetPosition*.