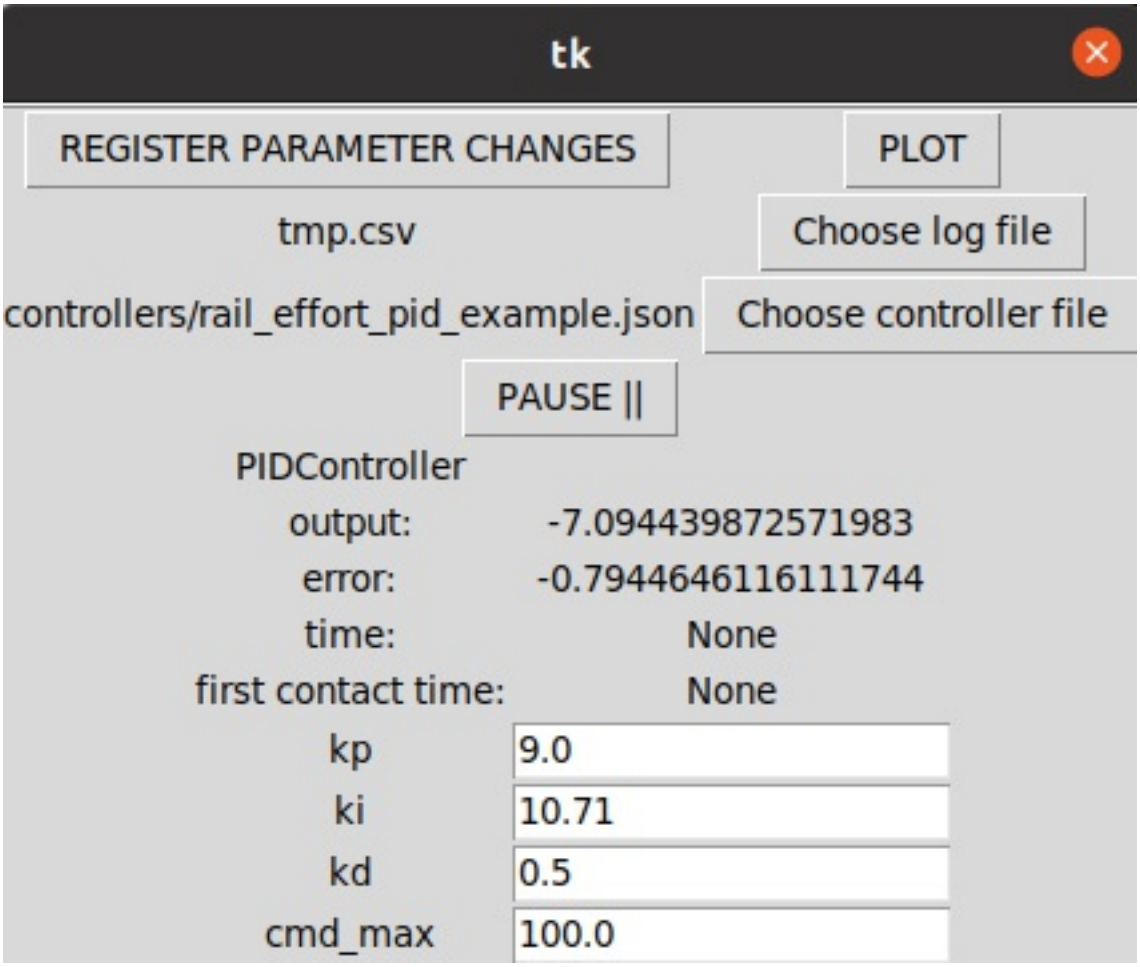
TP RMI - Lois de contrôle

Maxime BOUTHEROUE DESMARAIS
Titien CUBILIER
Thomas FOCHESATO
Florian GARDES
Abdelamnine MEHDAOUI
Luka MORAIZ
15/12/2020

Introduction

Le but de ce TP est de nous faire manipuler différents types des contrôleurs qui permettent de gérer le contrôle d'un robot sur une trajectoire donnée.

Afin de faciliter le réglage des paramètres de chaque contrôleur, nous avons développé une interface graphique simple qui se lance au-dessus de la simulation :
Cette interface permet de changer la valeur des paramètres. Nous pouvons aussi demander à ce qu'elle redémarre automatiquement la simulation avec ces nouvelles valeurs.
Un bouton "PLOT" permet de lancer l'affichage d'un graphique avec plotly à partir du fichier de log en cours d'utilisation.
Nous avons pu utiliser cette interface lors de l'application de la méthode Ziegler-Nichols qui sera développée dans la partie suivante.



Le robot ’rail’

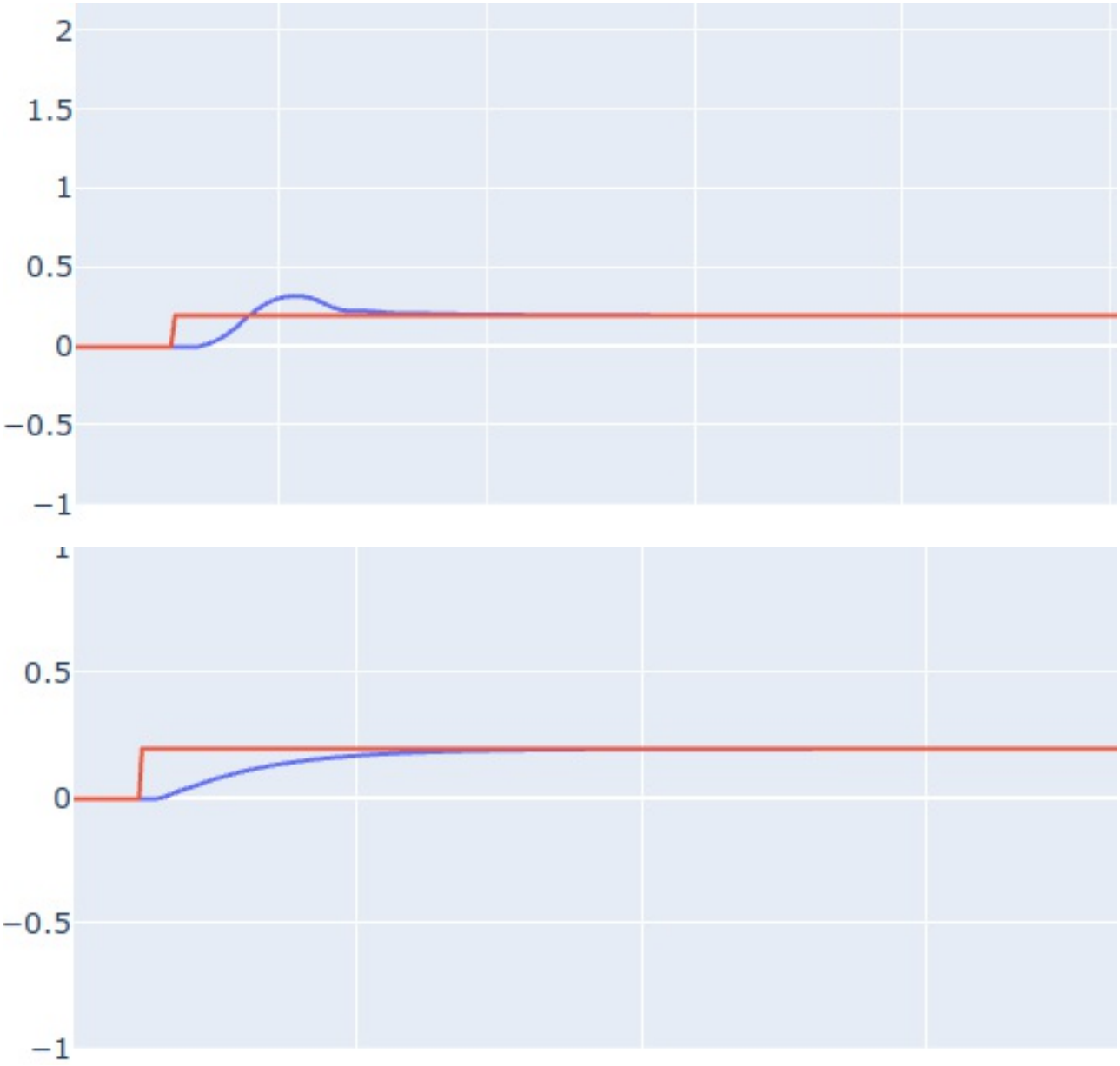
Contrôleur PID

Le contrôleur PID prend en entrée:

- Les valeurs mesurées de la position
- L'accélération
- Les valeurs initiales de position
- L'accélération
- La vélocité
- Le temps

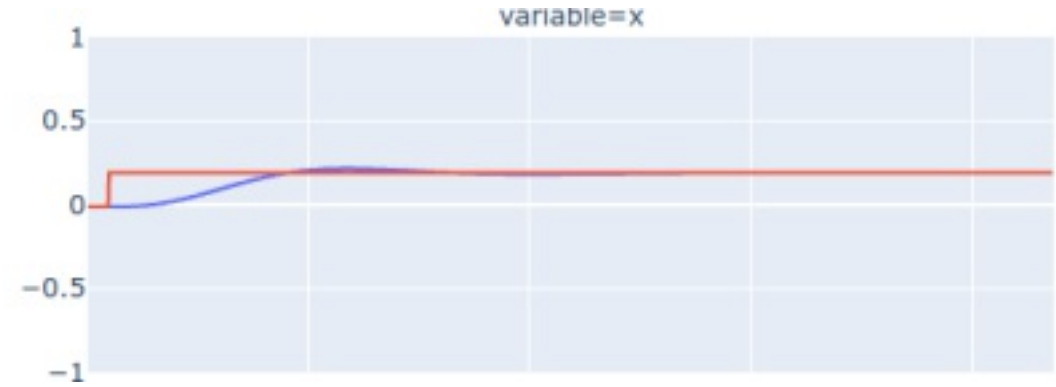
En sortie il retournera une commande avec une erreur corrigée qui sera appliquée sur un robot.

Nous avons observé que les actions proportionnelles et intégrales étaient des forces motrices dans la commande alors que l'action dérivée permet de la ralentir à l'approche de la cible.
Le contrôleur PID est basé sur trois paramètres : k_p, k_i et k_d respectivement utilisés pour les actions proportionnelles, intégrale et dérivée.
La valeur de ces paramètres peut être estimée grâce à la méthode de Ziegler-Nichols.
Pour se faire nous avons tout d'abord commencé par mettre tous ces paramètres à 0 et fait varier la valeur de k_p jusqu'à obtenir une oscillation continue. Cette situation apparait lorsque $k_u = 15$, la période est de 1.78.
En utilisant les formules données en cours, nous avons obtenu une valeur pour chaque paramètre mais le résultat obtenu n'était pas convainquant. Nous avons donc modifié la valeur de k_i pour lui donner une valeur plus petite.
Avec les paramètres $k_p = 9$, $k_i = 10.71$ et $k_d = 0.5$ nous arrivons à avoir une trajectoire qui se stabilise sur la cible après l'avoir une première fois dépassée.
En effectuant des recherches, nous sommes tombés sur ce lien (http://brettbeauregard.com/blog/2017/06/introducing-proportional-on-measurement/) qui présentait une méthode de calcul légèrement différente pour obtenir la valeur du step en fonction de t. L'idée générale est de considérer l'action proportionnelle comme une force résistante qu'il faudrait compenser avec l'intégrale et affiner avec la dérivée.
Cette nouvelle méthode permet de corriger le problème du dépassement de la cible : le résultat obtenu est une trajectoire qui rejoint progressivement la cible pour se stabiliser sans jamais la dépasser.



Pour le contrôle en couple, nous avons appliqué la même méthode que précédemment. Après avoir trouvé une oscillation continue avec $k_u = 2$ et une période d'environ 4.44, nous avons trouvé les valeurs suivantes : k_p = 0.6 * 2 = 1.2$, $k_i = 1.2 * 2 / 4.44 = 0.54$ et $k_d = 3 * 2 * 4.44 / 1.35 \rightarrow 1.31$.

Voici la courbe de position obtenue avec ces paramètres :



La méthode de mesure proportionnelle était également activée. Contrairement au cas précédent, le robot dépasse légèrement la cible avant de se stabiliser. Sans cette méthode la stabilisation est un peu plus lente.

Boucle ouverte

La boucle ouverte du robot "rail" fut assez simple à faire. Le robot se déplaçant sur un rail, la gravité est nulle.
De plus nous avons mis les frottements à zéro pour simplifier le contrôleur.
Il nous reste donc l'inertie à calculer :
L'inertie est égale à la masse que l'on peut trouver sur le URDF du robot multiplié par l'accélération.
Pour l'accélération vu que nous n'avons pas accès au capteur dans une boucle ouverte, elle est égale à l'accélération que l'on a supposément atteinte au step précédent.
Pour finir il suffit de multiplier la somme de l'inertie, la gravité et des frottements par le k_acc.

Feed forward

Concernant le feed forward, nous avons créé un objet ControllerPID et OpenLoopEffortController dans l'init grâce au JSON.
Dans la fonction step on appelle les fonctions step des 2 contrôleurs et on ajoute leurs résultats.

Pendulum

Boucle ouverte

La boucle ouverte du pendulum fonctionne d'une manière homologue à celle du rail sauf que cette fois il faut calculer la gravité.
Pour cela Nous avons utilisé la formule suivante : mass x cos(angle) x taille x 9.81.
Pour l'angle nous avons utilisé la position articulaire de référence que nous avons multiplié par la vitesse de référence et le temps entre 2 steps.

Feed forward

Le feed forward du pendulum fonctionne de la même manière que celui du rail sauf que l'on n'utilise pas un OpenLoopEffortController mais un OpenLoopPendulumEffortController

Conclusion

Le contrôleur OpenLoop ne nous a pas convaincu au vu des résultats obtenus, surtout pour le pendulum.
En effet, ce type de contrôleur n'est pas précis, il ne corrige pas l'erreur car il utilse pas de la valeur des capteurs, cela pourrait peut être utile dans certaines situations (économie de ressources processeur sur des robots bas de gamme).
Le PID grâce à ses trois actions propose plus de contrôle sur la trajectoire et permet de corriger une erreur.
Le FeedForward est une bonne alternative car il prend les avantages du PID et du OpenLoop pour obtenir un résultat assez précis.