Ecole Polytechnique fédérale de Lausanne

Babyfoot

Projet de semestre

Simulation: Passes entre les joueurs

Auteur :
Maxence PERRET

Professeur :
Christophe SALZMANN

JUIN 2020

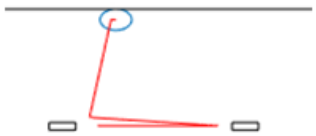
Table des matières

Résumé de ce qui a été fait

Mon but était de créer une simulation du babyfoot automatique, pour faire des passes entre des joueurs de la même ligne. Tout ceci à été fait étape par étape en ajoutant chaque fois des fonctionnalités voici les principaux points trouvés.

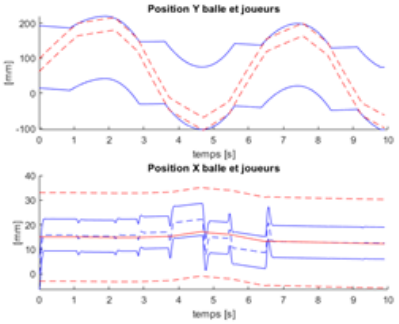
La balle

L'interaction de la balle et du joueur peut être assimilé à un ressort entre les deux corps. Cette balle roule et ne glisse pas, ce qui permet de trouver la dynamique. De plus un décalage entre le centre du joueur et celui de la balle va appliquer une force avec un certain angle, ce qui va faire dévier la balle. Le tout à été transformé en équations d'états et mis dans Simulink. Les états sont la position et la vitesse de la balle dans les deux directions ainsi que ceux des moteurs.



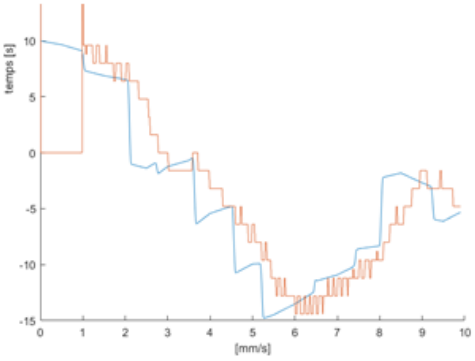
Les moteurs

Pour les moteurs, les paramètres déjà identifiés ont été utilisé et mis dans les équations d'états.



Le contrôle

Pour le contrôle de l'angle (et donc du décalage) j'utilise deux contrôleurs proportionnels, un sur la position et l'autre sur la vitesse en x. Plusieurs stratégies sont possible pour le contrôle du moteur linéaire, mais la meilleure option est d'amortir la balle avec une rampe de vitesse. Les deux mis ensemble arrivent bel et bien à se faire des passes.



Les capteurs

La caméra est simulée grâce à une discrétisation de la position de la taille des pixels. La vitesse est calculée en passant d'abord la position dans une moyenne mobile, puis en prenant la différence finie. Ceci permet de lisser le signal et d'éviter des pics de vitesse. Une fonction permet d'ajouter du bruit sur la trajectoire de la balle.

1 Introduction

Je suis heureux d'avoir pu travailler sur ce babyfoot automatique pour mon projet de semestre. Ce dernier est stimulant, il permet de mettre en pratique ses connaissances sur un jeu que tout le monde connaît. Il est multidisciplinaire, et diverses améliorations sur plusieurs aspects peuvent être apportées. Les projets sont facilement orientables vers ce qui nous intéresse le plus, software, hardware ou les deux.

Le Covid-19 nous a obligé à travailler à la maison et à orienter différemment le projet, pour qu'il soit entièrement faisable depuis chez nous. Donc après le premier mois, où mon but était de comprendre le code et ce qui avait déjà été fait sur le babyfoot, et les deux semaines suivantes, qui étaient plongées dans un flou sur l'avenir des projets, je me suis lancé dans la simulation.

Le but est de simuler et contrôler des passes entre les joueurs de la même ligne sur le babyfoot. Il s'agit l'étape suivante des projets précédents, qui ont réussi à bloquer la balle en l'amortissant. À cet effet, j'ai utilisé MATLAB et Simulink, qui permettent assez facilement de modéliser des systèmes physiques avec une programmation plus graphique et des fonctions adaptées. Cette simulation tentera de se rapprocher au mieux de la réalité, bien que certains paramètres ne peuvent être qu'estimés. Les passes et chocs n'ont été implémentés qu'entre deux attaquants.

Comme mon projet ne se déroule pas sur le babyfoot à proprement parlé, je vous laisse vous référer au dernier rapport en date, celui de Yann Morize en 2019 [3] pour connaître l'avancement de ce projet de babyfoot automatique. Mon rapport a pour but de présenter le travail effectué durant ce semestre, tout ce qui a été mis au point et intégré dans la simulation ainsi qu'une petite explication de la façon dont je suis arrivé à ce résultat et où cela se trouve dans mon programme.

Avant de commencer, j'aimerais particulièrement remercier Prof. Ch. Salzmann, qui m'a fait découvrir ce babyfoot et qui a suivi les avancées tout au long du semestre. J'aimerais aussi remercier et Dr. Philippe Müllhaupt pour son aide sur la modélisation du contact entre le joueur et la balle.

2 Guide d'utilisation de la simulation

Commençons par un petit guide de comment utiliser cette simulation. Le but est que l'utilisateur puisse facilement la lancer sans devoir entrer dans le programme. Le code est commenté du mieux possible pour que cela reste simple à comprendre. Ce rapport est un complément qui permet d'illustrer les choix qui ont été fais et de les expliquer. Quelques bouts de code se trouvent aussi dans ici.

Le programme offre un certain nombre de choix à l'utilisateur, facilement changeable pour tester les différents modes de fonctionnement (figure 2.1). Il suffit d'activer les booléens voulus. Puis vient les constantes et les paramètres dont la simulation à besoin. Ceux ci essayent d'être au plus proche de la réalité, mais certains ne restent que des estimations, voir des coefficients arbitraires. À la suite de ça, il y a les paramètres pour les contrôleurs. Ils ont été ajustés, mais peuvent facilement être changés, si on veut avoir une simulation différente. Vient enfin la section simulation qui se fait sur Simulink et qui est la majeure partie du programme.

```
%% Choix de l'utilisateur

video_bool = true; % crée et sauvegarde une vidéo de la simulation
control_bool = true; % active le contrôle sur l'angle du joueur
sensor_bool = false; % active les défauts des capteurs
bruit_bool = false; % applique du bruit aléatoire sur la trajectoire de la balle
strategie_type = 1; % stratégie pour les passes :
                    % {1:avec amortissement, 2: basique, défaut: séquence à définir}
video_name = 'fichiers/test.mp4'; % nom de la vidéo à enregistrer
simulation_time = 20; % temps de la simulation [s]
sampling_time = 1/10; % [s]

% conditions initiales
% x1 = x balle, x2 = y balle, x3 = x joueur, x4 = y joueur
% x5 = vx balle, x6 = vy balle, x7 = vx joueur, x9 = vy balle
x0 = [0,0,0,-80,0,0,0,0]';
```

FIGURE 2.1 – Début du code matlab, illustrant les choix que l'utilisateur peut faire

Le système général est défini à la figure 2.2. Un contrôleur calcul les références des moteurs suivant la stratégie appliquée et les informations sur les états qu'il reçoit. Les moteurs vont essayer de suivre cette référence et vont actionner le joueur. Ce dernier va interagir avec la balle pour la faire bouger. Le sous système balle contient les équations d'états avec deux cas possibles : quand la balle est en contact avec le joueur, ou quand elle roule librement. Les états sont ensuite mesurés par les capteurs, qui vont transmettre les valeurs aux contrôleur. À noter que tous les "enable crossing" des blocs Simulink, qui permettent d'intensifier les calculs autour des points d'intérêts, ont été enlevés pour ne pas trop ralentir la simulation.

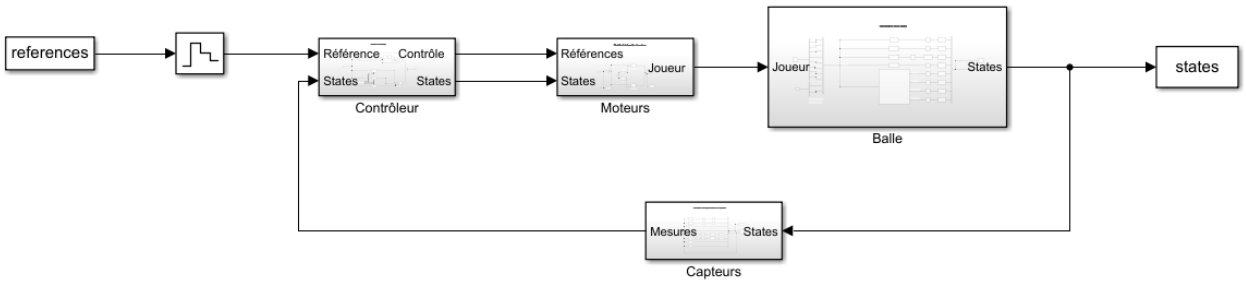


FIGURE 2.2 – Système général du simulink

3 Simulation de la balle

3.1 Théorie

3.1.1 Cinématique

Ici je vais étudier la balle, comment elle se déplace, quels sont les degrés de liberté et les hypothèses à poser sans me préoccuper des équations. Ceci me permettra d'avoir une idée plus générale du problème.

Tout corps simple (en une seule partie rigide) peut avoir au maximum 6 degrés de libertés. Un degré de liberté est mouvement basique et indépendant qui permet de décrire la trajectoire d'un objet. Il peut être représenté par une transformation géométrique. Comme dit précédemment, en 3 dimensions, on en dénombre 6 : une translation et une rotation pour chaque axe.

Revenons à la balle. Sur un baby foot elle peut se déplacer dans le plan du terrain, en X et Y mais pas en hauteur en Z (dans notre cas, en se faisant des passes). Par contre elle est libre de tourner autour de ces 3 axes. On a donc 5 possibilités de mouvements. Il faut faire quelques hypothèses pour continuer. La première est que notre balle est une sphère parfaite, semi-rigide et que le terrain est plat. Elle va garder sa forme tout au long de la trajectoire et n'aura qu'un seul point de contact avec le sol. La seconde est que notre balle roule et ne glisse pas. La rotation autour de l'axe X va créer un mouvement de translation suivant Y, et de la même manière la rotation autour de Y créera une translation en X, suivant les relations $\Delta Y = R * \Delta\theta_x$ et $\Delta X = R * \Delta\theta_y$ où R est le rayon de la balle. La dernière hypothèse est que la rotation en Z (ou yaw avec les angles d'Euler) a un effet minime et n'influence pas le mouvement. Avec tout ceci, il ne nous reste plus que 2 mouvements indépendants pour décrire la trajectoire de la balle, et donc 2 degrés de libertés (figure 3.1).

Maintenant intéressons nous au joueur. C'est lui qui va faire bouger la balle et qui on l'espère, pourra la contrôler. Il a une contrainte principale, il est relié à la barre du baby foot et aura les mêmes degrés de libertés. Cette dernière ne peut bouger que de deux manières : la translation et rotation en Y (d'après la définition des axes du baby foot). Chaque degré étant actionné par un moteur. Faisons aussi une hypothèse sur le joueur : celui-ci est un corps semi-rigide et à une surface plate en contact avec la balle. Ceci nous permet de dire que lorsqu'il est en contact avec la balle, il ne se déforme que légèrement et la touche qu'en un seul point. De plus quand il subit une rotation, le déplacement en X dans le plan de la balle est $\Delta X = L_j * sin(\Delta\theta_y)$ où L_j est la distance entre la barre et le point de contact avec la balle.

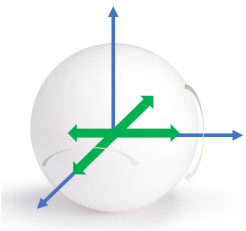


FIGURE 3.1 – Je considère la balle avec 2 degrés de libertés utiles indépendants (x, y) ou (R_x, R_y)

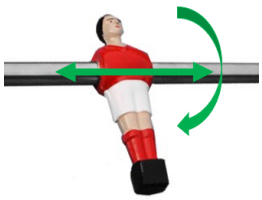


FIGURE 3.2 – Le joueur en a aussi 2 (y, R_y)

3.1.2 Dynamique

Regardons maintenant les forces et tentons de dériver les équations en découlant. La balle est une sphère et on peut donc facilement se placer dans le plan où la force agit, pour passer à un problème en 2 dimensions comme sur la figure 3.3. Grâce à la deuxième loi de Newton nous avons les équations suivantes :

$$\Sigma F_u = F - F_{frottement} = m \cdot \ddot{u} = 0 \tag{3.1}$$

$$\Sigma M_o = R \cdot F_{frottement} = J \cdot \ddot{\theta} \tag{3.2}$$

$$\ddot{u} = R \cdot \ddot{\theta} \tag{3.3}$$

$$\ddot{u} = \frac{R^2}{J} F \tag{3.4}$$

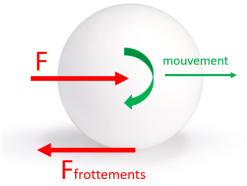


FIGURE 3.3 – Mouvement de roulement

Où u est la direction du mouvement et est parallèle à la force, R est le rayon, J l'inertie et M_o le moment. La balle ne glisse pas et c'est pour ceci que la somme des forces est égale à 0 (équation 3.1). La force externe est appliquée sur le centre de gravité et ne provoque donc aucun moment (3.2). Le déplacement linéaire et ses dérivés sont directement relié à l'angle et ses dérivés (3.3). Pour finir nous avons une équation reliant la force que l'on vient appliquer à l'accélération linéaire (3.4).

Nous savons maintenant quelle dynamique a la balle en 2 dimensions. Cette équation peut être appliquée dans la 3ème dimension si notre force agit au centre de la balle. Mais qu'en est-il si le joueur est décalé par rapport à la balle ? En observant une balle mise en mouvement par un objet qui est décalé, je n'arrivais pas à comprendre de manière intuitive quel était l'angle et quel repère prendre. Après quelques recherches, j'ai trouvé une étude concernant le billard [2] (figure 3.4), où la situation s'apparente fortement à celle que l'on a. Le vecteur T représente la direction de la force recherchée. Il est la somme du vecteur N qui relie le point d'impact et le centre et du vecteur F qui est la friction entre la queue de billard et la balle. Cette force de friction est dur à estimer et sera approximée par un coefficient réducteur de l'angle. Cet angle vaut donc

$$\alpha = arcsin(\frac{d}{R}) \cdot \eta_{angle} \tag{3.5}$$

Avec d la décalage entre le point d'impact et le centre de la balle et $\eta_{angle} \leq 1$ le coefficient qui prend en compte cette composante de friction

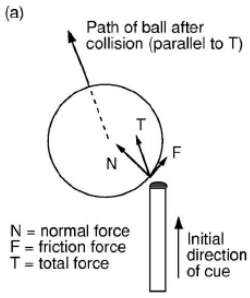


FIGURE 3.4 – Angle de la force appliquée dans le cas du billard

Deux possibilités s'offrent à nous à présent afin de décrire le système en 3 dimensions. Soit on calcule l'accélération, la vitesse et la position suivant la direction de la force, puis on projette ces vecteurs sur nos axes X et Y grâce à cet angle pour avoir la position. Ceci est fait en intégrant l'équation 3.4 pour trouver u

$$x = u \cdot sin(\alpha) \quad \text{et} \quad y = u \cdot cos(\alpha) \tag{3.6}$$

ou alors d'abord décomposer la force en deux composantes X et Y puis intégrer depuis là

$$F_x = F \cdot sin(\alpha) \quad \text{et} \quad F_y = F \cdot cos(\alpha) \tag{3.7}$$

$$\ddot{x} = F_x \frac{R^2}{J} \quad \text{et} \quad \ddot{y} = F_y \frac{R^2}{J} F_y \tag{3.8}$$

On gardera la seconde manière de faire par la suite, car elle sera plus arrangeante à mettre sous forme d'équations d'états et posera moins de problèmes avec les chocs. En effet avec la première façon, nous devrions mettre l'angle sous forme d'état, mais il n'a pas d'équation reliant sa dérivé.

Passons à la dernière partie, à savoir calculer cette force, qui jusqu'à présent a été laissée en inconnue. Tout se joue dans l'interaction entre la balle et le joueur. On a supposé que les deux étaient semi-rigides et peuvent légèrement se déformer. Cette hypothèse nous permet de les modéliser comme sur la figure 3.5 avec un ressort rigide dont la force va dépendre de la déformation.

Ces deux ressorts mis en série peuvent être assemblé en un seul ressort équivalent. La force vient du moteur et fournit le travail externe. Ce système est mis simplement en équation en faisant la somme des forces sur chaque corps.

$$M_{balle} \cdot \ddot{u}_{balle} = k_{eq}(u_{joueur} - u_{balle}) \tag{3.9}$$

$$M_{joueur} \cdot \ddot{u}_{joueur} = -k_{eq}(u_{joueur} - u_{balle}) + F_{moteur} \tag{3.10}$$

Où u est le déplacement dans le direction de la force et M_{joueur} est la masse équivalente au joueur avec la barre et l'inertie rapportée du moteur. Cette masse équivalente sera très grande par rapport à la balle et aura par conséquent plus d'effet sur la balle que la balle en a sur elle. La force et la dynamique du moteur sera calculée plus loin dans la section appropriée.

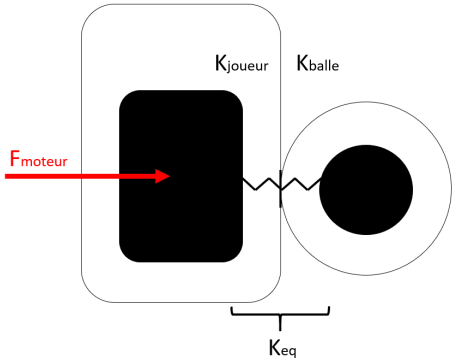


FIGURE 3.5

Si la balle n'est pas en contact avec le joueur ($u_{joueur} - u_{balle} > 0$), cette force de ressort n'est plus présente et la balle est alors simplement soumise aux frottements. (Ces frottement peuvent aussi être ajouté dans l'équation 3.9).

$$M_{balle} \cdot \ddot{u}_{balle} = -\eta_{air} \cdot \dot{u}_{balle} - F_{frott} \tag{3.11}$$

$$M_{joueur} \cdot \ddot{u}_{joueur} = F_{moteur} \tag{3.12}$$

Notre balle ne glisse pas comme les équations 3.9 et 3.10 le sous entendent, mais roule. Il suffit alors de remplacer la force dans l'équation 3.4 par la force de ressort récemment trouvée.

3.2 Implémentation

Pour mettre ceci dans Simulink, il faut transformer les équations précédemment trouvées en modèle d'espace d'états (state space model en anglais). Les états retenus sont ceux de la balle et ceux du joueurs car les deux s'influencent mutuellement. Il est composé de deux sous-systèmes qui vont être utilisés alternativement si la balle est en contact avec le joueur. Voici la définition des états avec leur équation associée. À noter que les définitions ne suivent pas la convention, état suivi de sa dérivé, pour des raisons graphiques dans Simulink.

$$\begin{cases} x_1 = x_{balle} \\ x_2 = y_{balle} \\ x_3 = x_{joueur} \\ x_{30} = \theta_{joueur} \\ x_4 = y_{joueur} \\ x_5 = \dot{x}_{balle} \\ x_6 = \dot{y}_{balle} \\ x_7 = \dot{\theta}_{joueur} \\ x_8 = \dot{y}_{joueur} \\ u_1 = F_{moteurAngulaire} \\ u_2 = F_{moteurLineaire} \end{cases} \tag{3.13}$$

$$\begin{cases} \dot{x}_1 = x_5 \\ \dot{x}_2 = x_6 \\ x_3 = L_{joueur} sin(x_{30}) \\ \dot{x}_{30} = x_7 \\ \dot{x}_4 = x_8 \\ \dot{x}_5 = \frac{R^2}{J} F sin(\alpha) - \eta_{air} x_5 - F_{frott} \\ \dot{x}_6 = \frac{R^2}{J} F cos(\alpha) - \eta_{air} x_6 - F_{frott} \\ \dot{x}_7 = \frac{-1}{M_{joueur}} F sin(\alpha) + u_1 \\ \dot{x}_8 = \frac{-1}{M_{joueur}} F cos(\alpha) + u_2 \end{cases} \tag{3.14}$$

avec

$$F = \begin{cases} k(x_{joueur} - x_{balle}) & \text{si joueur touche la balle} \\ 0 & \text{sinon} \end{cases}$$

$$\alpha = arcsin(\frac{x_{balle} - x_{joueur}}{R}) \cdot \eta_{angle}$$

x_{30} est une étape intermédiaire. Les équations du moteurs sont calculés avec l'accélération et la vitesse angulaire. L'état x_3, qui est la position X du joueur est directement utilisé, mais n'est une transformation géométrique de x_{30}. Il est plus simple de travailler en x_{joueur} pour le contrôleur.

Mis dans Simulink cela donne ceci

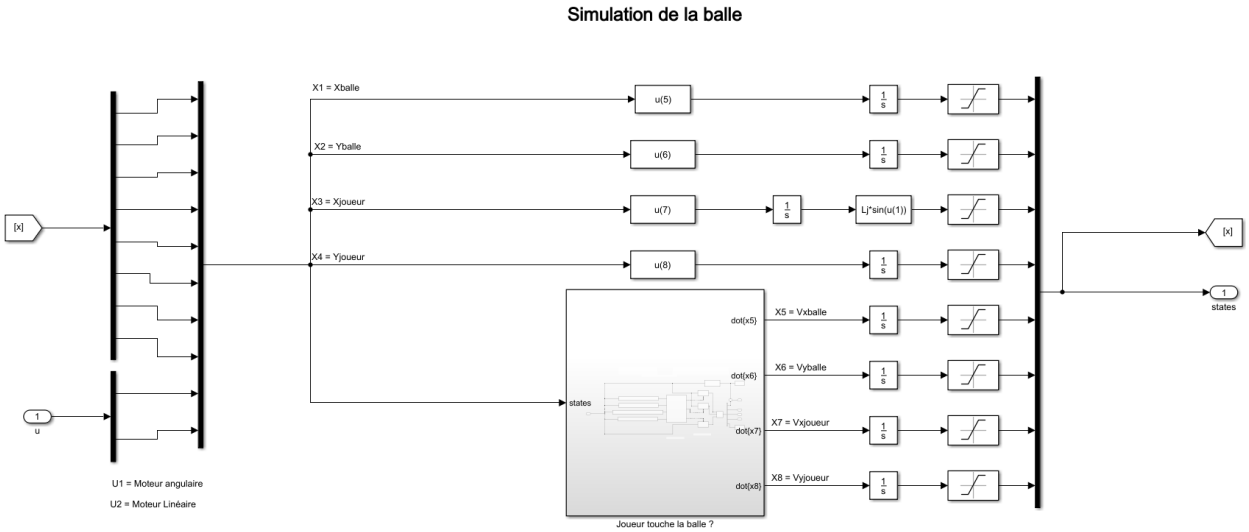


FIGURE 3.6 – implémentation du state space dans simulink

Le bloc "Joueur touche la balle" vérifie s'il y a contact et choisit le bon système à appliquer. Maintenant que l'on a réussi à modéliser la balle, on peut s'amuser et regarder si les résultats reflètent la réalité.

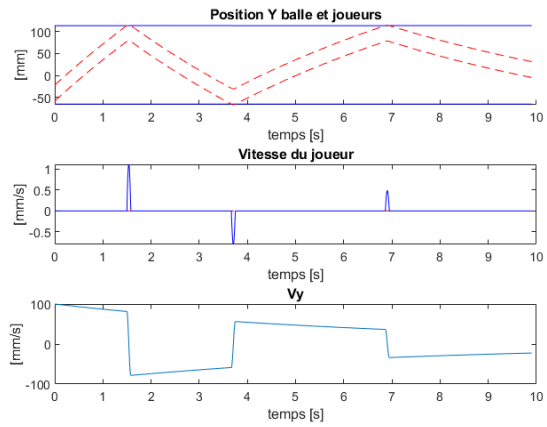


FIGURE 3.7 – Chocs de la balle avec les joueurs statiques et une vitesse initiale de 100mm/s

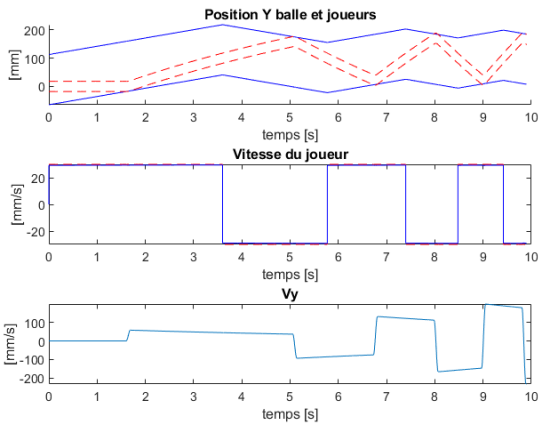


FIGURE 3.8 – Chocs de la balle avec les joueurs qui vont dans le sens opposé

Dans la simulation de la figure 3.7, la balle a été lancée avec une vitesse initiale contre les joueurs. On peut voir qu'elle rebondit bien entre les deux et qu'elle perd de la vitesse dû aux frottements. Le joueur est aussi impacté par ce choc, mais très faiblement (1/100 de la vitesse de la balle) à cause de la différence d'inerties. Elle lui cède un peu de quantité de mouvements.

Dans la figure 3.8, la balle part au repos et c'est le joueur qui vient lui donner un choc. Puis il va venir dans le sens opposé de la balle avec l'autre joueur afin de se faire des passes. Lors du choc, le joueur donne de la quantité de mouvement à la balle qui part plus vite que lui (toujours dû au rapport des inerties). Le moteur fournit le travail et à chaque collision la balle prend un peu plus de vitesse et d'énergie. Les chocs simulés sont avec conservation de l'énergie et de la quantité de mouvement (mais le moteur vient apporter de l'énergie externe) et donc la balle accélère.

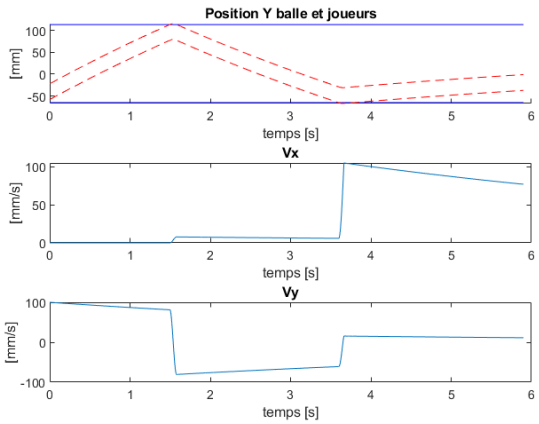


FIGURE 3.9 – Chocs de la balle avec les joueurs statiques, une vitesse initiale de 100mm/s et un décalage en X de 1mm

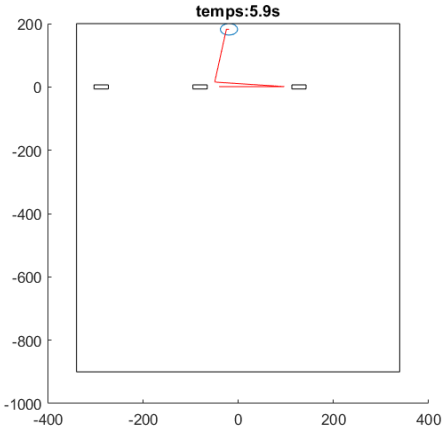


FIGURE 3.10 – Trajectoire obtenue

La simulation des figures 3.9 et 3.10 illustre cette fois-ci l'effet du décalage de la balle et du joueur. La balle a un décalage initial de 1mm et quand elle touche le joueur, ceci la dévie en lui donnant une composante X de la vitesse non nulle, qui a très rapidement de lourdes conséquences.

4 Simulation des moteurs

Les moteurs nous permettent d'actionner le joueur qui aura pour but de contrôler la balle. Ils ont une certaine dynamique et inertie qu'il faut prendre en compte. Ils sont inclus dans les équations d'états et donc important à modéliser. Le moteur angulaire est contrôlé en position et le linéaire en vitesse, afin d'être plus proche des commandes que nous allons utiliser. Sur le babyfoot, ils ont actuellement les deux un contrôle en position, mais d'après les donnés du fabricant ils peuvent aussi l'être en vitesse. Un contrôle un peu plus "manuel" de la vitesse peut être calculé en écrivant les équations la reliant à la position, comme l'a fait Yann dans son rapport [3] à la page 16.

4.1 Théorie

Moteur angulaire

Le moteur angulaire installé sur la babyfoot au moment de ce rapport est un Escon 50/5. Cyril Picard en 2013 [4] a déjà fait l'identification de système de ce moteur afin de trouver sa dynamique et ses paramètres. Les résultats paraissent concluant, je vais donc me référer à ce qu'il a déjà trouvé. Sans entrer dans les détails, voici l'équation basique du moteur

$$J^*\ddot{\theta}_m = K_m I - M_f\dot{\theta}_m \tag{4.1}$$

$$\theta_m = r\theta_p \tag{4.2}$$

Où θ_m est l'angle du moteur, θ_p l'angle du joueur, r le rapport de réduction du moteur, $J^* = 3.7 \cdot 10^{-4}[kgm^2]$ est l'inertie de la barre des attaquants rapportée au moteur, $K_m = 1.92 \cdot 10^{-2}[Nm/A]$ est la constante de couple du moteur et $M_f = 10^{-5}[Nms]$ est le moment de friction visqueux (estimé).

Quand la balle est en contact avec le joueur, elle vient appliquer une (légère) force sur celui ci. On peut ajouter cette force à l'équation précédente avec le couple qu'elle génère, rapporté au moteur, soit $\tau^* = F_{balle}L_{joueur}r^2$. On peut alors transformer cela toujours en jouant avec le rapport de réduction, pour l'avoir du coté du joueur et le mettre sous forme d'équation d'état.

$$\ddot{\theta}_m = r\ddot{\theta}_p = \frac{K_m I - M_f\dot{\theta}_p r - F_{balle}L_{joueur}r^2}{J^*} \tag{4.3}$$

$$\dot{x}_7 = \frac{-rM_f x_7 - r^2 F_{balle}L_{joueur} + K_m u_1}{rJ^*} \tag{4.4}$$

Moteur linéaire

Le moteur linéaire (Xenus XTL-230-18-S) est quand à lui plus récent et à été mis en place sur le babyfoot après le rapport de Cyril. En 2018, Yann Morize [3] a fait l'identification du système sur ce moteur, mais en prenant le système complet (avec les contrôleur), pour une sortie en position et avec des valeurs pour une fonction de transfert discrète. J'ai essayé de reprendre ce système, de le dériver (pour avoir la sortie en vitesse) et de le transformer en continu. Ceci n'est pas très fructueux, et implique des étapes qui n'ont pas de sens physique. Je n'ai pas réussi à trouver les valeurs des paramètres dans les datasheets du fabricant, j'ai donc préféré reprendre le moteur utilisé par Cyril. Il a une moins bonne dynamique que les nouveaux (200ms de temps de réponse contre 110ms trouvé par Léo Sibut en 2016 [5]), mais cette approximation reste raisonnable.

La même logique peut être appliquée, mais cette fois je suppose que le rapport de transmission permet de passer de la position angulaire à linéaire, en plus de réduire la vitesse. Le moment crée par la balle devient

une force. Nous avons pour finir l'équation suivante

$$\dot{x}_8 = \frac{-rM_f x_8 - r^2 F_{balle} + K_m u_2}{rJ^*} \tag{4.5}$$

Avec $J_m = 1.6794 \cdot 10^{-5}[kgm^2]$, $K_m = 2.05 \cdot 10^{-2}[Nm/A]$ et $M_f = 8 \cdot 10^{-3}[Nms]$. Les indices pour savoir de quel moteur on parle ont été laissés de côté afin d'alléger la lecture, mais ces paramètres restent différent pour chaque moteur.

4.2 Implémentation

L'implémentation n'est pas aussi compacte que des fonctions de transfert avec un feedback et un PID, car les états du moteurs font partie du state space. Néanmoins, elle reste équivalente. Les PIDs des moteurs ont été ajustés pour avoir des résultats satisfaisants, mais aucun calcul pour atteindre certaines performances n'ont été faits.

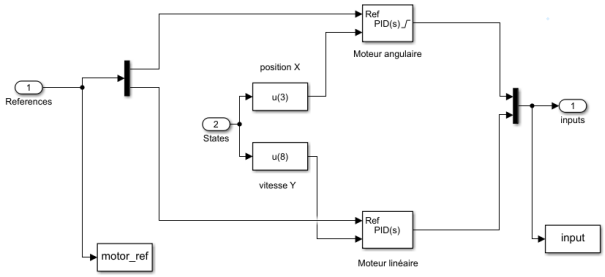


FIGURE 4.1 – Contrôle des moteurs

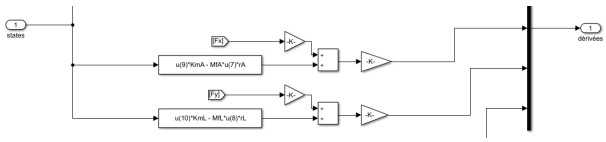


FIGURE 4.2 – Dans le state sapce, quand il y a contact avec la balle

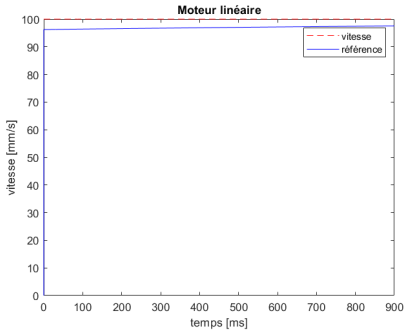


FIGURE 4.3 – Réponse du moteur linéaire pour un contrôle en vitesse

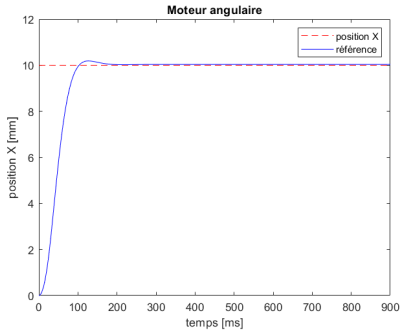


FIGURE 4.4 – Réponse du moteur angulaire pour un contrôle en position

Dans la figure 4.3, on peut voir que le moteur met très (voir trop) peu de temps pour atteindre une vitesse légèrement inférieure à 100 mm/s, puis intégre cette erreur gentiment pour suivre la référence. Le fait que la vitesse ne soit pas parfaite ne pose pas trop de problèmes pour la suite. Dans la figure 4.4, le moteur met plus de temps pour réagir avec un overshoot, mais après 180ms la réponse est stable. L'accent à été mis sur la rapidité pour que les moteurs soient réactifs et puissent au mieux corriger la trajectoire de la balle lors des moments de contacts joueurs/balle. La valeur finale des PIDs retenus est $[10, 5, 0.0]$ pour le moteur linéaire et $[2, 0.15, 0.1]$ pour l'angulaire. (À noter que c'est bien $[K_p, K_i, K_d]$ et non $[K_p, T_d, T_i]$ qui peut varier suivant les représentations).

5 Contrôle

Maintenant que la simulation de la balle et des moteurs fonctionne, on peut passer au contrôle. Le but est de pouvoir faire des passes entre deux joueurs. Pour cela il faudra garder la balle dans leur zone de travail, rejeter les perturbations sur la trajectoire dues au bruit de la table (petites aspérités, imperfections), corriger une mauvaise trajectoire, partir de n'importe quelles conditions initiales possibles et ne pas être trop sensible aux imprécisions des capteurs. Viens s'ajouter le fait que notre balle n'est pas contrôlable à tout moment, mais seulement quand elle est en contact avec un joueur.

5.1 Contrôle du décalage joueur balle

L'angle de la force appliquée et donc le ratio entre l'accélération en X et Y ne dépend que du décalage de la balle par rapport au joueur (d'après ce que l'on suppose de la simulation de la balle). Plus ce décalage sera important, plus la balle aura tendance à dévier et donc sortir de la zone de travail. On veut par conséquent tenter de supprimer toute différence en X entre la balle et le joueur et aussi la vitesse en X de la balle. Dans le cas idéal, on veut se faire des passes avec une balle qui a une composante X de la vitesse nulle et le plus proche possible de la ligne x = 0.

Pour le contrôle, deux simples termes proportionnels, un sur la position X et l'autre sur la vitesse X sont utilisés pour donner la référence au moteur angulaire. Il va tenter de maintenir ces deux états à 0. Je vais les appeler K_p et K_v respectivement, pour la suite des explications, même si ce n'est pas exactement équivalent à un PD. $K_p = 1$ tout seul signifie que joueur va se placer au centre de la balle sur la même ligne ($X_{joueur} = K_p \cdot (X_{balle} - référence) = X_{balle}$). Intuitivement, on va prendre un K_p (légèrement) supérieur ou égal à 1 afin de repousser la balle vers x = 0. Plus petit que ça l'angle entre la balle et le joueur tendra à pousser la balle vers l'extérieur et trop grand, il sera trop réactif, susceptible à des oscillations qui peuvent diverger.

On ne peut pas se représenter aussi facilement de manière simple et exacte le terme K_v. On sait de lui est qu'il doit réduire la vitesse à 0. Il est important qu'il le fasse le plus rapidement possible et lorsque notre balle n'est pas en contact avec le joueur, sa vitesse soit à peu près nulle. Sinon elle risque de ne pas être atteignable pour le joueur qui doit recevoir la passe (ou difficilement contrôlable). Il ne faut pas qu'il soit très grand, car il induirait de grand décalages du joueurs. Les figures suivantes sont là pour illustrer les effets des différentes valeurs du contrôleur. Ils sont un peu exagéré pour montrer que la trajectoire diverge en 2 passes . Même un petit défaut s'amplifie très vite avec le nombre de passes, la vitesse de la balle (relié au temps de contact) et le bruit.

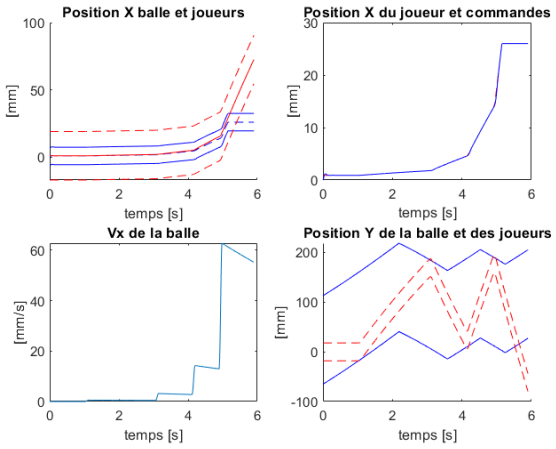


FIGURE 5.1 – $K_p = 0.9$, $K_d = 0$ $x_0 = 1$ et $v_{x0} = 0$

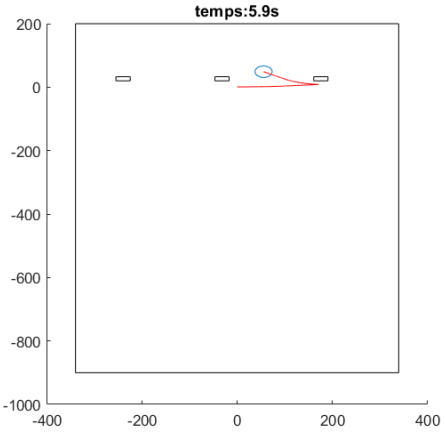


FIGURE 5.2 – Trajectoire de la balle

Les figure 5.1 et 5.2 nous montrent l'effet d'un K_p tout seul trop petit. En effet, il ne corrige pas assez la

position du joueur et tend à décaler encore plus la balle par rapport au joueur. La vitesse en X de la balle augmente et elle diverge assez rapidement, même avec juste un très léger décalage initial.

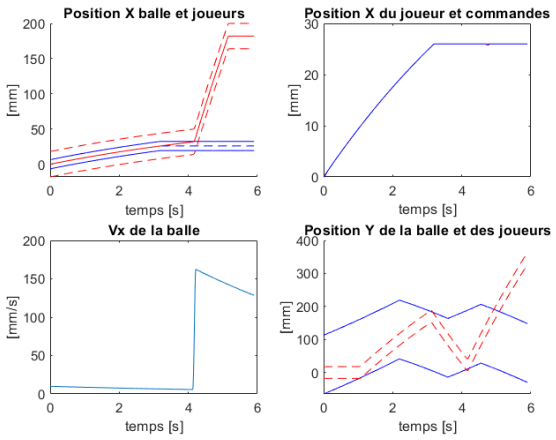


FIGURE 5.3 – $K_p = 1$, $K_d = 0$ $x_0 = 5$ et $v_{x0} = 10$

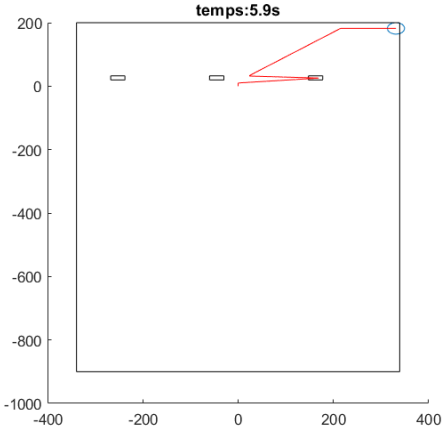


FIGURE 5.4 – Trajectoire de la balle

Ici (figure 5.3 et 5.4), K_p est juste suffisant, mais la vitesse n'est pas corrigée ($K_v = 0$, ce cas se généralise avec un K_v trop faible). Il n'y a pas ou peu de décalage entre le joueur et la balle, mais cette dernière à une vitesse qui va la sortir de la zone de travail. Le joueur ne peut pas aller plus loin et quand il veut recevoir la passe, il la prend trop décalée, ce qui l'éjecte.

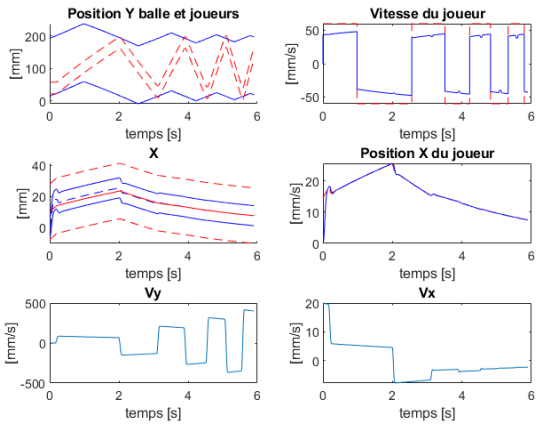


FIGURE 5.5 – PID final avec $x_0 = 10$ et $v_{x0} = 20$

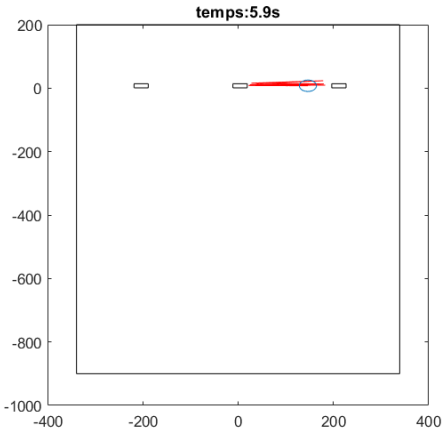


FIGURE 5.6 – Trajectoire de la balle

Les figures 5.5 et 5.6 ont été réalisées avec les valeurs du PID final, soit $K_p = 1.05$ et $K_v = 0.2$. La vitesse est rapidement corrigée pour éviter que la balle parte trop loin, tandis que la position tend vers 0 plus lentement. Même avec un décalage et une vitesse initiale assez importante par rapport aux autres, les 2 passes (et les suivantes) sont maîtrisées. Les paramètres de PIDs sont ajustés pour cette stratégie (stratégie "basique"), mais peuvent changer pour d'autres.

5.2 Contrôle de la vitesse du joueur et stratégie

Plus le temps de contact entre la balle et le joueur est long, plus il y aura de possibilités pour contrôler la balle et rectifier la trajectoire. Je vais essayer d'intégrer ce principe simple. 3 stratégies différentes ont été mise en place dans la simulation. La première offre la possibilité à l'utilisateur de choisir la référence d'entrée, sous

forme de vecteur par rapport au temps. La seconde, appelée "basique", va simplement regarder dans quelle zone la balle se trouve pour choisir le signe de la vitesse du joueur. Ce dernier va ainsi venir taper la balle dans le sens opposé. Cette technique a été utilisée dans les figures 5.1 à 5.6. Le contact entre le joueur et la balle n'est qu'un seul choc et par définition très bref. Dès que le choc est passé, la balle n'est plus contrôlable jusqu'au prochain. Faire des passes dans ce cas là est compliqué avec un simple PID, car il faut que le joueur soit très bien placé au moment du contact.

La dernière stratégie implémentée dans la simulation est celle avec amortissement (figure 5.9 et 5.10). Le principe est simple, amortir la balle quand elle arrive vers le joueur avec une rampe de vitesse et faire la passe avec une autre rampe. Ceci évite une trop grande différence de vitesses entre le joueur et la balle qui entraînerait un choc. Le temps de contact est maximisé et le contrôleur a plus d'opportunités pour corriger. Dans Simulink, ceci consiste en un switch case de 3 étapes. L'étape 1 se déroule lorsque la balle est suffisamment loin du joueur ou considérée comme immobile. Si la balle est immobile, le joueur va venir la pousser pour lui donner du mouvement (figure 5.7 de 0 à 3s). Puis quand elle a assez de vitesse il va venir se placer, c'est à dire que l'autre joueur va se rapprocher de la balle et attendre qu'elle soit assez proche de lui (de 3 à 6,5s). À ce moment, on passe à l'étape 2. Il enregistre la vitesse avec laquelle la balle arrive, puis en partant de cette dernière, il crée une rampe jusqu'à l'arrêt (6,5 à 7,5s). Puis il passe l'étape 3 qui consiste à repousser la balle avec une autre rampe dans le sens inverse (7,5 à 8,5s), pour atteindre une vitesse désirée et revenir à l'étape 1 (de 8,5 à 10s).

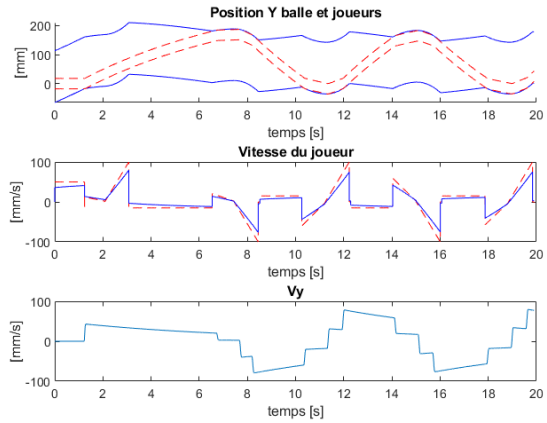


FIGURE 5.7 – illustration des 3 étapes de la stratégie avec amortissement

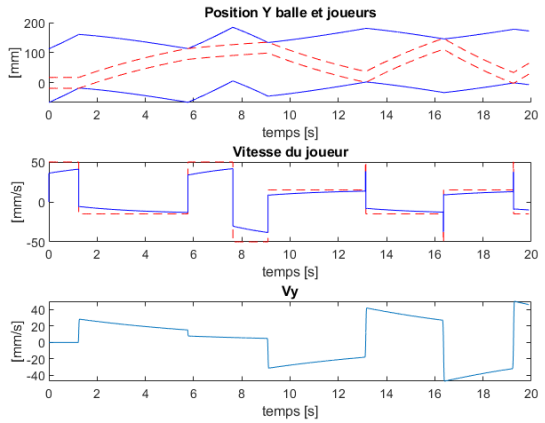


FIGURE 5.8 – Quand la rampe d'amortissement est lancée trop tard

Depuis le graphique des positions on remarque que la balle colle relativement bien au joueur. Cependant, le moteur ne respecte pas tout à fait la référence et la vitesse de la balle n'est pas linéaire comme souhaité. Des petits chocs se produisent et ralentissent la balle, mais la balle n'est pas continuellement en contact. Par contre si on lance la rampe trop tard, les moteurs n'ont pas le temps de se mettre à la vitesse de la balle et l'amortissement ne marche pas comme voulu (figure 5.8 à 6,5s). Lancer la rampe trop tôt va aussi avoir un effet non désiré, car la différence de vitesse entre la balle et le joueur sera trop grande. Les paramètres du PIDs ont varier par rapport à la stratégie basique et sont [1.05, 0, 2]. Il peut se permettre d'avoir un K_v plus grand sans être trop instable (si la balle ne va pas dans la direction voulue, il peut la corriger directement) grâce au plus long contact avec la balle.

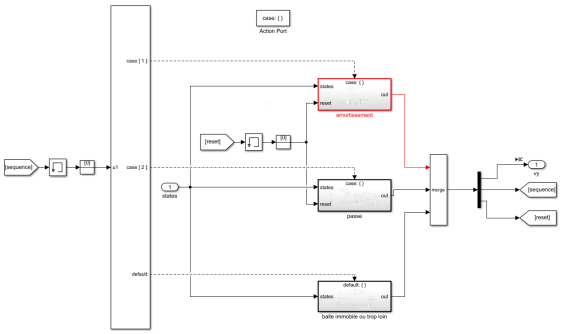


FIGURE 5.9 – Stratégie dans Simulink avec les 3 étapes

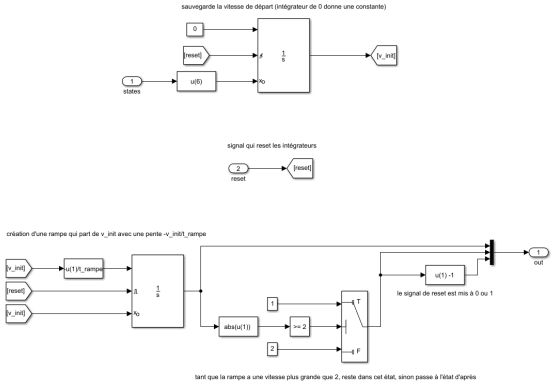


FIGURE 5.10 – Simulink de l'étape d'amortissement

6 Simulation des capteurs et du bruit

Maintenant que notre système sait comment contrôler la balle, on va lui ajouter les imperfections du monde réel et voir comment il réagit.

6.1 Capteurs

Tout capteur à ses caractéristiques et ses limites. Il a une certaine exactitude et précision (accuracy and precision en anglais), une certaine bande passante et fréquence acquisition des données et une certaine plage de valeurs dans laquelle il peut travailler. Tout ceci mène à des petites erreurs par rapport au cas parfait, qu'il faut tenir en compte, car nos mesures et données proviennent de capteurs. Afin d'être au plus proche de la réalité, il faut aussi les considérer dans la simulation. L'unique capteur considéré est la caméra.

Caméra

La caméra actuellement présente sur le babyfoot est Ximea MQ003CG CM, avec une résolution de 648x488 et une vitesse d'acquisition de maximum 500 images/s. C'est le capteur qui nous permet d'estimer la position de la balle et de calculer sa vitesse. La table du babyfoot a une taille de 1220x770mm et si la caméra est placée correctement, ceci nous donne un ratio d'environ 1.6 mm/pixel. La vision est par conséquent discrétisée en une grille de 1,6mm. On va supposer que la position de la balle à une résolution d'un pixel. Certaines techniques permettent une précision sub-pixel par du traitement d'images mais ceci n'est pas présent dans notre cas. De plus on applique une transformation géométrique à l'image acquise, pour des raisons de géométrie et de lentilles (d'où la nécessité d'une calibration si la caméra est bougée, voir rapport de Leila Afilal et al. [1], sur la vision et la calibration au printemps 2019). Cette transformation va modifier la précision de la position suivant l'emplacement sur le terrain, mais dans notre simulation, on approxime ceci comme constant.

La position estimée par le capteur est simplement une discrétisation de la position par pas de 1.6mm. La vitesse est calculée par différence finie ($\frac{P_t - P_{t-1}}{\Delta t}$, avec P_t la position à l'instant t). Comme la position est discrétisée, elle fait des changements abrupte, qui se traduisent par une forte différence finie. Pour éviter cela, la position est d'abord passée à travers une moyenne mobile qui permet de lisser le signal. Cette méthode à été implémentée par Yann en 2019 [3] avec une taille de fenêtre variable suivant la vitesse de la balle. Toute fois, dans la simulation elle est considéré comme constante et différente pour X et Y car v_x est censé être très faible.

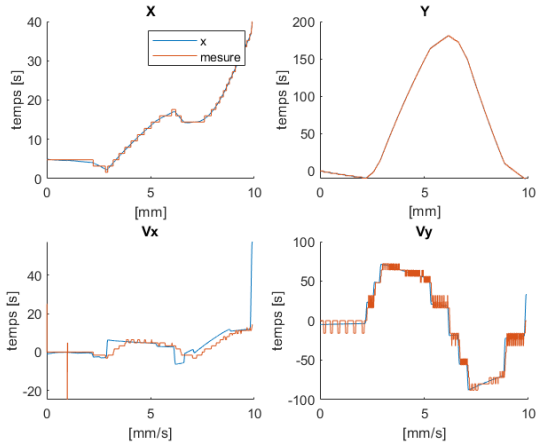


FIGURE 6.1 – Mesures de la caméra comparées aux vraies valeurs

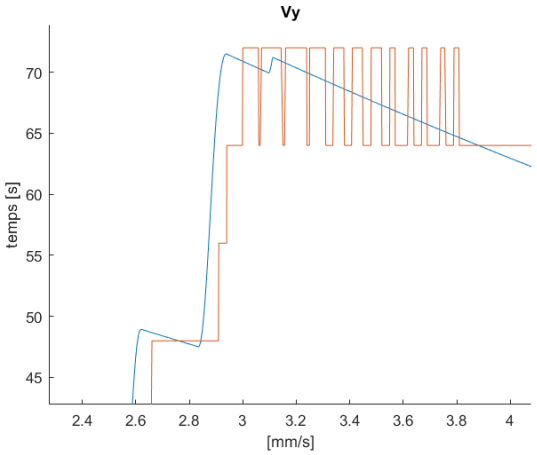


FIGURE 6.2 – Zoom sur la valeur de v_y

De la figure 6.1 on voit que le passage à travers la caméra modifie la perception des états. L'effet de quantification se fait plus resentir sur la position X que Y, car celle ci est gardée proche de zéro. Les mesures de la vitesse ne sont pas exactes. Grâce à la moyenne mobile, on a pu éliminer des sauts trop importants dans les valeurs. Mais elle ajoute aussi un retard au signal de N/2 échantillons, avec N la taille de la moyenne mobile. Sur la figure 6.3 un bloc de saturation de la vitesse x à été ajouté pour couper les trop grandes valeurs de la différence finie, qui sont du bruit et qui apparaissent au lancement de la simulation. Le contrôleur de l'angle a aussi été légèrement modifié et comprend une moyenne mobile pour éviter des sauts trop importants dans la commande que le moteur ne réussit pas à suivre.

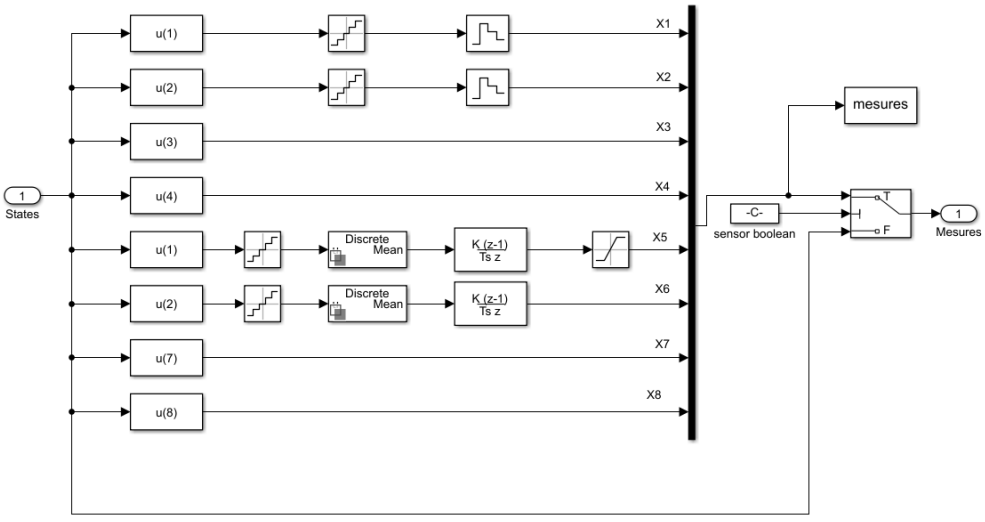


FIGURE 6.3 – implémentation des capteurs dans Simulink

6.2 Bruit

Puisque notre table n'est pas parfaitement plate et que notre balle n'est pas une sphère parfaite non plus, la trajectoire aura tendance à être déviée de manière stochastique et non voulue. Ceci est du bruit. Il est facile de s'en convaincre en observant une balle rouler sur un plan. De plus on peut aussi constater que plus la balle est lente, plus il y a de déviations. Le bruit dans les systèmes peut être quantifié par sa distribution, sa norme, son spectre de fréquence,... afin d'être reproduit au mieux. Malheureusement dans mon cas, celui ci ne pourra être qu'estimé à travers des approximations.

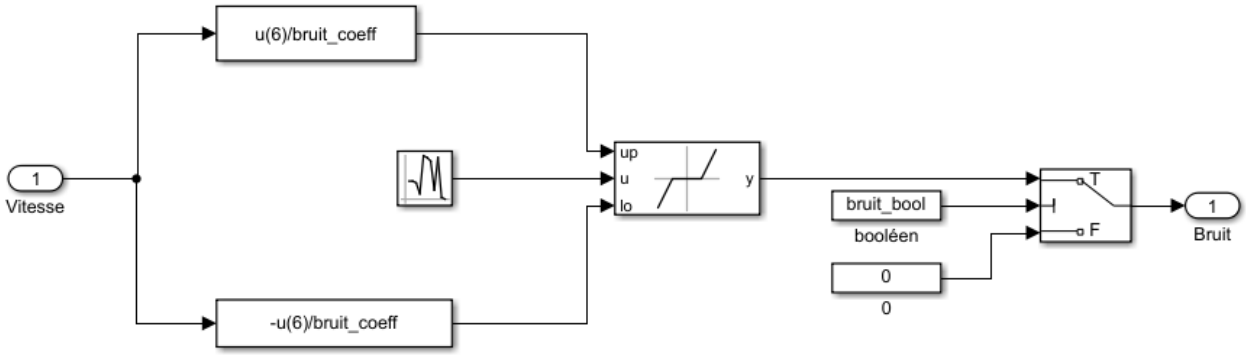


FIGURE 6.4 – génération du bruit dans Simulink

Pour générer le bruit, j'ai supposé qu'il était gaussien avec une moyenne à zéro, comme beaucoup de phénomènes naturels. Il passe ensuite dans un soft threshold avec des limites dynamiques, qui dépendent de la vitesse de la balle. Toutes les valeurs en dessous des limites sont mises à zéros et pour les autres, on leur soustrait la limite. Ceci permet de ne pas avoir du bruit tout le temps et d'en avoir plus de bruit si la vitesse est faible, comme on peut le voir sur la figure 6.5. Avec moins de vitesse, la balle est plus sujette au bruit. Sur la figure 6.4, on peut voir l'effet du bruit. Sans contrôle, la balle devient vite inaccessible.

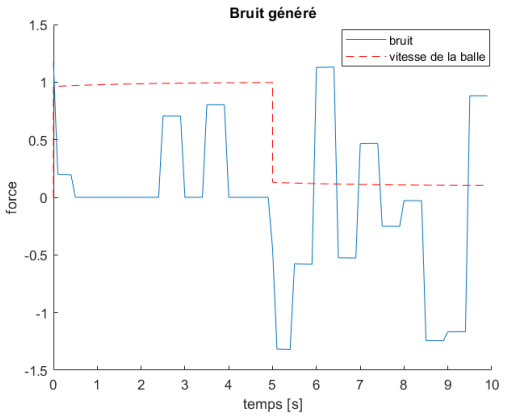


FIGURE 6.5 – Bruit généré

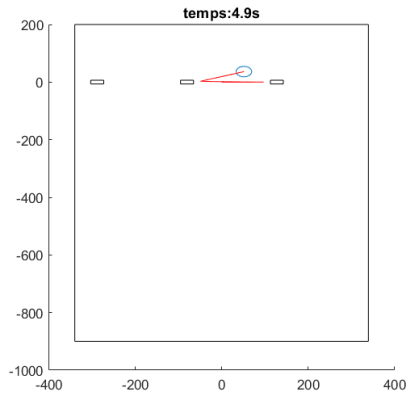


FIGURE 6.6 – influence du bruit sur la trajectoire avec les joueurs fixes

7 Résultats

Une fois la simulation est crée, il ne reste maintenant qu'à la tester !.On peut créer beaucoup de schémas de simulations possibles et aussi faire varier les paramètres, et les décisions dans les stratégies pour obtenir les meilleures performances possibles.

Par exemple, dans la figure 7.1 mon but était de réaliser un maximum de passes possibles, et le babyfoot à réalisé une quarantaine de passes, avant que la balle ne diverge. Le coefficient du bruit était à 200, avec les capteurs et des conditions initiales aléatoires (restant raisonnables). La vitesse était certes très basse (100mm/s) et ceci reste une simulation, mais cela montre qu'un contrôleur plutôt stable est possible. Après un certain temps, une combinaison de perturbations permet de dévier suffisamment la balle pour que le système ne la récupère pas. On peut voir des oscillations sur la position en X avec une longueur d'onde de plusieurs passes. Ceci est dû à notre contrôleur, qui comme un contrôleur classique peut osciller. Hormis que dans notre cas, il n'a pas accès en tout temps à la balle pour mieux se corriger. Avec un contrôleur plus agressif, ces oscillations ont une plus haute fréquence (figure 7.2). Si on va dans l'autre sens, en réduisant la valeur des paramètres, on arrive facilement à un système qui ne corrige pas assez et qui est moins robuste. (figure 7.3).

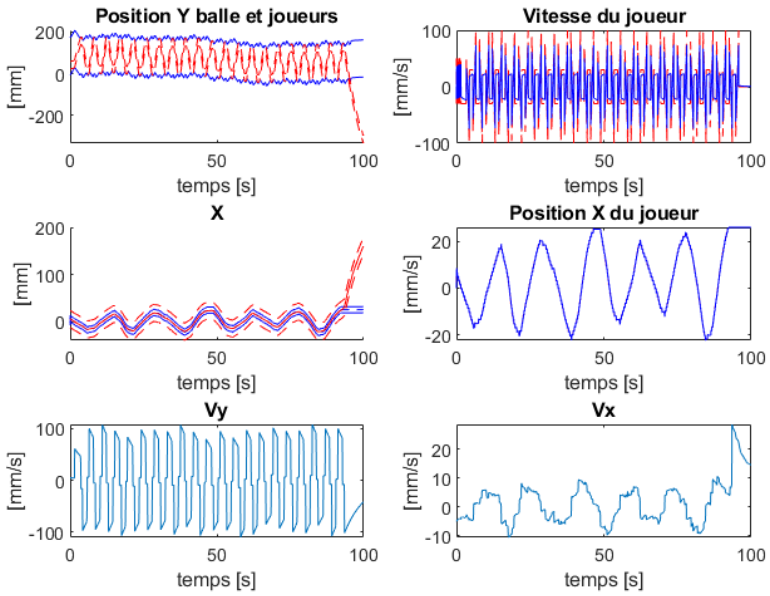


FIGURE 7.1 – essai pour faire le plus de passes possible

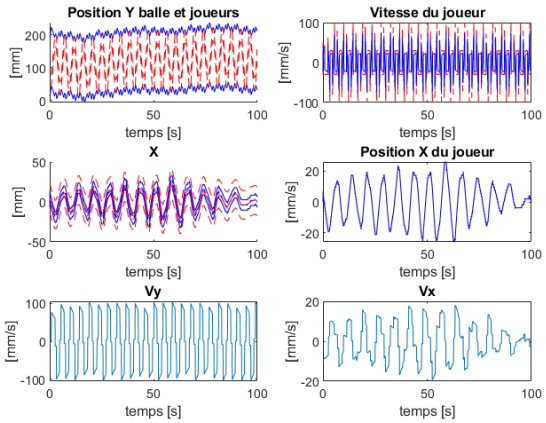


FIGURE 7.2 – stabilité avec un contrôleur plus agressif

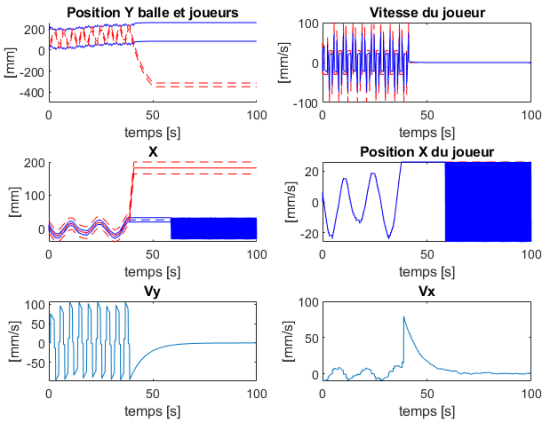


FIGURE 7.3 – stabilité avec un contrôleur trop faible

Un autre test réalisable qui est peut-être plus pertinent que de se faire le plus de passes , est de se faire des passes le plus rapidement possible. En effet, si on se remet dans la situation du vrai babyfoot, ceci est une arme fatale pour déstabiliser la défense adverse. C'est ce qui a été fait à la figure 7.4. La balle atteint une vitesse d'environ 1000mm/s et la rampe de vitesse à un temps relativement court (0.2s pour amortir et le même temps pour lancer), ce qui permet de réaliser deux passes, un aller-retour entre les joueurs en environ 1s (entre la seconde 3 et 4 sur la figure). Avec cette vitesse et cette configuration, il n'arrive pas à contrôler la balle plus que quelques passes, mais dans l'optique du babyfoot cela pourrait suffire.

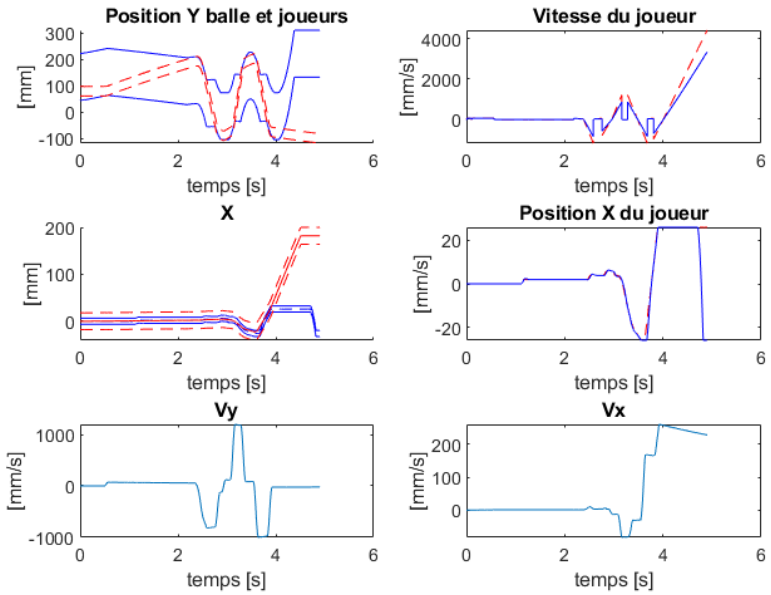


FIGURE 7.4 – essai de faire des passes relativement rapidement

8 Conclusion

Cette simulation n'est pas ce qui était réellement prévu au départ, mais je suis content d'avoir pu la réaliser et ce qu'elle m'a apprise. Elle fonctionne plutôt bien mais n'est pas encore parfaite.Il faut être conscient qu'elle n'est pas la réalité : faire des passes dans la vraie vie avec le PID trouvé ne sera pas si simple et les performances ne seront pas les mêmes. Néanmoins, j'ai essayé de faire de mon mieux pour qu'elle reflète la réalité, que cela soit au niveau des équations ou des limitations physiques. Si elle venait à être améliorée, elle pourrait servir à faire des tests plus simplement que sur le babyfoot et permettrait de confirmer ou non des stratégies élaborées sans y passer trop de temps. Elle pourrait aussi simplement être utilisée dans un but plus ludique et pédagogique. J'espère aussi qu'en la combinant avec mon rapport, cela puisse aider les futurs étudiants travaillant dessus à comprendre certains phénomènes physiques sans trop de problèmes, et qui pourraient être utiles pour d'autres fonctions.

8.1 Futures améliorations possibles

Beaucoup d'améliorations restent encore possibles que cela soit sur le babyfoot ou sur la simulation.

Simulation

Ceci est la première version de la simulation en espérant que des améliorations pourront suivre. Il reste beaucoup de points potentiellement améliorable. Dans un premier temps, on pourrait imaginer essayer de trouver tous les paramètres réels, pour les frottements, la constante de force, définir un peu mieux le bruit,... des tests pourrait être réalisés pour définir ces inconnues, en même temps de vérifier que la simulation se comporte correctement, et si nécessaire la modifier.

Les passes et contacts n'ont été implémenté que pour deux joueurs avant, on pourrait généraliser ceci aux autres joueurs et aux parois, en suivant les équations. Des stratégies plus complexes pourraient aussi être mise en place, notamment d'essayer de tirer, ou de faire quelques passes et de trouver le moment opportun pour placer un tir en fonction de la position de l'adversaire. Le contrôleur pourrait aussi être modifié en quelque chose de plus compliqué que le simple terme proportionnel à la position et à la vitesse. Le système pourrait calculer des prédiction de positions et trouver le décalage optimal pour renvoyer la balle. La simulation pourrait aussi être rendue plus interactive voir associée à un autre programme que Simulink.

Babyfoot

Comme je n'ai pas travaillé sur le babyfoot réel, je n'ai pas pu constater les problèmes par moi même. L'état d'avancement est sensiblement le même qu'avant mon arrivée et je vais vous laisser essentiellement vous référer aux autres rapports. Dans la suite de mon travail, il faudrait implémenter les passes avec les stratégies de bases de ma simulation, c'est à dire avec un contrôleur sur la position et vitesse X pour le moteur angulaire, ainsi qu'un amortissement sur le moteur linéaire. Si les paramètres de ma simulation sont suffisamment ajustés par rapport à la réalité on pourrait aussi imaginer utiliser ces équations pour prédire la position et créer un contrôleur qui trouve le décalage optimal (même contrôleur précédemment cité dans les amélioration de la simulation).

Références

[1] Leila Afilal, Kenza Benabderrazik, and David Nigri. Vision and calibration, 2019.

[2] Rod Cross. Cue and ball deflection (or “squirt") in billiards. *American Journal of Physics*, 2007.

[3] Yann Morize. Vision and strategy : Ball capture, 2019.

[4] Cyril Picard. Control system for babyfoot, 2013.

[5] Léo Sibut. Babyfoot mechanical electronic update, 2016.