Sorbonne University
Master 2 SAR

Robot Mobile: Suivi de chemin et application au contrôle en convoi

January 10, 2024

Compte Rendu

Par
LU Zhichen 21117174
YU Yifeng 21113616

Contents

1 Introduction

L'objectif de cette session pratique est d'appliquer des algorithmes de suivi de trajectoire pour les robots à roues, en particulier les robots monocycles, à l'aide de Matlab Simulink. Nous nous concentrerons sur l'illustration de l'application de ces algorithmes dans le cas de plusieurs robots monocycles voyageant le long du même chemin dans une formation de convoi l'un après l'autre.

Pendant la session, nous avons utilisé le code Simulink de Matlab, qui nous a fourni toute la section de simulation, de sorte que nous n'avons eu qu'à exécuter les commandes finales, qui seront mises en évidence plus loin dans ce rapport.

2 Partie I: Contrôle en position du véhicule de tête

L'objectif de cette partie est de synthétiser un contrôleur pour le véhicule de tête afin de faire du suivi de chemin en position.

1. Dans le fichier "$capteur.m$",d_f et d_b représentent respectivement les distances les plus courtes entre la position des capteurs des véhicules de tête et de suivi par rapport à la trajectoire. Les étapes spécifiques du calcul sont les suivantes

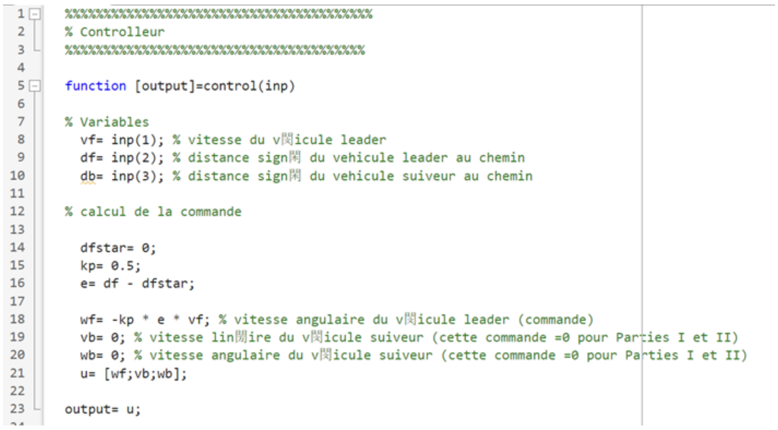
   (a) Calcul des points projetés sur la trajectoire : Tout d'abord, 200 points équidistants sont sélectionnés pour chaque véhicule à +/- 5 mètres de l'axe x, en utilisant la position du capteur comme origine. Ensuite, nous calculons pour chaque point la coordonnée transversale (notée "xc") et le carré de la distance entre le point de la courbe correspondant à cette coordonnée transversale ("xc") et l'origine (notée "$value$"). La "$value$" est ensuite comparée à un seuil prédéfini "opt". Lorsque la valeur "$value$" est inférieure à "opt", la valeur "opt" est mise à jour en "$value$" et la valeur "xc*" est mise à jour en "xc" actuel du point. Le résultat final est la distance la plus courte $d_f(d_b)$ pour chaque itération et l'emplacement du point projeté.

   (b) Détermination du signe de d_f et d_b : lors du calcul des distances d_f et d_b, si le véhicule est situé au-dessus de la trajectoire (c'est-à-dire $y_{f_{x=x_i}} > y_{c_{x=x_i}}$), la distance est la racine carrée de cette valeur. Inversement, si le véhicule est situé en dessous de la trajectoire, la distance est la racine carrée négative de la valeur.

2. Nous choisissons d'abord un contrôle de position, réalisé par le gain k_1 et telle que:

$$u_2 = -ku_1 d(k > 0) \tag{1}$$

Ici, $u_1 = v_f$, $u_2 = \omega_f$, $d = d_f - d_{f_{star}}$.



```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Controlleur
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [output]=control(inp)

% Variables
  vf= inp(1); % vitesse du v閲icule leader
  df= inp(2); % distance sign閲 du vehicule leader au chemin
  db= inp(3); % distance sign閲 du vehicule suiveur au chemin

% calcul de la commande

  dfstar= 0;
  kp= 0.5;
  e= df - dfstar;

  wf= -kp * e * vf; % vitesse angulaire du v閲icule leader (commande)
  vb= 0; % vitesse lin閲ire du v閲icule suiveur (cette commande =0 pour Parties I et II)
  wb= 0; % vitesse angulaire du v閲icule suiveur (cette commande =0 pour Parties I et II)
  u= [wf;vb;wb];

output= u;
```

Figure 1: Algorithme de loi de command ω_f

3. Lorsque la vitesse est positive, il y a des oscillations relativement importantes au début du mouvement du véhicule, mais celles-ci sont rapidement contrôlées, ce qui permet au véhicule de suivre la trajectoire. Cependant, à la fin de la trajectoire, il y a une section de très faible vitesse.

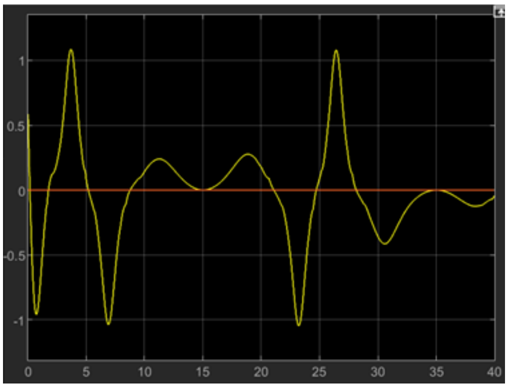


Figure 2: Scope U K_p normal

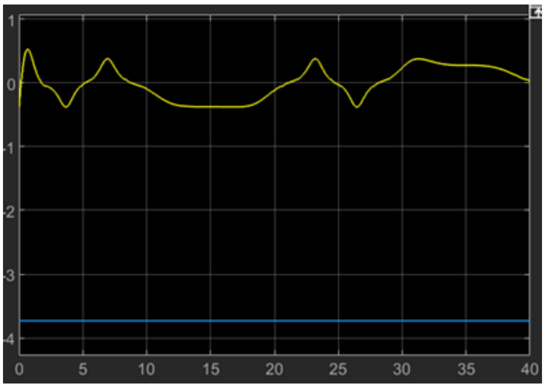


Figure 3: Scope d_f/d_b K_p normal

En modifiant la valeur de 'K_p', nous avons observé que lorsque 'K_p' est trop faible (0.1), le contrôle de la vitesse est insuffisant, et le véhicule ne peut pas suivre strictement le chemin, entraînant des oscillations latérales.

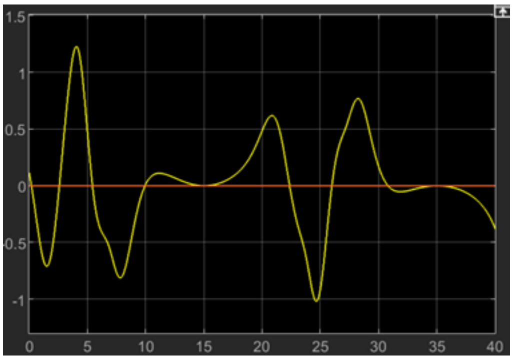


Figure 4: Scope U K_p petit

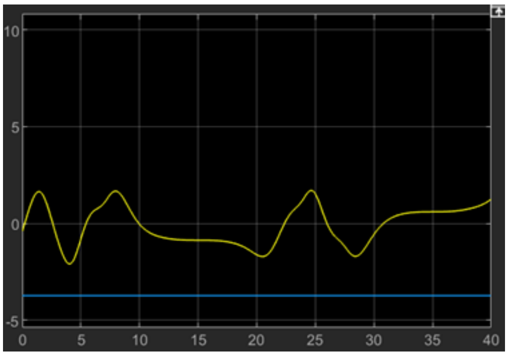


Figure 5: Scope d_f/d_b K_p petit

Lorsque 'K_p' est trop élevé (14), le contrôle de la vitesse est trop agressif. Bien que le véhicule suive la trajectoire, le réglage excessif perturbe sa stabilité et provoque de grandes oscillations

directionnelles.

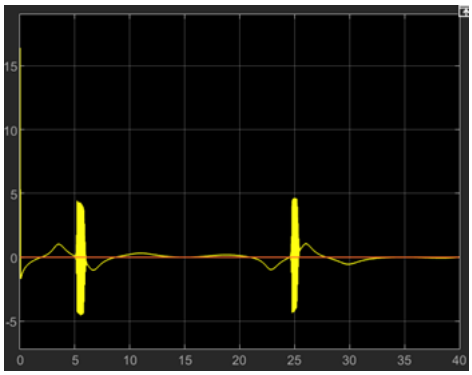


Figure 6: Scope U K_p grand

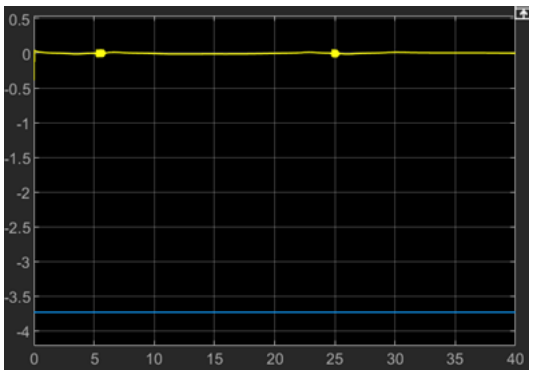


Figure 7: Scope d_f / d_b K_p grand

4. Lorsque nous avons tenté d'appliquer ce contrôle dans des conditions de vitesse aléatoire, dès qu'une vitesse inverse est apparue sur le chemin, le véhicule a perdu le contrôle et a commencé à tourner en rond.

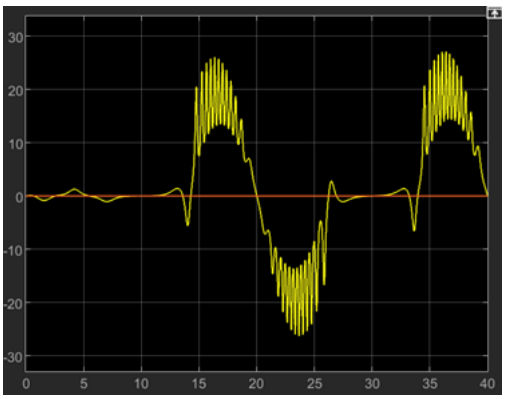


Figure 8: Scope U de Vitesse quelconque

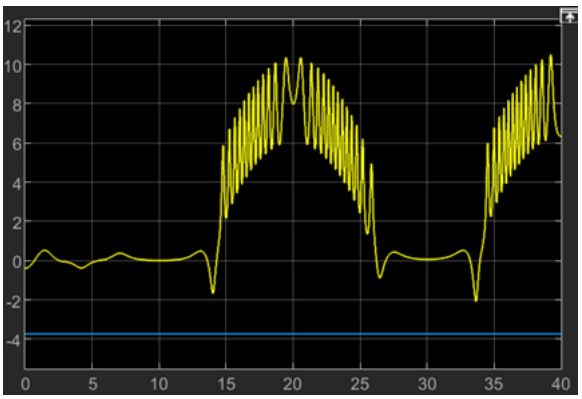


Figure 9: Scope d_f / d_b de vitesse quelconque

3 Partie II: Contrôle en position et orientation du véhicule de tête

1. Dans cette partie, nous approximons la voiture comme un point capable de se déplacer et de tourner. Dans la partie précédente, nous avons calculé la distance entre le devant de la voiture et la trajectoire, donc il était nécessaire de prendre en compte la distance "D" entre le point central des deux roues arrière et le devant de la voiture. Cependant, dans cette partie, nous souhaitons que 'D' soit égal à 0. Par conséquent, nous avons modifié la valeur de "D" à 0 dans les fichiers "$capteur$" et "$anim$".

2. La valeur non mesurée dont nous avons besoin est la différence entre l'angle d'orientation "θ_1" de la voiture à un point quelconque et l'angle de la tangente "θ_2" au point de projection correspondant sur le chemin. Cette différence est notée "θ_e".

3. À travers la formule, nous savons que pour calculer "$\dot{\hat{d}}$" et "$\dot{\hat{\theta}}$":

$$\begin{cases} \dot{\hat{d}} = u_1\hat{\theta}_e - k_1|u_1|(\hat{d}-d) \\ \dot{\hat{\theta}} = u_2 - k_2 u_1(\hat{d}-d) \end{cases} \tag{2}$$

Nous avons besoin de cinq paramètres : la vitesse angulaire "$w_f(u_2)$" de la voiture, la vitesse linéaire "$v_f(u_1)$", la distance la plus courte "$d_f(d)$" entre la voiture et le chemin, ainsi que "$\hat{d}$" et "$\hat{\theta}$". Par conséquent, nous avons configuré un tel observateur :

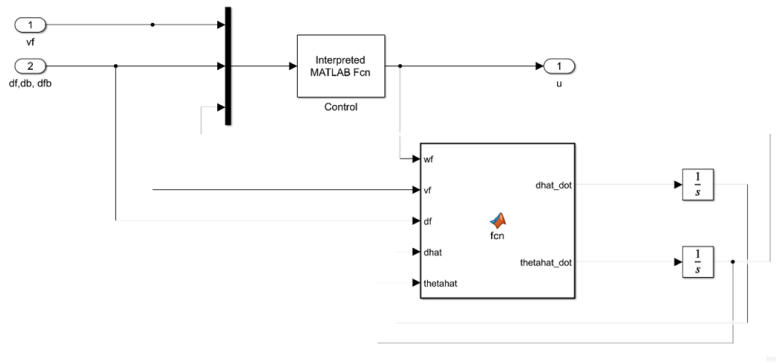


Figure 10: Observateur de $\hat{\theta}$ Partie 2

Dans le “fonction Matlab”, nous avons appliqué les équations pour calculer "$\hat{d}$" et "$\hat{\theta}$".

```
1   function [dhat_dot,thetahat_dot] = fcn(wf, vf, df, dhat, thetahat)
2
3   u1=vf;
4   u2=wf(1);
5   d=df(1);
6   k1=1;
7   k2=1;
8
9   dhat_dot=u1*thetahat - k1*abs(u1)*(dhat-d);
10  thetahat_dot= u2 - k2*u1*(dhat-d);
```

Figure 11: Fonction d’observateur $\hat{\theta}$

De plus, un intégrateur a été ajouté à la fin pour générer les valeurs finales requises de ’dhat‘ et ’thetahat‘. Finalement, ‘thetahat’ doit également être utilisé comme un paramètre d’entrée pour participer au calcul de u.

4. Pour la loi de commande, nous avons toujours choisi un contrôle proportionnel. Nous avons défini deux coefficients de gain ‘k_1’ et ‘$k2$’, et, en combinant la vitesse linéaire "v_f" de la voiture (u_1), la différence entre la distance réelle ‘d_f’ de la voiture par rapport au chemin et la distance souhaitée "d_f" (d), ainsi que la valeur non mesurable ‘thetahat’ (θ_e) calculée dans notre observateur, nous avons obtenu l’équation de contrôle de la vitesse angulaire ‘$w_f(u_2)$’ de la voiture en suivant la formule.

$$u_2 = -k_1 u_1 d - k_2 |u_1| \theta_e (k_1, k_2 > 0) \tag{3}$$

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Controlleur
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [output]=control(inp)

% Variables
  vf= inp(1); % vitesse du veicule leader
  df= inp(2); % distance signe du vehicule leader au chemin
  db= inp(3); % distance signe du vehicule suiveur au chemin
  thetahat= inp(5);

% calcul de la commande

  dfstar= 0;
  e= df - dfstar;
  k1 = 1;
  k2 = 1;

  wf= -k1*vf*e - k2 * abs(vf) * thetahat; % vitesse angulaire du veicule leader
  vb= 0; % vitesse lineaire du veicule suiveur (cette commande =0 pour Parties
  wb= 0; % vitesse angulaire du veicule suiveur (cette commande =0 pour Parties
  u= [wf;vb;wb];

output= u;
```

Figure 12: Algorithme de command Proportionnel partie 2

5. Dans cette partie, nous avons réalisé le contrôle à la fois de la position et de la direction du véhicule. Dans la partie précédente, dès que la vitesse devenait négative, le véhicule perdait le contrôle. Cependant, dans cette partie, nous avons intégré la valeur non mesurable 'thetae' dans le système, permettant ainsi un contrôle simultané de la direction du véhicule. Grâce à cette amélioration, le véhicule peut suivre parfaitement la trajectoire, que sa vitesse soit positive ou négative.

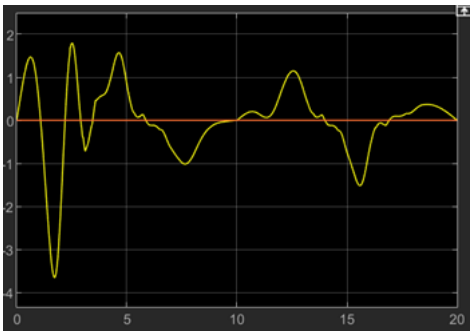


Figure 13: Scope U Partie 2

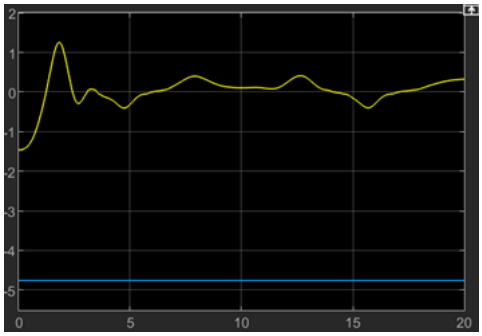


Figure 14: Scope d_f / d_b Partie 2

4 Partie III: Contrôle en convoi

Dans la troisième partie, nous utilisons la commande PID et introduisons quatre paramètres de gain k_3, k_4, k_5, k_{fb} pour contrôler la distance entre la voiture en attente et la voiture de tête.

Nous définissons la vitesse linéaire v_b du véhicule suiveur comme indiqué ci-dessous :

$$v_b = v_f + k_{fb}(d_{fb} - d_{fb_{star}}) \tag{4}$$

et la vitesse angulaire ω_b:

$$\omega_b = -k_3 v_b d - k_4 |v_b| \theta_e - k_5 |v_b| I_d, \quad \text{avec} \quad \dot{I}_d = v_b d \quad (k_3, k_4, k_5 > 0) \tag{5}$$

```
function [output]=control(inp)

% Variables
  vf= inp(1); % vitesse du v[illegible]icule leader
  df= inp(2); % distance sign[illegible] du vehicule leader au chemin
  db= inp(3); % distance sign[illegible] du vehicule suiveur au chemin
  dfb= inp(4); % distance entre le vehicule leader et le vehicule suiveur
  thetahatf= inp(5);
  thetahatb= inp(6)
  Id= inp(7);
% calcul de la commande

  dfstar= 1;
  dbstar= -1;
  dfbstar= 4;
  ef= df - dfstar;
  eb= db - dbstar;
  efb= dfb - dfbstar;
  k1 = 1;
  k2 = 1;
  k3 = 0.21;
  k4 = 1;
  k5 = 1e-3;
  kfb= 4;

  wf= -k1*vf*ef - k2 * abs(vf) * thetahatf; % vitesse angulaire du veicule leader (commande)
  vb=  vf + kfb*efb;  % vitesse lineaire du veicule suiveur
  wb= -k3*vb*eb - k4 * abs(vb) * thetahatb -k5*abs(vb)*Id;   % vitesse angulaire du veicule suiveur
  u= [wf;vb;wb];

output= u;
```

Figure 15: Partie 3 Algorithme de commande

Ajoutez le même observateur que celui de la voiture avant, qui est utilisé pour mettre en œuvre le suivi de la voiture suiveur.

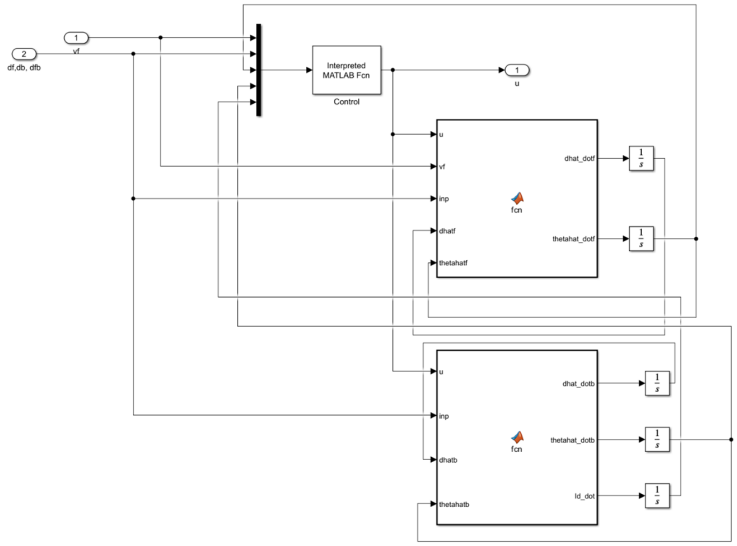


Figure 16: Observateur partie 3

À travers la formule, nous savons que pour calculer "$\dot{\hat{d}}$" et "$\dot{\hat{\theta}}$":

$$\begin{cases} \dot{\hat{d}} = u_1\hat{\theta}_e - k_1|u_1|(\hat{d}-d) \\ \dot{\hat{\theta}} = u_2 - k_2 u_1(\hat{d}-d) \end{cases} \tag{6}$$

```
function [dhat_dotf, thetahat_dotf] = fcn(u, vf, inp, dhatf, thetahatf)


% le veicule leader
u1=vf;
u2=u(1);
df=inp(1);
k1=1;
k2=1;

dhatdotf=u1*thetahatf - k1*abs(u1)*(dhatf-df);
thetahatdotf= u2 - k2*u1*(dhatf-df);

dhat_dotf= dhatdotf;
thetahat_dotf= thetahatdotf;
```

Figure 17: Fonction de l'observateur de f

```
function [dhat_dotb, thetahat_dotb, Id_dot]= fcn(u, inp, dhatb, thetahatb)

% le veicule suiveur
u1=u(2);
u2=u(3);
db=inp(2);
k1=0.5;
k2=0.5;

dhatdotb=u1*thetahatb - k1*abs(u1)*(dhatb-db);
thetahatdotb= u2 - k2*u1*(dhatb-db);

dhat_dotb= dhatdotb;
thetahat_dotb= thetahatdotb;
Id_dot=u1*db;
```

Figure 18: Fonction de l'observateur de b

Après avoir exécuté "$anim.m$", selon les images ci-dessous,nous constatons que la voiture arrière suit parfaitement la voiture avant, même pour une vitesse linéaire négative v_f.

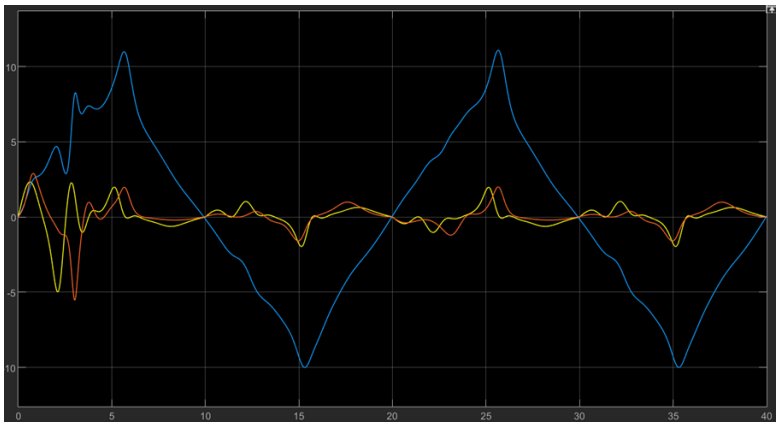


Figure 19: Scope U Partie 3

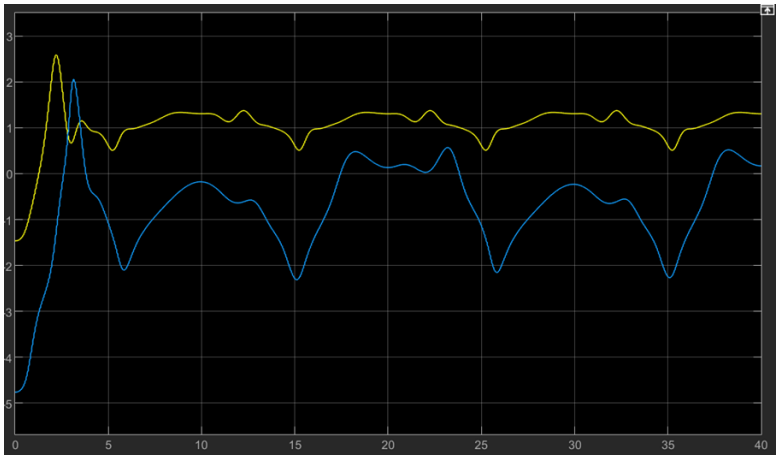


Figure 20: Scope d_f/d_b Partie 3

En écrivant le code, nous constatons que nous devons ajuster le paramètre pour les valeurs de k_3, k_4, k_5, k_{fb}. Les quatre valeurs de gain sont trop grandes, ce qui entraîne une vitesse angulaire wb du véhicule arrière trop importante, et la performance dans la situation de simulation est que le véhicule arrière tourne sur place. Si la valeur du gain est trop faible, le véhicule arrière ne pourra pas maintenir une distance suffisante avec le véhicule avant.