Commande avancée d'un manipulateur mobile

Nawel Himdi
Max Sonzogni

January 30, 2019

1 Modélisation

On choisit de modéliser un robot mobile surmonté d'un bras pilotable, avec les longueurs fixes $D = 1$ m, $l_1 = 0.8$ m et $l_2 = 0.55$ m (c.f. Figure 1).

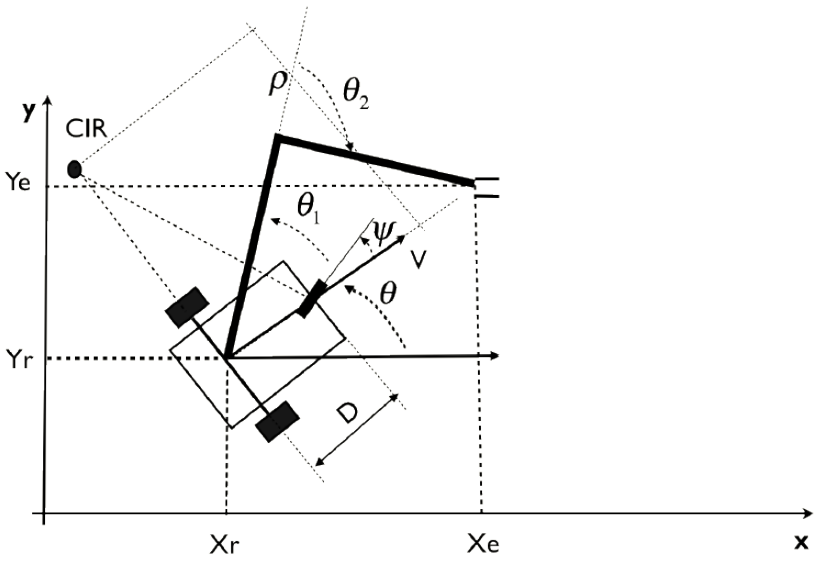


Figure 1: Représentation du manipulateur mobile

On donne la position $(X_e\ Y_e)^T$ de l'effecteur (bout du bras) en fonction de la position $(X_r\ Y_r)^T$ du robot (base du bras), de l'angle d'orientation du robot θ et des angles θ_1 et θ_2 de chaque partie du bras :

$$X_e = X_r + l_1 cos(\theta + \theta_1) + l_2 cos(\theta + \theta_1 + \theta_2) \tag{1}$$

$$Y_e = Y_r + l_1 sin(\theta + \theta_1) + l_2 sin(\theta + \theta_1 + \theta_2) \tag{2}$$

En dérivant (1) et (2), on obtient les vitesses $\dot{X}_e$ et $\dot{Y}_e$:

$$\dot{X}_e = -l_1(\dot{\theta}_1 sin(\theta + \theta_1)) - l_2(\dot{\theta_1} + \dot{\theta_2}) sin(\theta + \theta_1 + \theta_2) + V cos(\theta) \tag{3}$$

$$\dot{Y}_e = l_1(\dot{\theta}_1 cos(\theta + \theta_1)) + l_2(\dot{\theta_1} + \dot{\theta_2}) cos(\theta + \theta_1 + \theta_2) + V sin(\theta) \tag{4}$$

On peut alors écrire ces équations sous la forme d'un produit matriciel :

$$\begin{pmatrix} \dot{X}_e \\ \dot{Y}_e \\ \dot{\theta} \end{pmatrix} = \begin{pmatrix} a_1 & b_1 & cos(\theta) \\ a_2 & b_2 & sin(\theta) \\ 0 & 0 & \frac{tan(\psi)}{D} \end{pmatrix} \begin{pmatrix} \dot{\theta_1} \\ \dot{\theta_2} \\ V \end{pmatrix} \tag{5}$$

avec $a_1 = -l_1 sin(\theta+\theta_1) - l_2 sin(\theta+\theta_1+\theta_2)$, $a_2 = -l_2 sin(\theta+\theta_1+\theta_2)$, $b_1 = l_1 cos(\theta+\theta_1) + l_2 cos(\theta+\theta_1+\theta_2)$ et $b_2 = l_2 cos(\theta+\theta_1+\theta_2)$.

On réalise ensuite les approximations suivantes : $\dot{X}_e \approx X_{ed} - X_e$, $\dot{Y}_e \approx Y_{ed} - Y_e$, $V_d = \sqrt{\dot{X_e}^2 + \dot{Y_e}^2}$, $\theta_d(t) \approx arctan\left(\frac{\dot{Y}_e}{\dot{X}_e}\right)$, $\psi_d(t) = \theta_d - \theta$. D'après la forme matricielle (5), les valeurs de $\dot{\theta_1}$ et $\dot{\theta_2}$ sont déterminées par :

$$\begin{pmatrix} \dot{\theta_1} \\ \dot{\theta_2} \end{pmatrix} = \begin{pmatrix} a_1 & b_1 \\ a_2 & b_2 \end{pmatrix}^{-1} \begin{pmatrix} \dot{X}_e \\ \dot{Y}_e \end{pmatrix} \tag{6}$$

Les valeurs de $\dot{\theta_1}$, $\dot{\theta_2}$, V_d et ψ_d ainsi calculées sont alors transférées aux moteurs du robot pour qu'il actionne chacune des parties associées à ces variables.

Il est bon de remarquer que l'orientation θ_d est calculée de façon à ce qu'elle oriente le robot dans la direction pointant vers le point $(X_d\ Y_d)^T$. On la considère comme un des angles que formerait le triangle rectangle dont les deux côtés adjacents sont de longueur $\dot{Y}_e$ et $\dot{X}_e$.

L'angle de direction ψ_d du véhicule (angle de la roue) est lui déterminé en faisant la différence entre l'angle d'orientation désiré θ_d et l'angle d'orientation actuel θ. Il faut donc faire attention à ce que cette différence ne soit pas trop grande pour que l'angle de la roue ne diverge pas et devienne incontrolable .

2 Expérimentation

Pour représenter toute la modélisation du robot sous Matlab, un programme Simulink est réalisé (c.f. Figure 2). Le bloc `Controle` permet de déterminer les valeurs de $\dot{\theta_1}$, $\dot{\theta_2}$ V_d et $\dot{\theta}$ comme définies dans les équations précédentes. Le bloc `Vehicule` sert lui à recalculer les nouvelles valeurs de $\dot{X}_e$ et $\dot{Y}_e$ (voir équations 3 et 4). Ces blocs sont détaillés dans l'annexe 3.2.

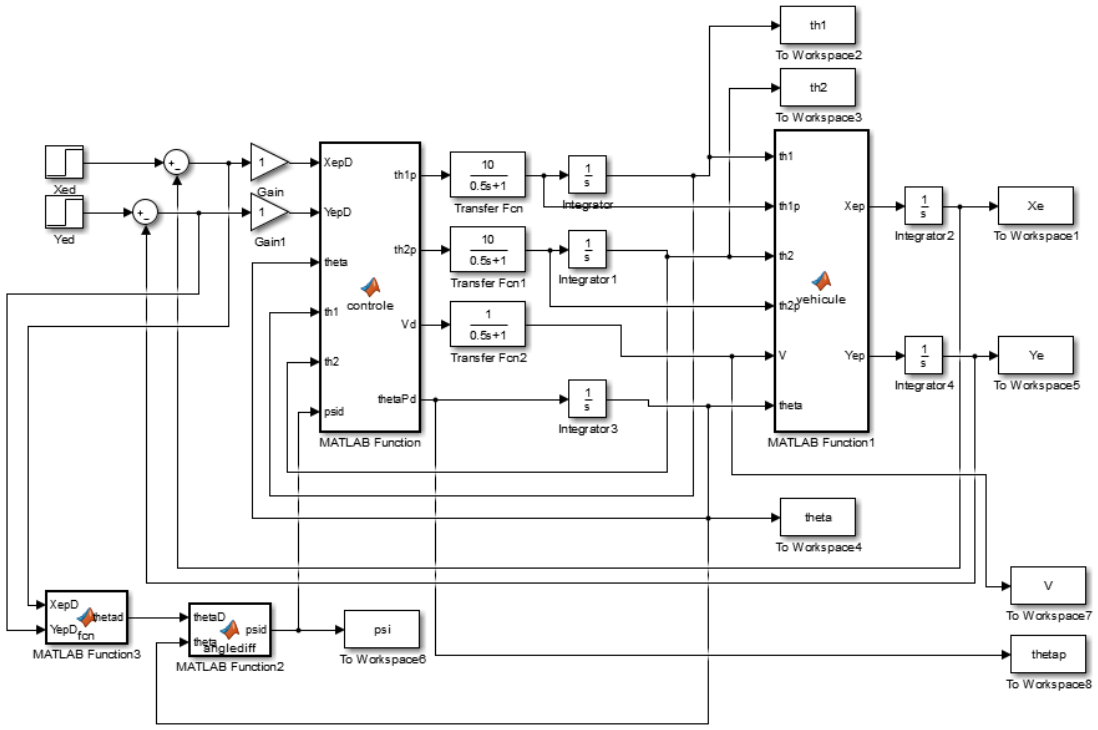


Figure 2: Simulink permettant la commande du robot

Dans les cas tests qui vont suivre, on fait en sorte que le robot ait toujours la même position initiale (voir Figure 3). La position de la base du bras est $(X_r\ Y_r)^T = (1\ 1)^T$, tandis que les angles qui définissent son orientation sont $\theta_1(0) = \theta_2(0) = \theta(0) = \frac{\pi}{3}$rd. On impose aussi que le robot commence sa manœuvre en étant arrêté, i.e. avec $V(0) = 0$ m/s.

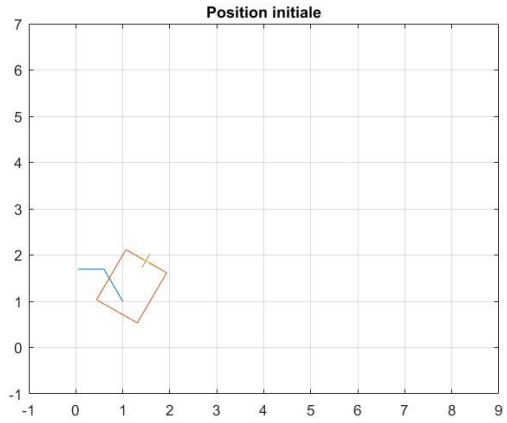
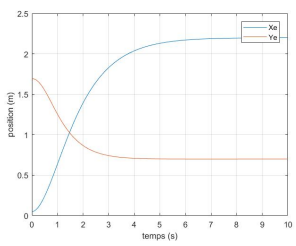


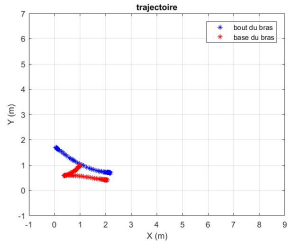
Figure 3: Position initiale du robot

2.1 Cas n°1 : $(X_{ed}\ Y_{ed})^T = (2.2\ 0.7)^T$

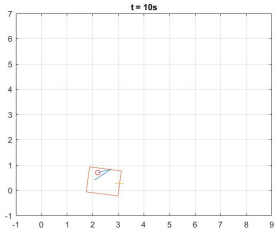
Pour ce premier cas, on souhaite que le bout du bras du robot atteigne la position $(X_{ed}\ Y_{ed})^T = (2.2\ 0.7)^T$. En lançant le programme défini précédemment, on obtient les courbes suivantes :



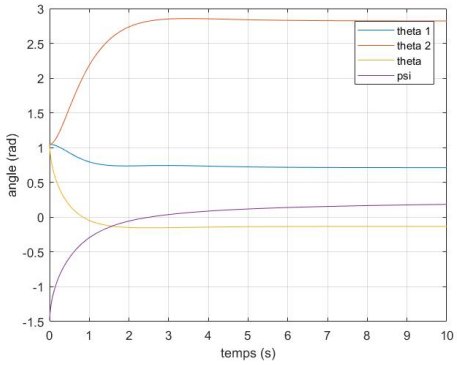
(a) Evolution de X_e et Y_e



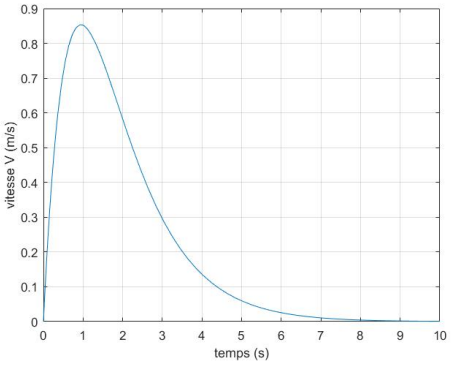
(b) Trajectoires



(c) Position finale du robot



(d) Angles du robot



(e) Vitesse du véhicule

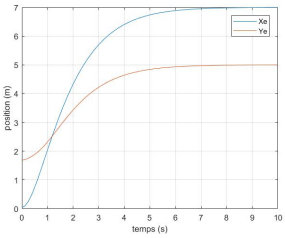
Figure 4: Courbes d'évolutions du robot pour le cas n°1

Les courbes de trajectoires montrent que le véhicule a dû reculer pour pouvoir bien s'orienter face à l'objectif tandis que le bout du bras suit une courbe assez lisse. Plus précisément, le véhicule n'a pas reculé, mais il a effectué une manœuvre qui imposait que sa roue pointe dans le sens arrière ($\psi_d < -\frac{\pi}{2}$). Il a ensuite avancé jusqu'à se positionner assez proche du point désiré pour correctement placer le bout du bras dessus.

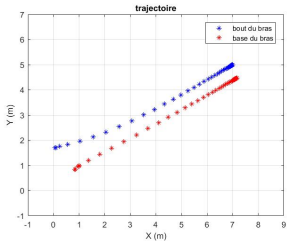
La vitesse évolue en augmentant rapidement au démarrage puis en décroîssant lentement jusqu'à être nulle. Le point correspondant au début de la réduction de la vitesse peut être mis en lien avec l'évolution de l'angle d'orientation du véhicule. Une fois que la voiture est bien orientée par rapport à l'objectif à atteindre, i.e. quand θ est quasiment constant, la vitesse de déplacement commence à diminuer. Cela s'explique par le fait que la vitesse est directement reliée avec la distance séparant la position actuelle et le point à atteindre. Ainsi, si la voiture est orientée directement vers l'objectif, cette distance diminuera forcément avec le temps.

2.2 Cas n°2 : $(X_{ed}\ Y_{ed})^T = (7\ 5)^T$

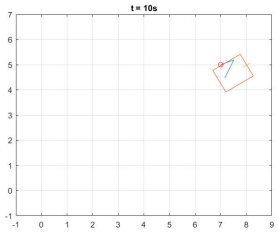
Le second cas test impose que le point à atteindre $(X_{ed}\ Y_{ed})^T$ soit situé à $(7\ 5)^T$, tout en gardant les mêmes conditions initiales. Les courbes définissant l'évolution de tous les paramètres du robot peuvent alors être calculées :



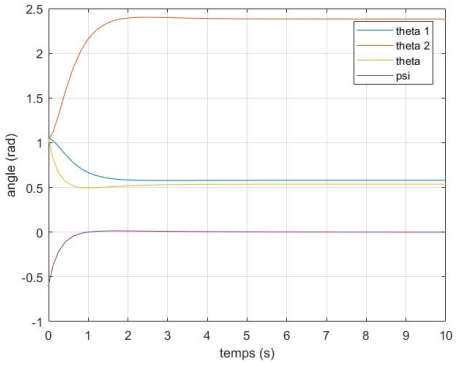
(a) Evolution de X_e et Y_e



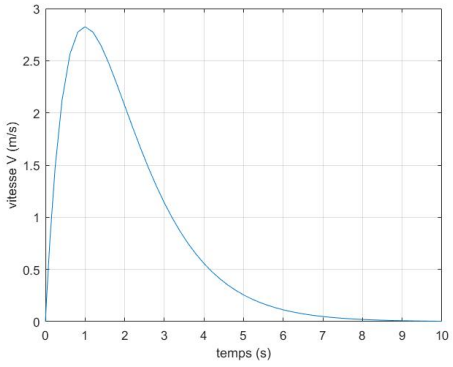
(b) Trajectoires



(c) Position finale du robot



(d) Angles du robot



(e) Vitesse du véhicule

Figure 5: Courbes d'évolutions du robot pour le cas n°2

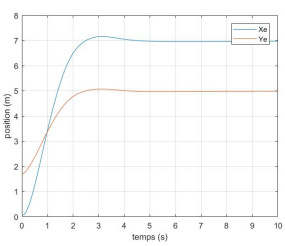
Ici, la position désirée est telle que le robot n'a pas besoin d'effectuer de manœuvre "*spéciale*", il lui suffit juste tourner de quelques degrés vers sa droite pour avoir l'objectif dans sa ligne de mire. Le fait de ne pas avoir à faire de manœuvre joue également sur la vitesse à laquelle le robot atteint l'objectif, vu que l'orientation initiale du véhicule est plus proche de son orientation finale. Cela se répercute aussi sur les angles qui convergent plus vite vers leurs valeurs finales. L'orientation rapide du véhicule et du bras au début du déplacement permet d'avoir une trajectoire quasi-rectiligne par la suite. Le robot n'aura donc plus qu'à se déplacer dans son sens avant pour atteindre l'objectif.

La courbe de la vitesse de déplacement a la même forme que dans le cas précédent, seules les valeurs qu'elle prend sont différentes. Elles sont plus élevées du fait que le point à atteindre est cette fois-ci plus éloigné du point de départ.

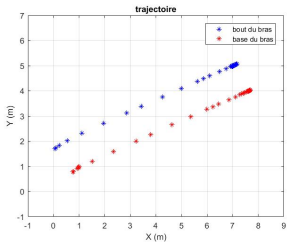
2.3 Modification de la commande

La méthode qui a été utilisée précédemment consistait à atteindre la position désirée tout en réglant en même temps la disposition du bras et le positionnement du véhicule. Une autre manière de procéder serait de faire en sorte que le bras ne bouge que lorsque le point à atteindre est dans le domaine atteignable du bras du robot. La seule modification à apporter dans le programme est qu'il faut rajouter une fonction Matlab appelée `Test_dist` après chaque sortie du bloc `Controle`. Cette fonction va calculer la distance entre la position désirée $(X_{ed}\ Y_{ed})^T$ et la position actuelle $(X_r\ Y_r)^T$ de la base du bras du robot. Si cette distance est supérieure au rayon du cercle que peut former le bras en étant totalement déplié, on choisit soit de garder la valeur de la variable étudiée, soit de la placer à 0. Cette fonction est détaillée dans l'annexe 3.2.3.

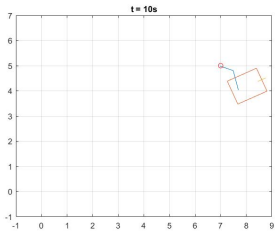
En reprenant la position désirée $(X_{ed}\ Y_{ed})^T = (7\ 5)^T$ du cas n°2 précédent, on obtient les courbes suivantes :



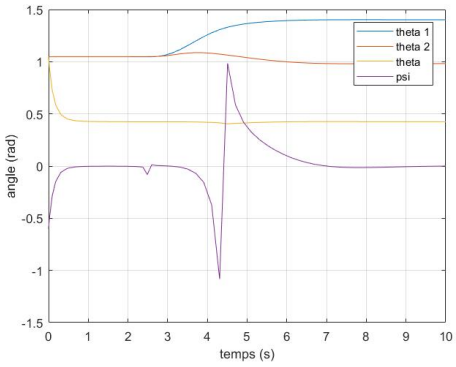
(a) Evolution de X_e et Y_e



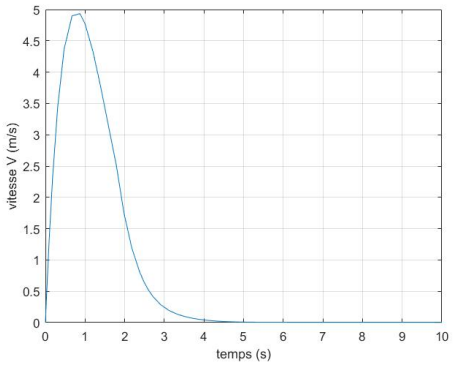
(b) Trajectoires



(c) Position finale du robot



(d) Angles du robot



(e) Vitesse du véhicule

Figure 6: Courbes d'évolutions du robot

Comme attendu, les angles θ_1 et θ_2 ne varient pas au démarrage du robot, celui-ci étant trop éloigné de la position à atteindre. Aux alentours de 3s, le robot semble s'être approché suffisamment: les angles qui jusqu'alors étaient constants commencent à s'activer pour faire en sorte que le bras soit bien positionné. C'est aussi à ce moment-là que le véhicule ne reçoit plus d'instructions concernant sa vitesse de déplacement ou sur son orientation. L'angle θ devient alors constant tandis que la vitesse diminue petit à petit. Le temps que la voiture s'arrête, le robot arrive à dépasser le point à atteindre (un peu après 4s), ce qui fait que la roue "*souhaite*" que le véhicule fasse demi-tour. Cela se remarque par l'angle d'orientation de la roue ψ_d qui s'affole à ce moment-là.

Le fait que le bras ne bouge qu'à partir d'un certain point peut se remarquer dans les tracés des trajectoires : contrairement au cas précédent, les trajectoires de la base et du bout du bras du robot sont quasiment identiques, ce qui indique que le bras ne bougeait pas pendant le déplacement.

De plus, il est important de noter que les valeurs finales des angles θ, θ_1 et θ_2 sont différentes par rapport au cas précédent : en effet, le programme cherche d'abord à optimiser le positionnement et l'orientation de la voiture avec θ et ψ_d avant de s'occuper du bras. Cela résulte dans le fait que l'angle θ final dans ce cas-ci est optimal si le bras ne bougeait pas, et vu que dès que le bras bouge cet angle reste fixe, la configuration finale garde cet angle-là.

3 Annexes

3.1 Programme principal

```
%dimensions robot
D = 1;
l1 = 0.8;
l2 = 0.55;
%position initiale
Xr0 = 1;
Yr0 = 1;
theta_0 = pi/3;
th1_0 = pi/3;
th2_0 = pi/3;
V0 = 0;
Xe0 = Xr0 + l1*cos(theta_0+th1_0) + l2*cos(theta_0+th1_0+th2_0);
Ye0 = Yr0 + l1*sin(theta_0+th1_0) + l2*sin(theta_0+th1_0+th2_0);
%a atteindre
Xd = 2.2;
Yd = 0.7;
%appel simulink
sim('simu')
%affichage du mouvement
figure;
for i=1:length(tout)
    %bras
    Xr = Xe(i) - l1*cos(theta(i)+th1(i)) - l2*cos(theta(i)+th1(i)+th2(i));
    Yr = Ye(i) - l1*sin(theta(i)+th1(i)) - l2*sin(theta(i)+th1(i)+th2(i));
    Xa = Xr + l1*cos(theta(i)+th1(i));
    Ya = Yr + l1*sin(theta(i)+th1(i));
    P = [Xr Yr; Xa Ya; Xe(i) Ye(i)];
    %voiture
    XA = Xr + D*cos(theta(i)) - (D/2)*sin(theta(i));
    YA = Yr + D*sin(theta(i)) + (D/2)*cos(theta(i));
    XB = Xr + D*cos(theta(i)) + (D/2)*sin(theta(i));
    YB = Yr + D*sin(theta(i)) - (D/2)*cos(theta(i));
    XC = Xr - (D/4)*cos(theta(i)) + (D/2)*sin(theta(i));
    YC = Yr - (D/4)*sin(theta(i)) - (D/2)*cos(theta(i));
    XD = Xr - (D/4)*cos(theta(i)) - (D/2)*sin(theta(i));
    YD = Yr - (D/4)*sin(theta(i)) + (D/2)*cos(theta(i));
    C = [XA YA; XB YB; XC YC; XD YD; XA YA];
    %roue
    XR1 = Xr + D*cos(theta(i)) + (D/6)*cos(theta(i)+psi(i));
    YR1 = Yr + D*sin(theta(i)) + (D/6)*sin(theta(i)+psi(i));
    XR2 = Xr + D*cos(theta(i)) - (D/6)*cos(theta(i)+psi(i));
    YR2 = Yr + D*sin(theta(i)) - (D/6)*sin(theta(i)+psi(i));
    R = [XR1 YR1; XR2 YR2];
    %affichage
    plot(P(:,1),P(:,2),C(:,1),C(:,2),R(:,1),R(:,2),Xd,Yd,'or');
    title(['t = ' num2str(tout(i)) 's']);
    grid on;
    axis([-1 9 -1 7]);
    pause(0.01);
end
```

3.2 Blocs Simulink

3.2.1 Bloc `Controle`

```
function [th1p,th2p,Vd,thetaPd] = controle(XepD,YepD,theta,th1,th2,psid)

Vd = sqrt(XepD^2+YepD^2);
thetaPd = Vd*tan(psid)/D;

A = [-l1*sin(theta+th1)-l2*sin(theta+th1+th2) -l2*sin(theta+th1+th2)
    l1*cos(theta+th1)+l2*cos(theta+th1+th2) l2*cos(theta+th1+th2) ];

res = inv(A)*[XepD;YepD];
th1p = res(1); th2p = res(2);
end
```

3.2.2 Bloc `Vehicule`

```
function [Xep,Yep] = vehicule(th1,th1p,th2,th2p,V,theta)

Xep = (-l1*sin(theta+th1)-l2*sin(theta+th1+th2))*th1p + ...
     (-l2*sin(theta+th1+th2))*th2p + V*cos(theta);
Yep = (l1*cos(theta+th1)+l2*cos(theta+th1+th2))*th1p + ...
     (l2*cos(theta+th1+th2))*th2p + V*sin(theta);

end
```

3.2.3 Bloc `Test_dist`

```
function var_sortie = test_dist(var_entree,Xe,Ye,th1,th2,theta,Xd,Yd)
Xr = Xe - l1*cos(theta+th1) - l2*cos(theta+th1+th2);
Yr = Ye - l1*sin(theta+th1) - l2*sin(theta+th1+th2);
dist = (Xr-Xd)^2+(Yr-Yd)^2;
if dist > l1^2+l2^2
    var_sortie = 0;
else
    var_sortie = var_entree;
end
```