Commande avancée
Examen

Max Sonzogni

31 janvier 2019

1 Modélisation

On choisit de modéliser un robot mobile surmonté d'un bras pilotable, avec les longueurs fixes $D = 1.2$ m, $l_1 = 0.6$ m et $l_2 = 0.75$ m (c.f. Figure 1).

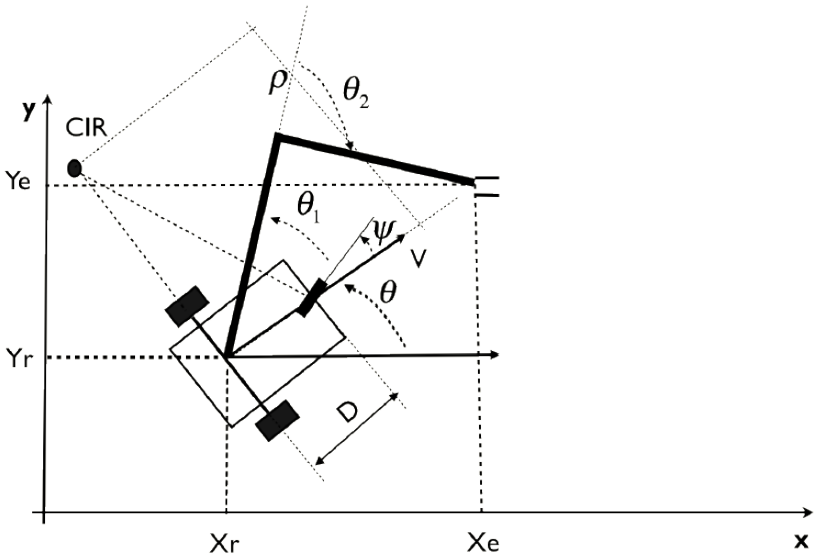


Figure 1: Représentation du manipulateur mobile

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) et en y rajoutant la vitesse de déplacement V de la voiture, 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 :

$$\begin{pmatrix} \dot{X}_e \\ \dot{Y}_e \end{pmatrix} = \begin{pmatrix} a_1 & b_1 & cos(\theta) \\ a_2 & b_2 & sin(\theta) \end{pmatrix} \begin{pmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \\ V \end{pmatrix} \tag{5}$$

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 : $\dot{X}_e \approx X_{ed} - X_e$, $\dot{Y}_e \approx Y_{ed} - Y_e$, $V_d = \sqrt{\dot{X_e}^2 + \dot{Y_e}^2}$, $\theta_d(t) \approx arctan\left(\frac{\dot{Y_e}}{\dot{X_e}}\right)$, $\psi_d(t) = \theta_d - \theta$ et $\dot{\theta} = \frac{tan(\psi_d)}{D} V_d$. 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. De plus, il ne faut pas que ψ_d soit égal à $\frac{\pi}{2} + k\pi$ ($k \in \mathbf{Z}$), c'est à dire que la roue n'indique pas à la voiture d'effectuer un déplacement latéral (différence entre θ_d et θ). Ce serait d'une part impossible physiquement, et cela donnerait à la vitesse angulaire $\dot{\theta}$ une valeur infinie, imposant alors une rotation quasi-immédiate de la voiture.

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.

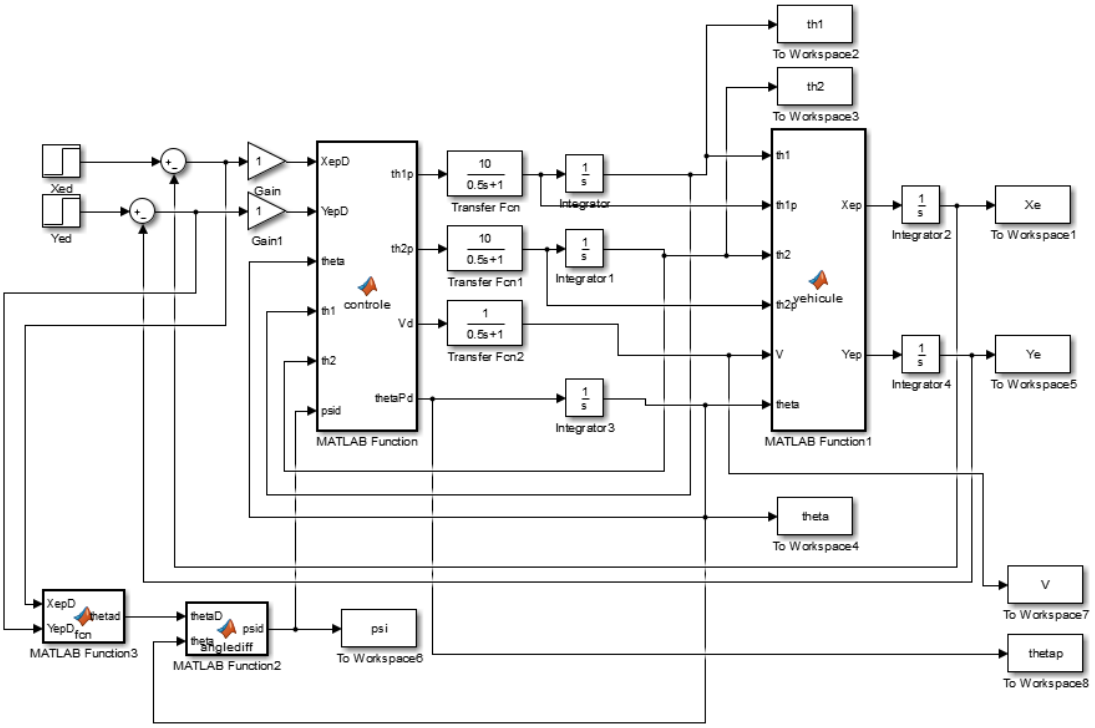


Figure 2: Simulink permettant la commande du robot

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 $\theta_1(0) = -\frac{\pi}{3}$, $\theta_2(0) = \frac{\pi}{3}$ et $\theta(0) = -\frac{\pi}{6}$. On impose aussi que le robot commence sa manœuvre en étant arrêté, i.e. avec $V(0) = 0$ m/s.

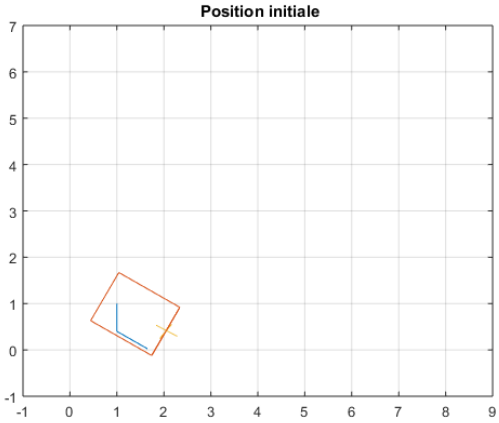
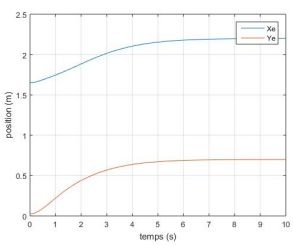


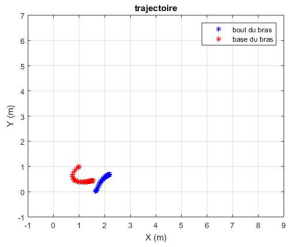
Figure 3: Position initiale du robot

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

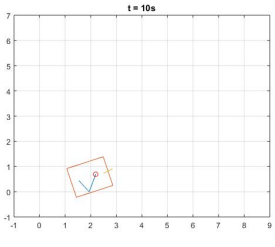
Pour ce premier cas, on souhaite que le bout du bras du robot atteigne la position $(X_{ed}\ Y_{ed})^T = (2.2\ 0.7)^T$. En lançant le programme défini précédemment, on obtient les courbes suivantes :



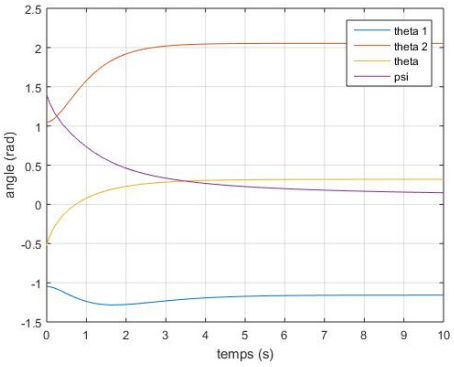
(a) Evolution de X_e et Y_e



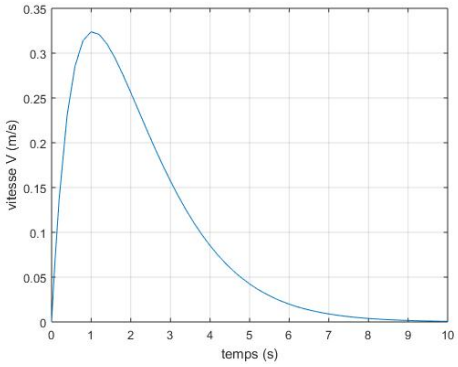
(b) Trajectoires



(c) Position finale du robot



(d) Angles du robot



(e) Vitesse du véhicule

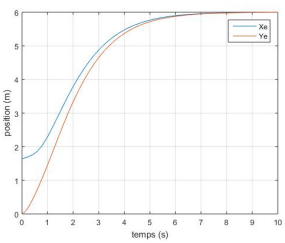
Figure 4: Courbes d'évolutions du robot pour le cas n°1

Les courbes de trajectoires montrent que la voiture a dû effectuer une manœuvre serrée pour pouvoir bien atteindre le point désiré avec le bout du bras. En effet, l'angle ψ est presque égal à $\frac{\pi}{2}$ au démarrage, ce qui indique que le véhicule a pris un virage plutôt court pour pouvoir bien s'orienter. Cette manœuvre a été d'autant plus facilitée par la faible vitesse que la voiture avait tout le long du déplacement, permettant ainsi un placement en précision.

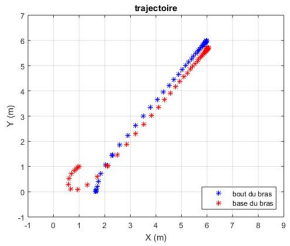
Le fait qu'une manœuvre ait lieu n'interfère quasiment pas dans la convergence des différents angles : il tendent tous vers leur valeur finale en suivant une fonction de la forme $y = ln(x)$. Seul l'angle θ_1 décroît puis recroît pendant le déplacement : cela vient du fait que le véhicule s'oriente plus vite que le bras, et donc le bras doit s'adapter aux variations imposées par la voiture.

2.2 Cas n°2 : $(X_{ed}\ Y_{ed})^T = (6\ 6)^T$

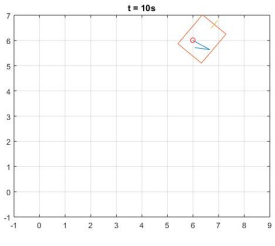
Le second cas test impose que le point à atteindre $(X_{ed}\ Y_{ed})^T$ soit situé à $(6\ 6)^T$, tout en gardant les mêmes conditions initiales. Les courbes définissant l'évolution de tous les paramètres du robot peuvent alors être calculées :



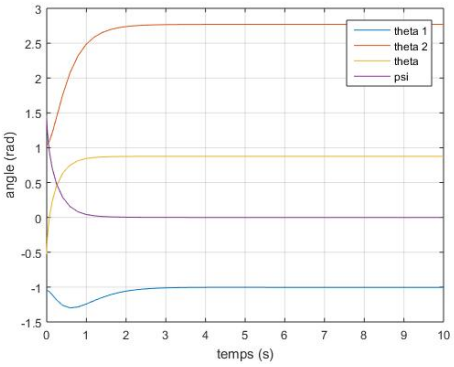
(a) Evolution de X_e et Y_e



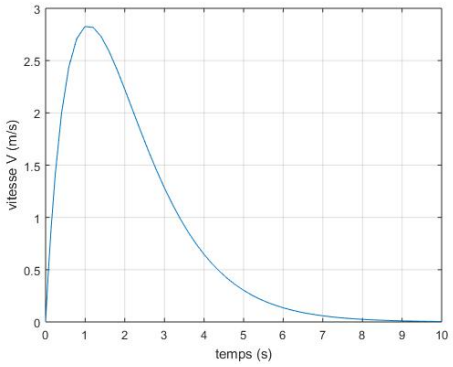
(b) Trajectoires



(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

Tout comme dans le cas précédent, la voiture s'oriente d'abord rapidement afin d'avoir le point à atteindre bien en face de lui. Pour ce faire, l'angle ψ de sa roue prend une valeur proche de $\frac{\pi}{2}$ afin de se tourner plus vite, puis diminue jusqu'à atteindre 0 lorsque l'objectif est bien en face. Il ne lui reste plus alors qu'à s'avancer tout droit pour toucher avec le bout du bras le point désiré.

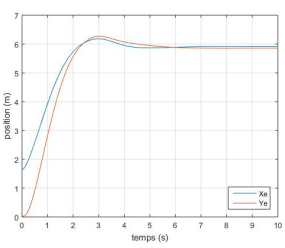
Pendant cette manœuvre, les angles d'orientation du bras ne restent pas fixes : ils commencent déjà à s'orienter pour faire en sorte que le robot n'ait plus qu'à avancer pour toucher l'objectif. Ainsi, les trajectoires de la base et du bout du bras sont quasi-rectilignes une fois la voiture bien orientée.

La courbe de la vitesse de déplacement a la même forme que dans le cas précédent : elle croît au démarrage jusqu'à atteindre son maximum lorsque la voiture est bien orientée par rapport au point à atteindre. A partir de là, elle va décroître lentement au fur et à mesure de l'avancement. La seule différence avec la courbe du cas précédent est que les valeurs prises sont plus élevées, du fait que le point à atteindre est cette fois-ci plus éloigné du point de départ.

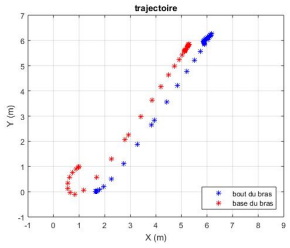
2.3 Modification de la commande

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.

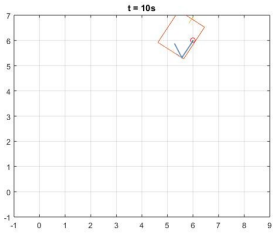
En reprenant la position désirée $(X_{ed}\ Y_{ed})^T = (6\ 6)^T$ du cas n°2 précédent, on obtient les courbes suivantes :



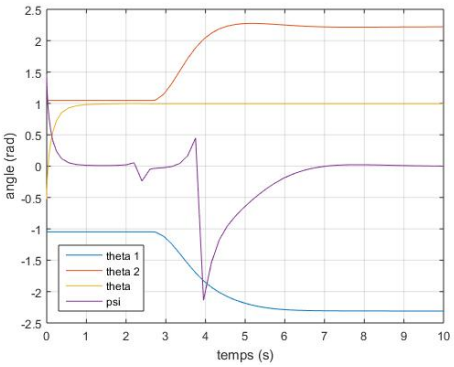
(a) Evolution de X_e et Y_e



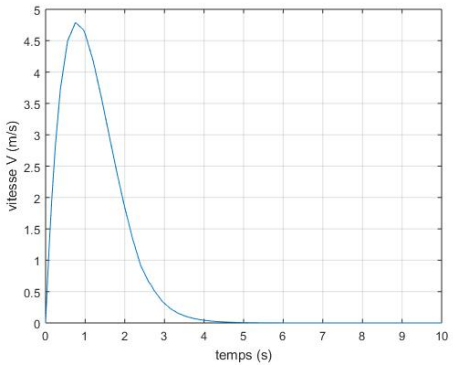
(b) Trajectoires



(c) Position finale du robot



(d) Angles du robot



(e) Vitesse du véhicule

Figure 6: Courbes d'évolutions du robot

Comme espéré, les angles θ_1 et θ_2 restent fixes au début du déplacement, car le point à atteindre est encore trop éloigné de la base du bras. C'est un peu avant la 3^e seconde que le robot est suffisament proche pour que l'ordre des choses s'inverse : la voiture ne reçoit plus d'ordres sur sa vitesse qui va alors décroître rapidement, et le bras va commencer à s'orienter de façon à atteindre le point voulu.

Cependant, avec la vitesse accumulée, le véhicule ne s'arrêtera pas immédiatement. Il va dépasser l'objectif avec sa roue, chose visible sur le graphe des angles : en effet, lorsque l'angle entre l'axe d'orientation de la voiture et l'axe roue-objectif est de $\frac{\pi}{2}$, il se crée une singularité dans le calcul de ψ_d. Cet angle est déterminé dans le bloc fonction `Anglediff` (voir Annexe 3.2.4) grâce aux angles θ_d (orientation désirée) et θ (orientation actuelle). Le calcul de θ_d implique que le point à atteindre soit situé dans l'espace frontal de la voiture. Ainsi, quand la voiture dépasse le point objectif, la valeur de θ_d est telle que le calcul de ψ_d donne une discontinuité de la courbe, et donc une singularité.

Il peut être 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 dans un premier temps à optimiser le positionnement et l'orientation de la voiture en faisant varier seulement θ et ψ_d. Le temps que la voiture soit suffisamment proche de l'objectif pour que le bras commence à bouger, l'angle θ est quasiment fixe, et vu qu'il ne recevra pas de nouvelles "*instructions*" une fois l'objectif assez proche, sa valeur ne changera plus. Ce sont donc les angles θ_1 et θ_2 qui vont devoir s'accomoder de cette contrainte afin de pouvoir positionner correctement le bout du bras sur le point désiré.

On peut remarquer que tout le long de la première phase de déplacement (quand $dist < l_1^2 + l_2^2$), les trajectoires du bout et de la base du bras sont identiques à une translation près. Cela est dû au fait que les angles du bras restent fixes pendant cette période, et donc la position du bras reste la même.

3 Annexes

3.1 Programme principal

```
%dimensions robot
D = 1.2;
l1 = 0.6;
l2 = 0.75;
%position initiale
Xr0 = 1;
Yr0 = 1;
theta_0 = -pi/6;
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 = 6;
Yd = 6;
%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 % ou dist < l1^2+l2^2
    var_sortie = 0;
else
    var_sortie = var_entree;
end
```

3.2.4 Bloc `Anglediff`

```
function y = anglediff(u,v)
if u<-pi
    u = mod(u,-pi);
end
if u>pi
    u = mod(u,pi);
end
if v<-pi
    v = mod(v,-pi);
end
if v>pi
    v = mod(v,pi);
end
y = u-v;
if y<-pi
    y = mod(y,-pi);
end
if y>pi
    y = mod(y,pi);
end
end
```