ÉCOLE NATIONALE SUPÉRIEURE DE TECHNIQUES AVANCÉES

ROB316 Planification et contrôle

TP - PID

Luiz Henrique MARQUES GONÇALVES

Palaiseau, France 2025

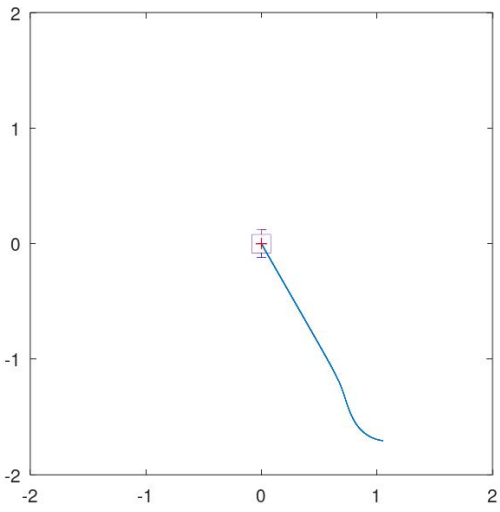
1. Contrôle de modèle unicycle

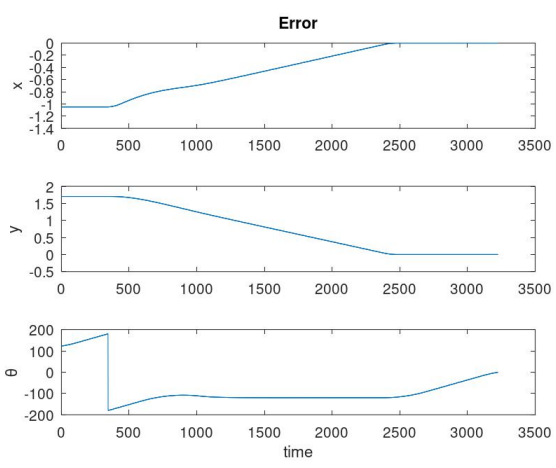
1.1. Question 1 - contrôle unicycle

Pour le contrôle du monocycle, la fonction suivante a été implémentée pour appliquer les commandes indiquées : une première correction proportionnelle de l'angle d'approche, suivie d'une correction proportionnelle de la vitesse angulaire et de l'avancement simultané, et enfin une correction finale de l'angle avec une correction proportionnelle. Dans toutes les questions, l'ajustement des gains a été réalisé empiriquement afin de minimiser les oscillations et d'assurer une convergence rapide.

```
function [ u ] = UnicycleToPoseControl( xTrue,xGoal )

alpha = atan2(xGoal(2) - xTrue(2), xGoal(1) - xTrue(1)) - xTrue(3);
alpha = AngleWrap(alpha);
rho = sqrt((xGoal(1) - xTrue(1))^2 + (xGoal(2) - xTrue(2))^2);

Krho = 30;
Kalpha = 10;
Kbeta = 10;
alpha_max = pi/3;

v = Krho * rho;

if(rho > 0.05)
  omega = Kalpha * alpha;
  if(abs(alpha) > alpha_max)
    v = 0;
  end
else

  beta = xGoal(3) - xTrue(3);
  omega = Kbeta * beta;
  v = 0;

end

u = [v; omega];

end
```

Voici la carte et le graphique des erreurs obtenus. La convergence vers la pose désirée est atteinte en moins de 4000 boucles. De plus, l'exécution de `UnicycleToPoseBenchmark` a retourné un score de 2055.7619, proche de la valeur attendue.





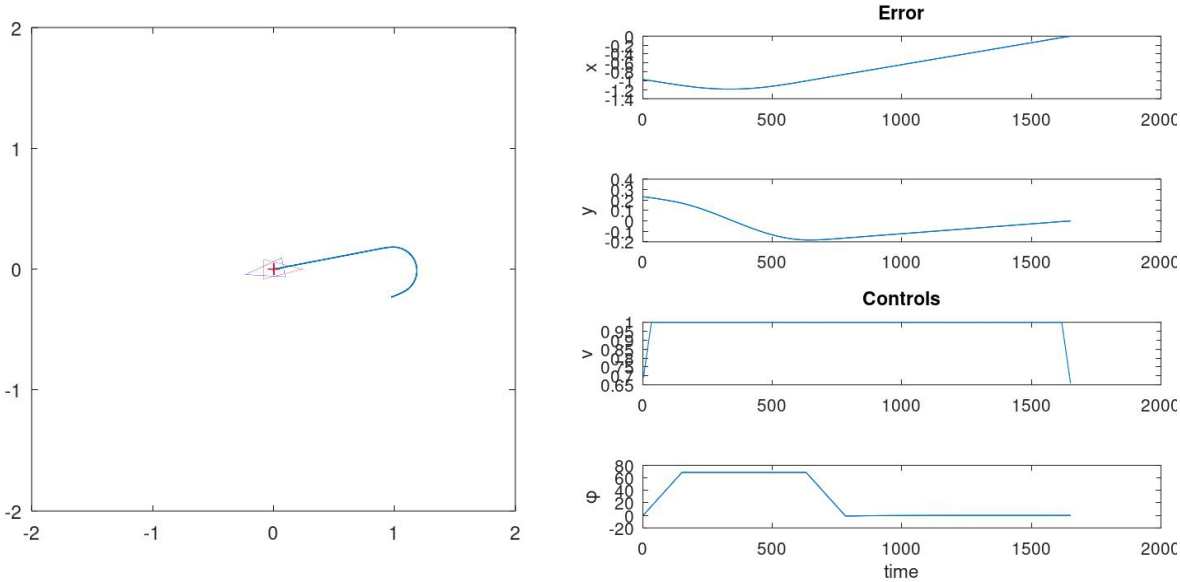
2. Contrôle de modèle bicyclette

2.1. Question 2 - contrôle de bicyclette vers un point

Pour le contrôle d'une bicyclette vers un point, un contrôleur proportionnel à la distance suffit pour régler la vitesse d'avancement, et un contrôleur proportionnel à l'angle d'approche nécessaire pour la vitesse angulaire.

```
function [ u ] = BicycleToPointControl( xTrue,xGoal )
%Computes a control to reach a pose for bicycle
%   xTrue is the robot current pose : [ x y theta ]'
%   xGoal is the goal point
%   u is the control : [v phi]'

alpha = atan2(xGoal(2) - xTrue(2), xGoal(1) - xTrue(1)) - xTrue(3);
alpha = AngleWrap(alpha);
rho = sqrt((xGoal(1) - xTrue(1))^2 + (xGoal(2) - xTrue(2))^2);

Krho = 30;
Kalpha = 5;

v = Krho * rho;
phi = Kalpha * alpha;

u = [v; phi];

end
```

Voici la carte et le graphique des erreurs obtenus. La convergence vers la pose désirée est atteinte en moins de 2000 boucles. De plus, l'exécution de `BicycleToPointBenchmark` a retourné un score de 1354.9048, au-dessus de la limite attendue.



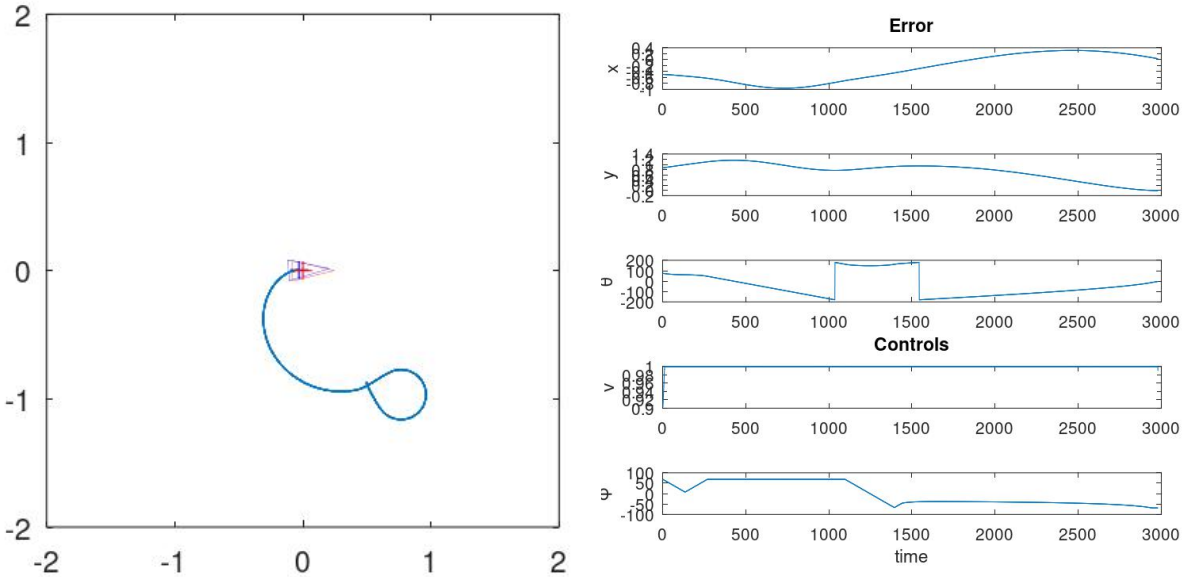
Ici, le contrôle doit se différencier de ce qui a été fait pour l'unicycle en raison de la dépendance entre la vitesse d'avancement et celle de rotation pour la bicyclette.

2.2. Question 3 - contrôle de bicyclette vers une position

Pour le contrôle de la bicyclette vers une position en respectant une orientation désirée, le contrôle angulaire est une combinaison entre une correction proportionnelle à l'angle d'avancement nécessaire, réduite par une correction proportionnelle à un angle relatif à la position finale. Le premier (alpha) tente de mettre la bicyclette dans la position désirée, tandis que le second (beta) force la bicyclette à adopter une position qui permet d'arriver à destination avec le bon angle.

```
function [ u ] = BicycleToPoseControl( xTrue,xGoal )
%Computes a control to reach a pose for bicycle
%   xTrue is the robot current pose : [ x y theta ]'
%   xGoal is the goal point
%   u is the control : [v phi]'

alpha = atan2(xGoal(2) - xTrue(2), xGoal(1) - xTrue(1)) - xTrue(3);
alpha = AngleWrap(alpha);
rho = sqrt((xGoal(1) - xTrue(1))^2 + (xGoal(2) - xTrue(2))^2);
beta = xGoal(3) - atan2(xGoal(2) - xTrue(2), xGoal(1) - xTrue(1));

Krho = 30;
Kalpha = 5;
Kbeta = -3;

v = Krho * rho;
phi = Kalpha * alpha + Kbeta * beta;

u = [v; phi];
end
```

Voici la carte et le graphique des erreurs obtenus. La convergence vers la pose désirée est atteinte en environ de 3000 boucles. De plus, l'exécution de `BicycleToPoseBenchmark` a retourné un score de 1706.1905, au-dessus de la limite attendue.



2.3. Question 4 - contrôle de bicyclette selon un chemin

Pour le contrôle selon un chemin, on profite de la fonction développée dans la question 2, qui conduit la bicyclette vers un point. L'approche adoptée est de représenter le chemin comme une séquence successive de points à atteindre. Dans le code suivant, cette idée a été implémentée : la bicyclette suit le point but actuel et, lorsqu'elle s'en rapproche, la fonction `UpdateGoal` met à jour le but en choisissant le point suivant.

```
function [ u ] = BicycleToPathControl( xTrue, Path )

% Store the current goal and step along the path
persistent current_goal;
persistent current_step;

% Initialize the current goal and step
if isempty(current_goal)
  current_goal = Path(:,1);
  current_step = 2;
end

% Distance from the current robot position to the current goal
distance = sqrt((current_goal(1) - xTrue(1))^2 + (current_goal(2) - xTrue(2))^2);

% If the robot is close enough to the current goal update the goal
if(distance < 0.3)
  [current_goal, current_step] = UpdateGoal(xTrue, current_goal, current_step, Path
    );
end

% Compute the control inputs to reach the current goal
u = BicycleToPointControl(xTrue, current_goal);

end


function [goal, step] = UpdateGoal( xTrue, current_goal, current_step, Path )

% Calculate the direction vector of the current path segment
current_line_start = Path(:, current_step - 1);
current_line_end = Path(:, current_step);
line_length = sqrt((current_line_start(1) - current_line_end(1))^2 + (
    current_line_start(2) - current_line_end(2))^2);
current_dir = (current_line_end - current_line_start) / line_length;

% New goal point along the current path segment
step_size = 0.3;
goal = current_goal + step_size * current_dir;

% If the robot is close to the current segment's end, update the goal and segment
distance_to_line_end = sqrt((current_line_end(1) - goal(1))^2 + (current_line_end
    (2) - goal(2))^2);
if(distance_to_line_end < 0.3 && current_step ~= size(Path, 2))
  goal = current_line_end;  % Set the goal to the segment's end
  current_step = current_step + 1;  % Move to the next path segment
end

step = current_step;

end
```

Voici la carte et le graphique des erreurs obtenues. La convergence vers le chemin est satisfaisante, avec des divergences dans les courbes accentuées, dues aux limitations du rayon de courbure de la bicyclette. De plus, l'exécution de `BicycleToPath` a retourné une erreur totale de 372.5945, inférieure à la limite acceptée.