Fundamentals of Mobile Robots

Tampere University
Tampere University of Applied Sciences

Exercise Report Number: 6

by:

Dong Le (151057054)

Date: 24.04.2023

THIS PAGE LEFT BLANK INTENTIONALLY

Dong Le

List of Terminology

u_gtg	Control input go-to-goal
u_avo	Control input avoidance
u_wf_cc	Control input wall-following counter clockwise
u_wf_cw	Control input wall-following clockwise
eps	Epsilon
d_safe	Area around obstacle
x_o_1 and x_o_2	Two sensors are chosen to calculate u_wf
v_wf_c and v_wf_cc	Two vectors of wall-following based on u_avo, the purpose to check condition only

Dong Le

Table of Contents

Dong Le

Introduction

This exercise is based on the concepts proportional go-to-goal, reference robot, and hard switching.

Proportional control is a control system technology based on a response in proportion to the difference between what is set as a desired process variable (or set point) and the current value of the variable.

Reference robot is to tracking the moving goal.

Hard switching is a method that changing the controller based on conditions and controller's state.

The purpose of this exercise is to learn how to design and implement controller using different methods to get the robot to reach the desired goal position and avoiding obstacles.

In the first problem, implement go-to-goal controller.
In the second problem, implement go-to-goal with moving goal.
In the third problem, use sensor reading to switch between methods.

Dong Le

Methodology

Problem 1

The theory of proportional control was used to design control input to reach the goal using different methods. The theory of unicycle mobile robot.

First, the initial robot state and desired state are defined. Then using zero matrices to store data from the process to plot after all.

For the first question, the control input was computed straightforwardly, the current control input was calculated based on the distance from the robot and the goal, and the error between angles. Ensure the error between angles in range -pi to pi.

For the second question, design the caster point to compute the unicycle robot as an omnidirectional robot, then calculate the control input as omnidirectional robot. Next, transfer the control input for unicycle one. One thing to consider is to design the caster to the robot center big enough to avoid omega getting too large which over the rotational speed of wheels.

Finally, a robot state at that time was made by adding time step times control input into the previous robot state.

Some libraries are used in the python code such as Matplotlib, numpy, a custom library called “visualize_mobile_robot”, and ctypes library to pop-up messages about the robot exceed the wheel speed limit.

Problem 2

The theory of reference robot was used to track the moving goal, designing proportional control based on position or posture of the final goal, and feedforward theory. The theory of unicycle mobile robot.

For the first task, before computing the control input for the system, turn the unicycle robot to single integrator by making a caster point, so the control input was computed like the same as omnidirectional mobile robot. Then a control input was computed by using feedforward and the new caster position. Next, transfer the control input of the single integrator to the control input of the unicycle robot.

For the second one, a reference robot was made for the actual robot follow it. Firstly, compute the speed and omega of the reference robot. Then, the control input was calculated by taking the error between the reference robot and the actual robot. Tuning parameters to ensure the wheel speed does not exceed the maximum speed.

Some libraries are used in the python code such as Matplotlib, numpy, a custom library called “visualize_mobile_robot”, and ctypes library to pop-up messages about the robot exceed the wheel speed limit.

Dong Le

Problem 3

The theory of proportional control for orientation was used to reach the desired goal. The theory of avoidance and wall-following to find avoid obstacles. The theory of omnidirectional and unicycle mobile robot.

To compute the control input, divided into 4 parts: preparing, checking condition, checking control state, and computing control input base on control state.

Firstly preparing, turning the unicycle robot into a single integrator by making a caster point. Then, defined six conditions as false, find the minimum reading distance and its sensor coordinate. Found out the distance from robot to the minimum sensor, and from robot to the goal. Computed the u_gtg and u_avo based on those data.

Secondly, found out the sensors nearest to the obstacles, and compute u_wf_cw and u_wf_cc.

Thirdly, checking the conditions and control state in order to change the control state, and whenever the control state changed to “wf_c” or “wf_cc”, a new value of radius $\|x(t_s) - x^d\|$ was set to compute the second stage.

Finally, based on the control state from the previous step, choosing the control input.

A robot state at that time was made by adding time step times control input into the previous robot state. The speed of the robot satisfy the robot’s limitation by tuning parameters.

Some libraries are used in the python code such as Matplotlib, numpy, a custom library called “visualize_mobile_robot”, a custom library called “detect_obstacle”, a pre-made python code called “ex6p3_obstacle”, and ctypes library to pop-up messages about the robot exceed the wheel speed limit.

Dong Le

Results and Discussion

In this exercise, an unicycle mobile robot was used with the position (p_x, p_y)

$$x = \begin{bmatrix} p_x \\ p_y \\ \theta \end{bmatrix} \quad (1)$$

And the control input:

$$u = \begin{bmatrix} v \\ \omega \end{bmatrix} \quad (2)$$

The robot has specifications:

- Robot’s radius 0.21m
- Robot’s wheel radius 0.1m
- Maximum wheel rotational speed 10 rad/s

Problem 1

In problem 1, the robot has an initial position, and the purpose of the task is to reach the goal, also ensure the wheel rotation speed does not exceed the maximum speed.

Initial position $x[0] = [0 \;\; 0 \;\; \frac{-\pi}{2}]^T$

The goal $x^d = [-1 \;\; 1 \;\; \frac{\pi}{2}]^T$

There are two methods to compute the go-to-goal controller which are implemented in question a and question b below.

Question a

In this task, the control input was definded as :

$$\begin{bmatrix} v \\ w \end{bmatrix} = \begin{bmatrix} k * \sqrt{\bar{u}_x^2 + \bar{u}_y^2} \\ k_o(\theta_d - \theta) \end{bmatrix} = \begin{bmatrix} k * \sqrt{\left(p_x^d - p_x\right)^2 + \left(p_y^d - p_y\right)^2} \\ k_o(\theta_d - \theta) \end{bmatrix} \quad (3)$$

With $\theta_d = \text{atan}\,(\frac{p_y^d - p_y}{p_x^d - p_x})$

Considering the robot’s size and limitation of the wheel speed, a constant value k was added to the formula of v in the control input. The proportional control k was changed based on the equation.

$$k = \frac{v_0\left(1 - e^{-\beta\|\bar{e}\|}\right)}{\|\bar{e}\|} \quad (4)$$

With $\|\bar{e}\|$ is the magnitude of the error between the desired state and the current state

$$\|\bar{e}\| = \sqrt{(p_x^d - p_x)^2 + (p_y^d - p_y)^2} \quad (5)$$

So, after several trials, the value of v_o and β to ensure the wheel speed does not exceed the maximum 10 rad/s.

$$v_o = 0.95$$

$$\beta = 1$$

Choosing $k_o = k$

There is a small notice when compute w in the control input in formula 3 is that the angle between θ_d and θ is not over the range $[-\pi, \pi]$.

Figures below shows the result of the inplement. Figure 1 displays the trajectory of the robot from the initial position to the goal.

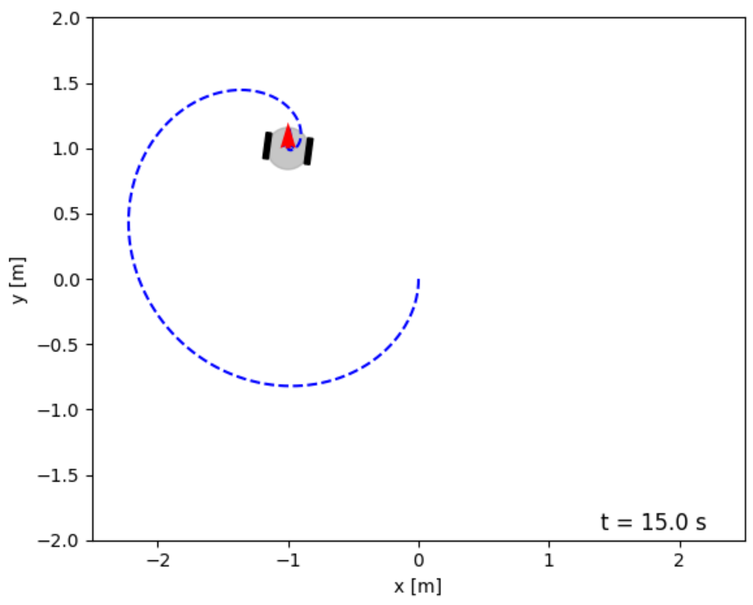


Figure 1: The robot's trajectory easy go-to-goal controller

The controller purpose was to reach the goal, so the orientation does not match with the final position. It was showed in figure 2, after 13 seconds, the speed went to 0 meaning it reach the goal, but w is different than 0.

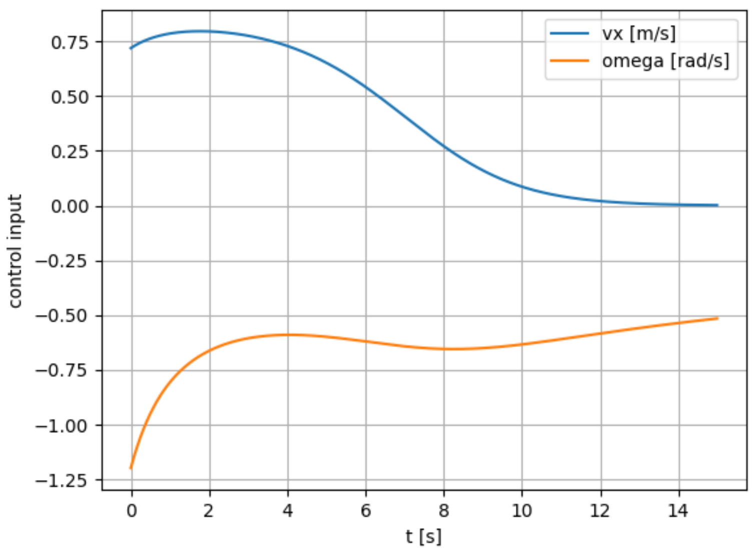


Figure 2: Control input of easy go-to-goal controller

One thing needs to be considered is the wheel speed during the travel. Figure 3 shows the rotational speeds of two wheels.

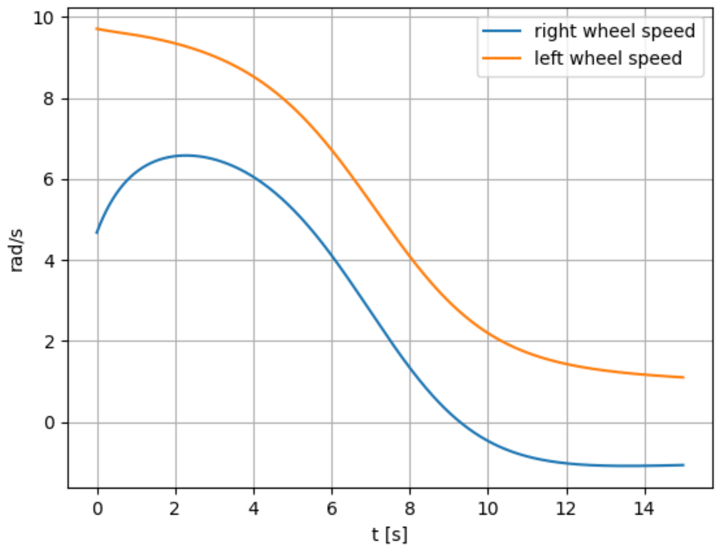


Figure 3: Wheel speeds of easy go-to-goal controller

At first the robot needs to speed-up a little to turn to right direction to the goal, however, it does not over 10 rad/s. The reason the wheel's speed does not go to 0, because the orientation of the robot and the goal is different, so the robot keeps spinning to reach the exact orientation, but the purpose of the controller is to reach the position, so it does not important to keep the controller running.

The time series of the robot's position in x and y compared to the position of the goal in x and y was showed in figure 4.

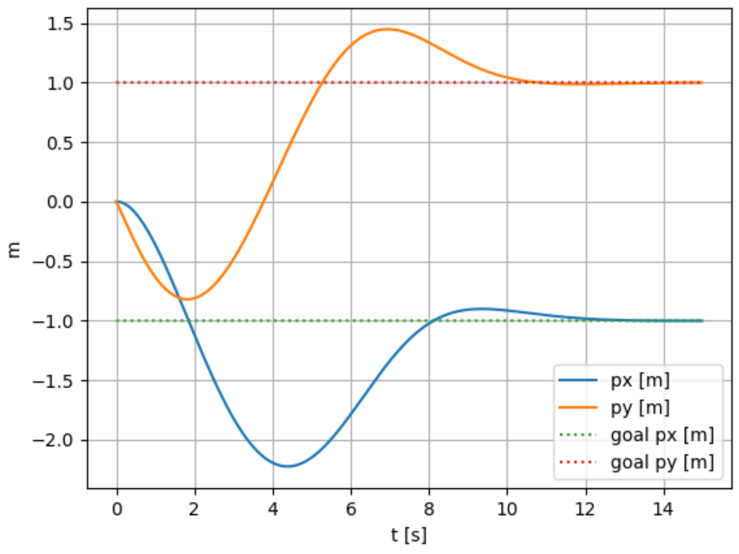


Figure 4: The position of robot and goal during easy go-to-goal controller

Finally, to confirm the robot reaches the goal, figure 5 shows the error between the position of the goal and the position of the robot during the travel.

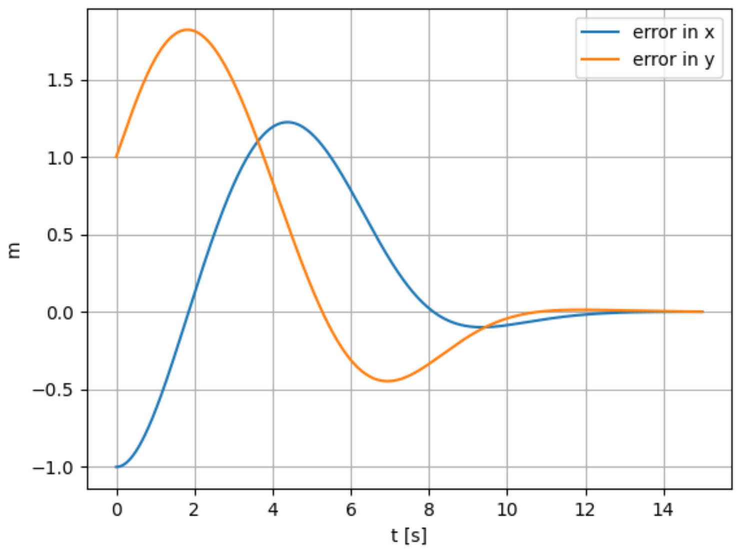


Figure 5: Time series of error of easy go-to-goal controller

Question b

In this task, instead of using the original position, a controller was applied for a single integrator, which was computed as :

$$\begin{bmatrix} S_x \\ S_y \end{bmatrix} = \begin{bmatrix} p_x \\ p_y \end{bmatrix} + \begin{bmatrix} l * cos\theta \\ l * sin\theta \end{bmatrix} \quad (6)$$

With l is a distance from a point (S_x, S_y) the position of the robot (p_x, p_y).

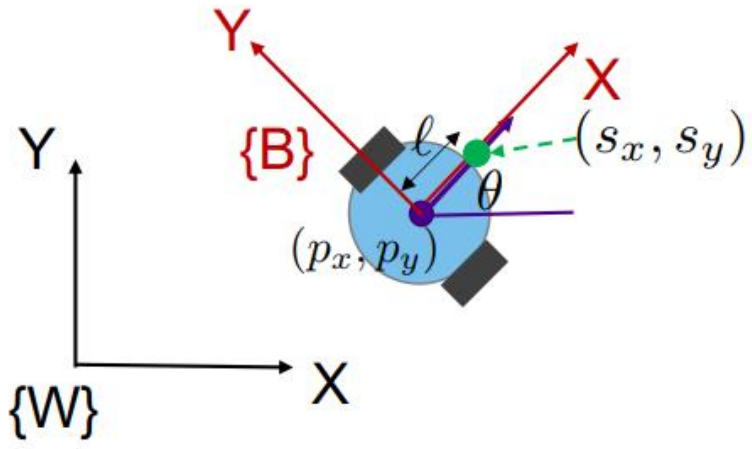


Figure 6: A caster (S_x, S_y)

Due to the new caster (S_x, S_y), the robot now acts as an omnidirectional robot, so the go-to-goal controller of the robot was computed:

$$\begin{bmatrix} \bar{u}_x \\ \bar{u}_y \end{bmatrix} = k * \begin{bmatrix} p_x^d - S_x \\ p_y^d - S_y \end{bmatrix} \quad (7)$$

For the controller 7, the omnidirectional mobile robot can go to the goal, but the robot here is a unicycle robot, so the control input needs to transfer to unicycle go-to-goal controller.

$$\begin{bmatrix} v \\ w \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & \frac{1}{l} \end{bmatrix} * \begin{bmatrix} cos\theta & sin\theta \\ -sin\theta & cos\theta \end{bmatrix} * \begin{bmatrix} \bar{u}_x \\ \bar{u}_y \end{bmatrix} \quad (8)$$

With θ is the angle of the robot.

For the caster (S_x, S_y), choosing $l = 0.06$. The proportional control k was changed based on the equation.

$$k = \frac{v_0\left(1 - e^{-\beta\|\bar{e}\|}\right)}{\|\bar{e}\|} \quad (9)$$

With $\|\bar{e}\|$ is the magnitude of the error between the desired state and the current caster state

$$\|\bar{e}\| = \sqrt{(p_x^d - S_x)^2 + (p_y^d - S_y)^2} \tag{10}$$

After several trials, the value of v_o and β in the formula 9 to ensure the wheel speed does not exceed the maximum 10 rad/s.

$$v_o = 0.2$$

$$\beta = 3.0$$

Figures below shows the result of the implement. Figure 7 displays the trajectory of the robot from the initial position to the goal.

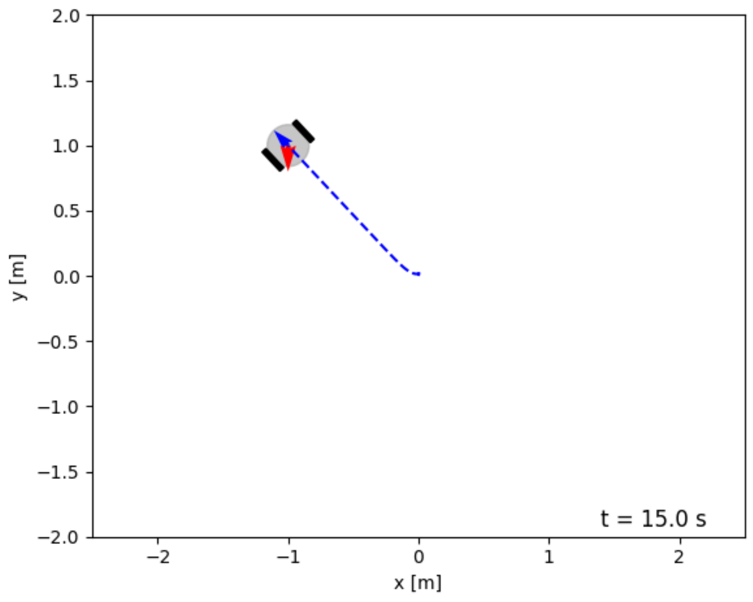


Figure 7: The control input of go-to-goal using caster.

The controller purpose was to reach the goal, so the orientation does not match with the final position. It was showed in figure 8, the control input of the unicycle mobile robot.

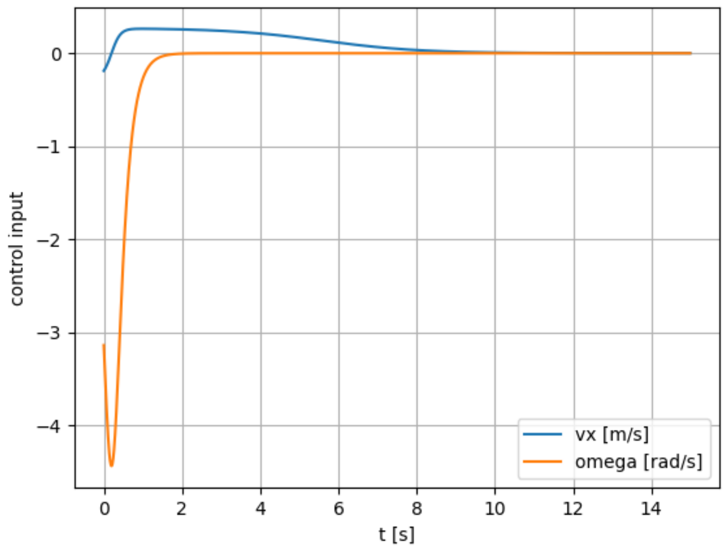


Figure 8: The control input of unicycle robot

Before computing the control input, a go-to-goal controller for the omnidirectional caster robot was also made. Figure 9 shows the go-to-goal $(\bar{u}_x, \bar{u}_y)$

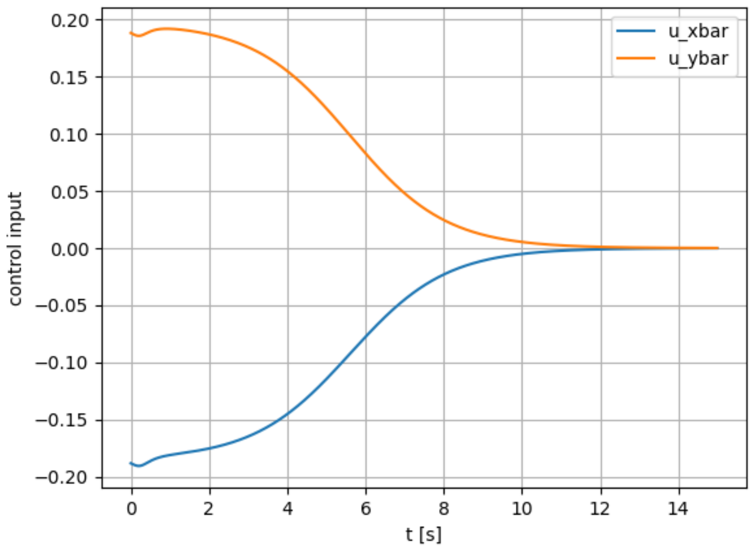


Figure 9: u_xbar, u_ybar of the robot

One thing also needs to be considered is the wheel speed during the travel. Figure 10 shows the rotational speeds of two wheels.

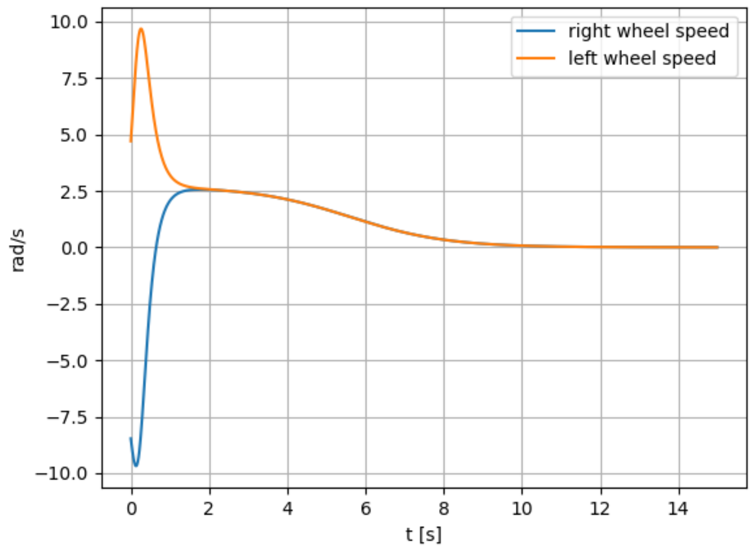


Figure 10: Wheel speed of question b problem 1

At first, the robot needs to turn the direction to the goal, so the speed rise rapidly, however, after the right orientation was reached, it reduced the speed until the robot at the goal.

The time series of the robot's position in x and y compared to the position of the goal in x and y was showed in figure 11.

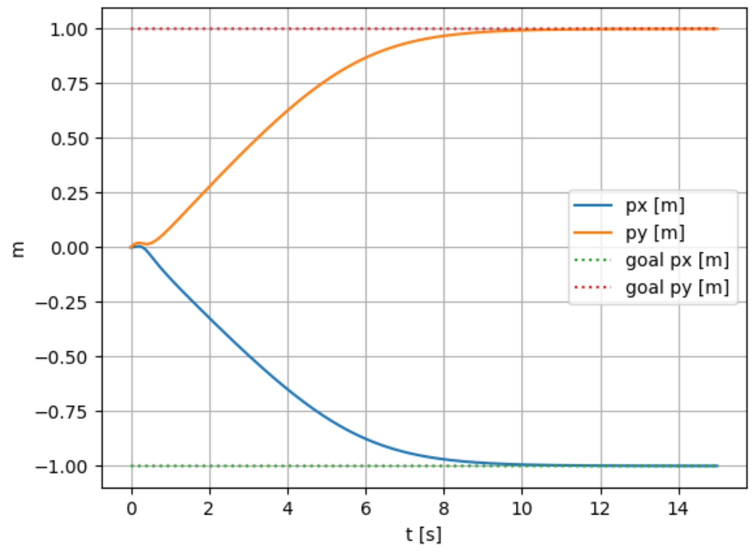


Figure 11: Time series of robot position and goal question b problem 1

Finally, to confirm the robot reaches the goal, figure 12 shows the error between the position of the goal and the position of the robot during the travel.

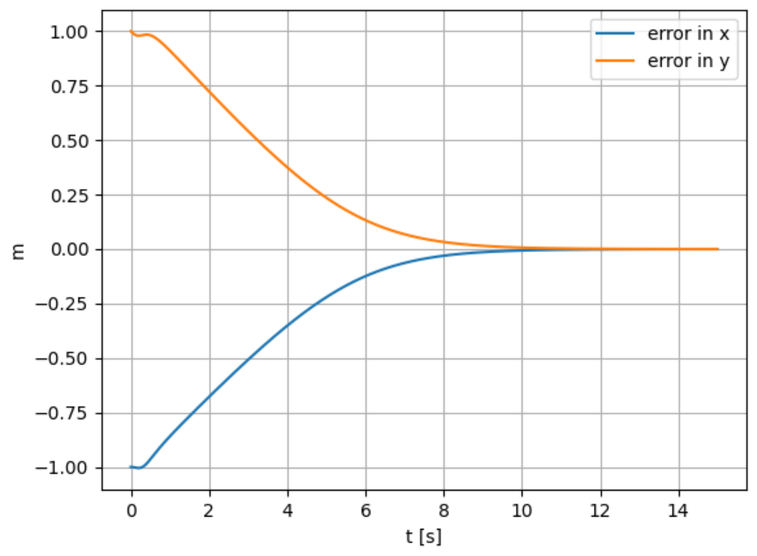


Figure 12: Time series of error between goal and robot's position in x and y

Problem 2

In problem 2, the robot has an initial position, and the purpose of the task is to reach the goal, also ensure the wheel rotation speed does not exceed the maximum speed.

Initial position $x[0] = [0 \;\; 0 \;\; \frac{-\pi}{2}]^T$

The moving goal

$$p_x^d[t] = -2\cos(0.25t)$$

$$p_y^d[t] = -sin(0.5t)$$

$$\theta^d[t] = atan2(\dot{p}_y^d, \dot{p}_x^d)$$

There are two methods to compute the go-to-goal controller: position and posture which are implemented in question a and question b below.

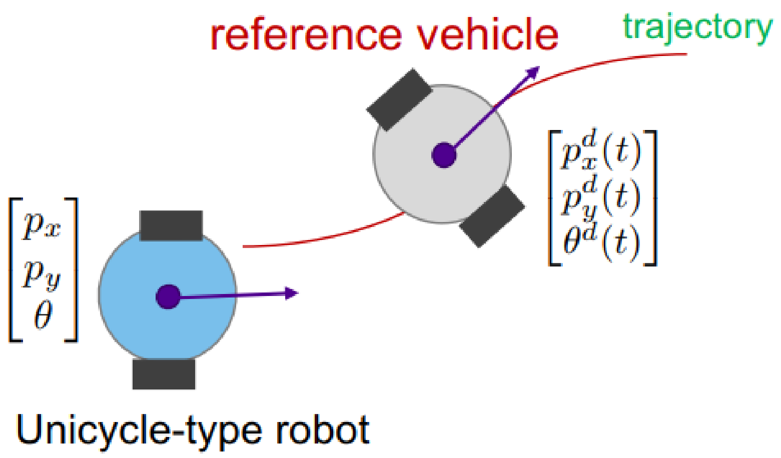

reference vehicle
trajectory
$\begin{bmatrix} p_x^d(t) \\ p_y^d(t) \\ \theta^d(t) \end{bmatrix}$
$\begin{bmatrix} p_x \\ p_y \\ \theta \end{bmatrix}$
Unicycle-type robot

Figure 13: The robot follows a virtual reference unicycle mobile robot

Question a

A new caster point was made to turn the robot into a single integrator.

$$\begin{bmatrix} S_x \\ S_y \end{bmatrix} = \begin{bmatrix} p_x \\ p_y \end{bmatrix} + \begin{bmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{bmatrix} * \begin{bmatrix} l_1 \\ l_2 \end{bmatrix} \tag{11}$$

With θ is the orientaion of the robot

Due to the new caster (S_x, S_y), the robot now acts as an omnidirectional robot, so the go-to-goal controller of the robot was computed using feedforward :

$$\bar{u}_x = k_1(p_x^d(t) - S_x) + \dot{p}_x^d \; with \; k_1 > 0 \tag{12}$$

$$\bar{u}_y = k_2\left(p_y^d(t) - S_y\right) + \dot{p}_y^d \; with \; k_2 > 0 \tag{13}$$

The feedforward was a derivertive of the reference robot, so

$$\dot{p}_x^d = 0.5\sin(0.25t) \tag{14}$$

$$\dot{p}_y^d = -0.5\cos(0.5t) \tag{15}$$

For the controller 12 and 13, the omnidirectional mobile robot can go to the goal, but the robot here is a unicycle robot, so the control input needs to transfer to unicycle go-to-goal controller.

$$\begin{bmatrix} v \\ w \end{bmatrix} = H^{-1} * \begin{bmatrix} \bar{u}_x \\ \bar{u}_y \end{bmatrix} \tag{16}$$

With matrix H

$$H = \begin{bmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{bmatrix} * \begin{bmatrix} 1 & 0 \\ 0 & l_1 \end{bmatrix} * \begin{bmatrix} 1 & -l_2 \\ 0 & 1 \end{bmatrix} \tag{17}$$

θ is the angle of the robot.

For the caster (S_x, S_y), choosing $l_1 = 0.08$ and $l_2 = 0$ to simplify the computing. The proportional control k was changed based on the equation.

$$k = \frac{v_0\left(1 - e^{-\beta\|\bar{e}\|}\right)}{\|\bar{e}\|} \tag{18}$$

With $\|\bar{e}\|$ is the magnitude of the error between the desired state and the current state

$$\|\bar{e}\| = \sqrt{(p_x^d - S_x)^2 + (p_y^d - S_y)^2} \tag{19}$$

After several trials, the value of v_o and β in the formula 18 to ensure the wheel speed does not exceed the maximum 10 rad/s.

$$v_o = 0.19$$

$$\beta = 5.0$$

Figures below shows the result of the inplement. Figure 14 displays the trajectory of the robot from the initial position to the goal.

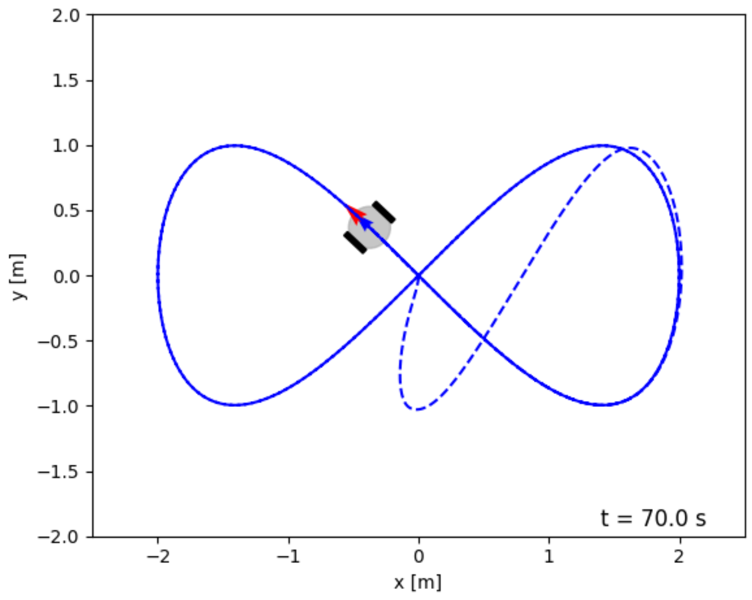


Figure 14: Robot trajectory using feedforward and single integrator.

The control input of the controller was showed in figure 15

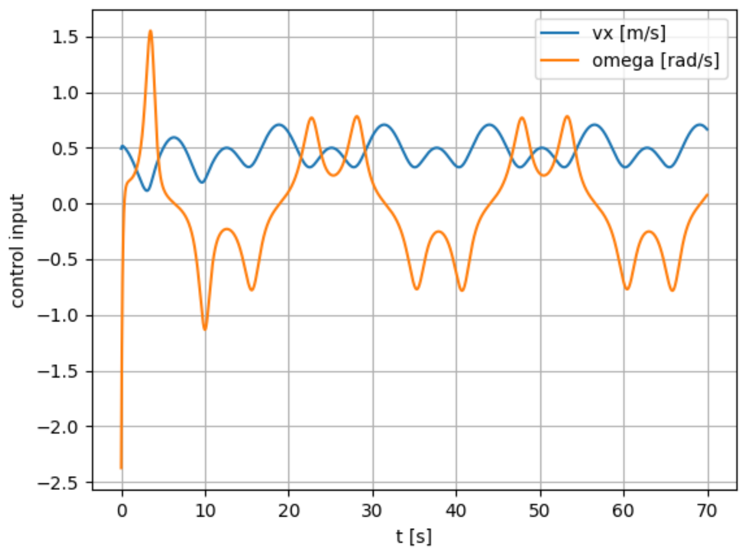


Figure 15: Control input using feedforward and single integrator.

One thing also needs to be considered is the wheel speed during the travel. Figure 16 shows the rotational speeds of two wheels.

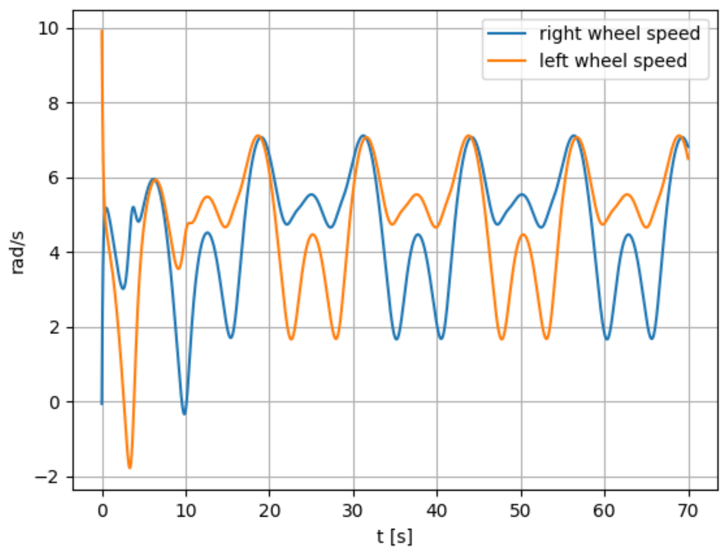


Figure 16: Rotational speed of two wheel using feedforward and single integrator.

In the figure 16, the robot starts the travel at very high speed of the left wheel, but it does not exceed the maximum speed. After the robot reach the moving goal, the control input and the speed of two wheel keeps repeating because the robot continues following the same path.

Figure 17 showed the time series of the robot position and the moving goal position.

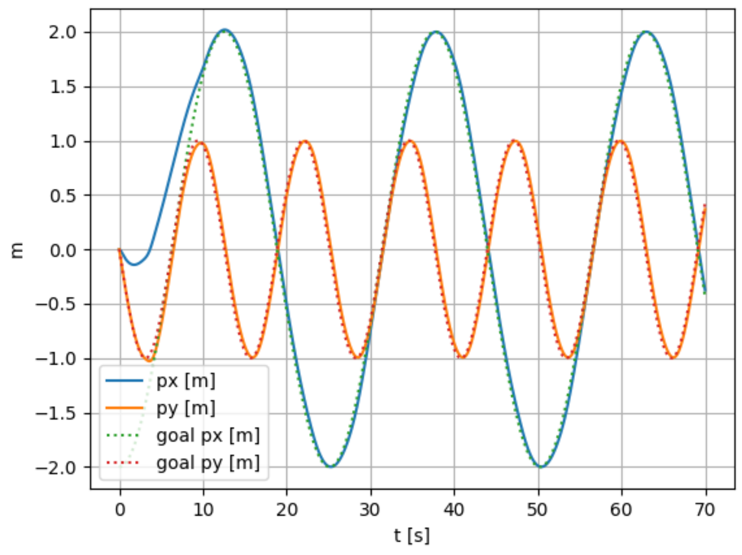


Figure 17: Time series of the robot and goal

In general, it looks like the lines in figure 17 are perfectly fit into each other, however, there is a slightly gap between them. To clarify the situation, the time series of the robot's position in x and y compared to the position of the goal in x and y was showed in figure 18.

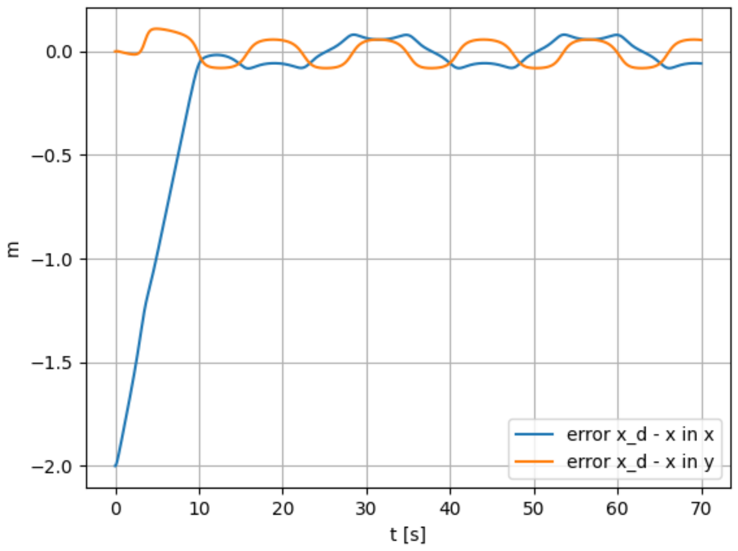


Figure 18: error between the robot position and goal

As it is showed in figure 18, the error between the moving goal and the robot is not 0. This happened because the single integrator point $[S_x, S_y]$ reaches the goal, there is a gap difference between the exact robot position $[p_x, p_y]$ and $[S_x, S_y]$, so that's why the graph in figure 18 does not look nice, and a small gap in figure 17. To clarify, figure 19 shows the error between the moving goal and the point $[S_x, S_y]$

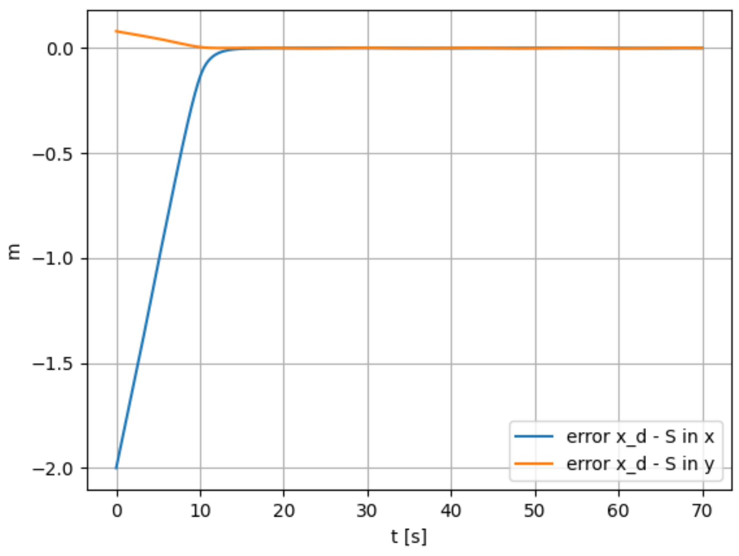


Figure 19: Error between the caster point and the goal.

As it shows the error went to 0 after 15 seconds.

Question b

Different than question a, question b focus on following the right posture of a reference virtual robot $[p_x^d(t), p_y^d(t), \theta^d(t)]$

As the reference robot is a unicycle mobile robot, then it follows the principal.

$$\dot{p}_x^d = v_d cos\theta^d$$

$$\dot{p}_y^d = v_d sin\theta^d$$

$$\dot{\theta}^d = w_d$$

From above,

$$v_d(t) = \sqrt{\left(\dot{p}_x^d(t)\right)^2 + \left(\dot{p}_y^d(t)\right)^2} \quad (20)$$

$$w_d(t) = \frac{\ddot{p}_y^d(t)\dot{p}_x^d(t) - \ddot{p}_x^d(t)\dot{p}_y^d(t)}{\left(\dot{p}_x^d(t)\right)^2 + \left(\dot{p}_y^d(t)\right)^2} \quad (21)$$

The error between the reference robot and the actual robot was defined and transfered from World-coordinate to Body-coordinate.

$$e = \begin{bmatrix} e_1 \\ e_2 \\ e_3 \end{bmatrix} = \begin{bmatrix} cos\theta & sin\theta & 0 \\ -sin\theta & cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} p_x^d - p_x \\ p_y^d - p_y \\ \theta^d - \theta \end{bmatrix} \quad (22)$$

From the error, the control input was calcaulated as :

$$\begin{bmatrix} v \\ w \end{bmatrix} = \begin{bmatrix} v_d \cos(e_3) - u_1 \\ w_d - u_2 \end{bmatrix} \tag{23}$$

The linear feedback

$$\begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} -k_1 e_1 \\ -k_2 e_2 - k_3 e_3 \end{bmatrix} \tag{24}$$

The constant value k_1, k_2, k_3 was defined based on

$$k_1 = k_3 = 2\eta a \tag{25}$$

$$k_2 = \frac{a^2 - w_d^2}{v_d} \; with\; a > 0, and\; \eta \in (0,1) \tag{26}$$

After several trials the value of k_1, k_2, k_3 in the formula 25 and 26 to ensure the wheel speed does not exceed the maximum 10 rad/s.

$$a = 0.6$$

$$\eta = 0.6$$

Figures below shows the result of the inplement. Figure 20 displays the trajectory of the robot from the initial position to the goal.

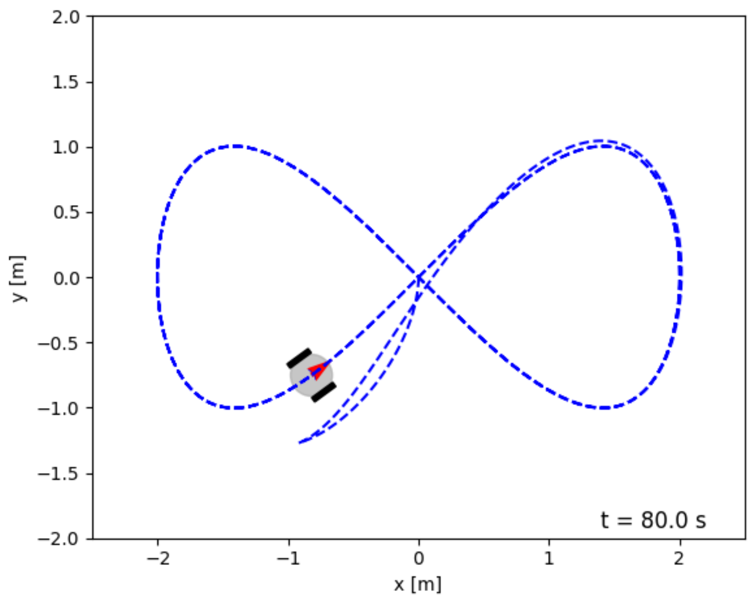


Figure 20: Robot trajectory following a reference robot

Based on figure 20, the robot path is not matching with the reference robot at first, however it reach the moving goal eventually. Figure 21 shows the control input of the robot in this task

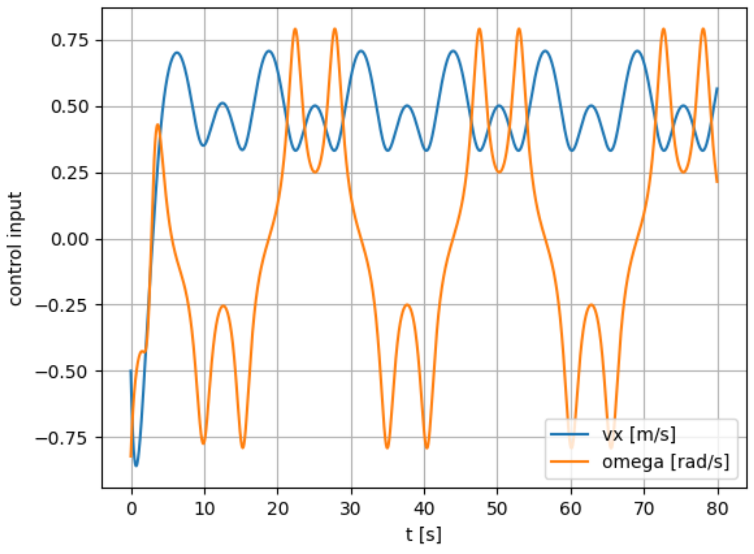


Figure 21: The control input of question b problem 2

The goal is moving so the control input continues until the time ends. Figure 22 shows the rotational speed of two wheels.

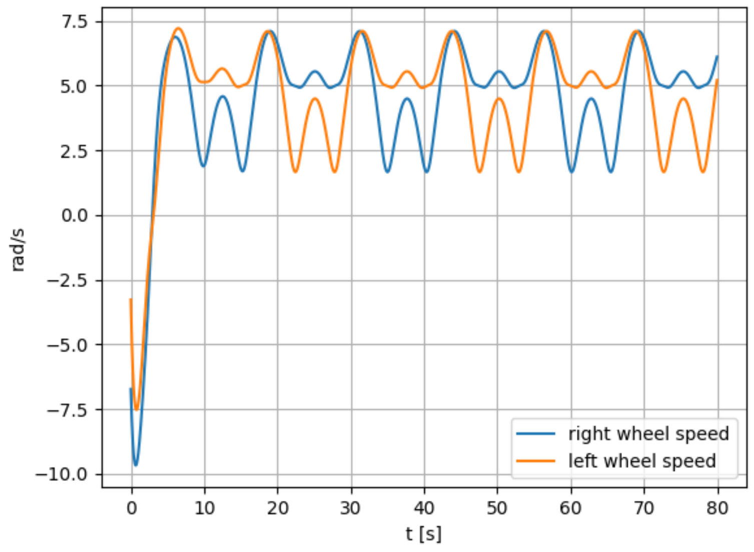


Figure 22: The rotational speed of two wheels question b problem 2

In the figure 22, the robot starts the travel at very high speed of two wheels, but it does not exceed the maximum speed. After the robot reach the moving goal about 20 seconds, the control input and the speed of two wheel keeps repeating because the robot continues following the same path.

To clarify the position of robot and the moving goal, figure 23 shows the p_x^d, p_y^d, p_x, p_y

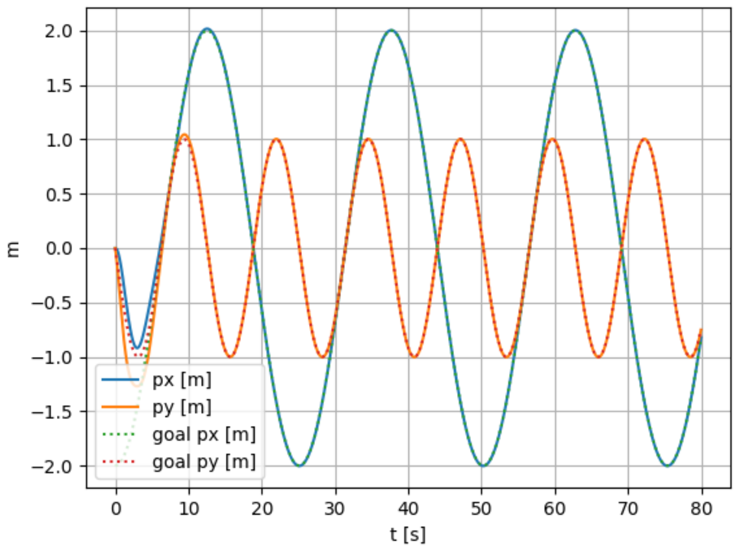


Figure 23: Position of goal and robot question b problem 2

As the figure above, after 20 seconds, the robot reaches the moving virtual robot, and it took 26 seconds for the robot to finish the loop path. Finally, to confirm the robot matches the virtual one, figure 24 shows the time series of the error between the position of the goal and the robot.

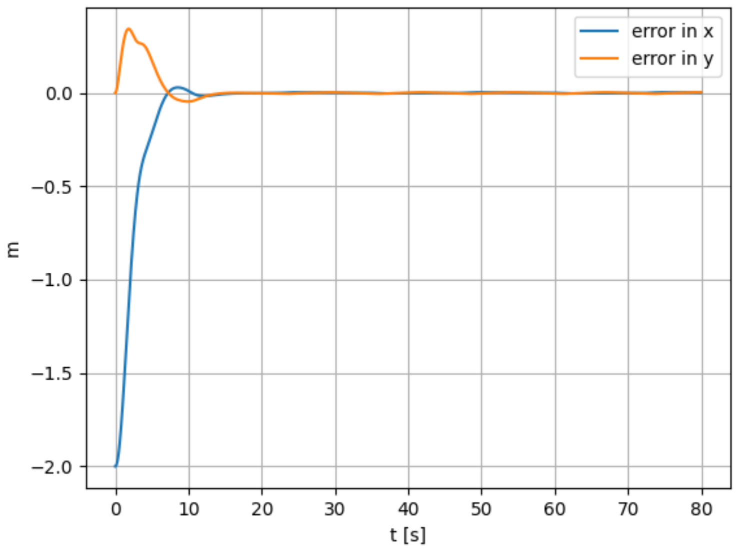


Figure 24: Error between the robot and the goal question b problem 2

The error should be 0 to prove that the robot reaches the goal, however there is a slightly difference, it does not affect the result and the robot behaviour so it will be neglected.

Problem 3

In this problem, uncycle mobile robot was used with:

Initial position $x[0] = [-4 \ \ -3.5 \ \ 0]^T$

There are three waypoints:

$$x_1^d = [4 \ \ 0 \ \ 0]^T$$

$$x_2^d = [-0.5 \ \ 3.7 \ \ 0]^T$$

$$x_3^d = [-4 \ \ -3.5 \ \ 0]^T$$

Figure 25 shows obstacle shape and the robot's sensor ring which has a range 1 m.

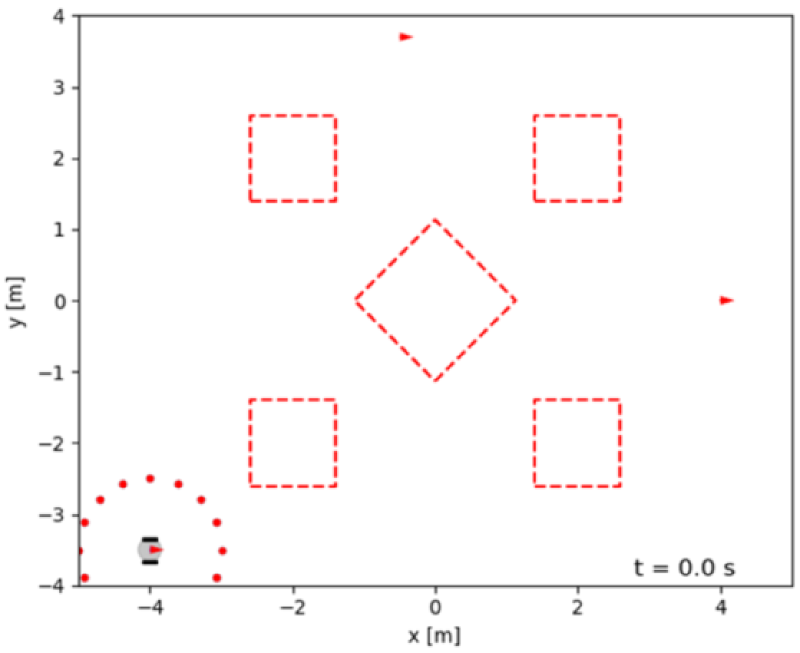


Figure 25: The robot and the obstacle and the goal

The purpose of the task is to reach three goals $x_1^d \rightarrow x_2^d \rightarrow x_3^d$ while avoiding collision with the obstacles and ensure the robot moves within the maximum rotational speed of two wheels.

The ring around the robot has 16 numbers which mean 16 sensors. From those sensors, there are 2 data which is needed in this problem. The first one is distance_reading, it has 16 elements, the maximum is 1, which means the sensor does not approach obstacle, the smaller value means the near the robot to the obstacle. The second one is obst_points, it is a matrix 2x16, which means the first array is x-coordinate and the second is y-coordinate of those sensors.

Changing goals

In every iteration, calculate the distance between the robot and the goal, if it is less than 0.2, change the goal to the next one.

Computing control input

For this problem, the robot can be controlled by four control input: go-to-goal, avoidance, wall-following clockwise and wall-following counter-clockwise. The robot started with u_gtg, when it approaches obstacle, it changed to u_wf, while the robot under u_wf, it can be deviate and cause collision with the obstacle, so u_avo will avoid it. Then the robot changed back to u_wf, until the path is clear to change to control input u_gtg.

To compute the control input of the robot, firstly a new caster point was made to turn the robot into a single integrator.

$$\begin{bmatrix} S_x \\ S_y \end{bmatrix} = \begin{bmatrix} p_x \\ p_y \end{bmatrix} + \begin{bmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{bmatrix} * \begin{bmatrix} l_1 \\ l_2 \end{bmatrix} \quad (27)$$

With θ is the orientaion of the robot, choosing $l_1 = 0.06$ and $l_2 = 0$

Based on the robot specification and speed limitation, choosing eps = 0.02, and d_safe = 0.3

Due to the new caster (S_x, S_y), the robot now acts as an omnidirectional robot, and the problem turns into a hard switching controller.

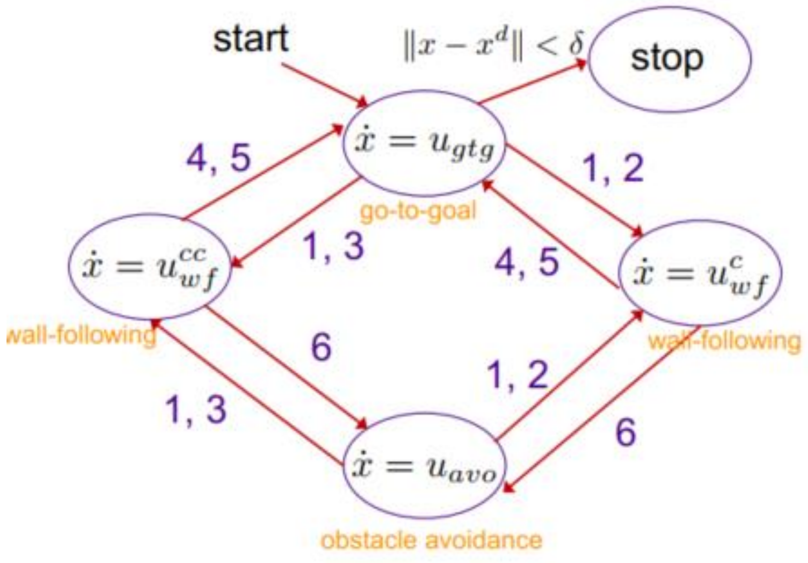
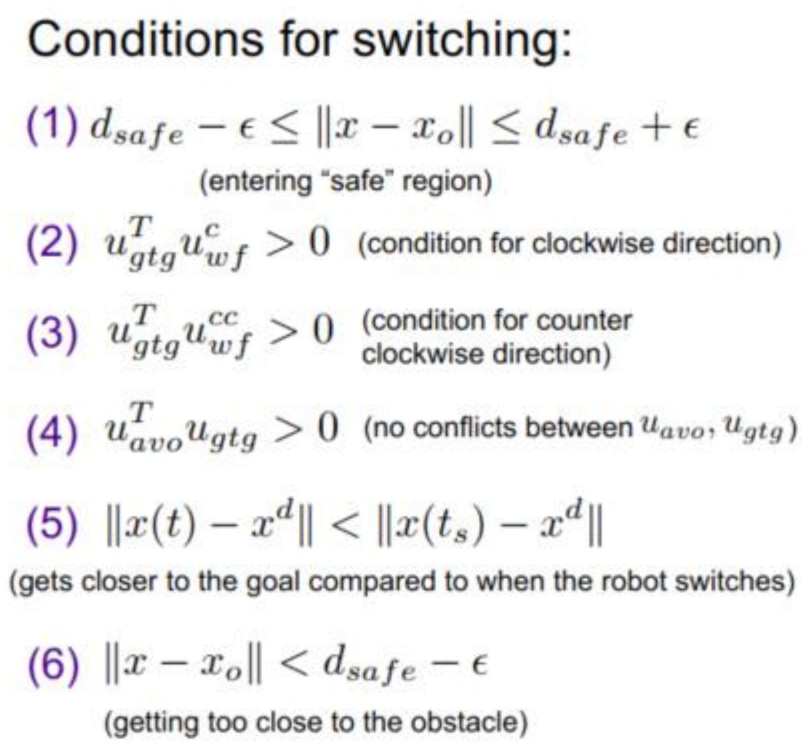


Figure 26: Hard switching method

The condition to switch is based on figure 27.



Conditions for switching:

(1) $d_{safe} - \epsilon \leq \|x - x_o\| \leq d_{safe} + \epsilon$

(entering "safe" region)

(2) $u_{gtg}^T u_{wf}^c > 0$ (condition for clockwise direction)

(3) $u_{gtg}^T u_{wf}^{cc} > 0$ (condition for counter clockwise direction)

(4) $u_{avo}^T u_{gtg} > 0$ (no conflicts between u_{avo}, u_{gtg})

(5) $\|x(t) - x^d\| < \|x(t_s) - x^d\|$

(gets closer to the goal compared to when the robot switches)

(6) $\|x - x_o\| < d_{safe} - \epsilon$

(getting too close to the obstacle)

Figure 27: Conditions for switching.

Therefore, in this problem, an approach for solving it is that firstly compute all controller u_gtg, u_avo, secondly, based on data from sensors, selecting suitable sensors, then compute u_wf_cc, u_wf_cw. Thirdly, checking all conditions based on computed controllers, then the control state was made. Finally, choosing the control input based on the control state, but the control input is for the omnidirectional mobile robot can go to the goal, but the robot here is a unicycle robot, so the control input needs to transfer to unicycle controller.

$$\begin{bmatrix} v \\ w \end{bmatrix} = H^{-1} * \begin{bmatrix} \bar{u}_x \\ \bar{u}_y \end{bmatrix} \tag{28}$$

With $\begin{bmatrix} \bar{u}_x \\ \bar{u}_y \end{bmatrix}$ is the control input based on control state, and matrix H

$$H = \begin{bmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{bmatrix} * \begin{bmatrix} 1 & 0 \\ 0 & l_1 \end{bmatrix} * \begin{bmatrix} 1 & -l_2 \\ 0 & 1 \end{bmatrix} \tag{29}$$

θ is the angle of the robot.

Here are the controllers were computed.

Control input go-to-goal u_gtg

For u_gtg control input, the proportional control k_g was designed based on the equation

$$k_g = \frac{v_0\left(1 - e^{-\beta\|\bar{e}\|}\right)}{\|\bar{e}\|} \tag{30}$$

With $\|\bar{e}\|$ is the magnitude of the error between the goal and the position of the caster

$$\|\bar{e}\| = \sqrt{(x_x^d - S_x)^2 + (x_y^d - S_y)^2} \tag{31}$$

v_0 was set to 0.2, and β was set to 1.5. The value of k_g was depended on the value of v_0 and β.

Control input avoidance u_avo

For u_avo control input, the proportional control k_o was designed based on the equation

$$k_o = \frac{1}{\|\bar{e}\|}\left(\frac{c}{\|\bar{e}\|^2 - eps}\right) \; with\; c > 0 \tag{32}$$

With c was set to 0.07 and $\|\bar{e}\|$ is the magnitude of the error between obstacle and the caster, which is taken from the distance_reading array

$$\|\bar{e}\| = \sqrt{(S_x - x_{ox})^2 + \left(S_y - x_{oy}\right)^2} \tag{33}$$

x_o was defined by distance_reading, at first, find out which sensor has value less than 1, which means it approached to the obstacle. Then find the index of its sensor and from that index, take the coordinate of the sensor by looking for at obst_points.

Computing x_o

The most important part of this method was to find two sensors to the obstacle. The robot can use control input u_wf clockwise or counter clockwise correctly depending on the selection of sensors.

From the number of elements in distance_reading, if the number of sensors contact to obstacles more than 1, then sorting the array distance_reading from lowest to highest value. So, taking the two lowest values as the two nearest sensor position from the sorted array. Depend on which control state to sort two sensors x_o_1 and x_o_2.

If the number of sensor which contact to obstacles is 1, then compute

$$u_{wf}^{cc} = k_{wf} * \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} * u_{avo} \tag{34}$$

$$u_{wf}^{cw} = k_{wf} * \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} * u_{avo} \tag{35}$$

Control input wall-following counter-clockwise u_wf_cc

Firstly, compute a vector tangential to the wall.

$$u_{wf,t} = x_{o2} - x_{o1} \tag{36}$$

$$\bar{u}_{wf,t} = \frac{u_{wf,t}}{\|u_{wf,t}\|} \tag{37}$$

Secondly, compute a vector perpendicular to the wall.

$$u_{wf,p} = (x_{o1} - x) - \left((x_{o1} - x) * \bar{u}_{wf,t}\right) * \bar{u}_{wf,t} \tag{38}$$

$$\hat{u}_{wf,p} = u_{wf,p} - \frac{d^{des}}{\|u_{wf,p}\|} * u_{wf,p} \tag{39}$$

With d^{des} is desired distance from the wall, in this case choosing $d^{des} = d\_safe$

Finally, combine two vectors.

$$u_{wf}^{cc} = \alpha_1 * \hat{u}_{wf,p} + \alpha_2 * \bar{u}_{wf,t} \tag{40}$$

Choosing $\alpha_1 = 0.4$ and $\alpha_2 = 0.25$

Control input wall-following clockwise u_wf_cw

Firstly, compute a vector tangential to the wall.

$$u_{wf,t} = x_{o1} - x_{o2} \tag{41}$$

$$\bar{u}_{wf,t} = \frac{u_{wf,t}}{\|u_{wf,t}\|} \tag{42}$$

Secondly, compute a vector perpendicular to the wall.

$$u_{wf,p} = (x_{o2} - x) - \left((x_{o1} - x) * \bar{u}_{wf,t}\right) * \bar{u}_{wf,t} \tag{43}$$

$$\hat{u}_{wf,p} = u_{wf,p} - \frac{d^{des}}{\|u_{wf,p}\|} * u_{wf,p} \tag{44}$$

With d^{des} is desired distance from the wall, in this case choosing $d^{des} = d\_safe$

Finally, combine two vectors.

$$u_{wf}^{cw} = \alpha_1 * \hat{u}_{wf,p} + \alpha_2 * \bar{u}_{wf,t} \tag{45}$$

Choosing $\alpha_1 = 0.4$ and $\alpha_2 = 0.25$

Result

From u_gtg, u_avo, u_wf_cc, and u_wf_c, a new control input was designed. Figure 30 shows the robot’s trajectory to three goals.

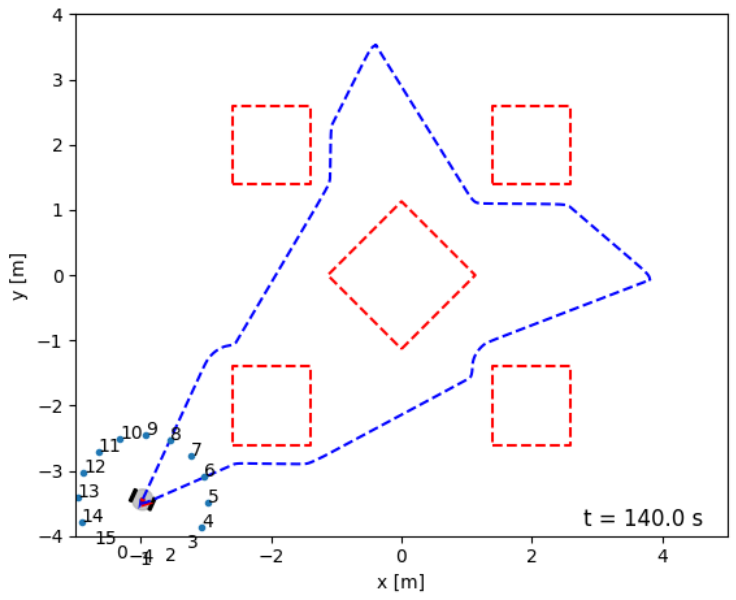


Figure 28: Robot's trajectory to three goals

Based on figure 30, the robot reaches 3 gateways and does not contact obstacles.

Figure 31 shows the control input of the robot.

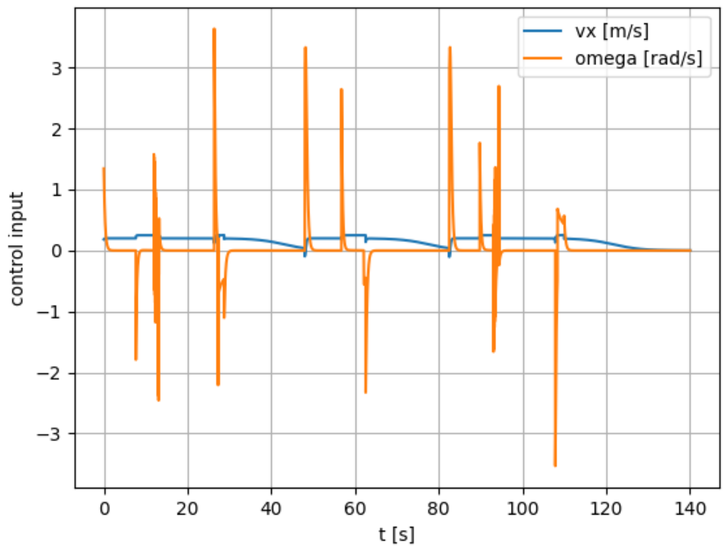


Figure 29: Control input during the travel

Figure 32 shows the goal state and robot state, the robot reaches goals one by one.

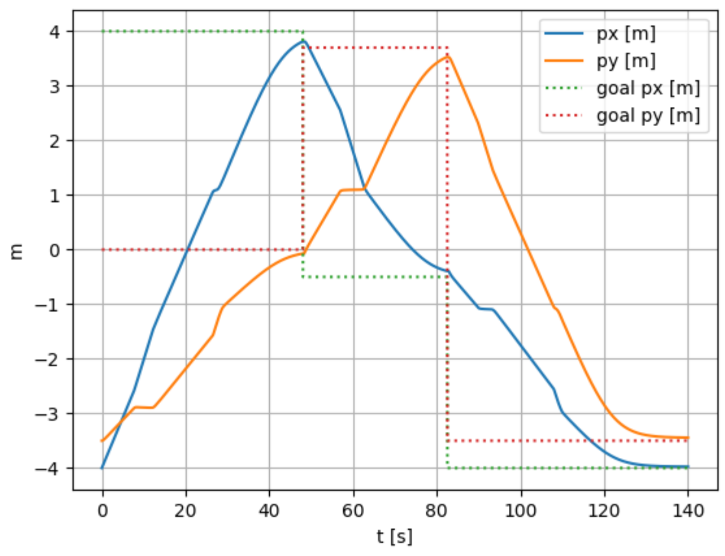


Figure 30: Robot state and goal states

There are small gaps between the goals and the robot when the robot approaches, to make the gaps easy to see, figure 33 shows the error between the goals and the robot state.

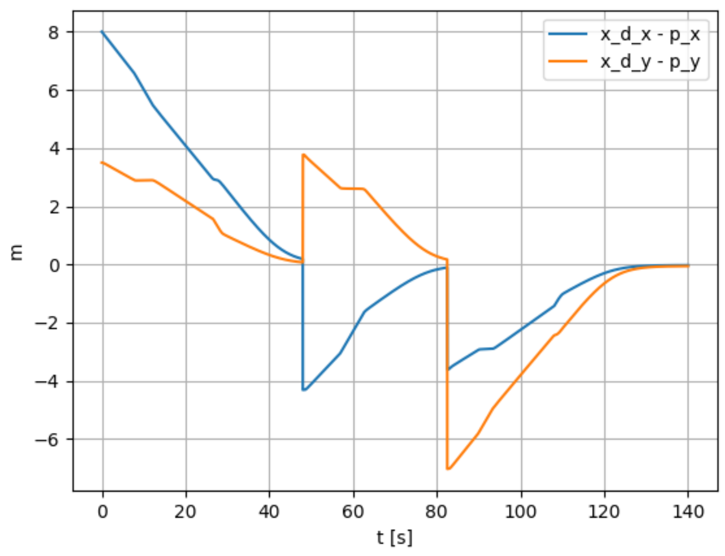


Figure 31: Error between the robot and the goals

It happens because a small distance 0.2 was made, so the robot did not reach the exact location of the goal, it went to 0.2 radius area then moved to next goal. Also, the caster point is the one reaching the goal, not the actual robot position, so to clarify the situation figure 34 shows the error of the goals and the caster point.

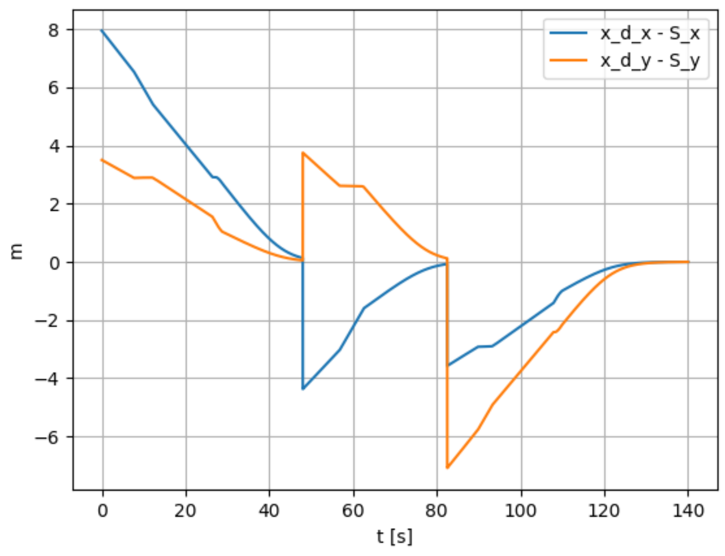


Figure 32: Error between the caster and the goals

Finally, figure 35 shows the rotational speed of two wheel to ensure the robot does not exceed the maximum speed limit.

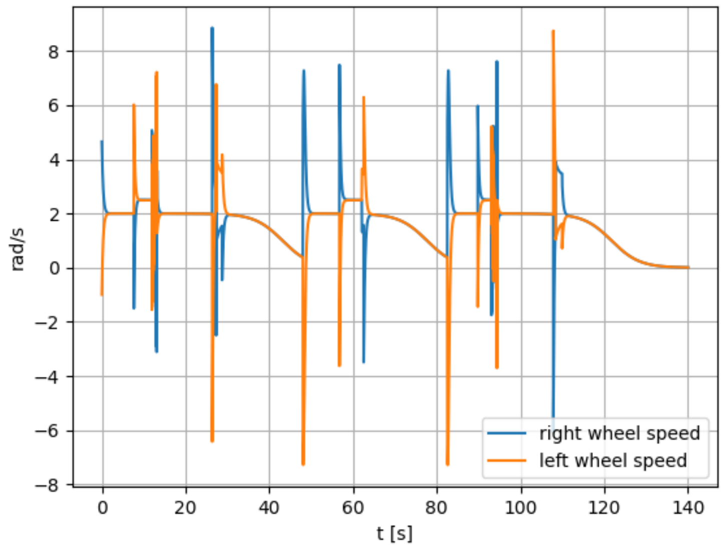


Figure 33: The rotational speed of two wheels

In conclusion, by using the wall-following method, the robot can overcome un-defined shape obstacles, changing the control input based on the condition and previous control state helps the ride smoother.

Dong Le

Conclusion

By implementing the first problem, the result found out that the robot can go to goal by using different methods like computing direct control input or using single integrator. For using the caster point, it is important to transfer the control input to unicycle control input, otherwise the result is wrong. Consider choosing l not too small, it can cause the large value of w, then the robot exceeds the maximum wheel speed. The choice of parameters affects the speed of wheels, therefore, carefully tuning it.

In the second problem the result found out that the trajectory tracking based on a reference robot depend on following position or posture of the reference and ensure that the reference trajectory is feasible. For the position case, the controller only considers about the position, so there is always an error between the angle of the robot and the angle of the goal. For the posture case, the reference robot needs to move constantly. The choice of parameters affects the speed of wheels, therefore, carefully tuning it.

In the third problem the result found out by applying everything in the part 2 of the course, from omnidirectional robot, to hard-switching, unicycle robot, sensor, obstacle avoidant controller, wall-following controller. The choice of parameters affects the speed of wheels, therefore, carefully tuning it.

Appendices

Appendix A

Program code is attached to the zip file.

THIS PAGE LEFT BLANK INTENTIONALLY