Homework 3 (Tasks 1-19) in EL2450 Hybrid and Embedded Control Systems

Devendra Sharma
950815-5497
devendra@kth.se

Kartik Chari
960807-0174
kartikc@kth.se

Merav Modi
19950120-0191
merav@kth.se

Khushdeep Singh
19940126-5955
ksmann@kth.se

March 5, 2019

Task 1

Given the translational velocity (v) and rotational velocity (ω), we can calculate u_r and u_l as explained below. Given equations:

$$v = \frac{u_r + u_l}{2} \tag{1}$$

$$\omega = u_r - u_l \tag{2}$$

Solving equation (1) and (2) by adding them, will give us u_r and u_l:

$$u_r = \frac{2v + \omega}{2}$$
$$u_l = \frac{2v - \omega}{2}$$

Task 2

The transition system $\mathcal{T}$ is defined as:

$$\mathcal{T} = (S, S^0, \sum, \rightarrow, AP, L)$$

where

- S is the set of the discretized states of the robot i.e. $S = \{R_1, R_2, ..., R_{\mathrm{K}}\}$
  Here, K is considered as 36. If we consider a bounding rectangle around our robot with length = 38 cm = breadth, the area occupied by it is 0.1444 m^2. So, the minimum area of each region in workspace should be 0.1444 m^2. For K = 36, the area of each region turned out to be 0.25 m^2 with length = 0.5m = breadth.(See fig.1)

- $S^0 = \{R_6\}$
- $\sum = \{up, down, right, left, stop\}$
- $\rightarrow = \{(R_1, stop, R_1), (R_1, up, R_2), (R_1, right, R_{12}),$
  $(R_2, stop, R_2), (R_2, up, R_3), (R_2, down, R_1), (R_2, right, R_{11}),$
  $(R_3, stop, R_3), (R_3, up, R_4), (R_3, down, R_2), (R_3, right, R_{10}),$
  $(R_4, stop, R_4), (R_4, up, R_5), (R_4, down, R_3), (R_4, right, R_9),$
  $(R_5, stop, R_5), (R_5, up, R_6), (R_5, down, R_4), (R_5, right, R_8),$
  $(R_6, stop, R_6), (R_6, down, R_5), (R_6, right, R_7),$
  $(R_7, stop, R_7), (R_7, down, R_8), (R_7, left, R_6), (R_7, right, R_{18}),$
  $(R_8, stop, R_8), (R_8, up, R_7), (R_8, down, R_9), (R_8, left, R_5), (R_8, right, R_{17}),$
  $(R_9, stop, R_9), (R_9, up, R_8), (R_9, down, R_{10}), (R_9, left, R_4), (R_9, right, R_{16}),$
  $(R_{10}, stop, R_{10}), (R_{10}, up, R_9), (R_{10}, down, R_{11}), (R_{10}, left, R_3), (R_{10}, right, R_{15}),$
  $(R_{11}, stop, R_{11}), (R_{11}, up, R_{10}), (R_{11}, down, R_{12}), (R_{11}, left, R_2), (R_{11}, right, R_{14}),$
  $(R_{12}, stop, R_{12}), (R_{12}, up, R_{11}), (R_{12}, left, R_1), (R_{12}, right, R_{13}),$
  $(R_{13}, stop, R_{13}), (R_{13}, up, R_{14}), (R_{13}, left, R_{12}), (R_{13}, right, R_{24}),$
  $(R_{14}, stop, R_{14}), (R_{14}, up, R_{15}), (R_{14}, down, R_{13}), (R_{14}, left, R_{11}), (R_{14}, right, R_{23}),$
  $(R_{15}, stop, R_{15}), (R_{15}, up, R_{16}), (R_{15}, down, R_{14}), (R_{15}, left, R_{10}), (R_{15}, right, R_{22}),$
  $(R_{16}, stop, R_{16}), (R_{16}, up, R_{17}), (R_{16}, down, R_{15}), (R_{16}, left, R_9), (R_{16}, right, R_{21}),$
  $(R_{17}, stop, R_{17}), (R_{17}, up, R_{18}), (R_{17}, down, R_{16}), (R_{17}, left, R_8), (R_{17}, right, R_{20}),$
  $(R_{18}, stop, R_{18}), (R_{18}, down, R_{17}), (R_{18}, left, R_7), (R_{18}, right, R_{19}),$
  $(R_{19}, stop, R_{19}), (R_{19}, down, R_{20}), (R_{19}, left, R_{18}), (R_{19}, right, R_{30}),$
  $(R_{20}, stop, R_{20}), (R_{20}, up, R_{19}), (R_{20}, down, R_{21}), (R_{20}, left, R_{17}), (R_{20}, right, R_{29}),$
  $(R_{21}, stop, R_{21}), (R_{21}, up, R_{20}), (R_{21}, down, R_{22}), (R_{21}, left, R_{16}), (R_{21}, right, R_{28}),$
  $(R_{22}, stop, R_{22}), (R_{22}, up, R_{21}), (R_{22}, down, R_{23}), (R_{22}, left, R_{15}), (R_{22}, right, R_{27}),$
  $(R_{23}, stop, R_{23}), (R_{23}, up, R_{22}), (R_{23}, down, R_{24}), (R_{23}, left, R_{14}), (R_{23}, right, R_{26}),$
  $(R_{24}, stop, R_{24}), (R_{24}, up, R_{23}), (R_{24}, left, R_{13}), (R_{24}, right, R_{25}),$
  $(R_{25}, stop, R_{25}), (R_{25}, up, R_{26}), (R_{25}, left, R_{24}), (R_{25}, right, R_{36}),$
  $(R_{26}, stop, R_{26}), (R_{26}, up, R_{27}), (R_{26}, down, R_{25}), (R_{26}, left, R_{23}), (R_{26}, right, R_{35}),$
  $(R_{27}, stop, R_{27}), (R_{27}, up, R_{28}), (R_{27}, down, R_{26}), (R_{27}, left, R_{22}), (R_{27}, right, R_{34}),$
  $(R_{28}, stop, R_{28}), (R_{28}, up, R_{29}), (R_{28}, down, R_{27}), (R_{28}, left, R_{21}), (R_{28}, right, R_{33}),$
  $(R_{29}, stop, R_{29}), (R_{29}, up, R_{30}), (R_{29}, down, R_{28}), (R_{29}, left, R_{20}), (R_{29}, right, R_{32}),$
  $(R_{30}, stop, R_{30}), (R_{30}, down, R_{29}), (R_{30}, left, R_{19}), (R_{30}, right, R_{31}),$
  $(R_{31}, stop, R_{31}), (R_{31}, down, R_{32}), (R_{31}, left, R_{30}),$
  $(R_{32}, stop, R_{32}), (R_{32}, up, R_{31}), (R_{32}, down, R_{33}), (R_{32}, left, R_{29}),$
  $(R_{33}, stop, R_{33}), (R_{33}, up, R_{32}), (R_{33}, down, R_{34}), (R_{33}, left, R_{28}),$
  $(R_{34}, stop, R_{34}), (R_{34}, up, R_{33}), (R_{34}, down, R_{35}), (R_{34}, left, R_{27}),$
  $(R_{35}, stop, R_{35}), (R_{35}, up, R_{34}), (R_{35}, down, R_{36}), (R_{35}, left, R_{26}),$
  $(R_{36}, stop, R_{36}), (R_{36}, up, R_{35}), (R_{36}, left, R_{25}),$
  $\}$
- $AP = \{"blue", "red", "green", "obstacle", "empty"\}$ This means each criterion checks whether the robot is in the region R or not.
- $L(r_1) = \{"red"\};\ L(r_2) = \{"green"\};\ L(r_3) = \{"obstacle"\};$
  $L(r_6) = \{"green"\};\ L(r_7) = \{"blue"\};\ L(r_8) = \{"red", "obstacle"\};$
  $L(r_9) = \{"green"\};\ L(r_{10}) = \{"obstacle"\};\ L(r_{11}) = \{"red", "obstacle"\};$
  $L(r_{19}) = \{"obstacle"\};\ L(r_{20}) = \{"red"\};\ L(r_{21}) = \{"blue"\};$
  $L(r_{22}) = \{"obstacle"\};\ L(r_{23}) = \{"blue"\};\ L(r_{25}) = \{"green"\};$

$L(r_{26}) = \{"red","obstacle"\}$; $L(r_{29}) = \{"blue","obstacle"\}$; $L(r_{31}) = \{"obstacle"\}$
$L(r_{32}) = \{"red"\}$; $L(r_{36}) = \{"blue"\}$; $L(r_{\text{rest}}) = \{"empty"\}$;
In fig. 1, L_i written inside the region means that the state $L(r_i)$ is True in that particular region

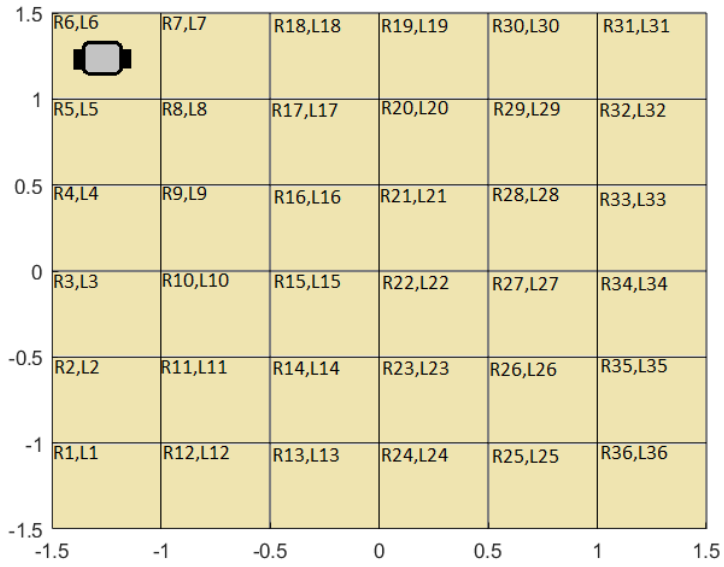


Figure 1: Workspace discretization in Rectangular regions

Task 3

The transition relation for the desired path is as follows:
(R_6,right,R_7),(R_7,right,R_{18}),(R_{18},down,R_{17}),(R_{17},right,R_{20}),[(R_{20},down,R_{21}),(R_{21},up,R_{20})]*
The path can be wirtten as: (R_6 R_7 R_{18} R_{17} [R_{20} R_{21}]*)
This is shown in the fig.(2) as follows:

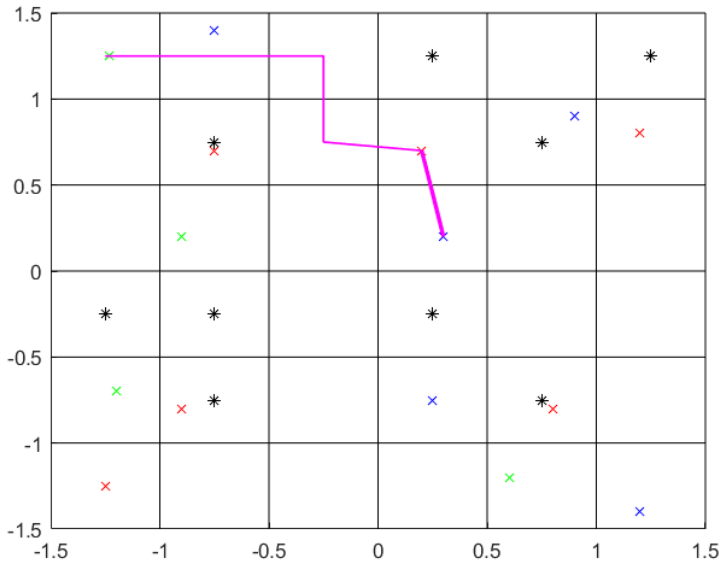


Figure 2: The 2D workspace with the path and obstacles

Task 4

During one transition, the robot must always remain in the two regions of the transition because if we simultaneously control both the relative orientation and distance of the robot to the goal region then instead of being on a straight line the robot will go move from initial points to goal points choosing a random zig-zag path. Also, since we have to move between two points as fast as possible, on simultaneous control it will take more time as compared to the normal situation. And, most importantly it is ***'safe'*** property to not give simultaneous control is because in this kind of control, the robot might collide with some obstacle in its path; causing failure.

Task 5

Since we have to design a proportional controller:

$$\omega[k] = K_\Psi(\theta^R - \theta[k]) \tag{3}$$

Given:

$$\dot{\theta} = \frac{R\omega}{L} \tag{4}$$

On using Euler Forward Algorithm, we can solve Equation (4), giving us:

$$\theta[k+h] = \dot{\theta}[k]\tau_s + \theta[k] \tag{5}$$

Substituting value of $\dot{\theta}$ from Equation(4) in Equation(6) gives us:

$$\theta[k+h] = \theta[k] + \frac{R\omega\tau_s}{L} \tag{6}$$

where τ_s is sampling time. And, we have already given controller that we have to design. Therefore,

$$\theta[k+h] = \theta[k] + \frac{R\tau_s K_\Psi}{L}(\theta^R - \theta[k]) \tag{7}$$

Simplifying Equation(8) will give us the below solution:

$$\theta[k+h] - \theta^R = [1 - \frac{R\tau_s K_\Psi}{L}](\theta^R - \theta[k]) \tag{8}$$

Now, we know for the system to be stable, the solution should exist inside the unit circle,

$$\left|(1 - \frac{RK_\Psi \tau_s}{L})\right| < 1 \tag{9}$$

$$0 < \frac{RK_\Psi \tau_s}{L} < 2 \tag{10}$$

Solving the above condition will give us the value range for K_Ψ:

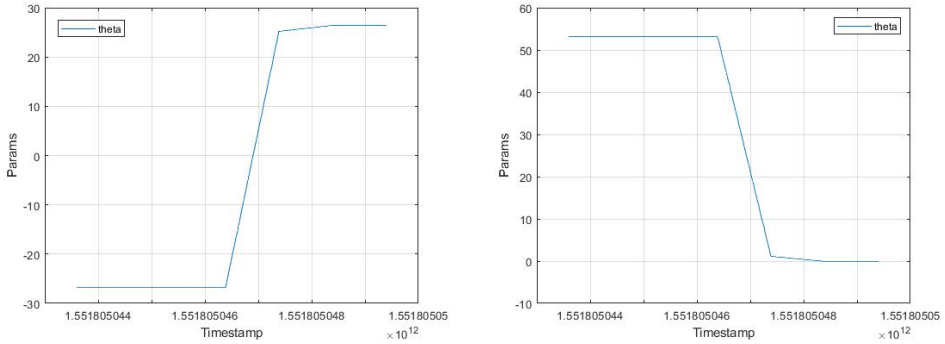
$$\boxed{0 < K_\Psi < \frac{2L}{R\tau_s}}$$

Thus, the maximum value of K_Ψ to achieve the goal is: $\frac{2L}{R\tau_s}$

Since we are dealing with proportional controller, we choose K_Ψ in order to get faster response i.e. minimum rise time, but also we need to keep in mind that we should get minimum oscillation as well. Therefore, we can choose K_Ψ by experimenting different values and hence keep on checking value for least error.

Task 6

On running the online simulator with the above controller, we see that the robot stays the x0 and y0 position and rotates and reaches the reference angle of the goal. When we change the reference goal the robot can be seen rotating. It can be seen that the robot does not move transitionally in any direction. In the simulation the robot is first placed at node 1 and final goal at node 6. The new start location is given as node 7. on setting the new position and the reference the robot rotates and aligns itself with the new reference direction. We can see that the error in the θ of the robot and the θRs reduces to zero.



(a) Theta v/s timestamp (b) Error in theta v/s timestamp

Figure 3: **Plots for evaluation of Rotation controller performance**

Task 7

For this task, we are considering a given assumption that $\theta[k+1] = \theta[k]$ i.e. θ does not change with time. We have to design the proportional controller:

$$v[k] = K_\omega d_0[k] \tag{11}$$

The equation of $d_0[k]$ is given by:

$$d_0[k] = cos(\theta[k])(x_0 - x[k]) + sin(\theta[k])(y_0 - y[k]) \tag{12}$$

Solving the Equation(12) by substituting the values of $x[k]$ and $y[k]$, with unit time shift, it will give us the below equation:

$$d_0[k] = (1 - \tau_s R K_\omega) d_0[k-1]$$
$$d_0[k+1] = (1 - \tau_s R K_\omega) d_0[k]$$

Now, we know for the system to be stable, the solution should exist inside the unit circle,

$$|(1 - \tau_s R K_\omega)| < 1$$
$$-1 < (1 - \tau_s R K_\omega) < 1$$
$$0 < \tau_s R K_\omega < 2$$

$$\boxed{0 < K_\omega < \frac{2}{R\tau_s}}$$

Thus, the maximum value of K_ω for $d_0[k]$ to converge to 0 is = $\dfrac{2}{R\tau_s}$

Since we are dealing with proportional controller, we choose K_ω in order to get faster response i.e. minimum rise time, but also we need to keep in mind that we should get minimum oscillation as well. Therefore, we can choose K_ω by experimenting different values and hence keep on checking value for least error.

Task 8

When executing the online simulator for this system, we first placed the robot at node 1 and gave the new start location as node 2 and ran the controller, the robot moves to node 2 and similar to node 3. While testing we gave the set point as node 2 and while the robot was moving from node 3, we immediately changed the set point back to node 1 and the robot came back to node 1. Hence the controller was working as expected.
Here in this case the robot was moved along a straight line i.e. by keeping the direction of motion along the θ the robot is pointed towards. If we change the theta and try to move the robot then it does not reach its location (x0, y0) as its rotational component is zero here. So to run this controller the heading of the robot must be along the goal/start location.

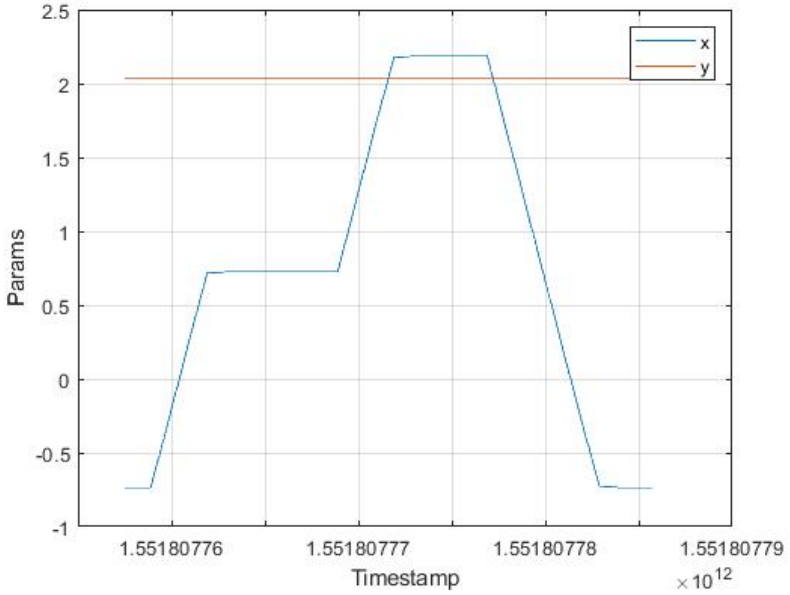
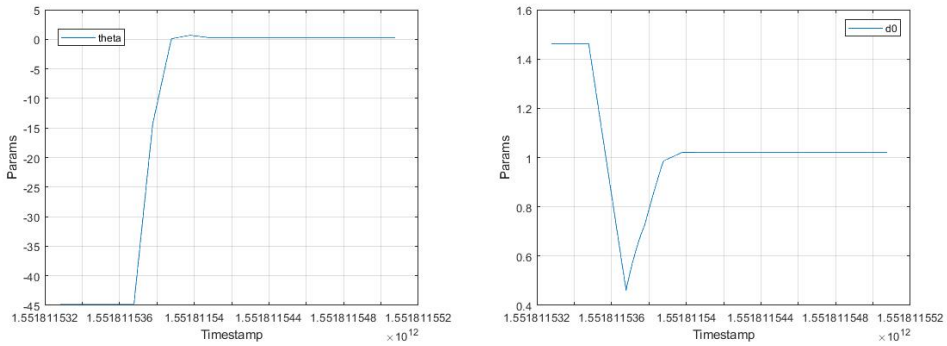


Figure 4: Evaluation of robustness of the controller to hold its start position

Task 9

On running both the controller together, we can see that the robot rotates and aligns itself w.r.t the reference line. The second controller reduces the error between the start set point and the robot location. The error plot for the rotational controller and the transnational controller can be seen in the graphs (figure 5).



(a) Error in Theta v/s timestamp (b) d0 v/s timestamp

Figure 5: **Plots for evaluation of controller performance**

When testing both the controllers together, we pushed the robot away from its initial position, while rotating, to test the robustness of the algorithm. We observed that the robot moves towards the start position to reduce the translational error and at the same time the rotational controller tries to rotate the robot towards the reference angle. After stabilizing the robot stops with some translational error although it has aligned itself along the reference angle. From the (b) plot we can see that the d0 asymptotically becomes stable at the value of 1 but due to the error in its position it does not become 0.(See

fig 6). This can be because, in order to reach its initial position, the robot will have to change its orientation first towards the goal location and then re-orient itself to the desired orientation which the rotational controller does not allow it to do. This can be avoided if the load change(the disturbance in placement of the robot) is of small magnitude or the Velocity controller is able to reach the initial position before the rotational controller reaches its steady state.

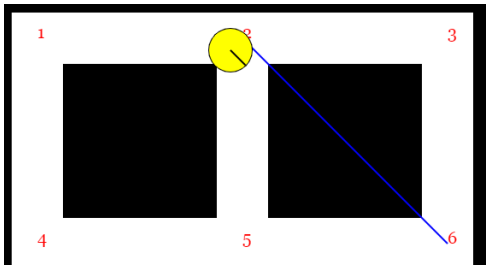


Figure 6: Location of the robot in the simulator

Task 10

For this task, we are considering a given assumption that $\theta[k] = \theta_g$ i.e. the robot is headed correctly. We have to design the proportional controller:

$$v[k] = K_\omega d_g[k] \tag{13}$$

The equation of $d_g[k]$ is given by:

$$d_g[k] = cos(\theta[k])(x_g - x[k]) + sin(\theta[k])(y_g - y[k]) \tag{14}$$

Using Euler forward equations that we have solved before i.e.

$$x[k+h] = x[k] + \tau_s R v cos\theta$$
$$y[k+h] = y[k] + \tau_s R v sin\theta$$

Solving the Equation(12) by substituting the above values, with unit time shift, it will give us the below equation:

$$d_g[k] = (1 - \tau_s R K_\omega) d_g[k-1]$$
$$d_g[k+1] = (1 - \tau_s R K_\omega) d_g[k]$$

Now, we know for the system to be stable, the solution should exist inside the unit circle,

$$|(1 - \tau_s R K_\omega)| < 1$$
$$-1 < (1 - \tau_s R K_\omega) < 1$$
$$0 < \tau_s R K_\omega < 2$$

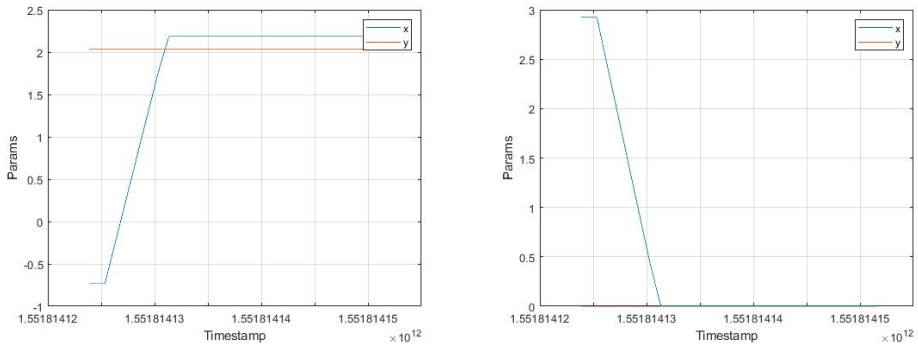
$$\boxed{0 < K_\omega < \frac{2}{R\tau_s}}$$

Thus, the maximum value of K_ω for $d_g[k]$ to converge to 0 is $= \frac{2}{R\tau_s}$

Since we are dealing with proportional controller, we choose K_ω in order to get faster response i.e. minimum rise time, but also we need to keep in mind that we should get minimum oscillation as well. Therefore, we can choose K_ω by experimenting different values and hence keep on checking value for least error.

Task 11

On running the online simulator we can see that the robot starts from node 1 to move towards node 3. It starts slowly and maintains the speed and as seen in the parameter d_g, the terms (x_g-x) and (y_g-y) will tend to become smaller as the robot approaches the goal and simultaneously reduce the speed of the robot. Eventually coming to a full stop at the goal coordinates of node 3. The change in the coordinate position of the robot and the error plot can be seen the fig 7 :



(a) Position v/s timestamp (b) Error in Position v/s timestamp

Figure 7: **Plots for evaluation of Transnational controller performance**

The error in position can be seen as max when the robot is at node 1 and the goal is at node 3. As it comes closer the error decreases to zero and the robot successfully reaches the goal location. Since the robot needs to be in the heading towards the goal location for this controller to work, the start and the goal position are taken as node 1 and node 3 for convenience.

Task 12

By taking into consideration that $\theta[k]$ is close to θ_g, we can write:

$$d_p[k] \approx p(\theta_g - \theta[k]) \tag{15}$$

Thus, we can also write $d_p[k+1]$ i.e.

$$d_p[k+1] \approx p(\theta_g - \theta[k+1]) \tag{16}$$

We have already solved for $\theta[k+1]$ and controller is given as:

$$\omega[k] = K_\Psi d_p[k] \tag{17}$$

Substituting the values in Equation(16) will give us,

$$d_p[k+1] = p(\theta_g - \theta[k] - \frac{\tau_s R K_\Psi d_p[k]}{L}) \tag{18}$$

Solving the Equation(18) will give us the following equation:

$$d_p[k+1] = [1 - \frac{p\tau_s R K_\Psi}{L}] d_p[k] \tag{19}$$

To be stable, the poles must lie within the unit circle:

$$\left|1 - \frac{p\tau_s R K_\Psi}{L}\right| < 1 \implies -1 < 1 - \frac{p\tau_s R K_\Psi}{L} < 1$$

$$0 < \frac{p\tau_s R K_\Psi}{L} < 2 \implies 0 < K_\Psi < \frac{2L}{pR\tau_s}$$

The maximum value of K_Ψ for $d_p[k]$ to converge to 0 is =

$$\boxed{K_\Psi < \frac{2L}{pR\tau_s}} \tag{20}$$

Task 13

We got the below condition on solving for K_Ψ :

$$K_\Psi < \frac{2L}{pR\tau_s}$$

Since we have designed a proportional controller, the value of p decides the robot's ability to follow the line.There are two possible conditions for p :

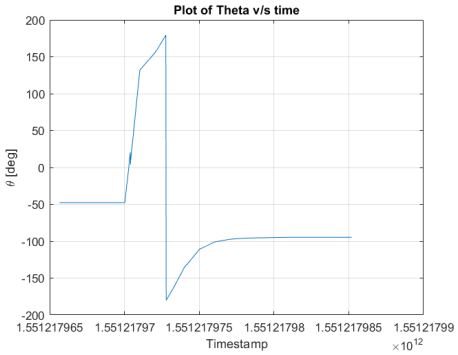
- When we take larger value of p that means K_Ψ will have ample range of values and hence, the robot's ability to follow the line will be better i.e. system is stable in this scenario
- On the other hand, on taking smaller value of p, system will not be stable and the robot's ability to follow the line will be poor

Therefore, in order to consider all the cases in mind, we should tune p for the best performance while working with the controller.

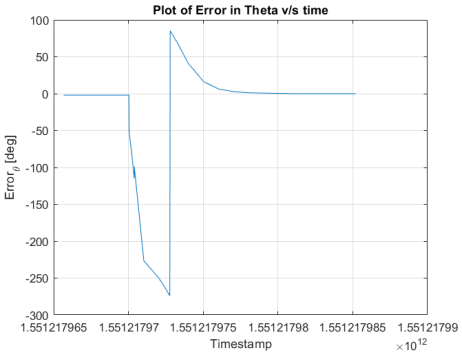
Task 14

On running the simulator we see that the robot rotates continuously at its location and when a reference location is given, the robot aligns itself with respect to the angle of the goal. The first node is the start point and the 5 as the goal. The robot aligns itself to the reference line. Now the start node is changed to node 2. On changing the reference line, we note that the robot starts rotating continuously and hence the variation can be seen in the curve of error. Now when the position of the robot is set to node 2. It stabilizes

itself as the reference line is found.
The main reason for this to happen is that the controller is only controlling the rotational velocity, keep the robot close to the straight line connecting x0,y0 and xg,yg. When it is not able to find the reference line it starts behaving erratically and when it finds the straight line it corrects itself and aligns in the right direction.



(a) Theta v/s timestamp



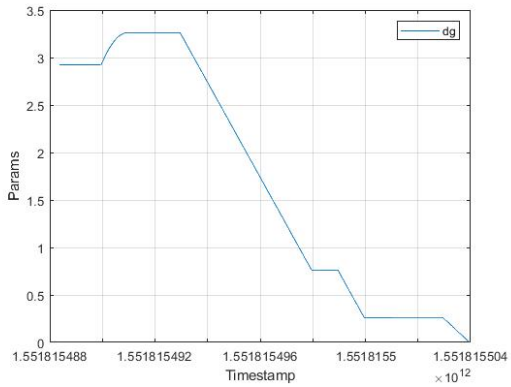
(b) Error in theta v/s timestamp

Figure 8: **Plots for evaluation of Rotation controller performance**

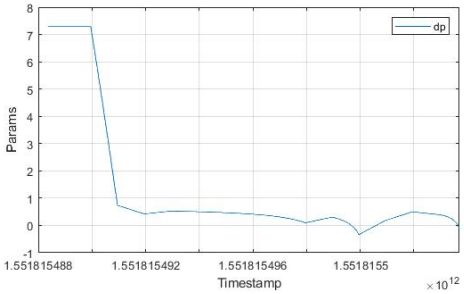
The value of the θg is -44.7degrees when the start is node 1 and goal is node 5. On changing the start node to 2 but not moving the robot to that location we can see that the θ changes continuously indicating continous rotation of the robot. When robot position is set to node 2, gradual change can be see and the error reduced to zero. Here the value of the θg is -90 degrees when the start is node 2 and goal is node 5.

Task 15

Controller can fix the small deviations of robot from desired straight line and heads towards the goal. A difference that exists now between this case and the cases where only one controller is active is that here the action from one controller acts as a disturbance to the other controller. Therefore, there are small levels of noise observed in the responses.



(a) Dg v/s timestamp



(b) Dp v/s timestamp

Task 16

The Hybrid Controller is modelled using Hybrid Automaton of 8 tuple:

$$H = (Q, X, Init, f, D, E, G, R)$$

here:

- $Q \equiv \{q1, q2, q3\}$ which denotes the set of discrete states. The state q1 represents rotational controller, q2 corresponds translational controller, and q3 represents stop condition.
- $Init \equiv \{q3\}$ denotes the initial state, in this case we have taken the stop condition.
- $X \equiv \{(x, y, \theta) \in \mathbb{R}^3 : \theta \in (-180°, 180°]\}$ denotes the continuous state space.
- Vector fields f given by:

$$f(q1, X) = \begin{bmatrix} \dot{x} = Rvcos\theta \\ \dot{y} = Rvsin\theta \\ \dot{\theta} = \dfrac{R\omega}{L} \end{bmatrix}$$

$$f(q2, X) = \begin{bmatrix} \dot{x} = Rvcos\theta \\ \dot{y} = Rvsin\theta \\ \dot{\theta} = \dfrac{R\omega}{L} \end{bmatrix}$$

$$f(q3, X) = \begin{bmatrix} \dot{x} = 0 \\ \dot{y} = 0 \\ \dot{\theta} = 0 \end{bmatrix}$$

- D shows conditions for automaton to stay in a state.
  $D(q1) = \{(\text{x,y},\theta) : x, y \in \mathbb{R}^2, |\theta_g - \theta| \ > \delta\}$
  $D(q2) = \{(\text{x,y},\theta) : x, y \in \mathbb{R}^2, \sqrt{(x_g - x)^2 + (y_g - y)^2} \ > \xi\}$
  $D(q3) = \{x, y \in \mathbb{R}^2, \sqrt{(x_g - x)^2 + (y_g - y)^2} \sim 0 \ \}$

- E: Edges show possible transitions.
  $E1 = \{q1, q2\}$
  $E2 = \{q2, q3\}$
  $E3 = \{q3, q1\}$
  $E = E1 \cup E2 \cup E3$

- $G : G$ is the guard conditions.
  $G(\{q1, q2\}) = \{|\theta_g - \theta| \ \leq \delta\}$
  $G(\{q2, q3\}) = \{\sqrt{(x_g - x)^2 + (y_g - y)^2} \ \leq \xi\}$
  $\text{G}(\{\text{q3,q1}\}) = \{\sqrt{(x_g - x)^2 + (y_g - y)^2} \ > \xi\}$

- R: denotes the reset map.
  $R = \{x, y, \theta\}$

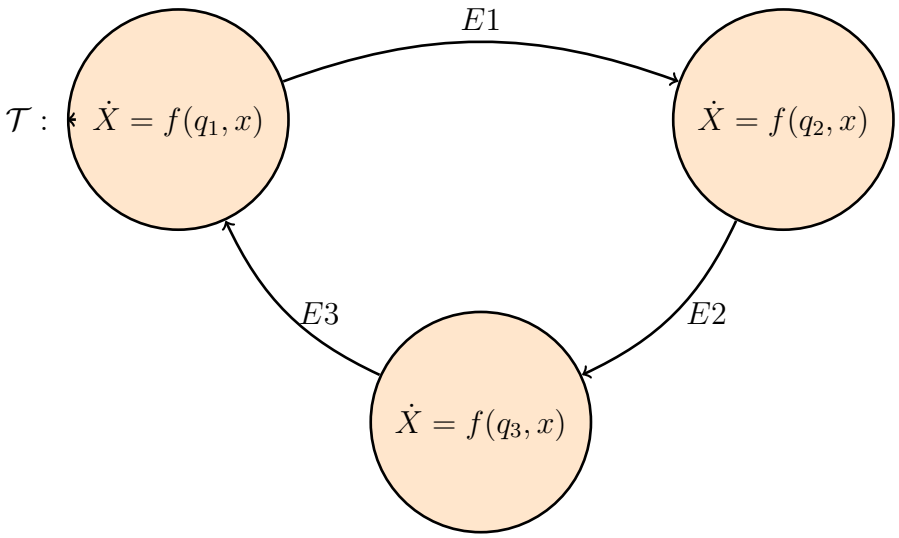
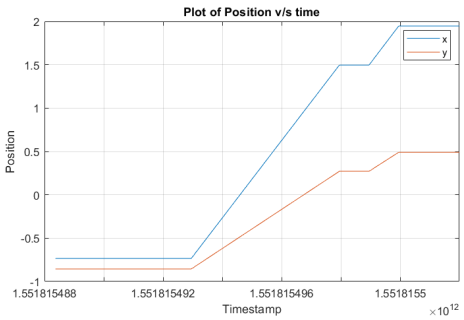


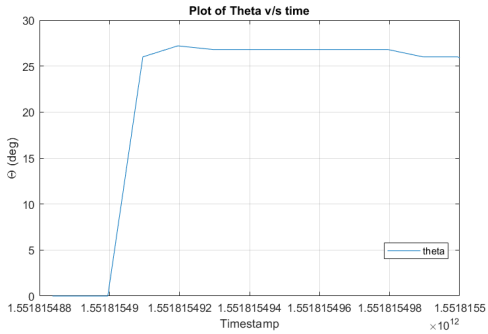
Figure 10: Product Interleaving Transition Systems $\mathcal{T}_{\text{int}}$.

Task 17

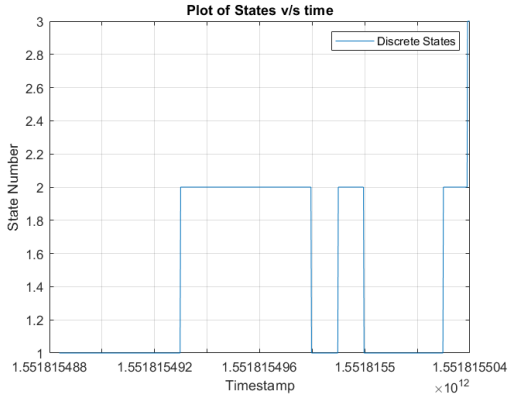
On running the simulation we can see that the robot moves successfully from the node 7 to node 6. It first corrects the rotational error and then moves the goal by using the go-to goal controller. On translational motion, some error is introduced in the alignment of the robot, which the rotational controller rectifies and aligns it again in the direction of the goal.



(a) Position v/s timestamp



(b) theta v/s timestamp



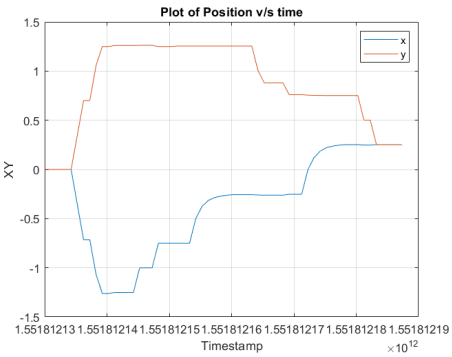
(c) Discrete State v/s timestamp

Figure 11: **Variation in the Position, theta and State of the robot when hybrid controller is implemented wrt timestamp**

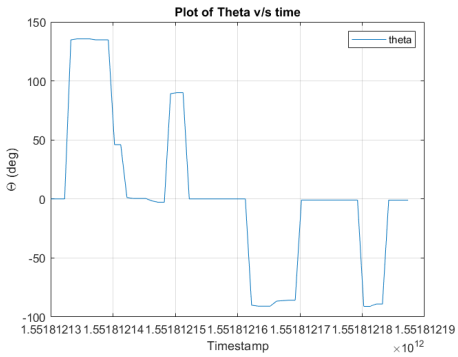
The Value of θg is 26.33deg and we can see from the simulated graphs as well that the robot reaches 26.33deg first and stabilizes itself till timestamp 1551815493 and then starts moving translationally. We note that around 1551815498 the state of the system again changes to rotational state in order to correct the small angle error. Here in the theta graph we cannot see any significant change which might be due to a very small correction in the rotational factor. Then the state again changes at 1551815 back to translational and reaches the goal.

Task 18

The way points from the Task 3 are taken as follows: [0;0;-1.25;1.25] [-1.25;1.25;-0.75;1.25] [-0.75;1.25;-0.25;1.25] [-0.25;1.25;-0.25;0.75] [-0.25;0.75;0.25;0.75] [0.25;0.75;0.25;0.25] The trajectory of the robot in the xy-plane while following all the way points one after the other can be seen in the figure (12)



(a) Position v/s timestamp



(b) theta v/s timestamp

Figure 12: **Trajectory of Position and Orientation for Way point Control**

The robot follows the desired trajectory by first orienting itself at the desired theta and then translates itself to reach the goal. In this way it ensures that safety condition is met and the robot does not venture into any BAD region.

Task 19

In case of hybrid controller, it ensures that robot must stay in the stated two regions i.e. the regions between which the robot is transitioning, and not go into any undetermined states. For the same, first the controller checks the orientation of the robot and ensure that before the straight line movement it is at the correct orientation, After the initial process, Go-To-Goal controller will ensure that when the robot is at the desired orientation, it should head in a straight line to the desired point in the transition goal region and it should not deviate from the line connecting the two points. This performance ensure the safe transition of the robot between two regions.

References

[1] Hassan K Khalil. *Nonlinear systems*. Prentice Hall, Upper Saddle river, 3. edition, 2002. ISBN 0-13-067389-7.

[2] Tobias Oetiker, Hubert Partl, Irene Hyna, and Elisabeth Schlegl. *The Not So Short Introduction to LaTeX 2ε*. Oetiker, OETIKER+PARTNER AG, Aarweg 15, 4600 Olten, Switzerland, 2008. http://www.ctan.org/info/lshort/.

[3] Shankar Sastry. *Nonlinear systems: analysis, stability, and control*, volume 10. Springer, New York, N.Y., 1999. ISBN 0-387-98513-1.