# Homework 3 in EL2450 Hybrid and Embedded Control Systems

Nadine Drollinger
890120-5420
nadined@kth.se

Alexandros Filotheou
871108-5590
alefil@kth.se

Roberto Sánchez-Rey
840616-9139
rosr@kth.se

Emma Thilén
930715-5466
ethilen@kth.se

## Task 1

Given the translational and rotational inputs $u_\omega$ and $u_\omega$, the individual wheel inputs can be computed by

$$\begin{aligned} u_\omega &= \frac{u_r + u_l}{2} \\ u_\Psi &= u_r - u_l \end{aligned} \quad \Leftrightarrow \quad \begin{aligned} u_l &= u_\omega - \frac{u_\Psi}{2} \\ u_r &= u_\omega + \frac{u_\Psi}{2} \end{aligned}$$

## Task 2

On the basis of the continuous state equations for translation.

$$\begin{aligned} \dot{x} &= R\, u_\omega\, cos\theta \\ \dot{y} &= R\, u_\omega\, sin\theta \end{aligned} \tag{1}$$

Since the robot itself uses centimeters as reference distance unit, the given 200 value for the speed turns to be: $u_\omega = 2\,[m/s]$. Furthermore, assuming:

1. $\dot{x} = \frac{\Delta x}{\Delta t}$
2. $\dot{y} = \frac{\Delta y}{\Delta t}$
3. $\Delta x = x_f - x_o$; $\Delta y = y_f - y_o$ $\Delta t = t_f - t_o$

and multiplying each term of (1) by itself and adding them toghether, one gets the following equation that can be solved with values of the variations in position coordinates and time, plus the above mentioned value of the speed.

$$\left.\begin{aligned} R &= \sqrt{\frac{1}{u_\omega^2}\left[\left(\frac{\Delta x}{\Delta t}\right)^2 + \left(\frac{\Delta y}{\Delta t}\right)^2\right]} \\ \Delta x &= 1.3\,[m] \\ \Delta y &= -0.07\,[m] \\ \Delta t &= 6.4250\,[s] \\ u_\omega &= 2\,[m/s] \end{aligned}\right\} R = 0.1013$$

For the calculation of L, the equation for the robot pose is used.

$$\dot{\theta} = \frac{R}{L} u_\Psi$$

As earlier, we have considered $\dot{\theta} = \frac{\Delta\theta}{\Delta t}$. From file `Rotation.cvs`, created for $u_\Psi = 400\,[cm/s]$ one gets $\Delta\theta = 150.27°$ , $\Delta t = 4.1667\,[s]$. Hence:

$$L = \frac{\Delta t}{\Delta\theta} R\, u_\Psi = 0.5113$$

## Task 3

In order to reach a conclusion about the stability of the angular displacement, it suffices to find a Lyapunov function $V(x)$ such that $V(0) = 0$, $V(x) > 0$ for all $x \neq 0$ and $\dot{V}(x) \leq 0$ for all x. Considering $x = \theta - \theta^G$ and $V(x) = x^2$:

$$V(0) = 0,\; V(x) > 0,\; \text{for all } x \neq 0, \text{ and}$$

$$\begin{aligned} \dot{V}(x) &= 2x\dot{x} = 2(\theta - \theta^G)\dot{\theta} \\ &= 2(\theta - \theta^G)\frac{R}{L}u_\Psi \end{aligned}$$

- When $\theta - \theta^G \leq 0$, $\dot{V}(x) = 2(\theta - \theta^G)\frac{R}{L} \leq 0$
- When $\theta - \theta^G > 0$, $\dot{V}(x) = -2(\theta - \theta^G)\frac{R}{L} < 0$

Hence $\dot{V}(x) \leq 0$ for all $x$, meaning that the system is stable for all $\theta \in (-180°, 180°]$.

Practically, as one can see in **??**, the system is stable but it exhibits Zeno behaviour, which is defined as an infinite number of switches in finite time.

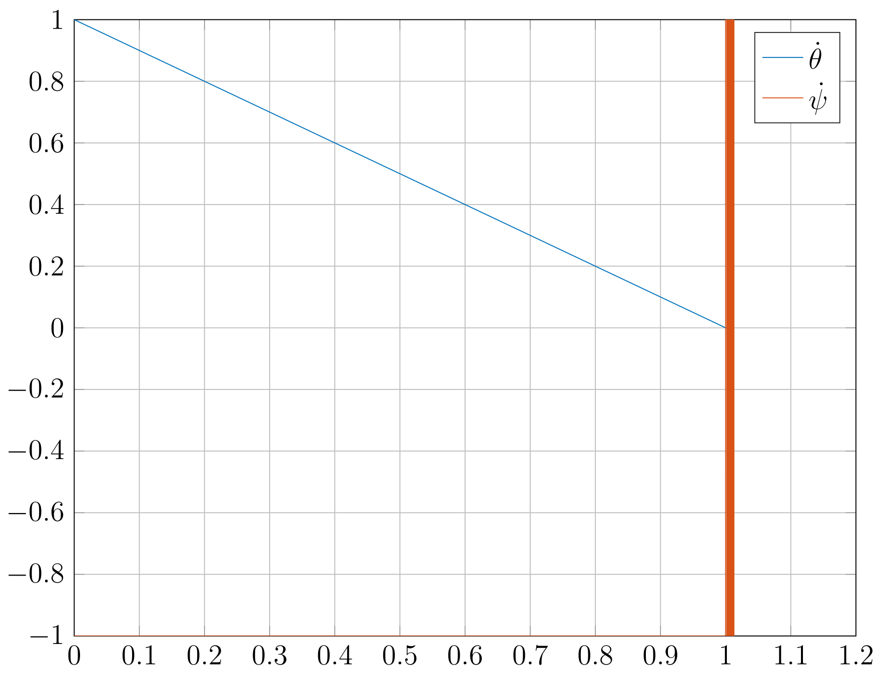


Figure 1

## Task 4

Examining file `rot2.slx` one sees that $\dot{\theta} = R/Lu_\Psi$ is stable. We can verify this analytically:

$$\begin{aligned}
\dot{u}_\Psi &= K_L(u^R_\Psi - u_\Psi)\\
\dot{u}_\Psi &= K_L u^R_\Psi - K_L u_\Psi\\
\dot{u}_\Psi \cdot e^{K_L t} &= K_L u^R_\Psi \cdot e^{K_L t} - K_L u_\Psi \cdot e^{K_L t}\\
\dot{u}_\Psi \cdot e^{K_L t} + K_L u_\Psi \cdot e^{K_L t} &= K_L u^R_\Psi \cdot e^{K_L t}\\
\frac{d}{dt}(u_\Psi e^{K_L t}) &= K_L u^R_\Psi e^{K_L t}\\
u_\Psi e^{K_L t} &= \int K_L u^R_\Psi e^{K_L t} = u^R_\Psi e^{K_L t} + \lambda\\
u_\Psi &= u^R_\Psi + \lambda e^{-K_L t}
\end{aligned}$$

Where, from inspection of the parameters of the integrator, the initial condition is $\lambda = 1$. Hence

$$u_\Psi(t) = \begin{cases} 1 + e^{-K_L t} & \theta - \theta^G \leq 0 \\ -1 + e^{-K_L t} & \theta - \theta^G > 0 \end{cases} \tag{2}$$

In order to reach a conclusion about the stability of the angular displacement, it suffices to find a Lyapunov function $V(x)$ such that $V(0) = 0$, $V(x) > 0$ for all $x \neq 0$ and $\dot{V}(x) \leq 0$ for all x. Considering $x = \theta - \theta^G$ and $V(x) = x^2$:

$$V(0) = 0,\ V(x) > 0,\ \text{for all } x \neq 0, \text{ and}$$

$$\begin{aligned}\dot{V}(x) &= 2x\dot{x} = 2(\theta - \theta^G)\dot{\theta} \\ &= 2(\theta - \theta^G)\frac{R}{L}u_\Psi \\ &= 2(\theta - \theta^G)\frac{R}{L}(u_\Psi^R + e^{-K_L t})\end{aligned}$$

- When $\theta - \theta^G \leq 0$, $\dot{V}(x) = 2(\theta - \theta^G)\frac{R}{L}(1 + e^{-K_L t}) \leq 0$
  With

$$\begin{aligned}0 < e^{-K_L t} \leq 1 \\ 0 < 1 < 1 + e^{-K_L t} \leq 2\end{aligned} \tag{3}$$

- When $\theta - \theta^G > 0$, $\dot{V}(x) = 2(\theta - \theta^G)\frac{R}{L}(-1 + e^{-K_L t}) < 0$
  With

$$\begin{aligned}0 < e^{-K_L t} \leq 1 \\ -1 < -1 + e^{-K_L t} \leq 0\end{aligned} \tag{4}$$

Hence, from inequalities **??** and **??** we conclude that $\dot{V}(x) \leq 0$ for all $x$, meaning that the system is stable for all $\theta \in (-180°, 180°]$.

From what one can see in the continuous and discrete state trajectory plot in figure **??**, one can say that the system enters into zeno behaviour, even though the system is asymptotically stable as $\dot{\theta} = R/Lu_\Psi$ goes to zero. So one can conclude that this control law is not practical as the discrete part enters zeno-behaviour even though the continuous part is asymptotically stable.

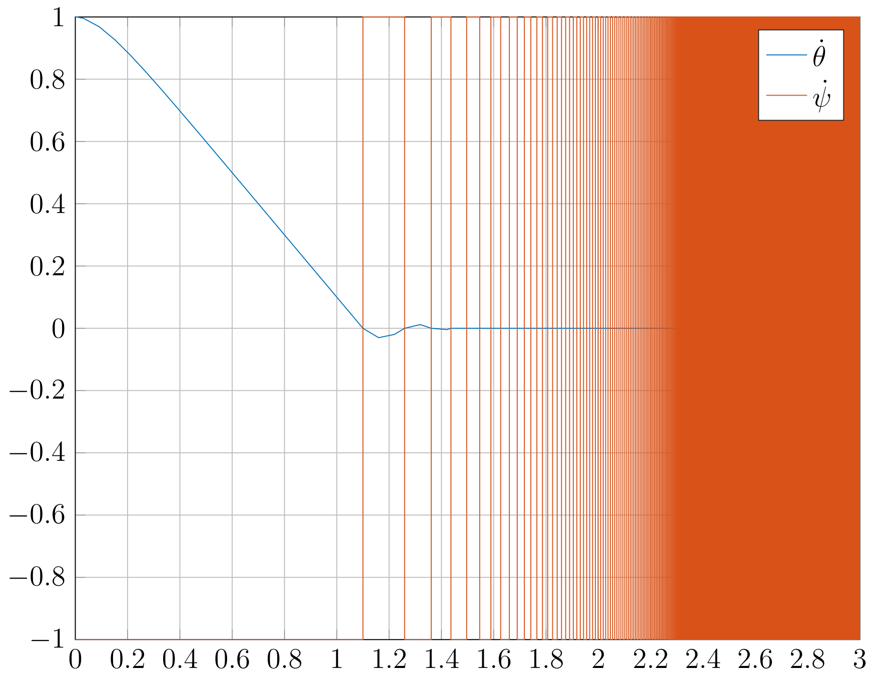


Figure 2

## Task 5

From what we can see in the continuous and discrete state trajectory plot in figure ??, there is no zeno behaviour. There may be an infinite amount of state switches, but they do not happen in an finite amount of time. The zero-order-hold implemented in this system holds the current value until the next sampling time instant. Therefore, the system's reaction time is limited by the ZOH and therefore it can not exhibit zeno behaviour.

$\dot{\theta} = R/Lu_{\Psi}$ is marginally stable, as it does not go to zero but as well not to instability.

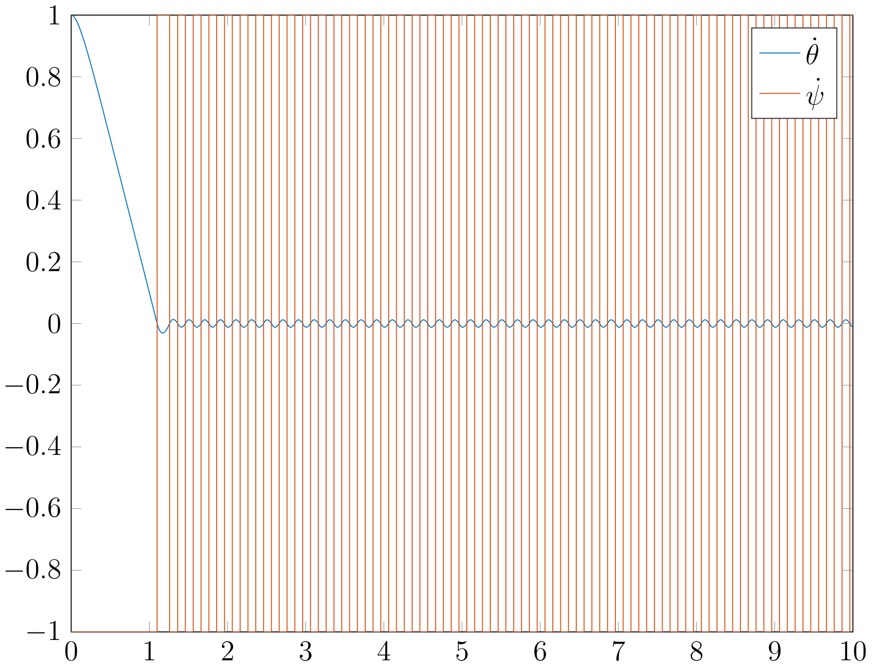


Figure 3

## Task 6

The system dynamics can be discretized using the Euler forward method. With sampling time $T_s$, the derivatives can then be approximated by

$$\frac{d}{dt}\chi(t) = \frac{q-1}{T_s}\chi(t) \tag{5}$$

Applying equation ?? to the continuous state-space equations gives

$$\frac{q-1}{T_s}x(t) = Ru_{\omega}(t)cos(\theta(t))$$
$$\frac{q-1}{T_s}y(t) = Ru_{\omega}(t)cos(\theta(t))$$
$$\frac{q-1}{T_s}\theta(t) = \frac{R}{L}u_{\psi}(t)$$

Multiplying both sides by $T_s$ together with some rearranging after letting the shift operator act upon the variables yields

$$
\begin{aligned}
x(t+T_s) &= x(t) + T_s R u_\omega(t) cos(\theta(t)) \\
y(t+T_s) &= y(t) + T_s R u_\omega(t) cos(\theta(t)) \\
\theta(t+T_s) &= \theta(t) + \frac{T_s R}{L} u_\psi(t)
\end{aligned}
$$

Letting $x[k], y[k], \theta[k]$ and $u_\omega[k], u_\psi[k]$ denote the robot's states and the control input respectively at time $t = kT_s, k = 0, 1, 2...$, the discretized system can be written as

$$
\begin{aligned}
x[k+1] &= x[k] + T_s R u_\omega[k] cos(\theta[k]) \\
y[k+1] &= y[k] + T_s R u_\omega[k] cos(\theta[k]) \\
\theta[k+1] &= \theta[k] + \frac{T_s R}{L} u_\psi(t)
\end{aligned}
$$

## Task 7

$$
\begin{aligned}
\theta[k+1] &= \theta[k] + \frac{T_s R}{L} u_\Psi^R[k] \\
&= \theta[k] + \frac{T_s R}{L} K_\Psi^R(\theta^R - \theta[k]) \\
&= \theta[k](1 - \frac{T_s R}{L} K_\Psi^R) + \frac{T_s R}{L} K_\Psi^R \theta^R
\end{aligned}
$$

By subtracting $\theta^R$ from both sides one gets

$$
\begin{aligned}
\theta[k+1] - \theta^R &= \theta[k](1 - \frac{T_s R}{L} K_\Psi^R) + (\frac{T_s R}{L} K_\Psi^R - 1)\theta^R \\
&= (\theta[k] - \theta^R)(1 - \frac{T_s R}{L} K_\Psi^R) \qquad (6)
\end{aligned}
$$

We now define state $\theta'$ as

$$\theta'[k] = \theta[k] - \theta^R$$

Then, equation **??** becomes

$$\theta'[k+1] = \theta'[k](1 - \frac{T_s R}{L} K_\Psi^R)$$

In order for this system to be stable, that is, $|\theta - \theta^R| \to 0$ as $k \to \infty$, the solution of the equation inside the parentheses should lie inside the unit circle:

$$\begin{aligned}
\left|1-\frac{T_sR}{L}K^R_\Psi\right| &< 1\\
-1 < 1-\frac{T_sR}{L}K^R_\Psi &< 1\\
-2 < -\frac{T_sR}{L}K^R_\Psi &< 0\\
0 < \frac{T_sR}{L}K^R_\Psi &< 2\\
0 < K^R_\Psi &< \frac{2L}{T_sR} \qquad (7)
\end{aligned}$$

Hence, the maximum value $K^R_\Psi$ can take for the system to be marginally stable is $K^R_{\Psi,max} = \dfrac{2L}{T_sR}$

Theoretically, the value of $K^R_\Psi$ can be chosen to be any value inside the interval defined in inequality **??**. However, first, it would be wise to choose a value that is far enough from the maximum value so as to avoid overshoot, but close enough to it, so that convergence happens in reasonable time. Hence, in practice, it is reasonable that one would need to experiment with different values and choose one that results in balancing a small angular error, a minimal overshoot, if any, and a quick enough settling time.

## Task 8

For the purpose of simulating this part of the controller, the initial point of the robot was taken to be $I(x_0, y_0) \equiv (0,0)$. The goal was set to $G(x_g, y_g) \equiv (-0.37, 1.68)$, which is node 1 in the simulation environment. The angle between the line connecting $I$ and $G$ and the x-axis is hence $\theta_R = tan^{-1}(1.68/-0.37) = 102.42$ degrees.

Figures **??**, **??**, **??**, **??**, **??** show the angular response of the robot for different values of $K^R_\Psi$ inside the interval set by inequality **??**. Figure **??** verifies that the upper limit for $K^R_\Psi$ is indeed $K^R_{\Psi,max} = \frac{2L}{T_sR}$ by showing that the angular response of the robot cannot converge for $K^R_\Psi > K^R_{\Psi,max}$.

Here, one can see that the smaller the value of $K^R_\Psi$ is, the larger the settling time, the lower the rise time and the smoother the response is. However, as the value of $K^R_\Psi$ increases, the steady-state response begins to oscillate, with the amplitude of this oscillation proportional to the value of $K^R_\Psi$.

Figures **??**, **??**, **??**, **??** and **??** focus on the steady-state value of the aforementioned responses. As it is evident, none of the responses converge to the value $\theta_R = 102.42$. This is reasonable since with only a purely proportional control signal, as the angular error, i.e. $e(\theta) = \theta^R - \theta$, tends to zero, the product of $K^R_\Psi$ and $e(\theta)$ isn't large enough to force the robot to rotate exactly $\theta^R$ degrees.

Another way to look at this is by looking at the steady-state response of the system, which is linear, for a step input of magnitude $\theta^R$. Figure **??** shows the structure of the system. The z-transform of the input is then $R(z) = \dfrac{\theta^R}{1-z^{-1}}$ and the equation of the closed-loop system is

$$Y(z) = \frac{K^R_\Psi G(z)}{1 + K^R_\Psi G(z)} R(z)$$

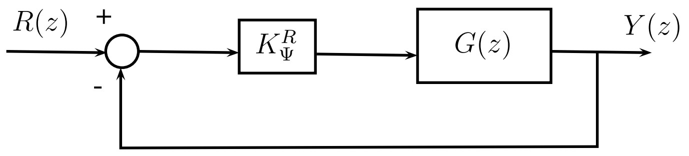


Figure 4: The structure of the system under rotational control. $G(z)$ is the discretized transfer function of the linear system whose state-space equation is $\dot{\theta} = \frac{R}{L} u_\Psi$

The steady-state response is

$$lim_{t\to\infty} y(t) = lim_{z\to 1}(1 - z^{-1}) \frac{\theta^R}{1 - z^{-1}} \frac{K_\Psi^R G(z)}{1 + K_\Psi^R G(z)} = \theta^R \cdot lim_{z\to 1} \frac{K_\Psi^R G(z)}{1 + K_\Psi^R G(z)}$$

The steady-state response cannot reach exactly $\theta^R$ as the above limit cannot converge to 1 under our limitations for $K_\Psi^R$ and the dynamics of $G(z)$.

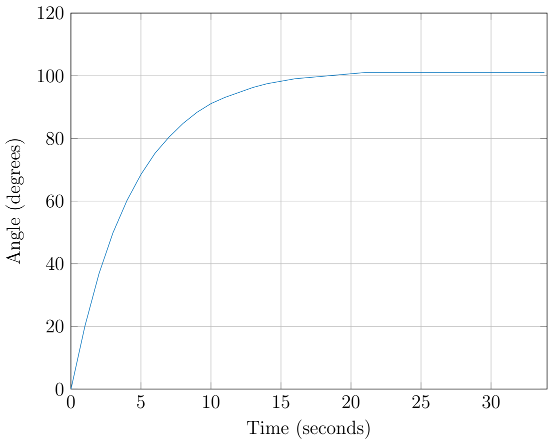


Figure 5: The orientation of the robot over time for $K_\Psi^R = 0.1 K_{\Psi,max}^R$

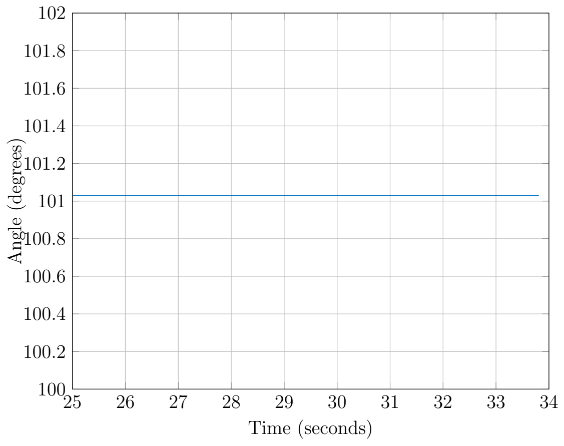


Figure 6: The steady state orientation of the robot for $K_\Psi^R = 0.1 K_{\Psi,max}^R$

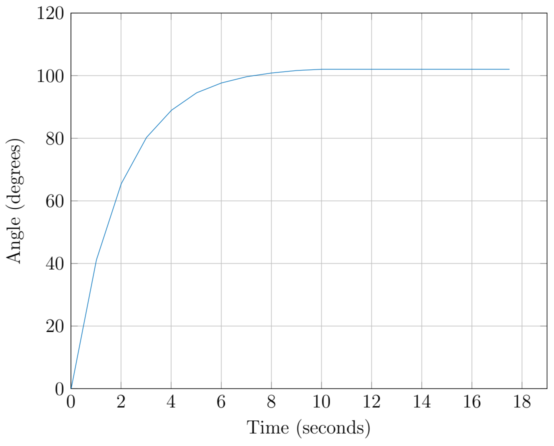


Figure 7: The orientation of the robot over time for $K_\Psi^R = 0.2 K_{\Psi,max}^R$

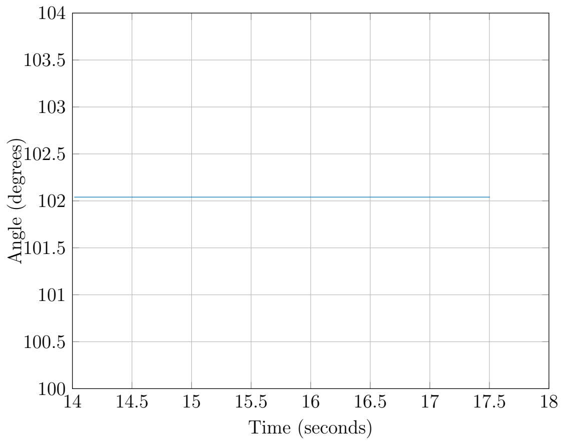


Figure 8: The steady state orientation of the robot for $K_\Psi^R = 0.2 K_{\Psi,max}^R$

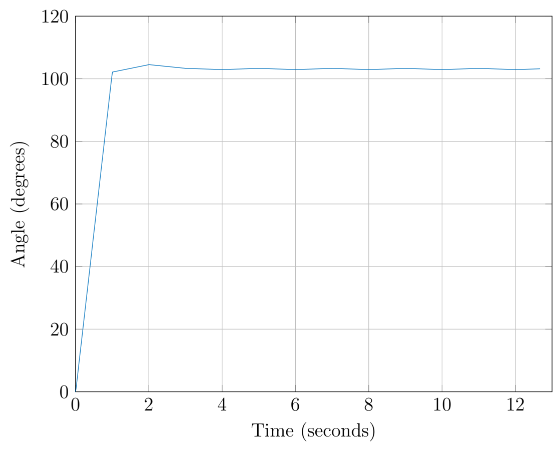


Figure 9: The orientation of the robot over time for $K_{\Psi}^{R} = 0.5K_{\Psi,max}^{R}$

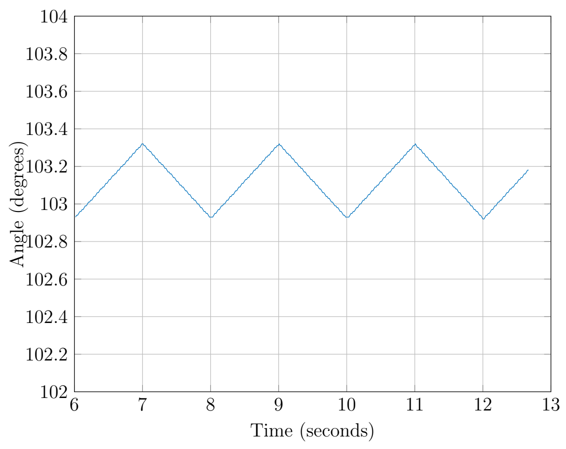


Figure 10: The steady state orientation of the robot for $K_{\Psi}^{R} = 0.5K_{\Psi,max}^{R}$

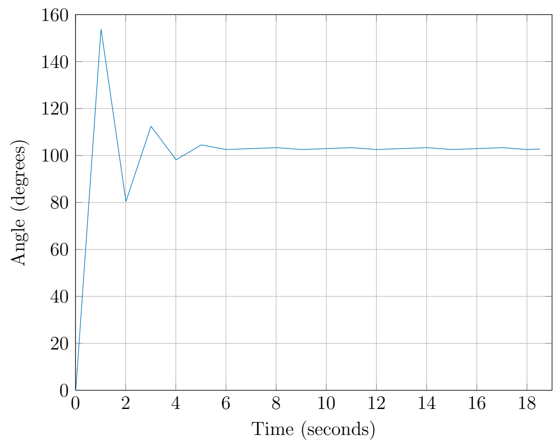


Figure 11: The orientation of the robot over time for $K_{\Psi}^{R} = 0.75K_{\Psi,max}^{R}$

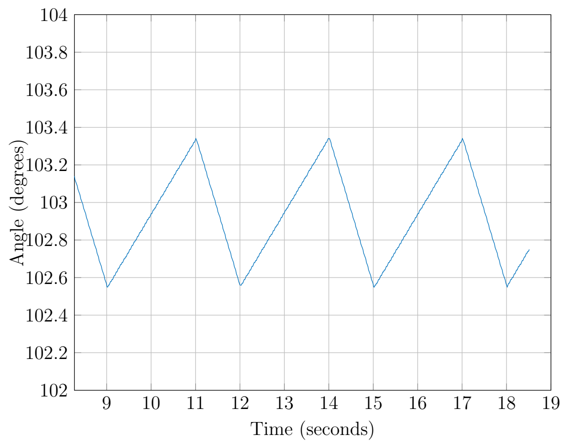


Figure 12: The steady state orientation of the robot for $K_{\Psi}^{R} = 0.75K_{\Psi,max}^{R}$

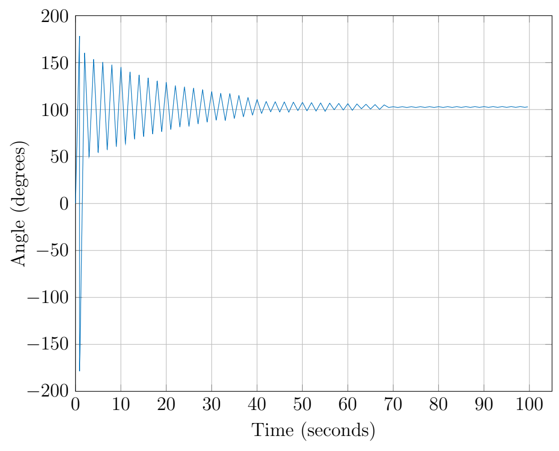


Figure 13: The orientation of the robot over time for $K_{\Psi}^{R} = K_{\Psi,max}^{R}$. This is the upper limit value of $K_{\Psi}^{R}$ before the system becomes unstable

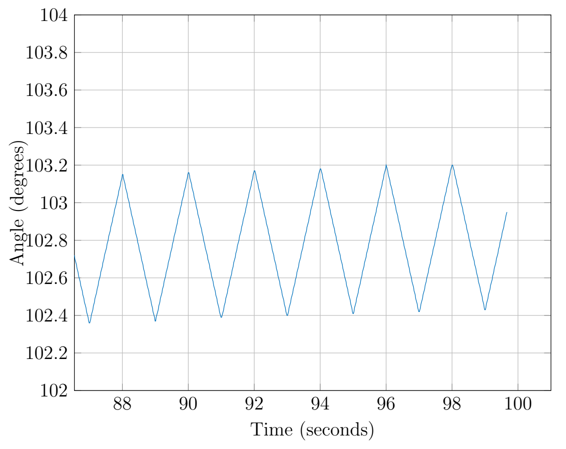


Figure 14: The steady state orientation of the robot for $K_{\Psi}^{R} = K_{\Psi,max}^{R}$

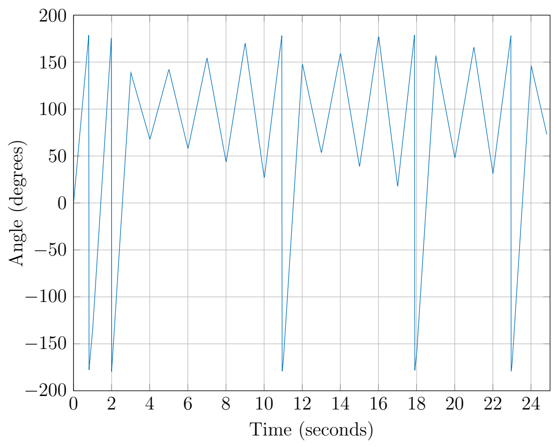


Figure 15: The orientation of the robot over time for $K_{\Psi}^{R} = K_{\Psi,max}^{R} + 1$. The system is indeed unstable

## Task 9

$$\begin{aligned}
d_0[k] &= cos(\theta[k])(x_0 - x[k]) + sin(\theta[k])(y_0 - y[k]) \\
&= cos(\theta[k])(x_0 - x[k-1] - T_s R u_{\omega}^{R}[k-1]cos(\theta[k-1])) \\
&+ sin(\theta[k])(y_0 - y[k-1] - T_s R u_{\omega}^{R}[k-1]sin(\theta[k-1])) \\
&= cos(\theta^R)(x_0 - x[k-1] - T_s R u_{\omega}^{R}[k-1]cos(\theta^R)) \\
&+ sin(\theta^R)(y_0 - y[k-1] - T_s R u_{\omega}^{R}[k-1]sin(\theta^R)) \\
&= cos(\theta^R)(x_0 - x[k-1]) + sin(\theta^R)(y_0 - y[k-1]) - T_s R K_{\omega}^{T} d_0[k-1] \\
&= d_0[k-1] - T_s R K_{\omega}^{T} d_0[k-1] \\
&= (1 - T_s R K_{\omega}^{R}) d_0[k-1]
\end{aligned}$$

Hence

$$d_0[k+1] = (1 - T_s R K_{\omega}^{R}) d_0[k]$$

In order for this system to be stable, that is, $d_0 \to 0$ as $k \to \infty$, the solution of the equation inside the parentheses should lie inside the unit circle:

$$\begin{aligned}
\left|1 - T_s R K_{\omega}^{R}\right| &< 1 \\
-1 < 1 - T_s R K_{\omega}^{R} &< 1 \\
-2 < -T_s R K_{\omega}^{R} &< 0 \\
0 < T_s R K_{\omega}^{R} &< 2 \\
0 < K_{\omega}^{R} &< \frac{2}{T_s R}
\end{aligned} \tag{8}$$

Hence, the maximum value $K_\omega^R$ can take for the system to be marginally stable is $K_{\omega,max}^R = \dfrac{2}{T_s R}$.

Theoretically, the value of $K_\omega^R$ can be chosen to be any value inside the interval defined in inequality **??**. However, first, it would be wise to choose a value that is far enough from the maximum value so as to avoid overshoot, but close enough to it, so that convergence happens in reasonable time. Hence, in practice, it is reasonable that one would need to experiment with different values and choose one that results in balancing a small angular error, a minimal overshoot, if any, and a quick enough settling time.

## Task 10

This part of the controller is responsible for compensating translational errors during rotation. Since the rotational speed $u_\Psi$ is zero, it is expected that the robot will not move away from its origin. Figure **??** plots the robot's distance from its origin over time for $K_\omega^R = 0.5K_{\omega,max}^R$.

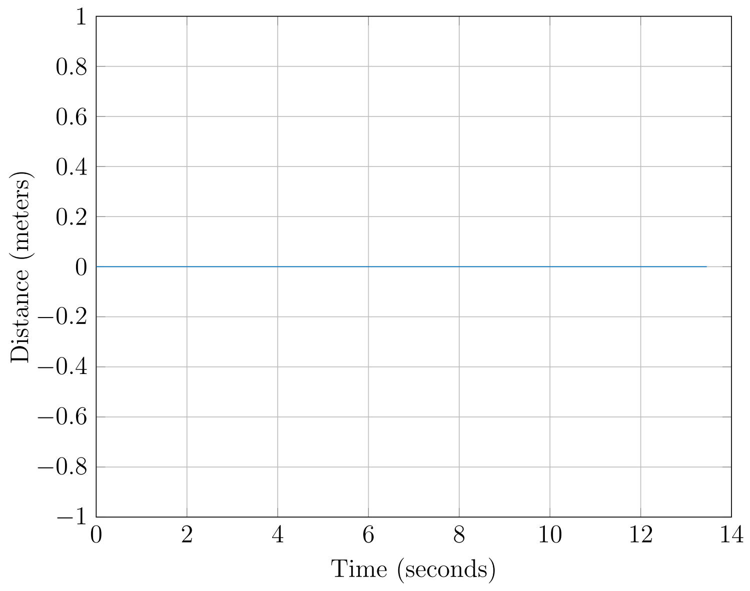


Figure 16: The distance of the robot from its origin position over time for $K_\omega^R = 0.5K_{\omega,max}^R$

## Task 11

Figures **??**, **??**, **??**, **??**, **??** show the bearing error of the robot for different values of $K_\Psi^R$ inside the interval set by inequality **??**. Figures **??**, **??**, **??**, **??** and **??** focus on the steady-state bearing error. Figure **??** illustrates the $d_0[k]$ error, which is at all times zero.

The evolution of the bearing and distance error is the same when both of the rotational controllers are enabled compared to when only one of them is enabled. This happens because the behaviour of each controller does not affect the behaviour of the other, since this is an ideal system. In reality, we expect that the distance error will be non-zero.

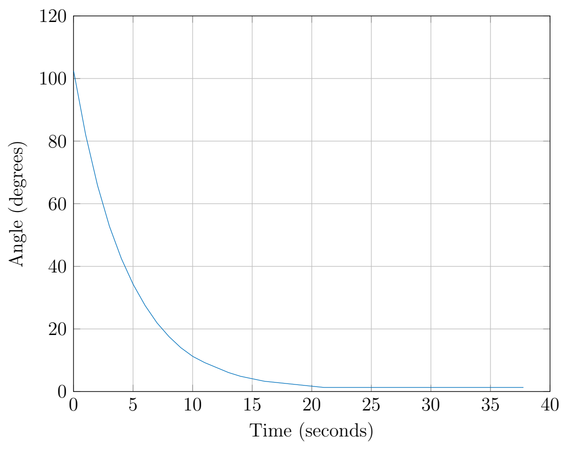


Figure 17: The error in orientation of the robot over time for $K^R_\Psi = 0.1K^R_{\Psi,max}$

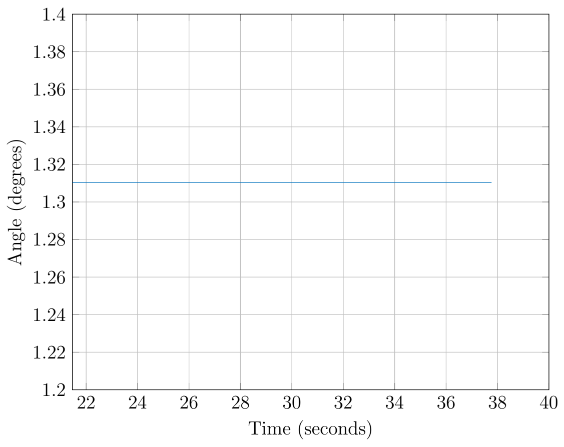


Figure 18: The steady state error in orientation of the robot for $K^R_\Psi = 0.1K^R_{\Psi,max}$

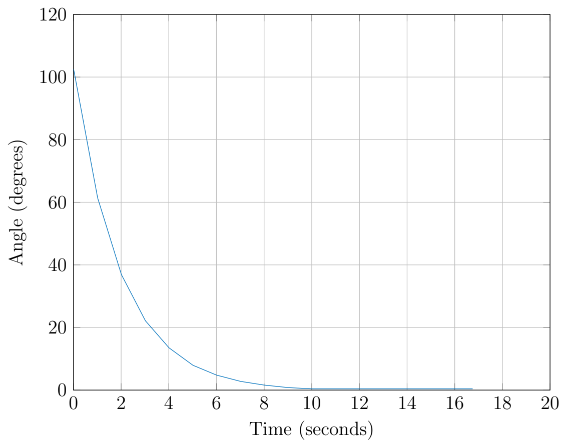


Figure 19: The error in orientation of the robot over time for $K^R_\Psi = 0.2K^R_{\Psi,max}$

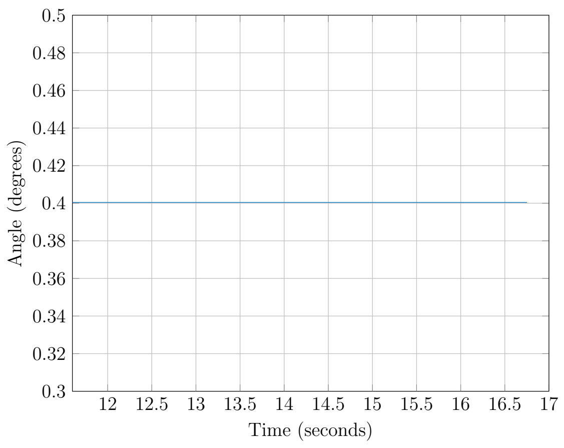


Figure 20: The steady state error in orientation of the robot for $K^R_\Psi = 0.2K^R_{\Psi,max}$

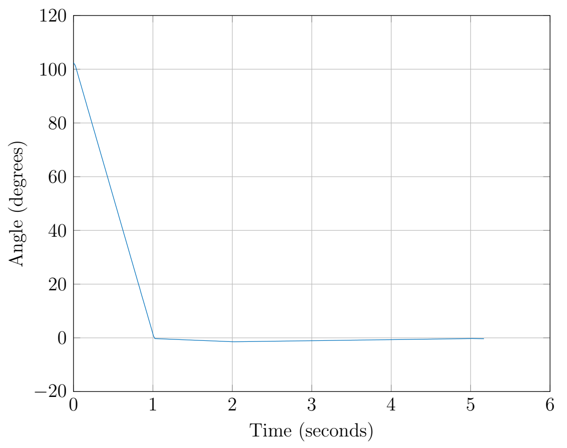


Figure 21: The error in orientation of the robot over time for $K^R_\Psi = 0.5K^R_{\Psi,max}$

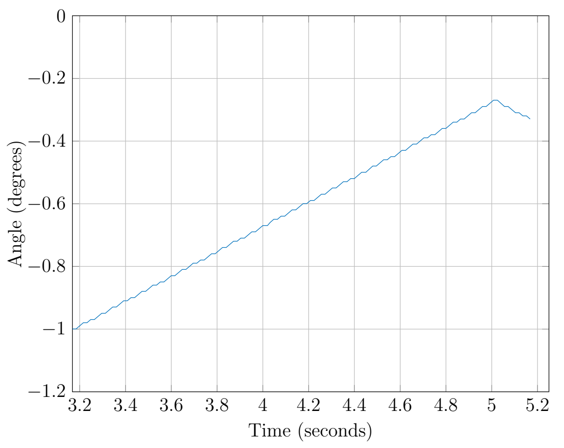


Figure 22: The steady state error in orientation of the robot for $K^R_\Psi = 0.5K^R_{\Psi,max}$

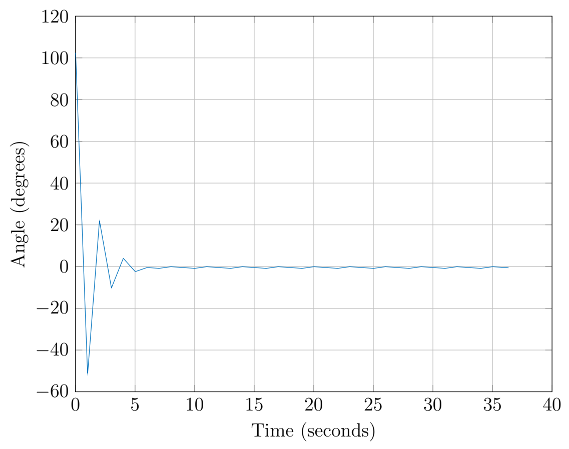


Figure 23: The error in orientation of the robot over time for $K^R_\Psi = 0.75K^R_{\Psi,max}$

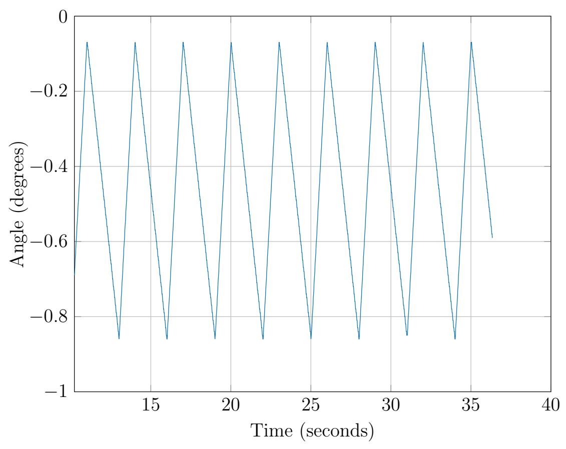


Figure 24: The steady state error in orientation of the robot for $K^R_\Psi = 0.75K^R_{\Psi,max}$

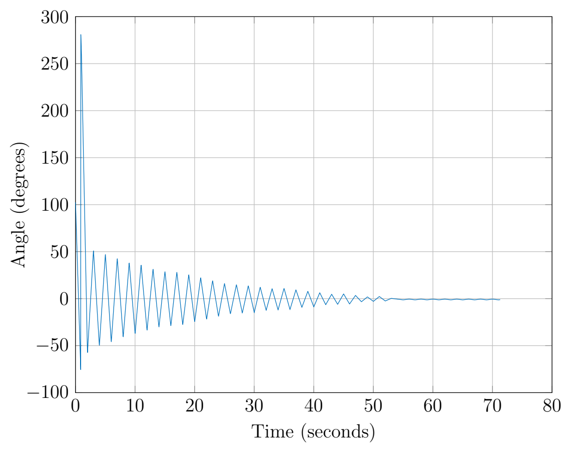


Figure 25: The error in orientation of the robot over time for $K^R_\Psi = K^R_{\Psi,max}$.

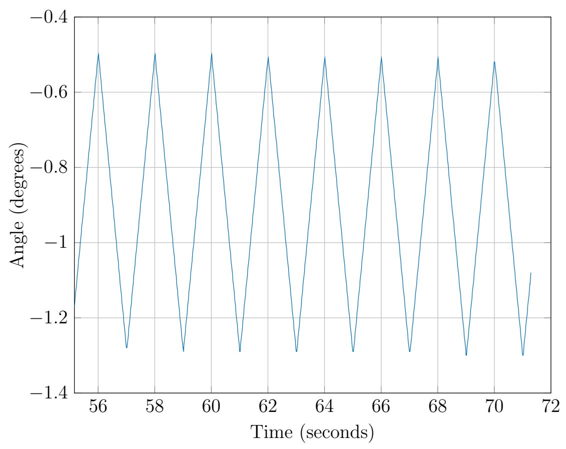


Figure 26: The steady state error in orientation of the robot for $K^R_\Psi = K^R_{\Psi,max}$

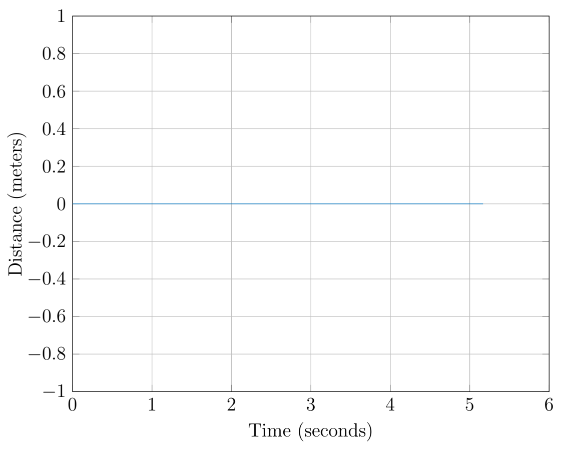


Figure 27: The distance of the robot from its origin position over time for all legitimate values or $K^R_\omega$

## Task 12

$$\begin{aligned}
d_g[k] &= cos(\theta_g)(x_g - x[k]) + sin(\theta_g)(y_g - y[k]) \\
&= cos(\theta_g)(x_g - x[k-1] - T_s R u_\omega^T[k-1] cos(\theta_g)) + \\
&sin(\theta_g)(y_g - y[k-1] - T_s R u_\omega^T[k-1] sin(\theta_g)) \\
&= cos(\theta_g)(x_g - x[k-1]) + sin(\theta_g)(y_g - y[k-1]) - T_s R K_\omega^T d_g[k-1] \\
&= d_g[k-1] - T_s R K_\omega^T d_g[k-1] \\
&= (1 - T_s R K_\omega^T) d_g[k-1]
\end{aligned}$$

Hence

$$d_g[k+1] = (1 - T_s R K_\omega^T) d_g[k]$$

In order for this system to be stable, that is, $d_g \to 0$ as $k \to \infty$, the solution of the equation inside the parentheses should lie inside the unit circle:

$$\begin{aligned}
\left|1 - T_s R K_\omega^T\right| &< 1 \\
-1 < 1 - T_s R K_\omega^T &< 1 \\
-2 < -T_s R K_\omega^T &< 0 \\
0 < T_s R K_\omega^T &< 2 \\
0 < K_\omega^T &< \frac{2}{T_s R} \qquad (9)
\end{aligned}$$

Hence, the maximum value $K_\omega^T$ can take for the system to be marginally stable is $K_{\omega,max}^T = \dfrac{2}{T_s R}$

Theoretically, the value of $K_\omega^T$ can be chosen to be any value inside the interval defined in inequality **??**. However, first, it would be wise to choose a value that is far enough from the maximum value so as to avoid overshoot, but close enough to it, so that convergence happens in reasonable time. Hence, in practice, it is reasonable that one would need to experiment with different values and choose one that results in balancing a small angular error, a minimal overshoot, if any, and a quick enough settling time.

## Task 13

For the purpose of simulating this part of the controller, the initial point of the robot was taken to be $I(x_0, y_0) \equiv (0, 0)$. The goal was set to $G(x_g, y_g) \equiv (1.0, 0.0)$.

Figures **??**, **??**, **??**, **??** and **??** show the displacemental response of the robot for various values of $K_\omega^T$ inside the interval set by inequality **??**. Figure **??** verifies that the upper limit for $K_\omega^T$ is indeed $\frac{2}{T_s R}$ by showing that the displacemental response of the robot cannot converge for $K_\omega^T > K_{\omega,max}^T$.

Here, one can see that the smaller the value of $K_\omega^T$ is, the larger the settling time, the lower the rise time and the smoother the response is. However, as the value of

$K_\omega^T$ increases, the steady-state response begins to oscillate, with the amplitude of this oscillation proportional to the value of $K_\omega^T$.

Figures **??**, **??**, **??**, **??** and **??** focus on the steady-state value of the aforementioned responses. In contrast to the proportional rotational controller, the robot *can* arrive to its reference signal. The equations that govern the robot's translational movement are non-linear, as opposed to the one that governs its rotational movement, which in turn means that the translational system's behaviour is not bounded within the laws that govern control with proportional controllers on linear systems.

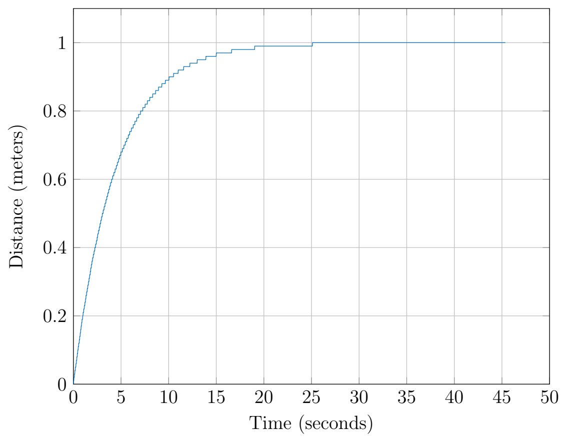


Figure 28: The orientation of the robot over time for $K_\omega^T = 0.1K_{\omega,max}^T$

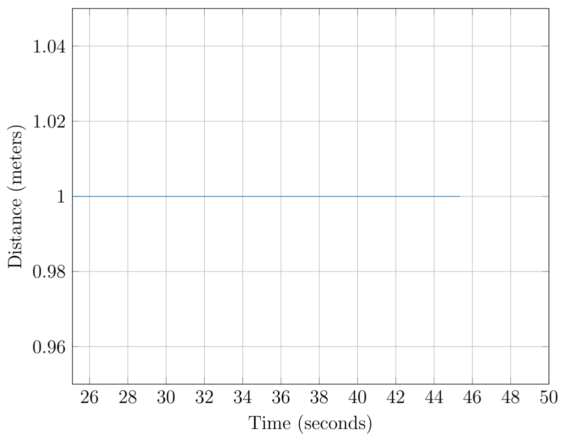


Figure 29: The steady state orientation of the robot for $K_\omega^T = 0.1K_{\omega,max}^T$

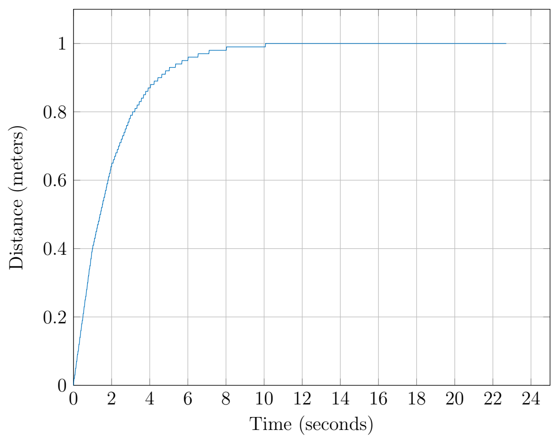


Figure 30: The orientation of the robot over time for $K_\omega^T = 0.2K_{\omega,max}^T$

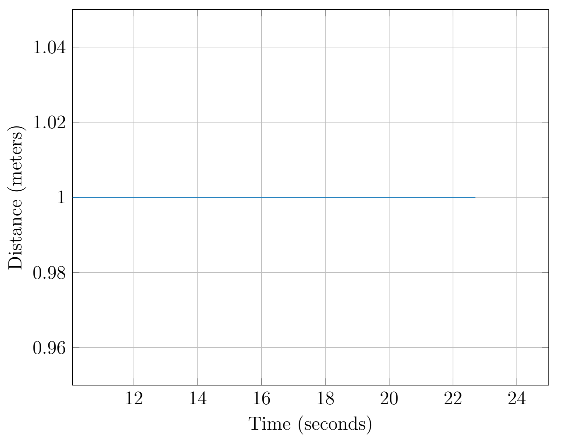


Figure 31: The steady state orientation of the robot for $K_\omega^T = 0.2K_{\omega,max}^T$

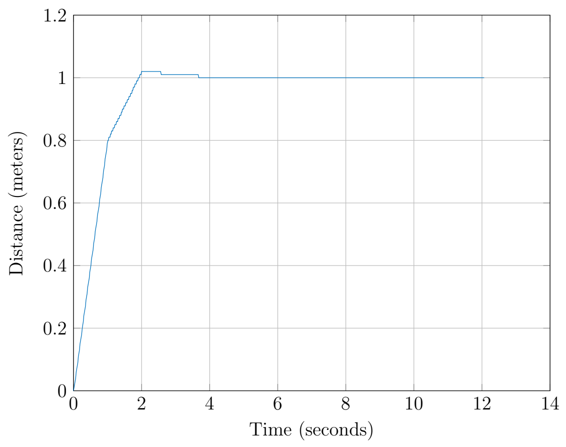


Figure 32: The orientation of the robot over time for $K^T_\omega = 0.5K^T_{\omega,max}$

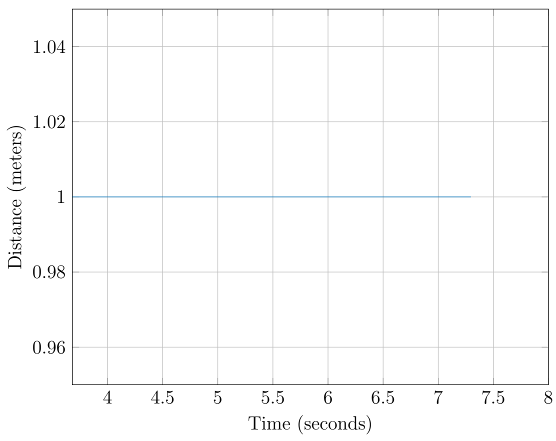


Figure 33: The steady state orientation of the robot for $K^T_\omega = 0.5K^T_{\omega,max}$

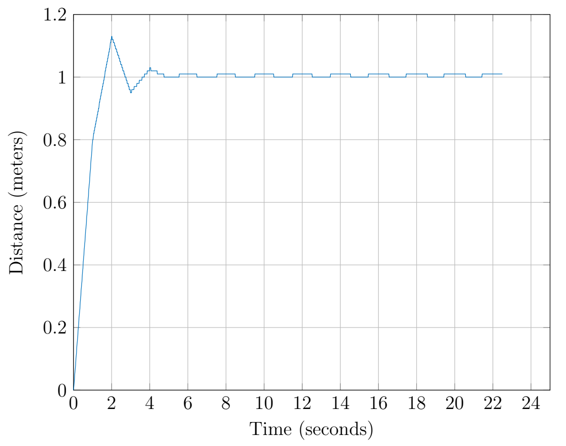


Figure 34: The orientation of the robot over time for $K^T_\omega = 0.75K^T_{\omega,max}$

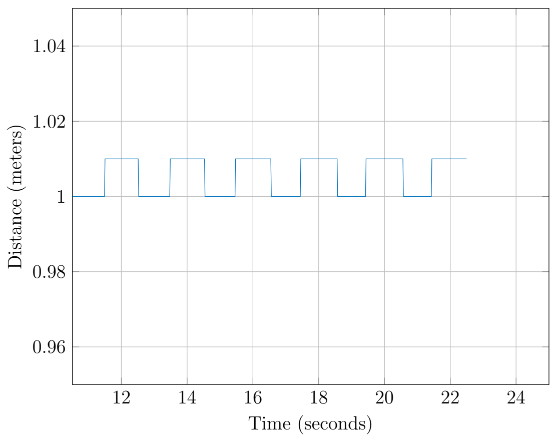


Figure 35: The steady state orientation of the robot for $K^T_\omega = 0.75K^T_{\omega,max}$

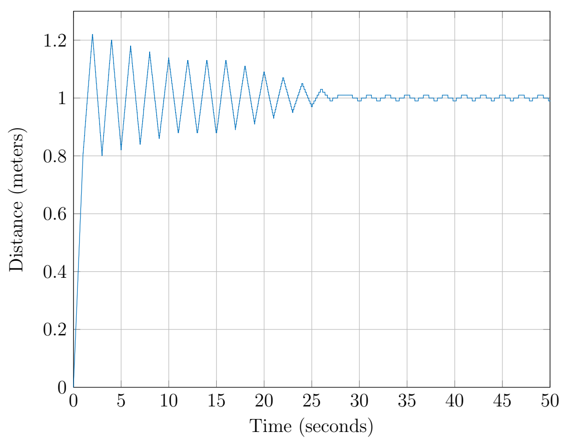


Figure 36: The orientation of the robot over time for $K^T_\omega = K^T_{\omega,max}$.

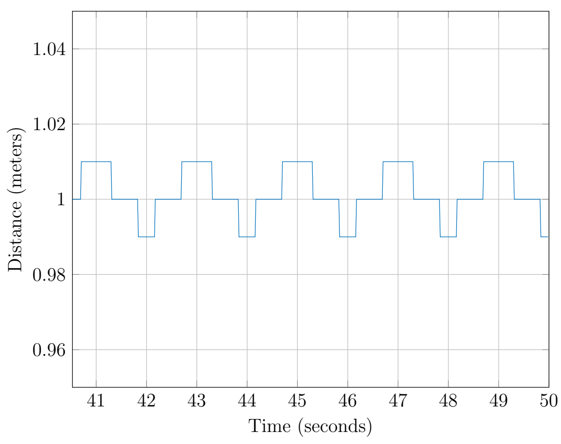


Figure 37: The steady state orientation of the robot for $K^T_\omega = K^T_{\omega,max}$

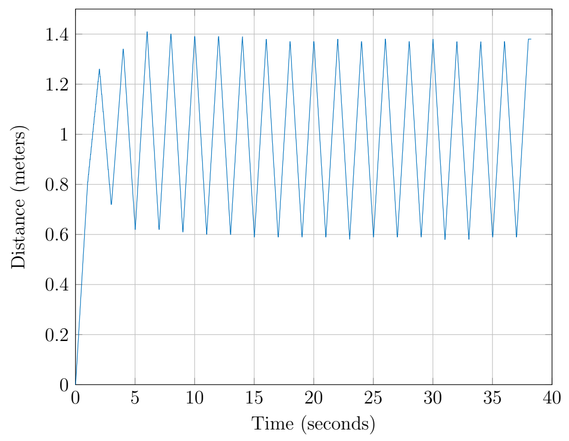


Figure 38: The orientation of the robot over time for $K_\omega^T = 1.1K_{\omega,max}^T$. The system is marginally stable

## Task 14

$$d_p[k+1] = p(\theta[k+1] - \theta_g) \tag{10}$$

But

$$\theta[k+1] = \theta[k] + T_s\frac{R}{L}u_\Psi^T[k]$$

Hence equation **??** becomes

$$\begin{aligned} d_p[k+1] &= p(\theta[k] + T_s\frac{R}{L}u_\Psi^T[k] - \theta_g) \\ &= p(\theta[k] + T_s\frac{R}{L}K_\Psi^T d_p[k] - \theta_g) \\ &= pT_s\frac{R}{L}K_\Psi^T d_p[k] + p(\theta[k] - \theta_g) \\ &= pT_s\frac{R}{L}K_\Psi^T d_p[k] + d_p[k] \\ &= d_p[k](1 + pT_s\frac{R}{L}K_\Psi^T d_p[k]) \end{aligned}$$

In order for this system to be stable, that is, $d_p \to 0$ as $k \to \infty$, the solution of the equation inside the parentheses should lie inside the unit circle:

$$\begin{aligned} \left|1+pT_s\frac{R}{L}K_\omega^T\right| &< 1 \\ -1 < 1+pT_s\frac{R}{L}K_\omega^T &< 1 \\ -2 < pT_s\frac{R}{L}K_\omega^T &< 0 \\ -\frac{2L}{pT_sR} < K_\omega^T &< 0 \end{aligned} \tag{11}$$

Hence, the maximum value $K_\omega^T$ can take for the system to be marginally stable is $K_{\omega,max}^T = 0$.

Theoretically, the value of $K_\omega^T$ can be chosen to be any value inside the interval defined in inequality **??**. However, first, it would be wise to choose a value that is far enough from the maximum value so as to avoid overshoot, but close enough to it, so that convergence happens in reasonable time. Hence, in practice, it is reasonable that one would need to experiment with different values and choose one that results in balancing a small angular error, a minimal overshoot, if any, and a quick enough settling time.

## Task 15

Inequality **??** tells us that the higher the value of $p$, the broader the region of values for $K_\Psi^T$ is so that the systems is stable. Hence, the lower the value of $p$ is, the worse the robot's ability to follow a line is.

## Task 16

This part of the controller is responsible for compensating for rotational errors during translation. Since the translational velocity $u_\omega$ is zero, it is expected that the robot will not rotate away from its original bearing. Figure **??** plots the robot's bearing with regard to its original bearing of 0 degrees over time for $K_\Psi^T = 0.5K_{\Psi,min}^T$.

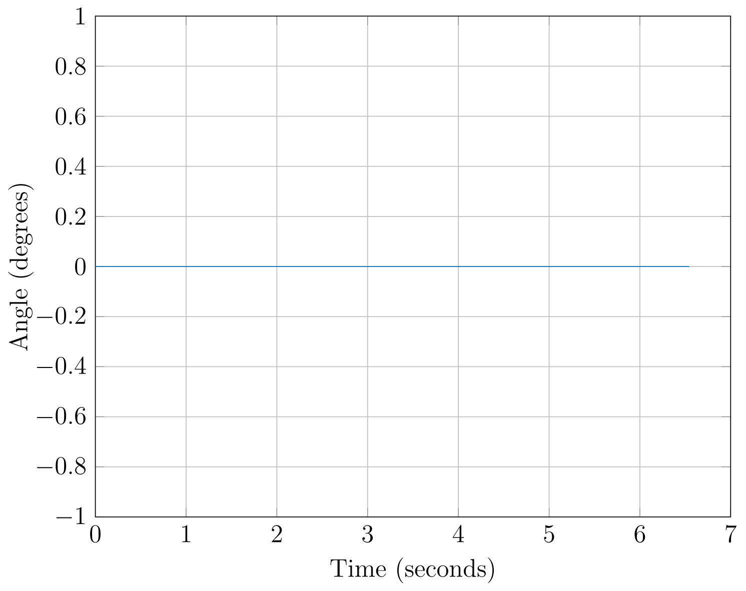


Figure 39: The angular displacement of the robot from its original bearing over time for $K_\Psi^T = 0.5K_{\Psi,min}^T$

## Task 17

Figures **??**, **??**, **??**, **??** and show the displacemental error of the robot for different values of $K_\omega^T$ inside the interval set by inequality **??**. Figures **??**, **??**, **??** and **??** focus on the steady-state displacemental error. Figure **??** illustrates the $d_0[k]$ error, which is at all times zero.

The evolution of the bearing and displacement error is the same when both of the translational controllers are enabled compared to when only one of them is enabled. This happens because the behaviour of each controller does not affect the behaviour of the other, since this is an ideal system. In reality, we expect that the angular error will be non-zero.

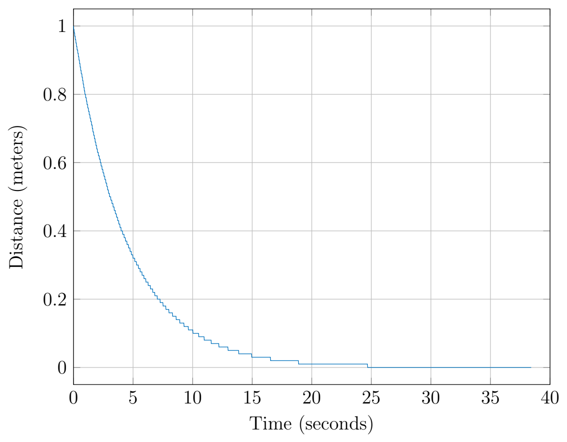


Figure 40: The error in displacement of the robot over time for $K_\omega^T = 0.1K_{\omega,max}^T$

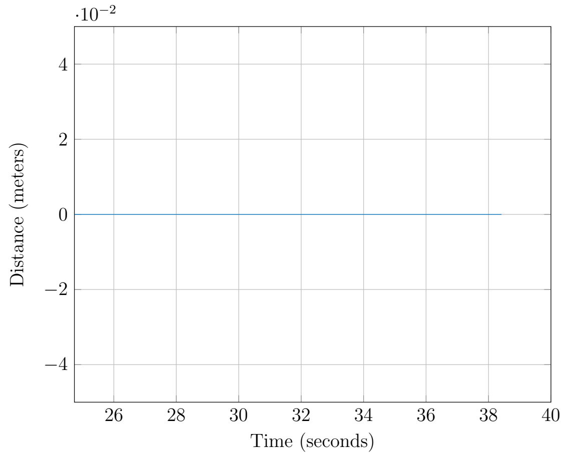


Figure 41: The steady state error in displacement of the robot for $K_\omega^T = 0.1K_{\omega,max}^T$

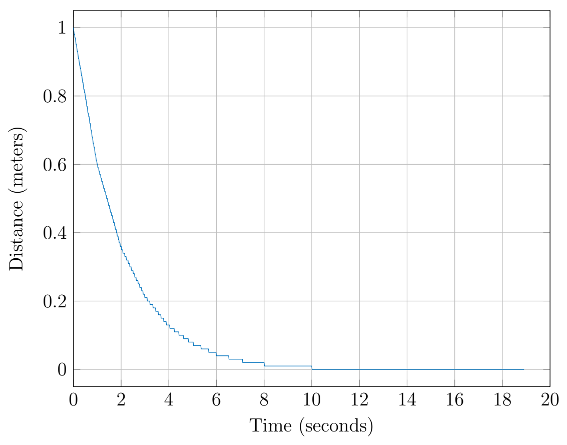


Figure 42: The error in displacement of the robot over time for $K_\omega^T = 0.2K_{\omega,max}^T$

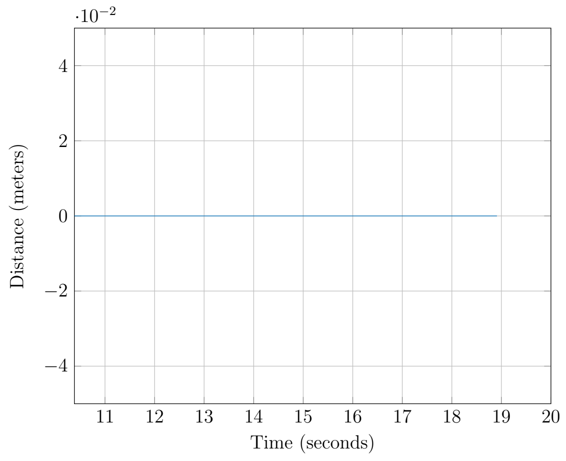


Figure 43: The steady state error in displacement of the robot for $K_\omega^T = 0.2K_{\omega,max}^T$

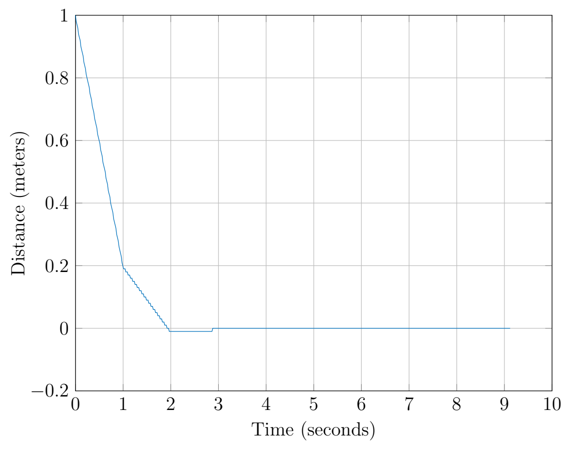


Figure 44: The error in displacement of the robot over time for $K^T_\omega = 0.5K^T_{\omega,max}$

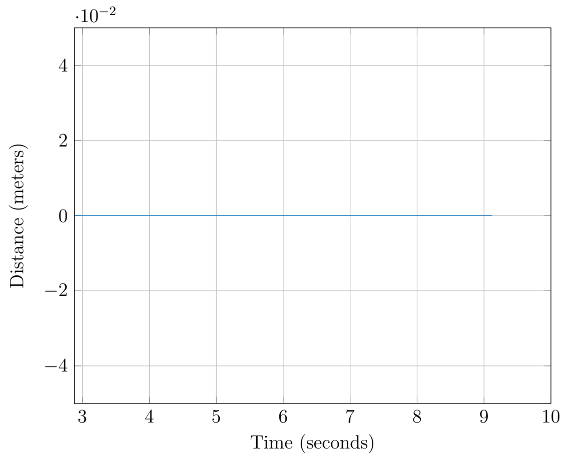


Figure 45: The steady state error in displacement of the robot for $K^T_\omega = 0.5K^T_{\omega,max}$

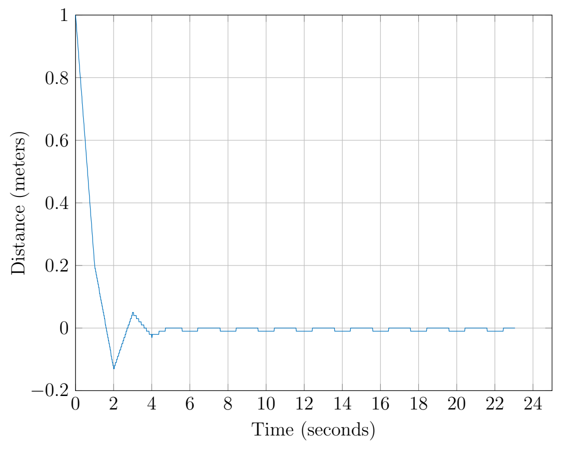


Figure 46: The error in displacement of the robot over time for $K^T_\omega = 0.75K^T_{\omega,max}$

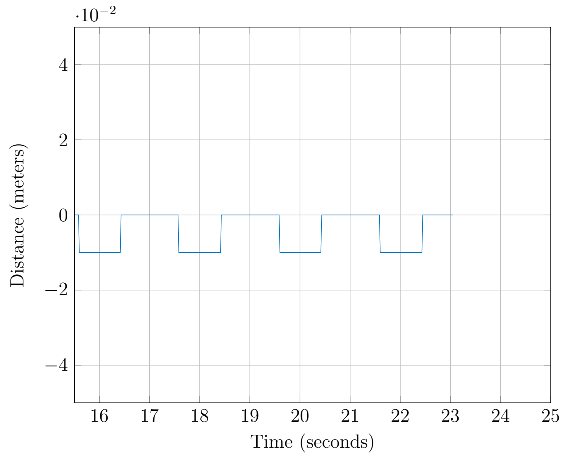


Figure 47: The steady state error in displacement of the robot for $K^T_\omega = 0.75K^T_{\omega,max}$

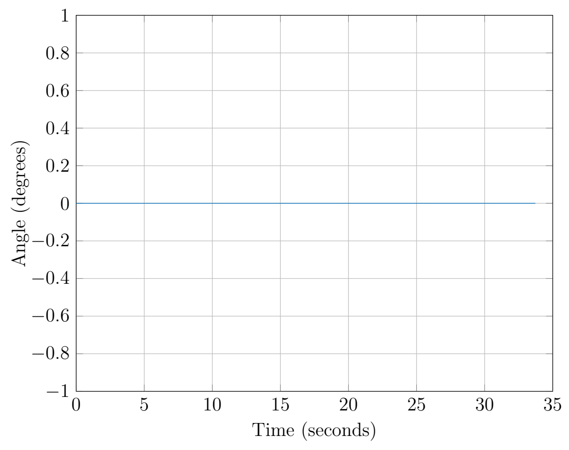


Figure 48: The steady state error in orientation of the robot for all legitimate values of $K^T_\omega$

## Task 18

The hybrid controller is modelled formally by an 8-tuple

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

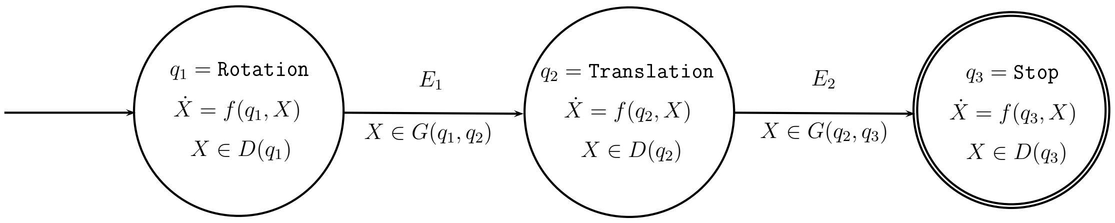


Figure 49: The hybrid automaton that controls the navigation of the robot from an initial state $(x_0, y_0, \theta_0)$ to a goal location $(x_g, y_g)$.

- $Q \equiv \{q_1, q_2, q_3\}$ denotes the set of discrete states. The robot is in state $q_1$ when executing a rotation, in state $q_2$ when executing line-following, and in state $q_3$ when it has stopped.
- $Init \equiv \{q_1\}$. The initial state is taken to be $q_1$.
- $X \equiv \{(x, y, \theta) : x, y \in \mathbb{R}^2, \theta \in (-180°, 180°]\}$ denotes the continuous states
- Vector fields $f$

$$f(q_1, X) = \begin{bmatrix} \dot{x} = Ru_\omega cos\theta \\ \dot{y} = Ru_\omega sin\theta \\ \dot{\theta} = \frac{R}{L}u_\Psi \end{bmatrix}$$

$$f(q_2, X) = \begin{bmatrix} \dot{x} = Ru_\omega cos\theta \\ \dot{y} = Ru_\omega sin\theta \\ \dot{\theta} = \frac{R}{L}u_\Psi \end{bmatrix}$$

$$f(q_2, X) = \begin{bmatrix} \dot{x} = 0 \\ \dot{y} = 0 \\ \dot{\theta} = 0 \end{bmatrix}$$

- $D$ shows what conditions need to be satisfied in order for the automaton to stay in a state.

  $D(q_1) = \{(x, y, \theta) : x, y \in \mathbb{R}^2, |\theta_g - \theta| > \delta\}$

  $D(q_2) = \{(x, y, \theta) : x, y \in \mathbb{R}^2, (x_g - x)^2 + (y_g - y)^2 > \xi, \theta \in (-180°, 180°]\}$

  $D(q_3) = \{x, y \in \mathbb{R}^2, \theta \in (-180°, 180°]\}$

- $E$: The edges show which transitions are possible.

  $E_1 = \{q_1, q_2\}$

  $E_2 = \{q_2, q_3\}$

  $E = E_1 \cup E_2$

- $G$: The guards show under what conditions the system can transition from one to another state.

  $G(\{q_1, q_2\}) = \{\theta_g - \theta \leq \delta\}$

  $G(\{q_2, q_3\}) = \{(x_g - x)^2 + (y_g - y)^2 \leq \xi\}$

- $R$: Resets illustrate the values that the state takes when transitioning between states.

  $R = \{x, y, \Theta\}$

## Task 19

Evaluating the performance of the hybrid automaton primarily involves the evaluation of the steady-state errors regarding the position and angle of the robot with regard to the selected goal position. In order to obtain a broader understanging of how the four $K_*$ gains influence the trajectory of the robot, twelve combinations were considered for the 4-tuple

$$(K_\Psi^R, K_\omega^T, K_\omega^R, K_\Psi^T)$$

where

$$K_\Psi^R \in \{0.1, 0.2, 0.5\} \cdot K_{\Psi,max}^R$$

$$K_\omega^T \in \{0.1, 0.2, 0.5, 0.75\} \cdot K_{\omega,max}^T$$

$$K_\omega^R = 0.5K_{\omega,max}^R, \text{ and } K_\Psi^T = -0.5K_{\Psi,min}^T$$

Notice the minus sign for $K_\Psi^T$. Although theoretically (and using the approximative form for $d_p[k]$) this gain should be negative, simulations showed that the robot achieved lower levels of both displacemental and angular errors when considered positive[1]. For purposes of homogeneity we shall denote $K_{\Psi,max}^T = -K_{\Psi,min}^T$, so $K_\Psi^T = 0.5K_{\Psi,max}^T > 0$.

Figures **??** - **??** illustrate the evolution of the continuous and discrete state trajectories, over time, for the aforementioned combinations of values of the gains $K_*$. The goal was set to be node 1 $N1(-0.37, 1.68)$, and the robot's initial pose was $(x_0, y_0, \theta_0) \equiv (0, 0, 0)$. Hence, the distance to the goal was $d_g = 1.7203$ meters and the angle to the goal $\theta^R = 102.42°$. The distance and angle tolerance thresholds where taken to be $\delta = 2$ cm and $\xi = 2°$ respectively.

Since the actual final pose is not defined deterministically, five simulations of each possible combination of settings for the aforementioned 4-tuple were conducted. Hence, all the following figures express the mean continuous and discrete state trajectories.

Table **??** illustrates the steady-state errors $e_d$ and $e_\theta$ regarding the distance and the angle that the robot had to travel, respectively.

[1]Analysis on Task 14 showed that $K_{\Psi,min}^T$ is considered negative

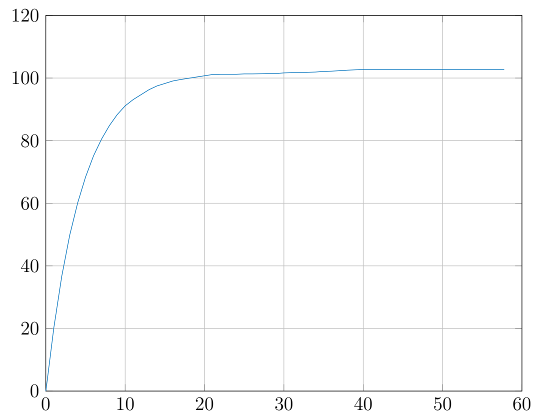
| $K^R_\Psi / K^R_{\Psi,max}$ | $K^T_\omega / K^T_{\omega,max}$ | $e_d$ $(cm)$ | $e_\theta$ (deg) |
|---|---|---|---|
| 0.1 | 0.1 | 1.73 | 0.329 |
| 0.1 | 0.2 | 0.60 | 0.144 |
| 0.1 | 0.5 | 0.90 | 0.330 |
| 0.1 | 0.75 | 1.9 | 0.054 |
| 0.2 | 0.1 | 1.47 | 0.419 |
| 0.2 | 0.2 | 1.21 | 0.308 |
| 0.2 | 0.5 | 0.67 | 0.116 |
| 0.2 | 0.75 | 2.12 | 0.026 |
| 0.5 | 0.1 | 1.26 | 0.303 |
| 0.5 | 0.2 | 0.75 | 0.585 |
| 0.5 | 0.5 | 1.05 | 0.695 |
| 0.5 | 0.75 | 0.86 | 0.347 |

Table 1: Mean steady-state errors regarding the distance and bearing to Node 1 for the 12 different combinations considered for the 4-tuple $(K^R_\Psi, K^T_\omega, K^R_\omega, K^T_\Psi)$

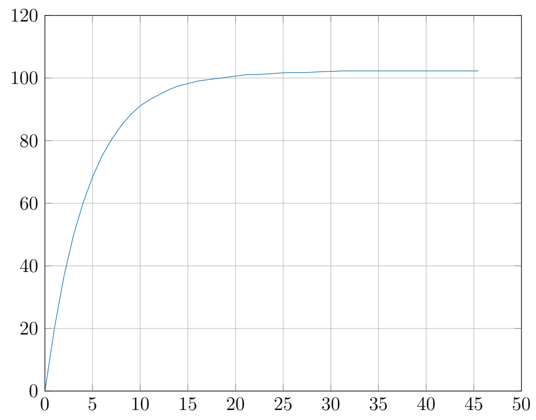
The combination of the two control components made it possible for the robot to move to any location inside the simulation environment. In particular, for the single goal considered here, the controller made it possible for the robot to approach node 1 within a radius of length $\delta$ (with one exception, when $(K^R_\Psi / K^R_{\Psi,max}, K^T_\omega / K^T_{\omega,max}) \equiv (0.2, 0.75)$) and with a bearing well within the $2°$ upper threshold. The second component of the line-following controller was partly responsible for reducing the bearing errors compared to the case when only the rotational controller was enabled. However, in order for the second component's input to be able to kick in in the first place, it was necessary to limit the translational velocity of the robot to a value of $u_\omega = 400$, so as to permit the correctional input of this component to become active, since there is an upper limit to the velocities the robot can achieve, either in simulation or in real-life.

The discrete state trajectory figures express what was expected: as the value of a gain $K_*$ increases, the time that the robot stays in the corresponding state decreases. What was not expected, though, was the temporary spikes into state `Stop` for all cases when $K^T_\omega / K^T_{\omega,max} = 0.75$. This makes sence, since that high a value for the gain of the translational part of the line-following controller makes the robot overshoot the goal. Its momentum is such that while it approaches the goal within $\delta$ m and its state transitions from `Translation` to `Stop`, it cannot physically stop and steps out of the circle with radius $\delta$, and thus its state goes back to `Translation`. The duration the robot stays at state `Stop` is the duration of time it spends inside this virual circle with radius $\delta$. This time interval is observed to be unequal among all three cases, due to the fact that the robot travels forwards (part I of the line-following controller) and rotates (part II of the line-following controller) simultaneously.

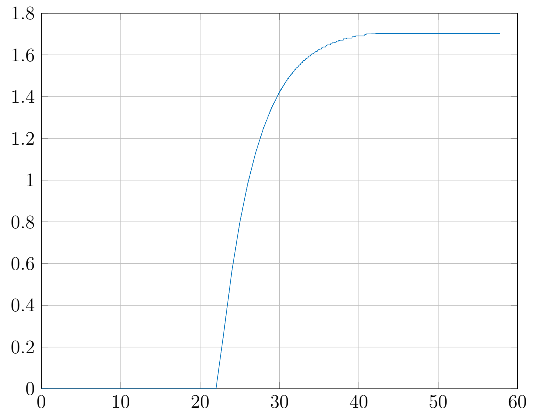
Furthermore, we note that the robot does not start to translate immediately after the state switch to $q_2$.



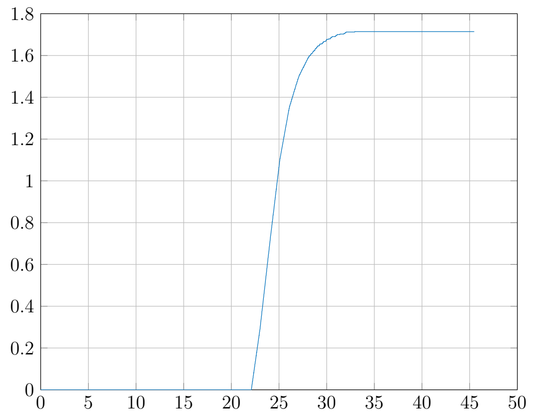
(a) The bearing of the robot over time for $(K_\Psi^R, K_\omega^T) \equiv (0.1K_{\Psi,max}^R, 0.1K_{\omega,max}^T)$



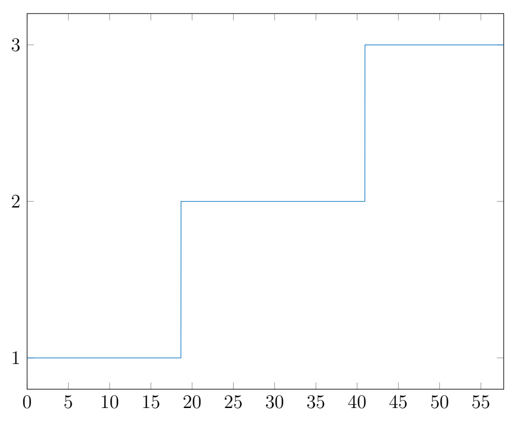
(d) The bearing of the robot in over time for $(K_\Psi^R, K_\omega^T) \equiv (0.1K_{\Psi,max}^R, 0.2K_{\omega,max}^T)$



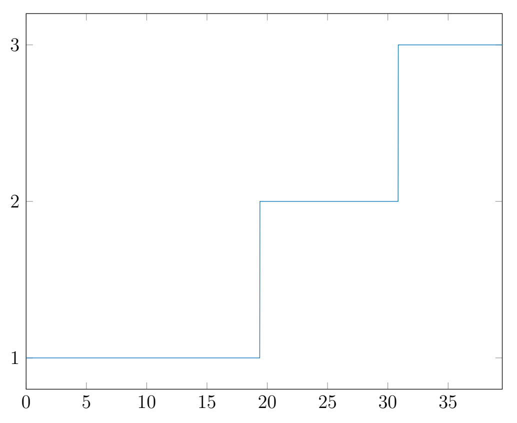
(b) The distance of the robot to the origin over time for $(K_\Psi^R, K_\omega^T) \equiv (0.1K_{\Psi,max}^R, 0.1K_{\omega,max}^T)$



(e) The distance of the robot to the origin over time for $(K_\Psi^R, K_\omega^T) \equiv (0.1K_{\Psi,max}^R, 0.2K_{\omega,max}^T)$

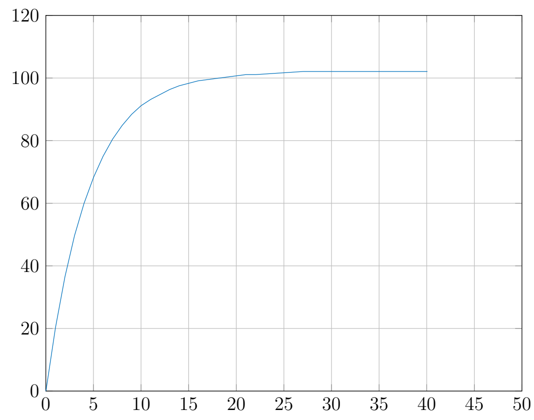


(c) The discrete state trajectory for $(K_\Psi^R, K_\omega^T) \equiv (0.1K_{\Psi,max}^R, 0.1K_{\omega,max}^T)$. 1 denotes `Rotation`, 2 denotes `Translation` and 3 denotes `Stop`

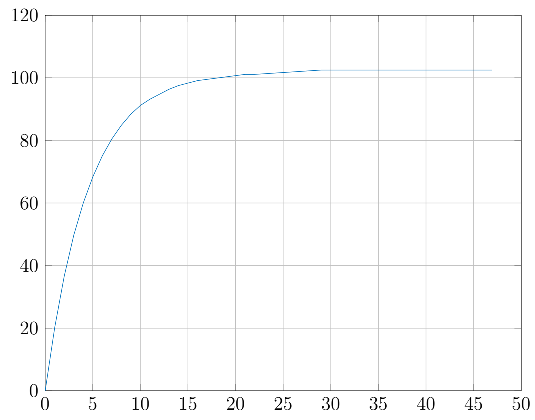


(f) The discrete state trajectory for $(K_\Psi^R, K_\omega^T) \equiv (0.1K_{\Psi,max}^R, 0.2K_{\omega,max}^T)$. 1 denotes `Rotation`, 2 denotes `Translation` and 3 denotes `Stop`

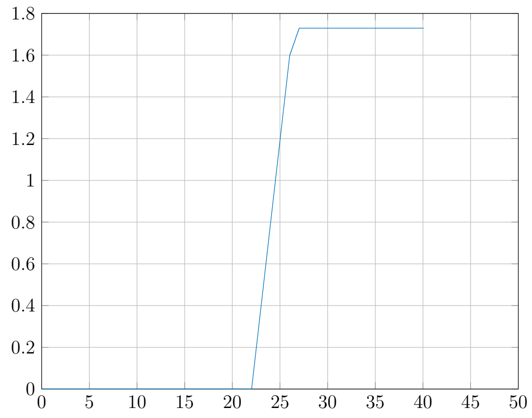
Figure 50



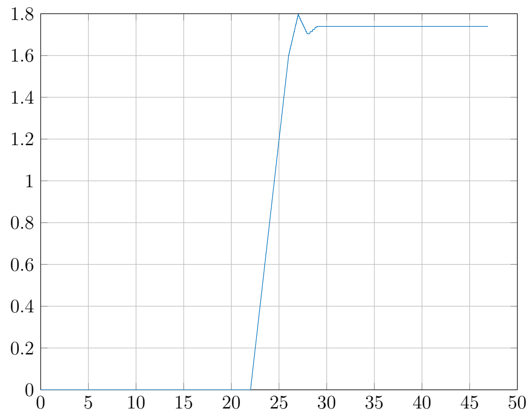
(a) The bearing of the robot over time for $(K^R_\Psi, K^T_\omega) \equiv (0.1K^R_{\Psi,max}, 0.5K^T_{\omega,max})$



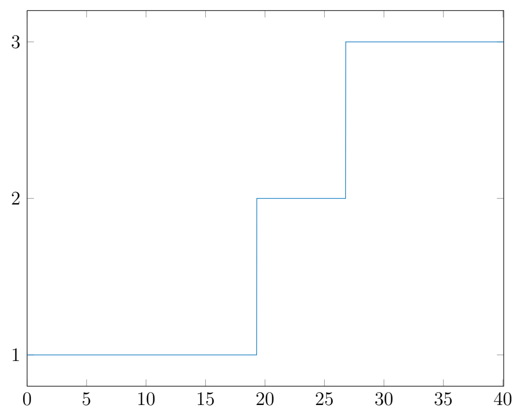
(d) The bearing of the robot in over time for $(K^R_\Psi, K^T_\omega) \equiv (0.1K^R_{\Psi,max}, 0.75K^T_{\omega,max})$



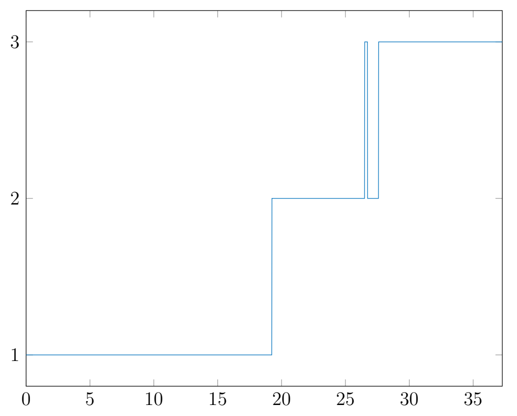
(b) The distance of the robot to the origin over time for $(K^R_\Psi, K^T_\omega) \equiv (0.1K^R_{\Psi,max}, 0.5K^T_{\omega,max})$



(e) The distance of the robot to the origin over time for $(K^R_\Psi, K^T_\omega) \equiv (0.1K^R_{\Psi,max}, 0.75K^T_{\omega,max})$

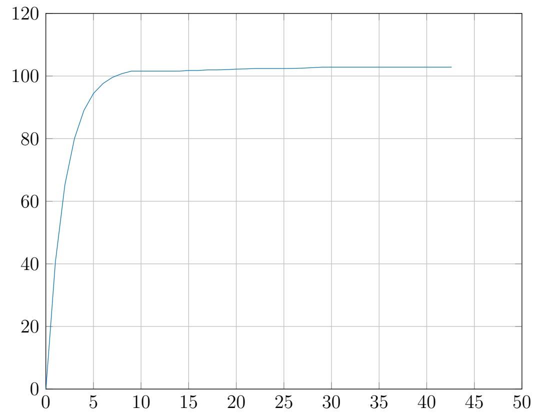


(c) The discrete state trajectory for $(K^R_\Psi, K^T_\omega) \equiv (0.1K^R_{\Psi,max}, 0.5K^T_{\omega,max})$. 1 denotes `Rotation`, 2 denotes `Translation` and 3 denotes `Stop`

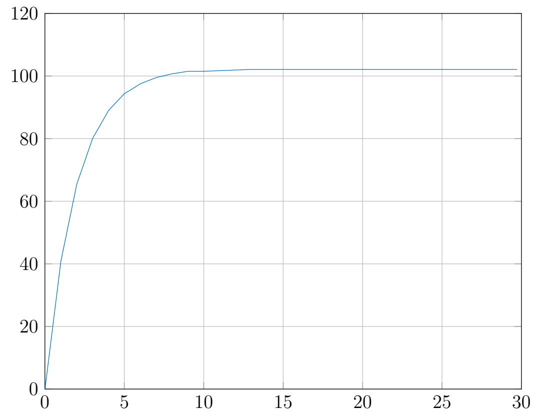


(f) The discrete state trajectory for $(K^R_\Psi, K^T_\omega) \equiv (0.1K^R_{\Psi,max}, 0.75K^T_{\omega,max})$. 1 denotes `Rotation`, 2 denotes `Translation` and 3 denotes `Stop`

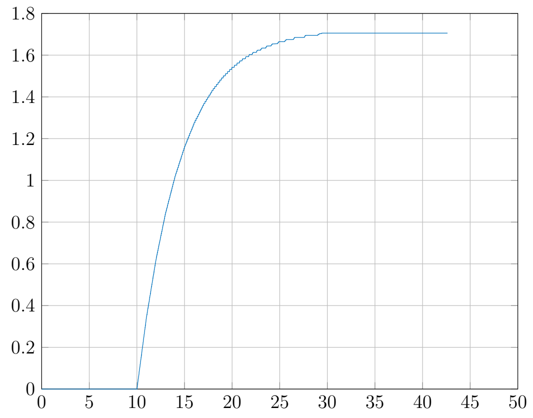
Figure 51



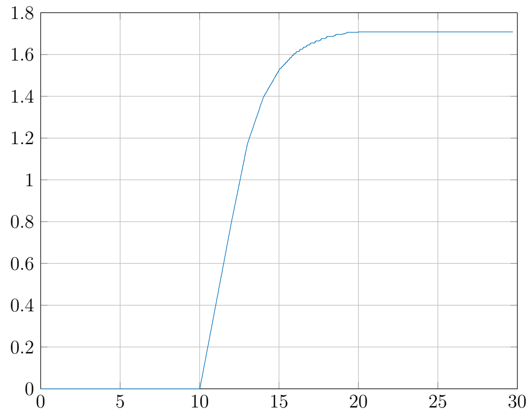
(a) The bearing of the robot over time for $(K^R_\Psi, K^T_\omega) \equiv (0.2K^R_{\Psi,max}, 0.1K^T_{\omega,max})$



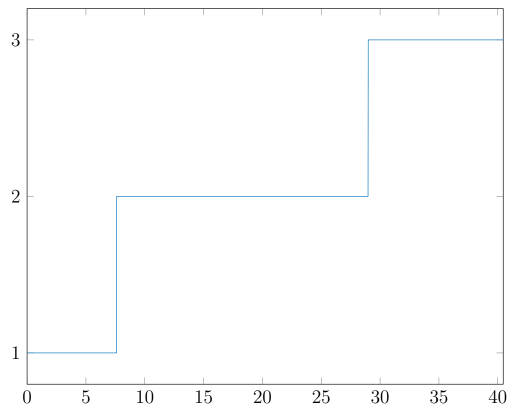
(d) The bearing of the robot in over time for $(K^R_\Psi, K^T_\omega) \equiv (0.2K^R_{\Psi,max}, 0.2K^T_{\omega,max})$



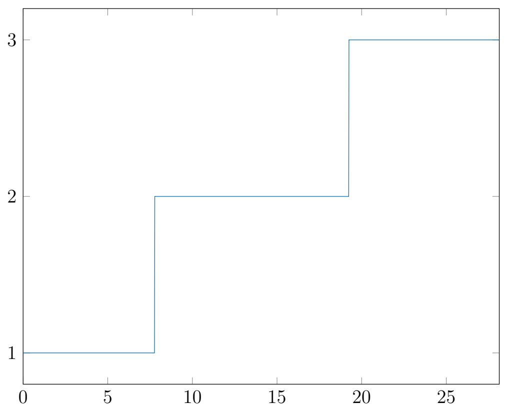
(b) The distance of the robot to the origin over time for $(K^R_\Psi, K^T_\omega) \equiv (0.2K^R_{\Psi,max}, 0.1K^T_{\omega,max})$



(e) The distance of the robot to the origin over time for $(K^R_\Psi, K^T_\omega) \equiv (0.2K^R_{\Psi,max}, 0.2K^T_{\omega,max})$

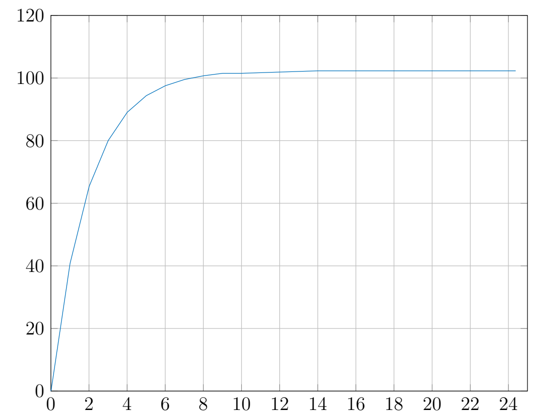


(c) The discrete state trajectory for $(K^R_\Psi, K^T_\omega) \equiv (0.2K^R_{\Psi,max}, 0.1K^T_{\omega,max})$. 1 denotes `Rotation`, 2 denotes `Translation` and 3 denotes `Stop`

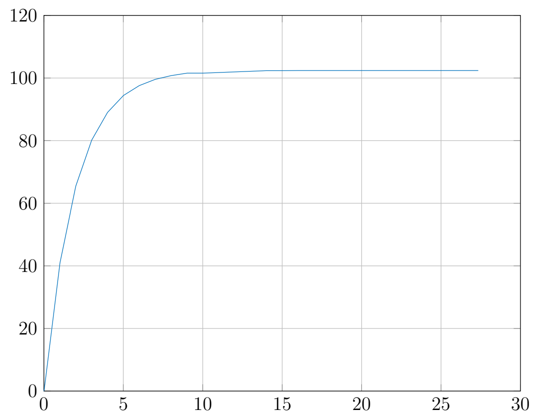


(f) The discrete state trajectory for $(K^R_\Psi, K^T_\omega) \equiv (0.2K^R_{\Psi,max}, 0.2K^T_{\omega,max})$. 1 denotes `Rotation`, 2 denotes `Translation` and 3 denotes `Stop`

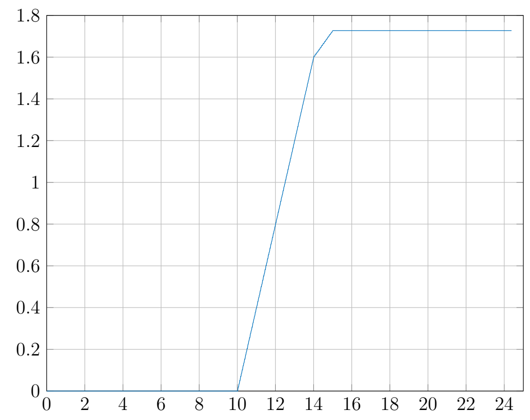
Figure 52



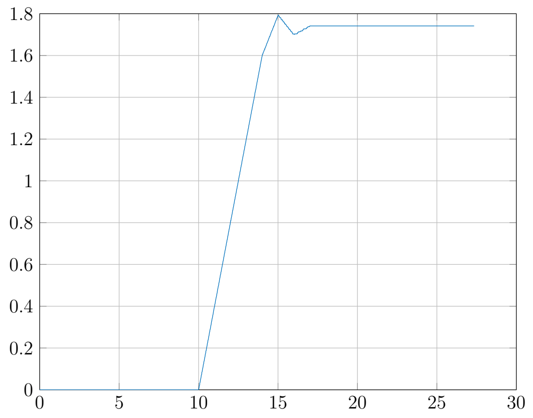
(a) The bearing of the robot over time for $(K^R_\Psi, K^T_\omega) \equiv (0.2K^R_{\Psi,max}, 0.5K^T_{\omega,max})$



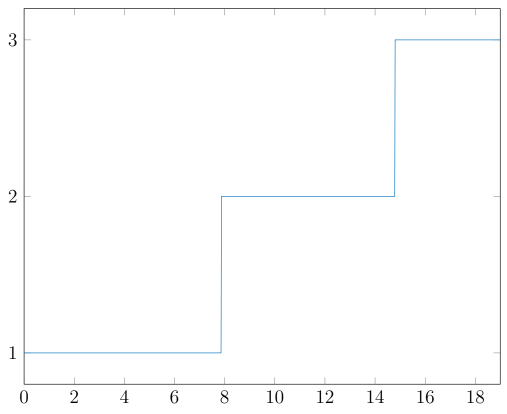
(d) The bearing of the robot in over time for $(K^R_\Psi, K^T_\omega) \equiv (0.2K^R_{\Psi,max}, 0.75K^T_{\omega,max})$



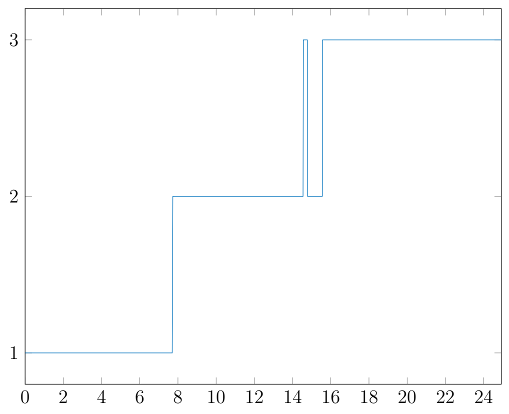
(b) The distance of the robot to the origin over time for $(K^R_\Psi, K^T_\omega) \equiv (0.2K^R_{\Psi,max}, 0.5K^T_{\omega,max})$



(e) The distance of the robot to the origin over time for $(K^R_\Psi, K^T_\omega) \equiv (0.2K^R_{\Psi,max}, 0.75K^T_{\omega,max})$

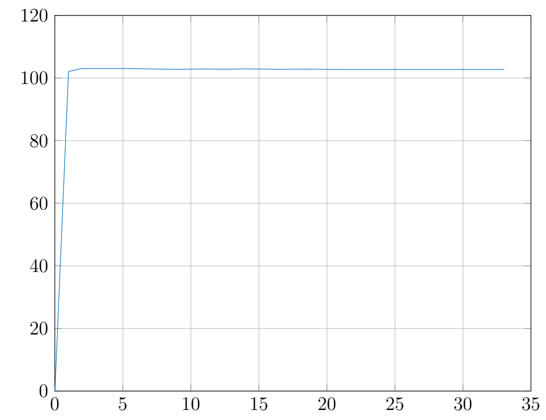


(c) The discrete state trajectory for $(K^R_\Psi, K^T_\omega) \equiv (0.2K^R_{\Psi,max}, 0.5K^T_{\omega,max})$. 1 denotes `Rotation`, 2 denotes `Translation` and 3 denotes `Stop`

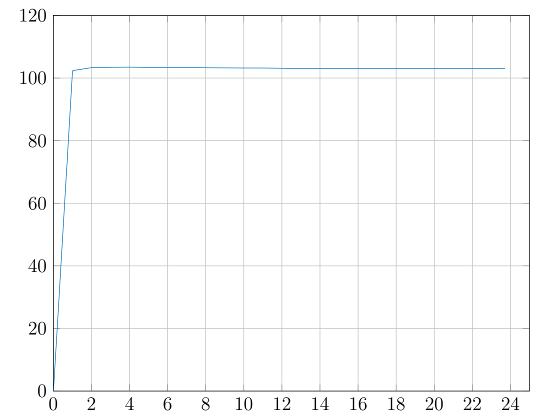


(f) The discrete state trajectory for $(K^R_\Psi, K^T_\omega) \equiv (0.2K^R_{\Psi,max}, 0.75K^T_{\omega,max})$. 1 denotes `Rotation`, 2 denotes `Translation` and 3 denotes `Stop`

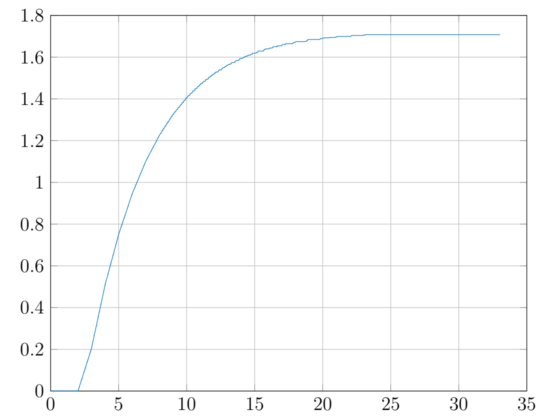
Figure 53



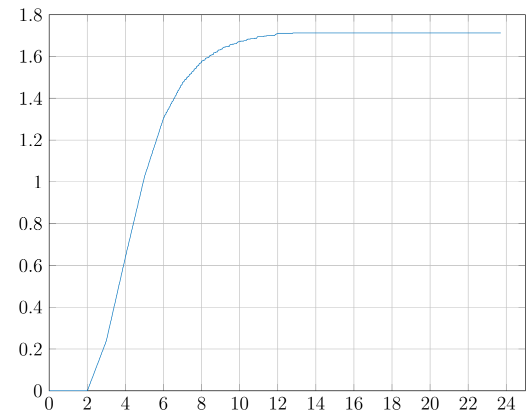
(a) The bearing of the robot over time for $(K^R_\Psi, K^T_\omega) \equiv (0.5K^R_{\Psi,max}, 0.1K^T_{\omega,max})$



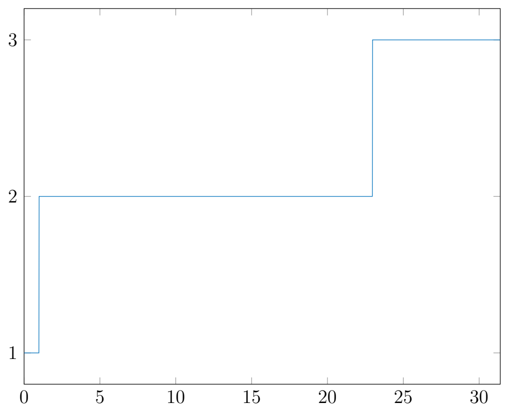
(d) The bearing of the robot in over time for $(K^R_\Psi, K^T_\omega) \equiv (0.5K^R_{\Psi,max}, 0.2K^T_{\omega,max})$



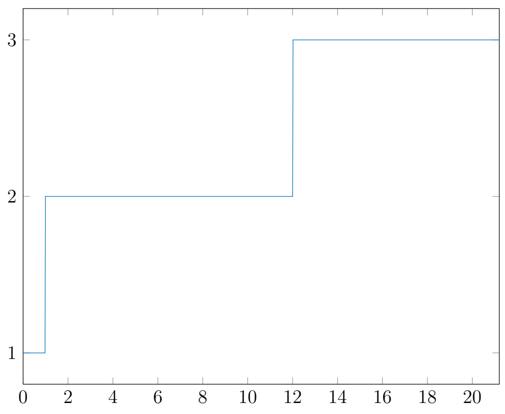
(b) The distance of the robot to the origin over time for $(K^R_\Psi, K^T_\omega) \equiv (0.5K^R_{\Psi,max}, 0.1K^T_{\omega,max})$



(e) The distance of the robot to the origin over time for $(K^R_\Psi, K^T_\omega) \equiv (0.5K^R_{\Psi,max}, 0.2K^T_{\omega,max})$

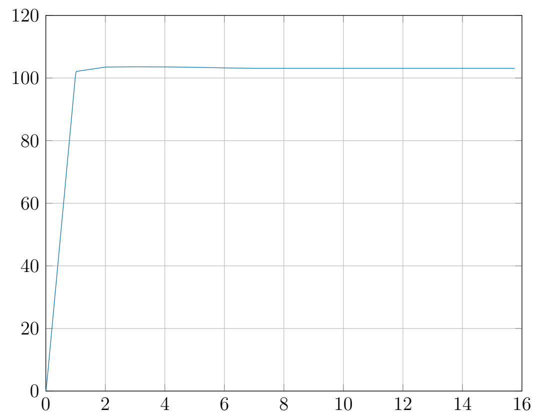


(c) The discrete state trajectory for $(K^R_\Psi, K^T_\omega) \equiv (0.5K^R_{\Psi,max}, 0.1K^T_{\omega,max})$. 1 denotes `Rotation`, 2 denotes `Translation` and 3 denotes `Stop`

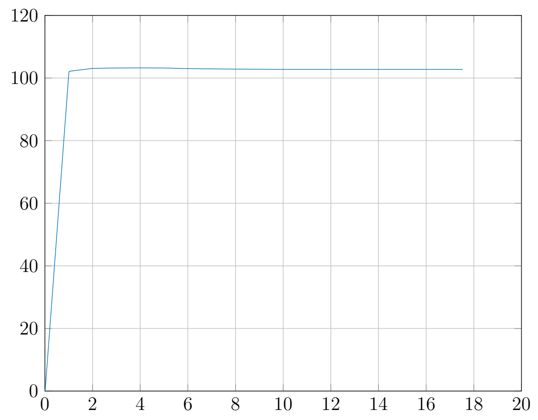


(f) The discrete state trajectory for $(K^R_\Psi, K^T_\omega) \equiv (0.5K^R_{\Psi,max}, 0.2K^T_{\omega,max})$. 1 denotes `Rotation`, 2 denotes `Translation` and 3 denotes `Stop`

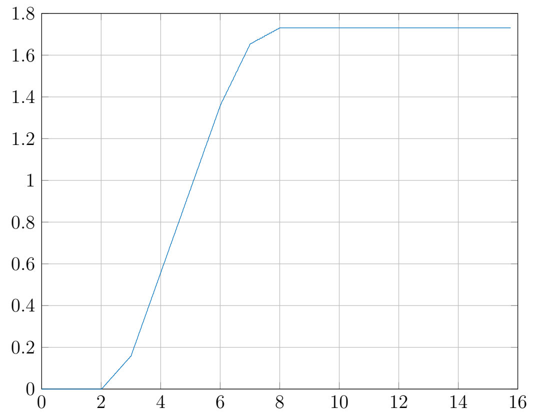
Figure 54



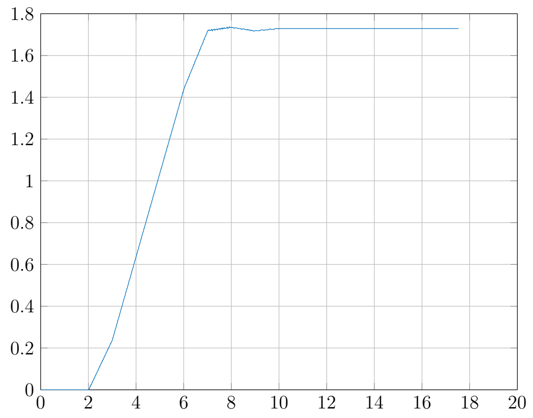
(a) The bearing of the robot over time for $(K^R_\Psi, K^T_\omega) \equiv (0.5K^R_{\Psi,max}, 0.5K^T_{\omega,max})$



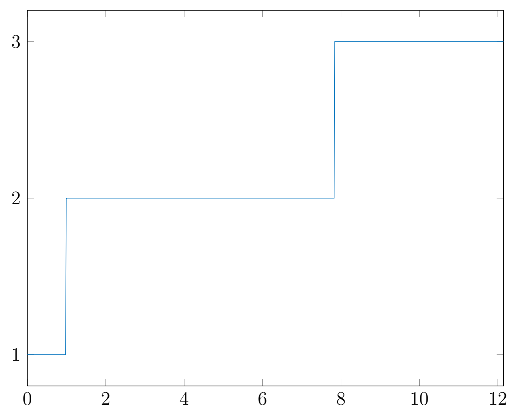
(d) The bearing of the robot in over time for $(K^R_\Psi, K^T_\omega) \equiv (0.5K^R_{\Psi,max}, 0.75K^T_{\omega,max})$



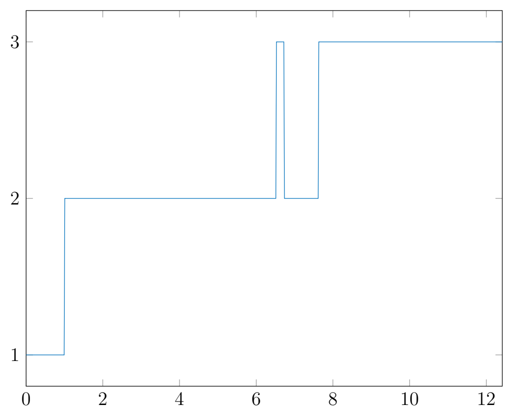
(b) The distance of the robot to the origin over time for $(K^R_\Psi, K^T_\omega) \equiv (0.5K^R_{\Psi,max}, 0.5K^T_{\omega,max})$



(e) The distance of the robot to the origin over time for $(K^R_\Psi, K^T_\omega) \equiv (0.5K^R_{\Psi,max}, 0.75K^T_{\omega,max})$



(c) The discrete state trajectory for $(K^R_\Psi, K^T_\omega) \equiv (0.5K^R_{\Psi,max}, 0.5K^T_{\omega,max})$. 1 denotes `Rotation`, 2 denotes `Translation` and 3 denotes `Stop`



(f) The discrete state trajectory for $(K^R_\Psi, K^T_\omega) \equiv (0.5K^R_{\Psi,max}, 0.75K^T_{\omega,max})$. 1 denotes `Rotation`, 2 denotes `Translation` and 3 denotes `Stop`

Figure 55

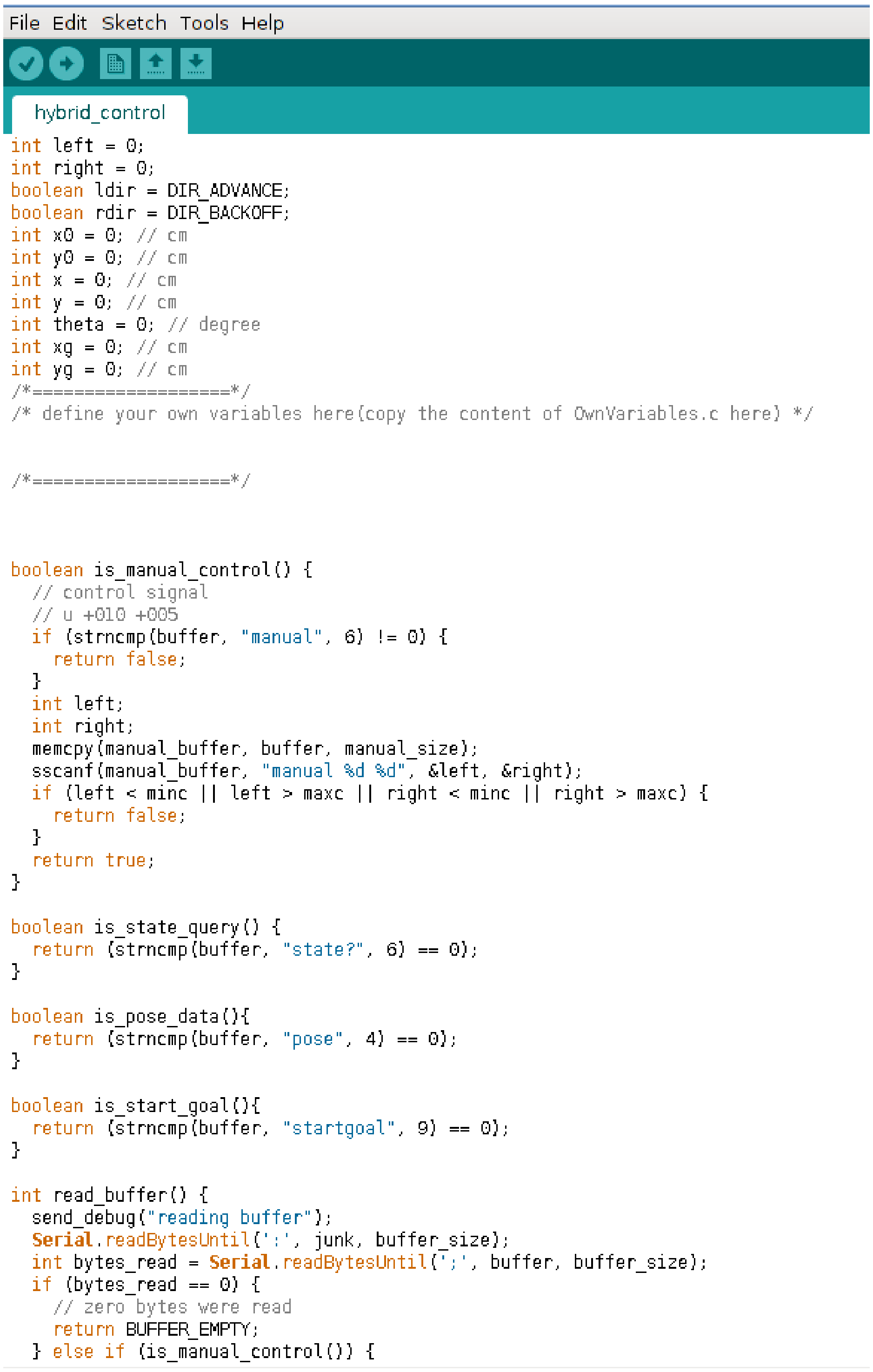
## Task 20

First, we copy files `Controller.c`, `OwnVariables.c` and `RenewControllerState.c` by opening each one, pressing `Ctrl+A` and then `Ctrl+C` inside the editor.

We then navigate to the directory

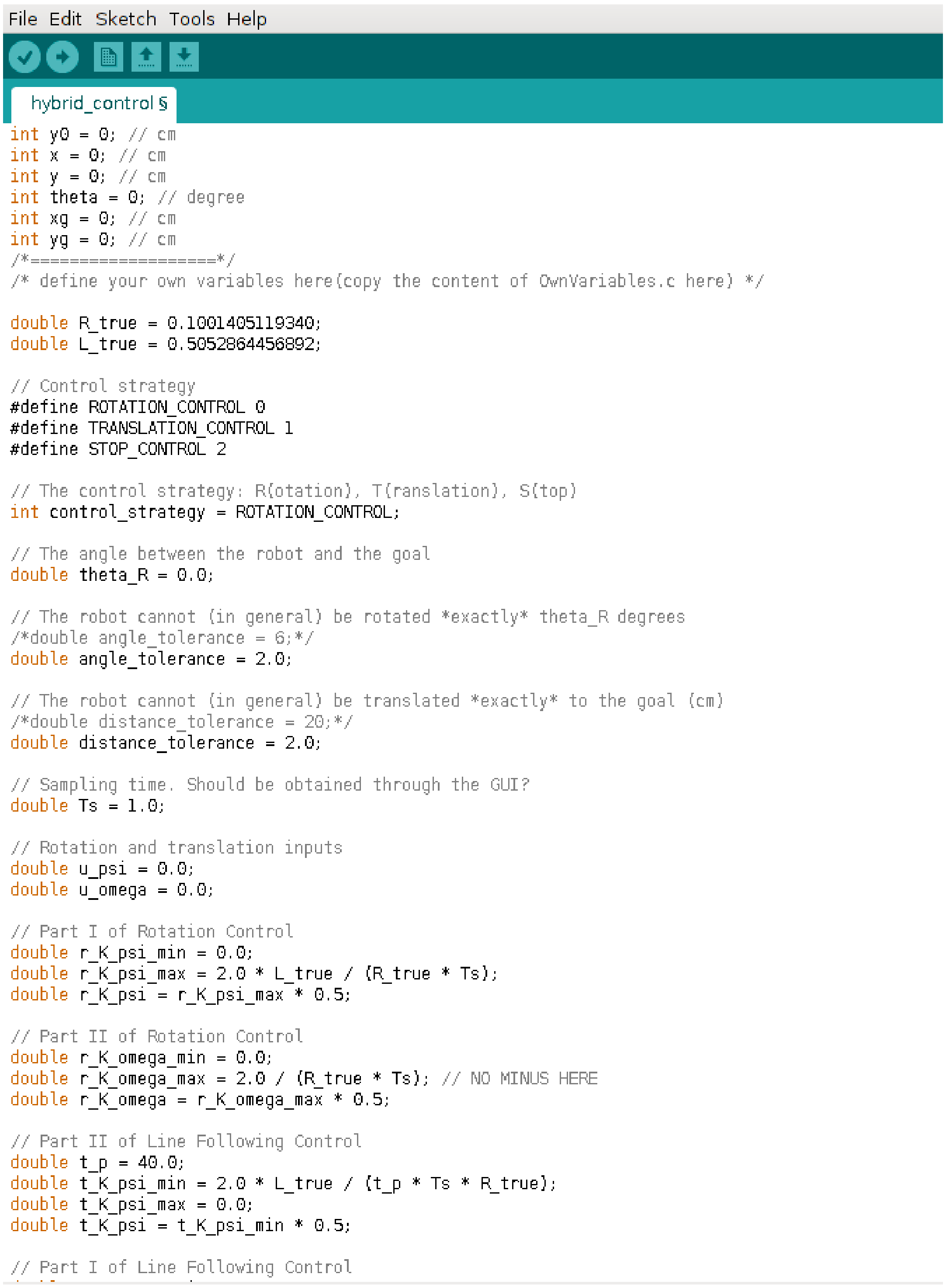
`SOURCE_RESIDENCE/Simulation Environment/ArduinoFiles/hybrid_control.ino`

and double-click on `hybrid_control.ino`. We scroll down until we locate the parts where our code should be pasted. Upon finding them we immediately hit `Ctrl+V`. Figure **??** shows a part of the process before pasting the relevant copied file and figure **??** shows the outcome of applying the action `Ctrl+V`.



```
int left = 0;
int right = 0;
boolean ldir = DIR_ADVANCE;
boolean rdir = DIR_BACKOFF;
int x0 = 0; // cm
int y0 = 0; // cm
int x = 0; // cm
int y = 0; // cm
int theta = 0; // degree
int xg = 0; // cm
int yg = 0; // cm
/*===================*/
/* define your own variables here(copy the content of OwnVariables.c here) */


/*===================*/



boolean is_manual_control() {
  // control signal
  // u +010 +005
  if (strncmp(buffer, "manual", 6) != 0) {
    return false;
  }
  int left;
  int right;
  memcpy(manual_buffer, buffer, manual_size);
  sscanf(manual_buffer, "manual %d %d", &left, &right);
  if (left < minc || left > maxc || right < minc || right > maxc) {
    return false;
  }
  return true;
}

boolean is_state_query() {
  return (strncmp(buffer, "state?", 6) == 0);
}

boolean is_pose_data(){
  return (strncmp(buffer, "pose", 4) == 0);
}

boolean is_start_goal(){
  return (strncmp(buffer, "startgoal", 9) == 0);
}

int read_buffer() {
  send_debug("reading buffer");
  Serial.readBytesUntil(':', junk, buffer_size);
  int bytes_read = Serial.readBytesUntil(';', buffer, buffer_size);
  if (bytes_read == 0) {
    // zero bytes were read
    return BUFFER_EMPTY;
  } else if (is_manual_control()) {
```

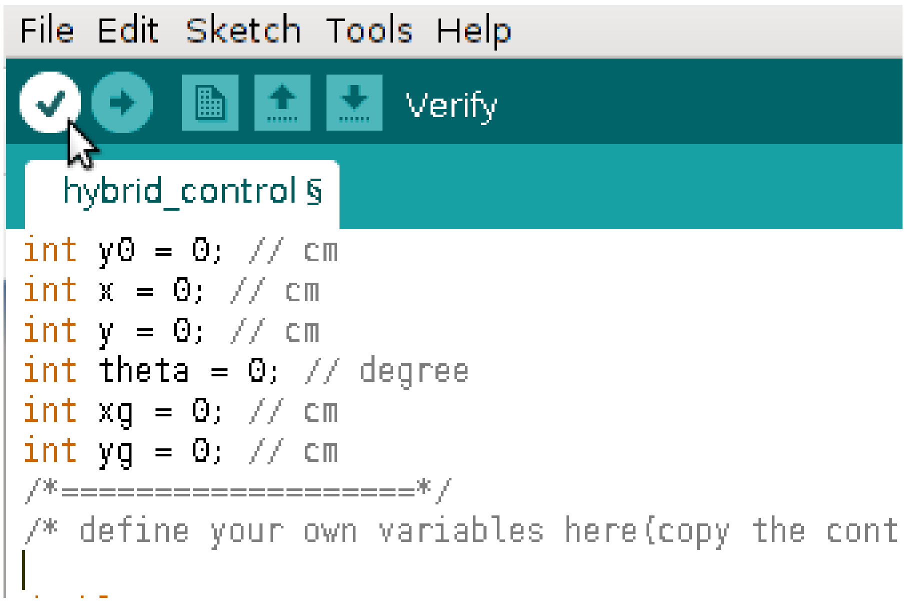
Figure 56: `hybrid_control.ino` before copying the contents of `OwnVariables.c` in it.



```
int y0 = 0; // cm
int x = 0; // cm
int y = 0; // cm
int theta = 0; // degree
int xg = 0; // cm
int yg = 0; // cm
/*===================*/
/* define your own variables here(copy the content of OwnVariables.c here) */

double R_true = 0.1001405119340;
double L_true = 0.5052864456892;

// Control strategy
#define ROTATION_CONTROL 0
#define TRANSLATION_CONTROL 1
#define STOP_CONTROL 2

// The control strategy: R(otation), T(ranslation), S(top)
int control_strategy = ROTATION_CONTROL;

// The angle between the robot and the goal
double theta_R = 0.0;

// The robot cannot (in general) be rotated *exactly* theta_R degrees
/*double angle_tolerance = 6;*/
double angle_tolerance = 2.0;

// The robot cannot (in general) be translated *exactly* to the goal (cm)
/*double distance_tolerance = 20;*/
double distance_tolerance = 2.0;

// Sampling time. Should be obtained through the GUI?
double Ts = 1.0;

// Rotation and translation inputs
double u_psi = 0.0;
double u_omega = 0.0;

// Part I of Rotation Control
double r_K_psi_min = 0.0;
double r_K_psi_max = 2.0 * L_true / (R_true * Ts);
double r_K_psi = r_K_psi_max * 0.5;

// Part II of Rotation Control
double r_K_omega_min = 0.0;
double r_K_omega_max = 2.0 / (R_true * Ts); // NO MINUS HERE
double r_K_omega = r_K_omega_max * 0.5;

// Part II of Line Following Control
double t_p = 40.0;
double t_K_psi_min = 2.0 * L_true / (t_p * Ts * R_true);
double t_K_psi_max = 0.0;
double t_K_psi = t_K_psi_min * 0.5;

// Part I of Line Following Control
```

Figure 57: `hybrid_control.ino` after copying the contents of `OwnVariables.c` in it.

After the successful insertion of the contents of the aforementioned files, we move the mouse to upper-left corner where we find a tick symbol, illustrated in figure ??. We then press it using the left button of the mouse.



```
int y0 = 0; // cm
int x = 0; // cm
int y = 0; // cm
int theta = 0; // degree
int xg = 0; // cm
int yg = 0; // cm
/*==================*/
/* define your own variables here(copy the cont
```

Figure 58: After the injection of the contents of the three files we locate the tick symbol and click it

## Task 21

Reaching a certain accuracy is difficult. First one has to orientate and then to translate. As there is no automatic line-following, one has to correct many times to reach a certain point, while reaching a certain area should be easier. The accuracy in both bearing and distance is related to the sensitivity of the motors driving the wheels to their input signals and their output rotation precision.

## Task 22

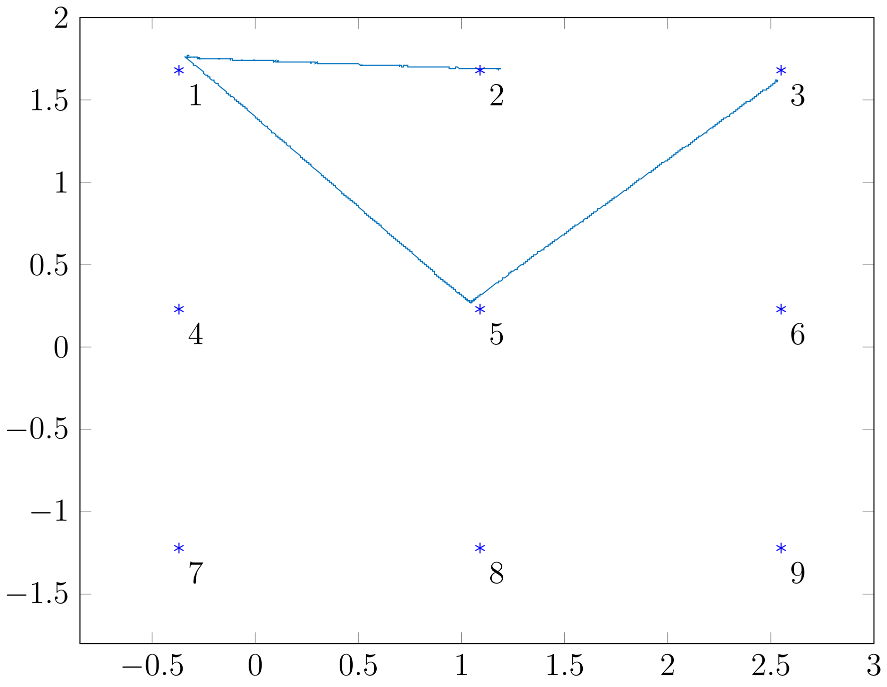


Figure 59