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

Rene Garcia
20010124-5512
reneogt@kth.se

First name2 Last name2
person number
email

Joel Aggefors
20030301-4575
joelag@kth.se

Jenis Jain
20030424-T490
jjjain@kth.se

First name5 Last name5
person number
email

February 26, 2026

Task 1: Compute u_r and u_l from (v, ω)

The robot inputs are defined as :

$$v = \frac{u_r + u_l}{2}, \qquad \omega = u_r - u_l. \tag{1}$$

From (1), multiply the first equation by 2:

$$2v = u_r + u_l. \tag{2}$$

Now add (2) and the second equation in (1):

$$2v + \omega = (u_r + u_l) + (u_r - u_l) = 2u_r \quad \Rightarrow \quad u_r = v + \frac{\omega}{2}. \tag{3}$$

Similarly, subtract the second equation in (1) from (2):

$$2v - \omega = (u_r + u_l) - (u_r - u_l) = 2u_l \quad \Rightarrow \quad u_l = v - \frac{\omega}{2}. \tag{4}$$

Therefore, the wheel speeds corresponding to (v, ω) are:

$$\boxed{u_r = v + \frac{\omega}{2} \quad [1^\circ/s], \qquad u_l = v - \frac{\omega}{2} \quad [1^\circ/s]} \tag{5}$$

Task 2

(a) Discretization, numbering , and atomic propositions

Workspace: $[-1.5, 1.5] \times [-1.5, 1.5]$ (meters). A discretization with $K = 36$ gives a 6×6 grid.

R_6 G,S	R_7 B	R_{18} -	R_{19} O	R_{30} -	R_{31} R,G,O
R_5 -	R_8 R,O	R_{17} -	R_{20} R	R_{29} B,O	R_{32} R
R_4 -	R_9 G	R_{16} -	R_{21} B	R_{28} -	R_{33} -
R_3 O	R_{10} O	R_{15} -	R_{22} O	R_{27} -	R_{34} -
R_2 G	R_{11} R,O	R_{14} -	R_{23} B	R_{26} R,O	R_{35} -
R_1 R	R_{12} -	R_{13} -	R_{24} -	R_{25} G	R_{36} B

Figure 1: Workspace discretization with $K = 36$ regions. Labels indicate where atomic propositions hold.

regions are numbered in a *column-wise snake* pattern: start at the bottom-left with R_1, go *up* in the first column, then *down* in the next column, and so on. Thus, the 6×6 numbering is:

R_6	R_7	R_{18}	R_{19}	R_{30}	R_{31}
R_5	R_8	R_{17}	R_{20}	R_{29}	R_{32}
R_4	R_9	R_{16}	R_{21}	R_{28}	R_{33}
R_3	R_{10}	R_{15}	R_{22}	R_{27}	R_{34}
R_2	R_{11}	R_{14}	R_{23}	R_{26}	R_{35}
R_1	R_{12}	R_{13}	R_{24}	R_{25}	R_{36}

Atomic propositions: Let $AP = \{\mathsf{red}, \mathsf{blue}, \mathsf{green}, \mathsf{obstacle}\}$. The propositions are specified by the following positions (meters):

- $\mathsf{obstacle}$ centers (spheres of radius 0.05 m) at:

  $$(-0.75, 0.75),\ (0.25, 1.25),\ (0.75, 0.75),\ (0.25, -0.25),\ (0.75, -0.75),\ (-0.75, -0.75),\ (-0.75, -0.2$$

- red holds at:

  $$(-0.75, 0.7),\ (0.2, 0.7),\ (1.25, 1.25),\ (1.2, 0.8),\ (0.8, -0.8),\ (-0.9, -0.8),\ (-1.25, -1.25).$$

- blue holds at:

  $$(-0.75, 1.4),\ (0.9, 0.9),\ (0.3, 0.2),\ (0.25, -0.75),\ (1.2, -1.4).$$

- green holds at:

  $$(-1.23, 1.25),\ (1.25, 1.25),\ (-0.9, 0.2),\ (-1.2, -0.7),\ (0.6, -1.2).$$

Labeling rule: a proposition is true in the region that contains its given position, i.e.

$$p \in L(R_i) \iff (x_p, y_p) \in R_i, \qquad p \in AP.$$

The start position is $(-1.25, 1.25)$, hence $S_0 = \{R_6\}$.

(b) Cell size dx, dy

Since the side length is 3 m and there are 6 cells per side,

$$dx = \frac{3}{6} = 0.5 \text{ m}, \qquad dy = \frac{3}{6} = 0.5 \text{ m}.$$

(c) Comment on the choice $K = 36$

A 6×6 grid provides a finer abstraction than coarse grids (better separation of obstacles/colored areas), but increases the number of states and transitions compared to smaller K.

(d) Transition system $T = (S, S_0, \Sigma, \to, AP, L)$

$S = \{R_1, \dots, R_{36}\}$, $\quad S_0 = \{R_6\}$, $\quad \Sigma = \{\mathsf{Up}, \mathsf{Down}, \mathsf{Left}, \mathsf{Right}\}, AP = \{\mathsf{red}, \mathsf{blue}, \mathsf{green}, \mathsf{obstacle}\}$.

The transition relation $\to$ is the 4-neighborhood relation on the grid:

$$R_i \xrightarrow{\sigma} R_j \quad \Longleftrightarrow \quad R_j \text{ is the adjacent region to } R_i \text{ in direction } \sigma \in \Sigma.$$

The labeling function $L : S \to 2^{AP}$ is defined using the point-in-region rule above.

Q3: Find an infinite path satisfying the specification

Specification: (i) visit red infinitely often, (ii) whenever the robot is in a red region, the *next* region is blue, (iii) never enter a region labeled obstacle.

Chosen red–blue pair: From the labeling in Q2, $R_{20} \in \mathsf{red}$ and $R_{21} \in \mathsf{blue}$, and they are adjacent in the 6×6 grid (R_{21} is directly below R_{20}). Moreover, $R_{20}, R_{21} \notin \mathsf{obstacle}$.

A valid infinite path: Starting from $S_0 = \{R_6\}$, one feasible prefix to reach R_{20} without entering obstacles is

$$R_6 \to R_7 \to R_{18} \to R_{17} \to R_{16} \to R_{21} \to R_{20}.$$

Then, repeat the 2-cycle (R_{20}, R_{21}) forever:

$$\pi = \underbrace{(R_6, R_7, R_{18}, R_{17}, R_{16}, R_{21}, R_{20})}_{\text{prefix}} \cdot \underbrace{(R_{21}, R_{20})^{\omega}}_{\text{suffix repeated forever}} .$$

Why π satisfies the specification:

- π visits R_{20} infinitely often, and $R_{20} \in \mathsf{red}$, hence red is visited infinitely often.
- Every time π is in R_{20} (a red region), the next state is R_{21} and $R_{21} \in \mathsf{blue}$, so the "after red next is blue" condition holds.
- All regions used in π are chosen outside the set of obstacle regions, hence obstacles are never entered.

Q4

The hybrid strategy prevents entering unintended regions by separating the motion into two simple phases. First, the robot uses a rotation mode with (approximately) zero forward speed, i.e., $v \approx 0$, so it turns in place to align its heading with the straight line connecting the center of the current region to the center of the target (neighbor) region. Because the robot does not translate during this phase, it stays close to the current region center and does not drift into adjacent regions. Second, the robot switches to a line-following mode and drives forward while tracking that same center-to-center line. For neighboring regions, this line segment lies within the union of the two adjacent cells. Therefore, if the tracking error is kept small, the robot remains inside only the current and target regions during the transition, and it avoids passing through any other region that could contain an obstacle.

Q5

During the rotation mode, the controller is

$$\omega[k] = K_{\Psi,1}\big(\theta_R - \theta[k]\big). \tag{6}$$

The robot yaw dynamics satisfy

$$\dot{\theta}(t) = \frac{R}{L}\,\omega(t) \quad [1^\circ/s]. \tag{7}$$

Using forward Euler discretization with sampling time h,

$$\theta[k+1] = \theta[k] + h\frac{R}{L}\omega[k]. \tag{8}$$

Substituting (6) into (8) and defining the error $e[k] = \theta_R - \theta[k]$, we obtain

$$e[k+1] = \Big(1 - \frac{hR}{L}K_{\Psi,1}\Big)e[k]. \tag{9}$$

For asymptotic stability of the discrete-time error dynamics, we require

$$\left|1 - \frac{hR}{L}K_{\Psi,1}\right| < 1 \iff 0 < \frac{hR}{L}K_{\Psi,1} < 2 \iff \boxed{0 < K_{\Psi,1} < \frac{2L}{hR}}. \tag{10}$$

A practical choice is to pick $K_{\Psi,1}$ well inside this interval (e.g., $K_{\Psi,1} = \alpha\frac{L}{hR}$ with $\alpha \in (0,1)$) to avoid oscillations and actuator saturation.

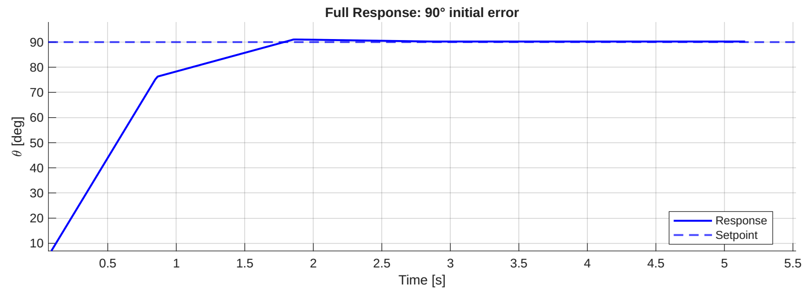
Task 6

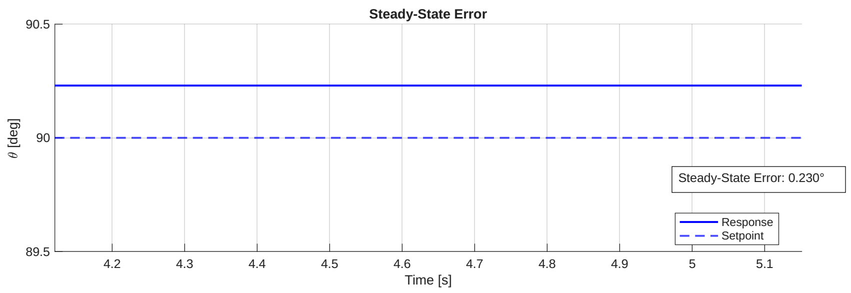
First we define the proportional constant value. Given eq. (10), we could aim for a deadbeat controller by setting $K_{\Psi,1} = \frac{2L}{hR}$, nonetheless, this would force the controller to correct the error in one sampling step, completely ignoring physical limits and inertia. A safer value to chose is the middle of the stability range:

$$K_{\Psi,1} = \frac{1}{2}\frac{2L}{hR} = \frac{L}{hR} = \frac{0.16}{0.033 * 1} = 4.85 \quad [1/s]. \tag{11}$$

To test the performance of the controller, we run 2 different scenarios:

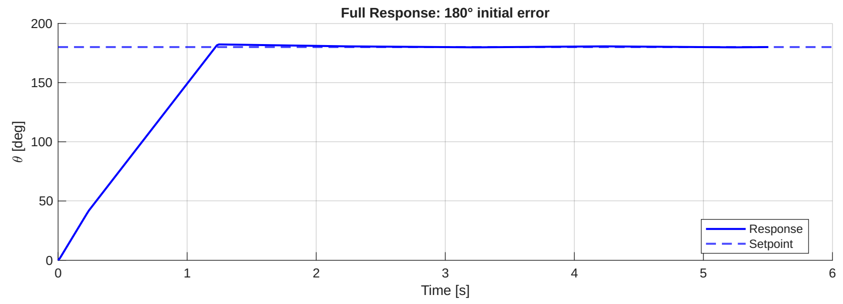
- **Scenario 1:** $\theta_R = 90^\circ$, $\theta[0] = 0^\circ$ (90 degree turn)

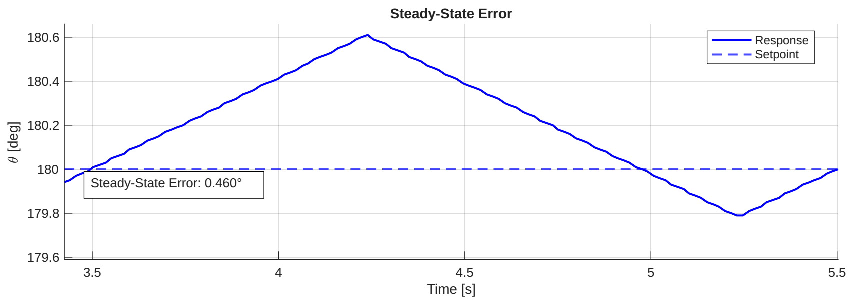




For this case, the wheels stop turning at around $1.85\,s$, and there is a **steady-state** error of 0.23°.

- **Scenario 2:** $\theta_R = 180^\circ$, $\theta[0] = 0^\circ$ (180 degree turn)





For this case, there is a constant ripple that slowly rotates the robot back and forth around the target angle. This is a case of limit cycle caused by quantization. For this case, we consider the largest error, that is, the peak of the ripple. Then, the **steady-state** error is 0.46°.

In general, can see that the controller **asymptotically stabilizes** the error in both scenarios, as we don't see divergence over time.

Task 7

Assume $\theta[k+1] = \theta[k] = \theta$ so $v_c = \begin{bmatrix}\cos\theta & \sin\theta\end{bmatrix}^\top$ is constant and $v_c^\top v_c = 1$. Using $\dot{p} = Rvv_c$ with $p = \begin{bmatrix}x & y\end{bmatrix}^\top$ and forward Euler,

$$p[k+1] = p[k] + hR\,v[k]\,v_c \;\Rightarrow\; \Delta_0[k+1] = \Delta_0[k] - hR\,v[k]\,v_c.$$

Premultiplying by $v_c^\top$ yields

$$d_0[k+1] = v_c^\top \Delta_0[k+1] = d_0[k] - hR\,v[k].$$

With $v[k] = K_{\omega,1} d_0[k]$,

$$d_0[k+1] = \bigl(1 - hRK_{\omega,1}\bigr) d_0[k].$$

Asymptotic stability requires $|1 - hRK_{\omega,1}| < 1$, hence

$$\boxed{0 < K_{\omega,1} < \frac{2}{hR}}.$$

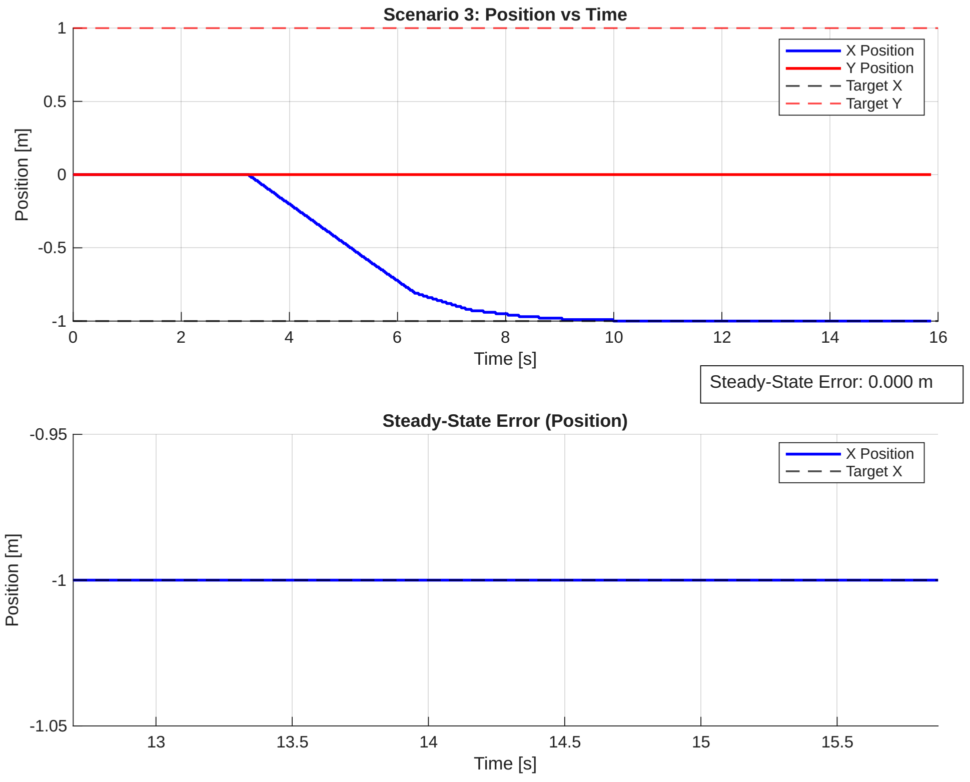
Task 8

Similarly to Task 6, we select $K_{\omega,1}$ in the middle of the stability interval:

$$K_{\omega,1} = \frac{1}{2}\frac{2}{hR} = \frac{1}{hR} = \frac{1}{0.033 * 1} = 30.3 \quad [\frac{1}{m \cdot s}]. \tag{12}$$

To test the performance of the controller, we run 1 single scenario:

- **Scenario 1: Diagonal Error**
  Starting position: $(x_0, y_0) = (0, 0)$, $\theta = 0°$ Target: $(x_g, y_g) = (-1, 1)$

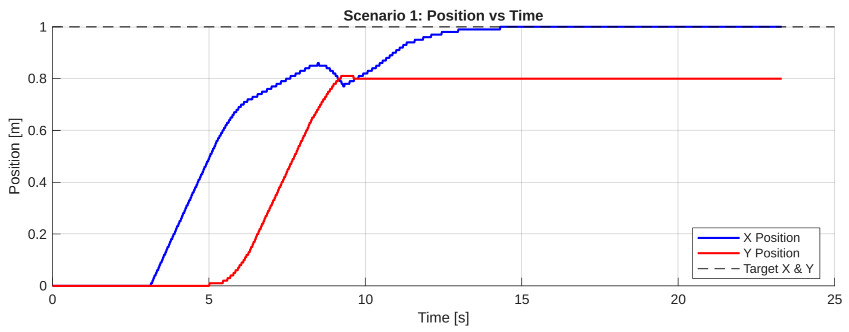


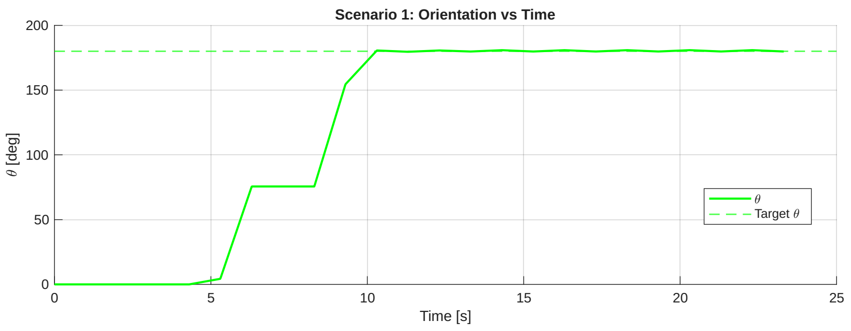
As we can see, the robot moves only in the x direction, as this is the direction it is facing initially. The controller successfully reduces the X error to zero, but the Y error remains.

Task 9

To test both controllers together, we simulate an scenario where we have error in the position and angle, and enable both controllers.

- **Scenario 1:** Starting position: $(x_0, y_0) = (0, 0)$, $\theta = 0°$ Target: $(x_g, y_g) = (1, 1)$, $\theta^R = 180°$





As we can see, the angle error is corrected first, and then blocking the posiblitiy of correcting the position error. This can become a race condition.

Task 10

Assume $\theta[k] = \theta_g$, hence $v_g = \begin{bmatrix} \cos\theta_g & \sin\theta_g \end{bmatrix}^\top$ is constant and $v_g^\top v_g = 1$. With $\dot{p} = Rvv_g$ and forward Euler,

$$p[k+1] = p[k] + hR\,v[k]\,v_g \;\Rightarrow\; \Delta_g[k+1] = \Delta_g[k] - hR\,v[k]\,v_g.$$

Premultiplying by $v_g^\top$ yields

$$d_g[k+1] = v_g^\top \Delta_g[k+1] = d_g[k] - hR\,v[k].$$

Using $v[k] = K_{\omega,2} d_g[k]$,

$$d_g[k+1] = \big(1 - hRK_{\omega,2}\big) d_g[k].$$

Asymptotic stability requires $|1 - hRK_{\omega,2}| < 1$, hence

$$\boxed{0 < K_{\omega,2} < \frac{2}{hR}}.$$

A practical choice is to select $K_{\omega,2}$ well inside this interval (e.g., $K_{\omega,2} = \alpha \frac{1}{hR}$ with $\alpha \in (0, 1)$) to ensure monotone convergence and robustness to discretization/actuator limits.

Task 11

Solution to the task

Task 12

The controller for line-following (part II) is

$$\omega[k] = K_{\Psi,2}\, d_p[k].$$

Assume the robot is on the line from (x_0, y_0) to (x_g, y_g) and θ is close to θ_g so that

$$d_p[k] \approx p(\theta_g - \theta[k]), \qquad p > 0.$$

Let $e[k] = \theta_g - \theta[k]$, hence $d_p[k] \approx p\, e[k]$. From the yaw dynamics $\dot{\theta} = \frac{R}{L}\omega$ and forward Euler with sampling time h,

$$\theta[k+1] = \theta[k] + h\frac{R}{L}\omega[k].$$

Therefore,

$$e[k+1] = \theta_g - \theta[k+1] = e[k] - h\frac{R}{L}\omega[k] = e[k] - h\frac{R}{L}K_{\Psi,2}d_p[k] \approx \Big(1 - h\frac{R}{L}K_{\Psi,2}p\Big)e[k].$$

Multiplying by p gives the discrete-time dynamics of d_p:

$$d_p[k+1] \approx \Big(1 - h\frac{R}{L}K_{\Psi,2}p\Big)d_p[k].$$

Thus $d_p[k]$ is asymptotically stabilized in 0 iff

$$\left|1 - h\frac{R}{L}K_{\Psi,2}p\right| < 1 \quad \Longleftrightarrow \quad \boxed{0 < K_{\Psi,2} < \frac{2L}{hRp}}.$$

Choice of $K_{\Psi,2}$. Select $K_{\Psi,2}$ strictly inside the stability interval, e.g.

$$K_{\Psi,2} = \alpha\,\frac{L}{hRp}, \qquad \alpha \in (0,1),$$

which yields the contraction factor $1 - \alpha$ (monotone convergence) and keeps the closed-loop behavior consistent when the sampling time h changes.

Task 13

Solution to the task

Task 14

Solution to the task

Task 15

Solution to the task

Task 16 : Hybrid automaton

Define the hybrid automaton $H = (Q, X, Init, f, D, E, G, R)$.

Discrete states.

$$Q = \{q_{\text{rot}},\ q_{\text{line}}\},$$

where q_{rot} aligns the robot with the goal direction and q_{line} performs line-following / go-to-goal.

Continuous state and initialization.

$$X = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix}, \qquad Init = \left\{ (q_{\text{rot}}, X) \;\middle|\; X = \begin{bmatrix} x_s \\ y_s \\ \theta_s \end{bmatrix} \right\}.$$

Continuous dynamics (closed-loop). Robot kinematics:

$$\dot{x} = R\,v\cos\theta, \qquad \dot{y} = R\,v\sin\theta, \qquad \dot{\theta} = \frac{R}{L}\omega.$$

Let $\theta_R = \text{atan2}(y_g - y,\ x_g - x)$.

Mode q_{rot}:

$$\omega = K_{\Psi,1}(\theta_R - \theta), \qquad v = K_{\omega,1} d_0.$$

Mode q_{line}:

$$v = K_{\omega,2} d_g, \qquad \omega = K_{\Psi,2} d_p.$$

Domains.

$$D(q_{\text{rot}}) = D(q_{\text{line}}) = \mathbb{R}^2 \times (-180^\circ, 180^\circ].$$

Edges and guards.

$$E = \{(q_{\text{rot}}, q_{\text{line}}), (q_{\text{line}}, q_{\text{rot}})\}.$$

Using thresholds $\varepsilon_\theta > 0$ and $\varepsilon_g > 0$:

$$G(q_{\text{rot}}, q_{\text{line}}) = \{X :\ |\theta_R - \theta| \le \varepsilon_\theta\}, \qquad G(q_{\text{line}}, q_{\text{rot}}) = \{X :\ \|(x_g - x,\ y_g - y)\| \le \varepsilon_g\}.$$

Resets. No state jump at switching (identity reset):

$$R(q_{\text{rot}}, q_{\text{line}}) :\ X^+ = X, \qquad R(q_{\text{line}}, q_{\text{rot}}) :\ X^+ = X.$$

(If multiple waypoints are used, the next goal (x_g, y_g) is updated externally when ε_g is reached.)

Task 17

Solution to the task

Task 18

Solution to the task

Task 19

Solution to the task

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.