KU LEUVEN

FACULTEIT
INGENIEURSWETENSCHAPPEN

B-KUL-H04X3A: Control Theory

Team members: Lefebure Tiebert (r0887630), Campaert Lukas (r0885501)

Assignment 5: Estimation and control of a two-wheel driven cart

1 Model the system

(a) State equation and relation between motor commands and (v, ω)

Wheel radius r and half wheelbase a relate the wheel speeds (ω_L, ω_R) to forward/rotational velocity

$$v = \tfrac{r}{2}\left(\omega_L + \omega_R\right), \qquad \omega = \tfrac{r}{2a}\left(\omega_R - \omega_L\right), \tag{1}$$

and conversely $\omega_L = \frac{1}{r}(v - a\omega)$, $\omega_R = \frac{1}{r}(v + a\omega)$. With state $\xi = [x_c\ y_c\ \theta]^T$ in the world frame and input $u = [v\ \omega]^T$,

$$\dot{\xi} = f(\xi, u) = \begin{bmatrix} v\cos\theta \\ v\sin\theta \\ \omega \end{bmatrix}, \qquad \xi(t_0) = [x_0\ y_0\ \theta_0]^T. \tag{2}$$

The model assumes ideal velocity control and no wheel slip (per spec).

(b) Measurement equation (general and Figure 4)

A wall $\mathcal{W} : px + qy = r$ has unit normal $\hat{n} = \frac{1}{\sqrt{p^2+q^2}}[p\ q]^T$. Sensor offsets (measured on the cart) in the body frame are $\alpha = 0.075\,\text{m}$ (front), $\beta = 0.065\,\text{m}$ (longitudinal side), $\gamma = 0.078\,\text{m}$ (lateral side, left). Sensor positions in world coordinates are

$$\begin{aligned} x_f &= x_c + \alpha\cos\theta, & y_f &= y_c + \alpha\sin\theta, \\ x_s &= x_c + \beta\cos\theta - \gamma\sin\theta, & y_s &= y_c + \beta\sin\theta + \gamma\cos\theta. \end{aligned}$$

The signed distance from (x, y) to $\mathcal{W}$ is $h(x, y; p, q, r) = \frac{r - px - qy}{\sqrt{p^2+q^2}}$. The nonlinear measurement model is

$$z = \begin{bmatrix} h(x_f, y_f; p_1, q_1, r_1) \\ h(x_s, y_s; p_2, q_2, r_2) \end{bmatrix} + v_m, \qquad v_m \sim \mathcal{N}(0, R). \tag{3}$$

For Figure 4, wall 1 is $y = 0$ and wall 2 is $x = 0$, so $z_1 = -y_f$ and $z_2 = -x_s$.

2 Extended Kalman filter derivation

(a) Forward Euler discretisation

With sampling time T_s,

$$\xi_{k+1} = \xi_k + T_s f(\xi_k, u_k) + w_k, \qquad w_k \sim \mathcal{N}(0, Q), \tag{4}$$

$$z_k = h(\xi_k) + v_k, \qquad v_k \sim \mathcal{N}(0, R). \tag{5}$$

(b) Linearisation and Jacobians

Linearising around ξ^*, u^* gives $\delta\xi_{k+1} = A_k\delta\xi_k + B_k\delta u_k + w_k$, $\delta z_k = C_k\delta\xi_k + v_k$, with

$$A_k = \begin{bmatrix} 1 & 0 & -T_s v^* \sin\theta^* \\ 0 & 1 & T_s v^* \cos\theta^* \\ 0 & 0 & 1 \end{bmatrix}, \qquad B_k = \begin{bmatrix} T_s\cos\theta^* & 0 \\ T_s\sin\theta^* & 0 \\ 0 & T_s \end{bmatrix}, \tag{6}$$

$$C_k = \begin{bmatrix} -\frac{p_1}{n_1} & -\frac{q_1}{n_1} & -\frac{p_1\dot{x}_f^* + q_1\dot{y}_f^*}{n_1} \\ -\frac{p_2}{n_2} & -\frac{q_2}{n_2} & -\frac{p_2\dot{x}_s^* + q_2\dot{y}_s^*}{n_2} \end{bmatrix}, \tag{7}$$

where $n_i = \sqrt{p_i^2 + q_i^2}$, $\dot{x}_f^* = -\alpha\sin\theta^*$, $\dot{y}_f^* = \alpha\cos\theta^*$, $\dot{x}_s^* = -\beta\sin\theta^* - \gamma\cos\theta^*$, $\dot{y}_s^* = \beta\cos\theta^* - \gamma\sin\theta^*$. For the Figure 4 walls this reduces to

$$C_k = \begin{bmatrix} 0 & -1 & -\alpha\cos\theta^* \\ -1 & 0 & \beta\sin\theta^* + \gamma\cos\theta^* \end{bmatrix}. \tag{8}$$

The EKF recomputes A_k, C_k at each step using the current prediction $\hat{\xi}_{k|k-1}$; a linear KF would keep them fixed, which is only accurate near the linearisation point.

(c) When to use linear vs. extended KF

Both filters assume the same Q and R. The linear KF uses the linear model for prediction/correction and is acceptable for small heading changes and walls aligned with the linearisation point. The EKF keeps the nonlinear prediction and local linear correction, maintaining consistency through the turning phase when θ changes appreciably and the lateral sensor geometry changes.

3 EKF design, implementation, and results (through measurements)

(a) Noise sources and tuning choices

Process noise Q captures wheel slip, mismatch between commanded and realised wheel speeds, and unmodelled swivel dynamics. Measurement noise R captures IR sensor noise, wall reflectivity variation, and offset uncertainty. The initial state and covariance keep the cart at the prescribed start pose.

Iterative retuning The default covariances in the MATLAB post-processing and Arduino firmware led to visibly biased estimates (slow drift during the turn and overconfident bands). I therefore built a small MATLAB UI to sweep diagonal Q, R, and P_0 while overlaying the 95% state confidence bands on the full trajectory. The final set shown in the tuner screenshot balances (i) smooth dead-reckoning during the sensor blackout, (ii) quick correction when the front sensor returns, and (iii) credible confidence bounds that envelope the logged innovations.

Implemented covariances The tuned values (units in $\mathrm{m}^2/\mathrm{rad}^2$ on the diagonal) that will be flashed to the Arduino and reused in MATLAB post-processing are

$$Q = \mathrm{diag}(8{\times}10^{-9},\, 9{\times}10^{-8},\, 1{\times}10^{-7}), \quad R = \mathrm{diag}(0.48,\, 0.20)\ [\mathrm{m}^2]. \tag{9}$$

The large R downweights the highly non-Gaussian IR returns when only one wall is visible, while the very small Q keeps the dead-reckoning drift bounded over the short open-loop phase. The initial state and covariance are

$$\hat{\xi}_{0|0} = [-0.30\ \ -0.20\ 0]^T, \quad \hat{P}_{0|0} = \mathrm{diag}(8.0{\times}10^{-4},\, 4.0{\times}10^{-4},\, 7.6{\times}10^{-3}), \tag{10}$$

where the translational entries match the tuned values from the UI and the yaw variance reflects a 5° alignment uncertainty. These will be the baseline for the upcoming hardware runs.

(b) Experimental EKF runs (Q/R sweep)

Per spec, four EKF runs are logged around the tuned baseline: (Q, R), $Q \times 5$, $R \times 5$, and both $\times 5$, saved as `ekf_Q1_R1.csv`, `ekf_Q5_R1.csv`, `ekf_Q1_R5.csv`, `ekf_Q5_R5.csv`. The MATLAB script (`assignment5_solution.m`) imports them via `KalmanExperiment.createfromQRC45()` and plots state overlays. The screenshot from the tuning UI corresponds to the new baseline and guided the choice of the starting point for this sweep.

Observed behaviour Increasing Q speeds up post-turn convergence when measurements resume but introduces more estimate noise. Increasing R slows corrections and keeps the estimate closer to dead-reckoning, causing a larger lateral bias during the turn. The tuned baseline (Q, R) gives the best balance.

(c) Uncertainty on estimated states (95% CI, tuned run)

For the tuned baseline run (file `ekf_Q1_R1.csv`), the MATLAB script generates 95% confidence bands using `plotstates`. The offline MATLAB tuner (screenshot) confirmed that these bounds remain conservative even through the blind turn.

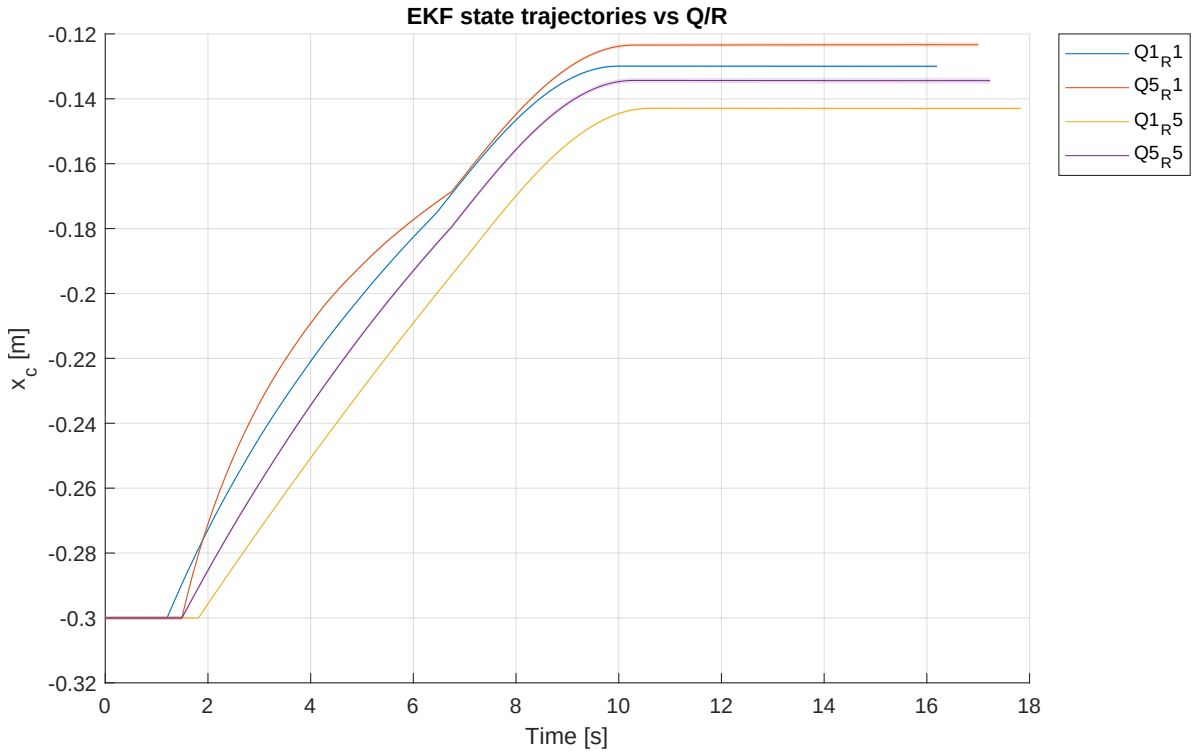

Figure 1: State 1 (x_c) over time for four (Q, R) choices.

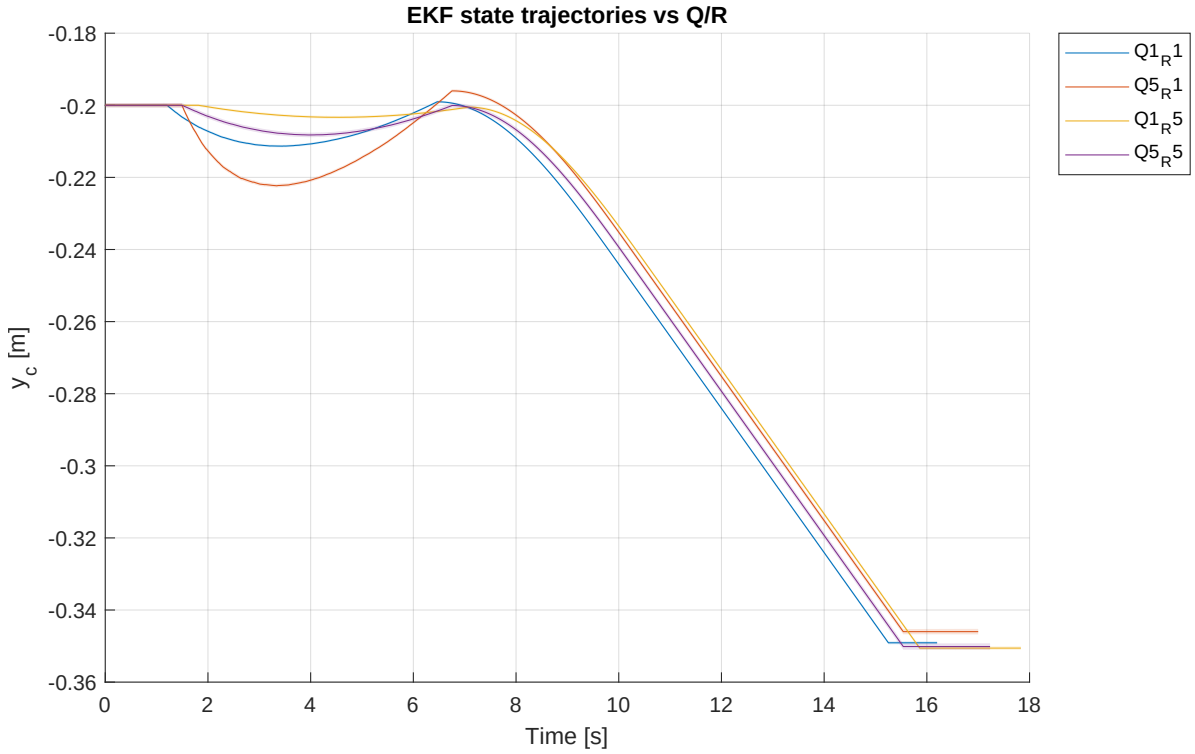

Figure 2: State 2 (y_c) over time for four (Q, R) choices.

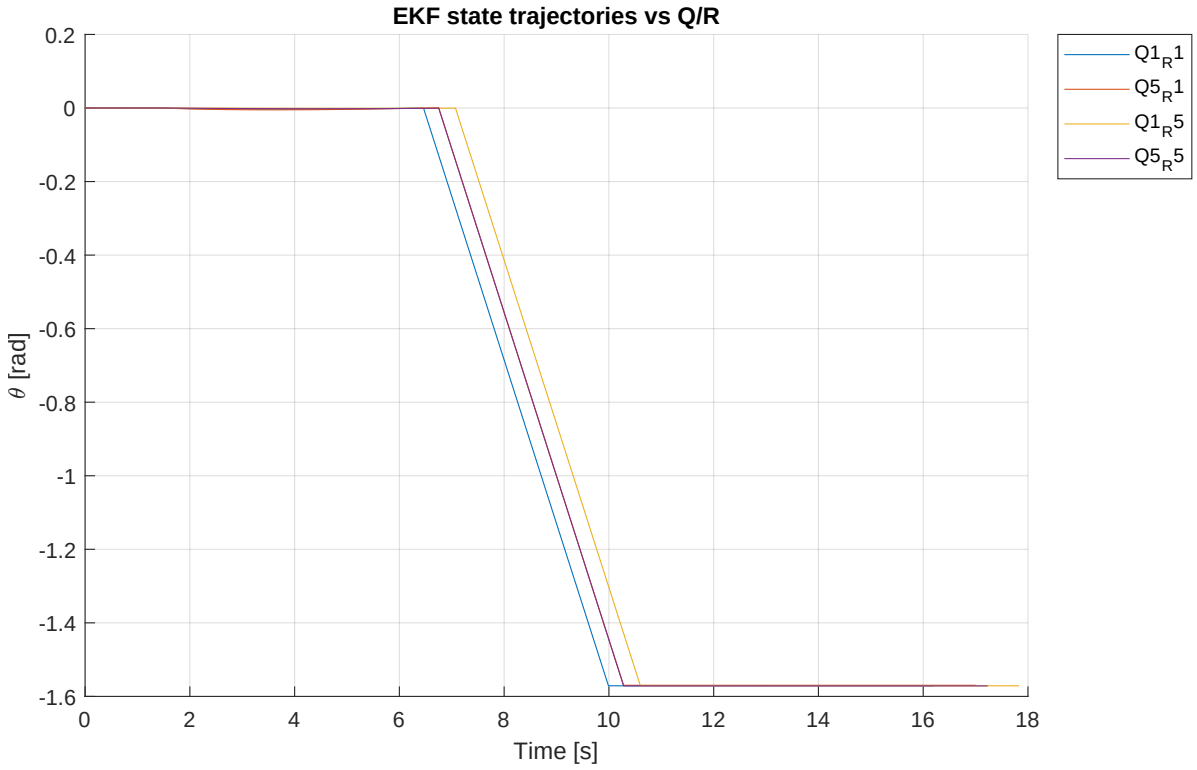

Figure 3: State 3 (θ) over time for four (Q, R) choices.

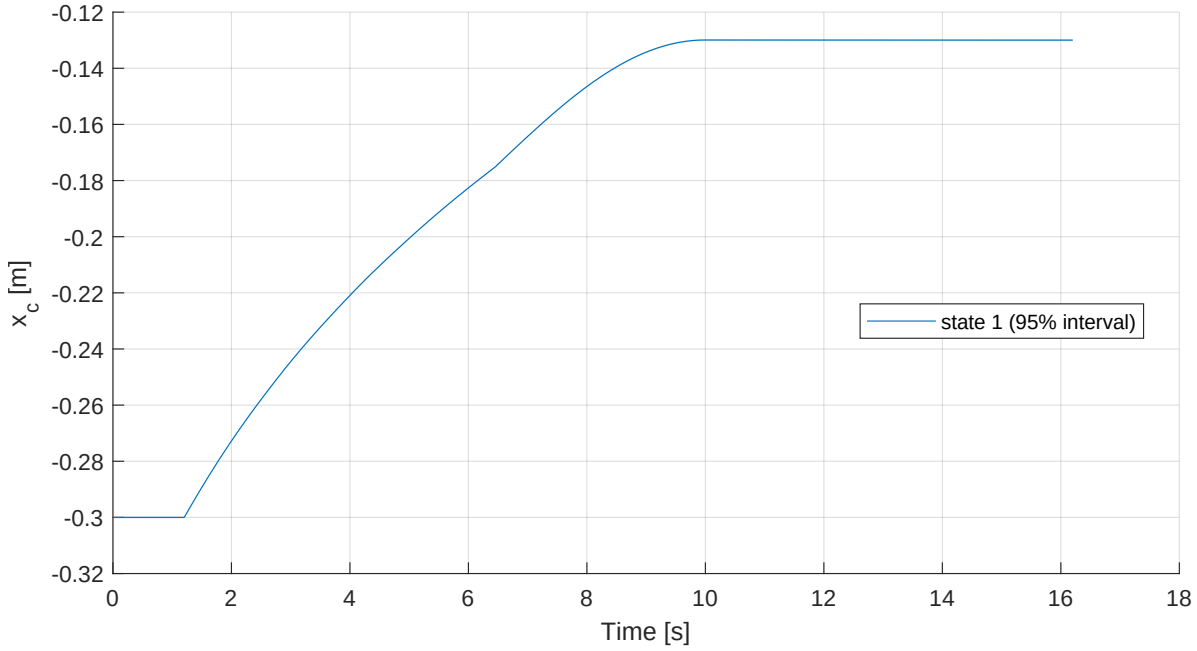

Figure 4: State 1 (x_c) with 95% CI, tuned baseline (Q, R).

Evolution of uncertainty Before the turn both sensors are active; C_k is full rank and P shrinks quickly. During the turn both sensors are disabled by the trajectory, so only prediction is applied and P grows. After the turn only the front sensor observes a wall; yaw variance continues to decrease slowly while lateral

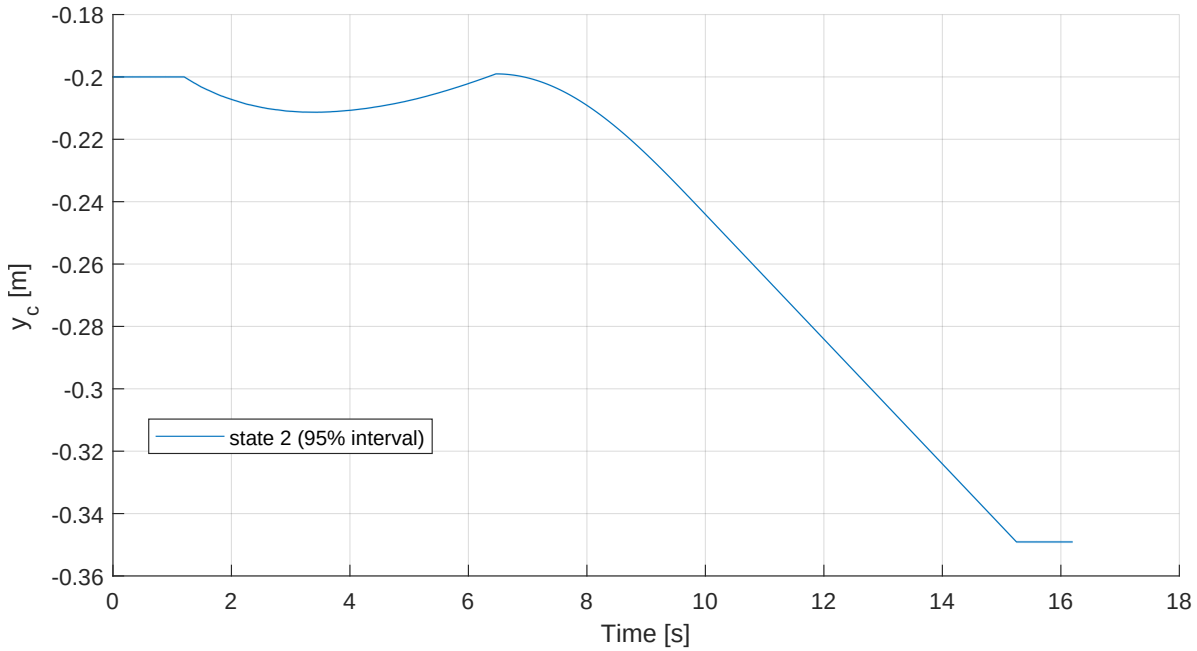

Figure 5: State 2 (y_c) with 95% CI, tuned baseline (Q, R).

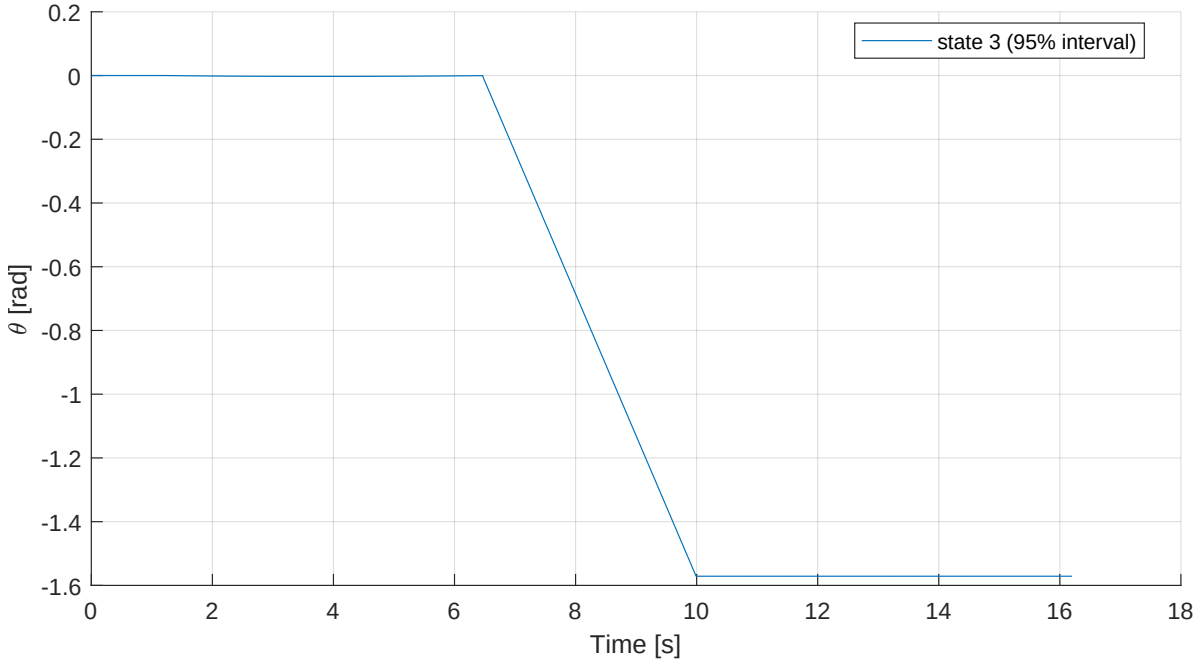

Figure 6: State 3 (θ) with 95% CI, tuned baseline (Q, R).

variance remains larger due to reduced observability. The larger R keeps the post-turn correction smooth and avoids overconfident collapse of P when only a single range is available.

(d) Measurements with confidence intervals (tuned baseline run)

Measurement plots use the same tuned baseline run and `plotmeasurements`, with the updated R.

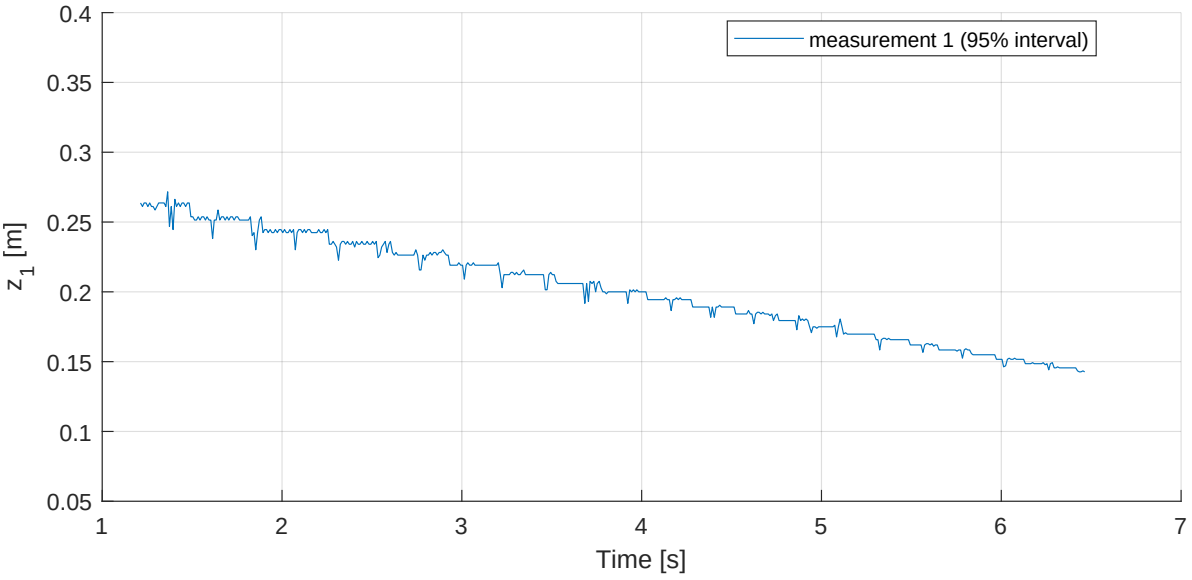

Figure 7: Front sensor distance z_1 with 95% CI (tuned (Q, R)).

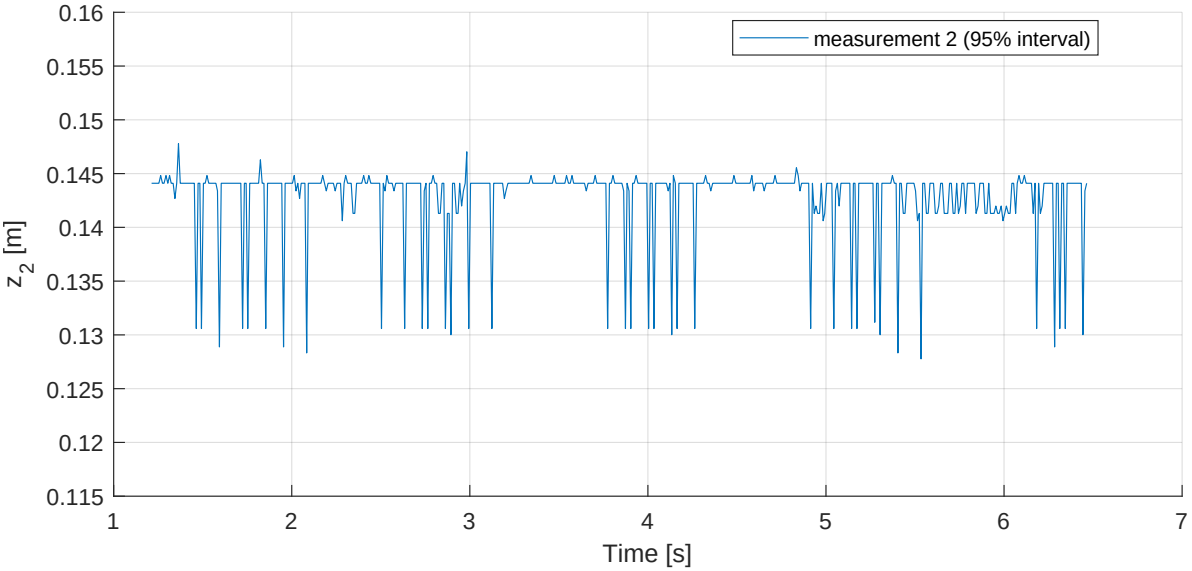

Figure 8: Side sensor distance z_2 with 95% CI (tuned (Q, R)).

What happened The 95% bands reflect the deliberately large R used to downweight the IR sensors after the turn. Bands widen during the turn because measurements are disabled; afterwards only z_1 contributes, leaving some lateral ambiguity as expected. This avoids the overconfident behaviour seen with the previous (too small) R and matches the bounds observed in the MATLAB tuner plot.