Exercises for Course on Modeling and System Identification (MSI)
Albert-Ludwigs-Universität Freiburg – Winter Term 2019-2020

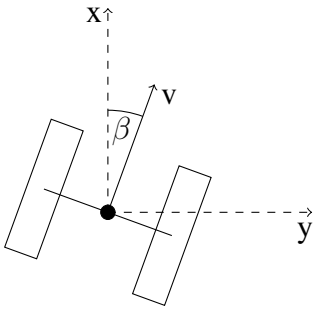
Exercise 8: Nonlinear Least Squares + Linear Kalman Filter

(to be returned on Jan 27th, 2021, 9:00)

Prof. Dr. Moritz Diehl, Katrin Baumgärtner, Naya Baslan, Jakob Harzer, Doga Can Öner

Parameter estimation for output error minimization (5 points + 2 bonus points)

In this exercise we will consider the model of a differential drive robot with unicycle dynamics. The movement of the robot depends on the angular velocities of the left and the right wheel ω_{L} and ω_{R}, as well as on their radii R_{L} and R_{R}. Differing radii influence the behaviour of the robot.



The system can be described by a state space model with three internal states. The state vector $\mathbf{x} = [x,\ y,\ \beta]^\top$ contains the position of the robot in the $X-Y$ plane and the deviation β from its initial orientation. The system can be controlled by the angular velocities of the wheels: $\mathbf{u} = [\omega_{\mathrm{L}},\ \omega_{\mathrm{R}}]^\top$. The output of the system is the position of the robot: $\mathbf{y} = [x,\ y]^\top$. The model follows as

$$\dot{\mathbf{x}} = \begin{bmatrix} v \cdot \cos\beta \\ v \cdot \sin\beta \\ \frac{\omega_{\mathrm{L}} R_{\mathrm{L}} - \omega_{\mathrm{R}} R_{\mathrm{R}}}{L} \end{bmatrix} \qquad \mathbf{y} = \begin{bmatrix} x \\ y \end{bmatrix} \tag{1}$$

with L being the length of the axis between the two wheels and the velocity v being

$$v = \frac{\omega_{\mathrm{L}} \cdot R_{\mathrm{L}} + \omega_{\mathrm{R}} \cdot R_{\mathrm{R}}}{2}.$$

The system state is $\mathbf{x} = [x, y, \beta]^\top$ and is equal to the robot's pose. The system can be controlled by the angular velocities of the wheels: $\mathbf{u} = [\omega_L\ \omega_R]^\top$. The output $\mathbf{y}$ is the position of the robot and measured with a sampling time of $\Delta t = 0.01$ s.

We already provided you some functions to simulate the position of the two-wheel-robot using the state space model. For reference use chapter 6.2 'Numerical Integration Methods' from the lecture notes. Please go through all of the provided functions and try to understand what they are doing.

- `[xdot] = robot_ode(x,u,p)` evaluates the right-hand side of the ODE $\dot{x} = f(x, u, p)$, with parameters $p = [R_L, R_R, L]$. Use the following values: R_L = 0.2 m, R_R = 0.2 m and L = 0.6 m.
- `[x_next] = RK4_step(h, x0, u, ode, p)` performs one RK4 integration step for a general ODE $\dot{x} = f(x, u, p)$ starting at x_0, with input u, parameters p and step size h.
- `[x_sim] = sim_RK4(t,x0, u, ode, p)` simulates the robot's behaviour at times $\mathbf{t}$ given a set of inputs $\mathbf{u}$, starting at $x_0 = [0\ 0\ 0]^\top$.

In this task, we would like to estimate the dimensions of the robot $\theta = [R_L, R_R, L]^\top$ using `lsqnonlin`[1]. Assuming that the robot system has only output errors, and that these errors are Gaussian with zero mean and variances $\sigma_x^2 = 1.6 \cdot 10^{-3}$ m^2 and $\sigma_y^2 = 4 \cdot 10^{-4}$ m^2, then the Maximum Likelihood Estimation problem to estimate θ is:

$$\theta^* = \arg\min_{\theta \in \mathbb{R}^3} \sum_{k=0}^{N} \|\mathbf{y_k} - M_k(\mathbf{U}, \mathbf{x_0}, \theta)\|^2_{\mathbf{\Sigma}_y^{-1}},$$

where $\mathbf{y_k} = (x, y)^\top \in \mathbb{R}^2$ with x and y being the coordinates of the robot and N is the number of measurements; $\mathbf{\Sigma}_y$ is the weighing matrix containing the variances on the x and y measurements,

$$\mathbf{\Sigma}_y = \begin{bmatrix} \sigma_x^2 & 0 \\ 0 & \sigma_y^2 \end{bmatrix};$$

$M_k(\mathbf{U}, \mathbf{x_0}, \theta)$ denotes the modeled position at timestep k for given $\mathbf{U}, \mathbf{x_0}, \theta$ where $\mathbf{U} \in \mathbb{R}^{(N-1)\times 2}$ is a matrix that contains all applied control inputs $u_1, \ldots, u_N$, each consisting of the angular velocity of the left and right wheel respectively (ω_L and ω_R); $\mathbf{x_0}$ contains the robot's initial pose $\mathbf{x_0} = [x_0, y_0, \beta_0]^\top = [0, 0, 0]^\top$ which we assume to be perfectly known.

1. ON PAPER: Formulate the output model

   $$M_k : \mathbb{R}^{(N-1)\times 2} \times \mathbb{R}^3 \times \mathbb{R}^3 \to \mathbb{R}^2,\ (\mathbf{U}, \mathbf{x_0}, \theta) \mapsto \hat{\mathbf{y}}_\mathbf{k}$$

   where you may use a function $F : \mathbb{R}^3 \to \mathbb{R}^3$ to denote the discretized system dynamics. (1 point)

2. MATLAB: Implement a function

   ```
   residual(theta, x0, U, t, y, sigma_y)
   ```

   which computes the residual vector between the given measured location $\mathbf{y_k}$ and the modeled location $M_k(\mathbf{U}, \mathbf{x_0}, \theta)$. Keep in mind to incorporate the measurement variances $\mathbf{\Sigma}_y$ correctly, i.e. weight the residual and to perform the right number of integration steps. Check the provided code for additional information on the parameters. (1 point)

3. MATLAB: Use `lsqnonlin` to estimate θ^*. (1 point)

4. MATLAB: Compute the simulated trajectory using θ^* and use the provided code to plot it versus the measurements and a 4th order polynomial fit.
   ON PAPER: What do you observe? (1 point)

5. ON PAPER: Check if the assumptions made on the noise were correct by plotting a histogram for the residual in x and y (using θ^*). (1 point)

6. MATLAB: Approximate the covariance matrix $\mathbf{\Sigma}_{\theta^*}$ of your estimate θ^* (check page 52 of the lecture notes). (2 bonus points)

[1] `lsqnonlin` takes as input a vector function $f(\theta) = [f_1(\theta), \ldots, f_N(\theta)]$, and minimizes $\|f(\theta)\|_2^2$ with respect to θ. Thus, you have to stack the residuals obtained for different timesteps to obtain a *single* residual vector.

Kalman Filter for State Estimation of Omni Wheel Robot **(5 points)**

In this task, we consider a robot with omni wheels which means it can instantaneously move in any direction. We assume that the robot's orientation does not change and thus we model the robot's state $x \in \mathbb{R}^4$ by

$$x = \begin{bmatrix} p \\ v \end{bmatrix}$$

where $p = (p_1, p_2)^\top \in \mathbb{R}^2$ denotes the position of the robot in m, $v = (v_1, v_2)^\top \in \mathbb{R}^2$ denotes its velocity in $\frac{\mathrm{m}}{\mathrm{s}}$. The time derivatives of p and v are given by $\dot{p} = v$ and $\dot{v} = u - \mu v$ with control inputs $u = (u_1, u_2)^\top \in \mathbb{R}^2$ and constant parameter $\mu = 0.001 \frac{1}{\mathrm{s}}$. We assume that the control inputs u are perfectly known.

1. ON PAPER: Specify the continuous time state-space model, i.e. define matrices $A_c \in \mathbb{R}^{4\times4}$ and $B_c \in \mathbb{R}^{4\times2}$ such that the following holds:

   $$\dot{x} = f(x, u) = A_c x + B_c u$$

   Formulate the corresponding discrete time model for the robot's dynamics using a one-step Euler integrator with step length $h = 0.5$ s, i.e. specify matrices $A_d \in \mathbb{R}^{4\times4}$ and $B_d \in \mathbb{R}^{4\times2}$ such that

   $$x_{k+1} = F(x_k, u_k) = A_d x_k + B_d u_k \qquad \text{(1 point)}$$

2. ON PAPER: We assume that the discrete time state dynamics are perturbed by additive zero-mean Gaussian noise. Note that we cannot observe the state directly, but can only measure the robot's position p using a GPS sensor. These GPS measurements are perturbed by additive zero-mean Gaussian noise with covariance matrix Σ_{γ_p}.

   Summarizing, our state and measurement model has the form

   $$x_{k+1} = A_d x_k + B_d u_k + \chi_k, \tag{2}$$
   $$y_k = C x_k + \gamma_k, \tag{3}$$

   where $\chi_k \sim \mathcal{N}(0, \Sigma_\chi)$ and $\gamma_k \sim \mathcal{N}(0, \Sigma_\gamma)$. We assume

   $$\Sigma_{\chi_p} = 2 \cdot 10^{-2} \cdot \mathbb{I}\,\mathrm{m}^2, \qquad \Sigma_{\chi_v} = 4 \cdot 10^{-3} \cdot \mathbb{I}\,\frac{\mathrm{m}^2}{\mathrm{s}^2}, \qquad \Sigma_{\gamma_p} = 16 \cdot \mathbb{I}\,\mathrm{m}^2.$$

   Specify the matrix $C \in \mathbb{R}^{2\times4}$ such that $y_k \in \mathbb{R}^2$ corresponds to the noisy GPS measurement. Also write down the covariance matrices Σ_χ and Σ_γ. (1 point)

3. MATLAB: Write two functions

   ```
   [x_predict, P_predict]   = predict(x_estimate, P_estimate, A, b, W)
   [x_estimate, P_estimate] = update(y, x_predict, P_predict, C, V)
   ```

   that implement the prediction and update step of the Kalman filter.

   Hint: See equations (9.21) to (9.24) on page 103 of the lecture notes. Here `x_predict` corresponds to $x_{[k|k-1]}$ and `x_estimate` corresponds to $x_{[k|k]}$. (1 point)

4. MATLAB: For the given measurement and control trajectories, $y = (y_0, \ldots, y_N)$ and $u = (u_0, \ldots, u_{N-1})$, compute the state estimates $x_{[k|k]}$ and state predictions $x_{[k|k-1]}$ where we assume an estimated initial state $x_0 \sim \mathcal{N}(0, \Sigma_0)$ where $\Sigma_0 = 10^{-5} \cdot \mathbb{I}$, i.e. we assume to know the initial state almost exactly.

   (1 point)

5. MATLAB: We already provided code to plot the estimated trajectory, the predicted position $p_{[k|k-1]}$ and the corresponding confidence ellipsoids.
   You only have to compute $\Sigma_{p_{[k|k-1]}}$ from $P_{[k|k-1]}$. (1 points)

This sheet gives in total 10 points plus 2 bonus points.