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

Exercise 8: Nonlinear Least Squares

(to be returned by Jan 15th, 2020, 8:30 in HS 00 036 (Schick-Saal), or in building 102, 1st floor, 'Anbau' until 10:00)

Prof. Dr. Moritz Diehl, Tobias Schöls, Naya Baslan, Jakob Harzer, Bryan Ramos

In this exercise we will consider again the model of a robot moving on a plane as presented in exercise sheet 7. It's kinematic model is given by the state-space equations

$$\dot{\mathbf{x}} = \begin{bmatrix} v \cdot \cos\beta \\ v \cdot \sin\beta \\ \frac{\omega_L R_L - \omega_R R_R}{L} \end{bmatrix}, \qquad \mathbf{y} = \begin{bmatrix} x \\ y \end{bmatrix} \tag{1}$$

where the robot's velocity v is given by $v = \frac{\omega_\mathrm{L} \cdot R_\mathrm{L} + \omega_\mathrm{R} \cdot R_\mathrm{R}}{2}$. 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.

Exercise Tasks

1. **Forward simulation of the robot's state** **(5 points)**

   In this task you will 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.

   (a) MATLAB: Given the state-space model, implement a function

   ```
   [xdot] = robot_ode(x,u,p)
   ```

   which 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. (1 point)

   (b) MATLAB: Implement a function

   ```
   [x_next] = euler_step(h, x0, u, ode, p)
   ```

   which performs one Euler 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. (1 point)

   (c) MATLAB: Implement a function

   ```
   [x_next] = rk4_step(h, x0, u, ode, p)
   ```

   which performs one Runge-Kutta (of order 4) 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. (1 point)

   (d) MATLAB: Write a function

   ```
   [x_sim] = sim(t,x0,u, integrator, ode, p)
   ```

   which simulates the robot's behaviour at times $\mathbf{t}$ given a set of inputs $\mathbf{u}$, starting at $x_0 = [0\ 0\ 0]^\top$ where you use the given integrator. (1 point)

   (e) MATLAB: Plot the results you obtain from the `sim` function when calling it with `euler_step` and `RK4_step`.

   ON PAPER: Are there any difference? Why (not)? (1 point)

2. **Parameter estimation for output error minimization** **(8 points)**

After observing the movement of a different sized robot, you would like to estimate the dimensions of this 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 defined as:

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

$M_k(\mathbf{U}, \mathbf{x_0}, \theta)$ 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.

(a) ON PAPER: First formulate a discrete time model for the robot's dynamics $F : \mathbb{R}^3 \to \mathbb{R}^3$ using a one-step Euler integrator and the kinematic model given in (1). Then formulate the output model

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

Hint: You may use F for the formulation of M_k. (1 point)

(b) 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 paramters. (2 points)

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

(d) 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)

(e) 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)

(f) MATLAB: Approximate the covariance matrix $\mathbf{\Sigma}_{\theta^*}$ of your estimate θ^* (check page 48 of the lecture notes). (2 points)

This sheet gives in total 13 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.