SE 380 Introduction to Feedback Control

Gennaro Notomista

PROJECT DESCRIPTION

1   Introduction

Mobile robotic platforms are becoming increasingly important in a variety of applications, including warehouse automation, medical assistance, cleaning services, agricultural robotics, and environmental monitoring, to cite a few.

1.1   Mobile robot control

Prototypical mobile robot tasks consist in visiting a sequence of locations in space. Once a path has been planned which passes through these locations, a controller has to be designed to make the robot follow this path. In this project we consider the problem of designing a controller to drive a ground mobile robot to a desired location.
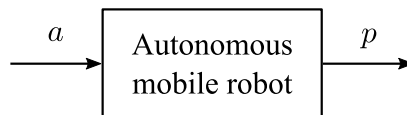

Figure 1: Block diagram of the considered ground mobile robot. The control input of the robot is its acceleration vector, $a \in \mathbb{R}^2$, and we have access to a measurement of its position $p \in \mathbb{R}^2$.

The block-diagram model of the considered mobile robotic system is given in Fig. 1. The control input to the system is the robot acceleration vector $u = [u_1, u_2]^T = [a_x, a_y]^T = a \in \mathbb{R}^2$, where a_x and a_y are the accelerations of the robot along the directions of the axes of the reference frame defined on the plane where the robot moves. The output of the system is the robot position vector $y = [y_1, y_2]^T = [p_x, p_y]^T = p \in \mathbb{R}^2$, where p_x and p_y are the components of the position of the robot along the axes of the reference frame. *Driving the robot to a desired fixed location corresponds to regulating the output of the system to a given constant reference value.*

The mathematical model of the system is not known but it is possible to probe the system by supplying an acceleration input signal and measuring the corresponding position output signal. This way, one can build a mathematical model of the system to be used to design a controller that fulfills the desired specifications.

1.2   Code

The code provided in the file `system_model.py` contains the Python class `SystemModel` that implements the dynamics of the mobile robot. This class allows us to faithfully simulate the motion of the robot and obtain the output response to a given input signal. The latter can be done by calling the class method `sim` as follows:

$$\texttt{output\_signal = sm.sim(input\_signal)},$$

where `sm` is an object of the class `SystemModel`, and `input_signal` and `output_signal` are the input and output signals, respectively, represented as `numpy` arrays whose shape is `(L, 2)`. The variable `L` is the length of the signal measured in number of time steps, where one time step lasts $\Delta t = 0.001$s. This way, if a signal lasts for 1s, the length of the `numpy` array representing the signal will be 1000.

The class `SystemModel` has one another method, `step`, which can be used as follows to obtain the measured output vector $\bar{y}$ at time $t + \Delta t$ in response to an input signal $u(\tau) = \bar{u}, \forall \tau \in [t, t + \Delta t]$:

$$y\_bar = sm.step(u\_bar),$$

where `u_bar` and `y_bar` are the input and measured output vectors, represented as `numpy` arrays whose shape is `(2,)`.

2 Project tasks

Task 1 consists of collecting and processing input-output data of the robot to control, which will be used in Task 2 to build a mathematical model of the robot, which, in turn, will be used in Task 3 to design a controller for the robot.

2.1 Task 1

Complete the items described below:

- **Item 1 (1 point)** Define constant input signals u_1, u_2 of amplitude 1 over a time horizon of 15s. Generate the corresponding output signals y_1, y_2.

- **Item 2 (1 point)** Plot the signals u_1, u_2, y_1, y_2.

- **Item 3 (2 points)** Plot the frequency spectrum of the signals y_1, y_2.

- **Item 4 (2 points)** Choose an appropriate cut-off frequency ω_c to filter out noise—by removing frequencies larger than ω_c—from the signals y_1, y_2.

- **Item 5 (3 points)** Design a third order low-pass filter with real and coincident poles and cut-off frequency ω_c.

- **Item 6 (1 point)** Filter the signals y_1, y_2 using the designed low-pass filter. Plot the filtered signals.

2.2 Task 2

In this task we will build a mathematical model to describe the relation between the inputs, u_1 and u_2, and the outputs, y_1 and y_2, of the system simulated in the `SystemModel` class. The following design choices are made:

- We choose to model the system using transfer functions. Thus, we have:

$$\begin{bmatrix} Y_1(s) \\ Y_2(s) \end{bmatrix} = \begin{bmatrix} G_{11}(s) & G_{12}(s) \\ G_{21}(s) & G_{22}(s) \end{bmatrix} \begin{bmatrix} U_1(s) \\ U_2(s) \end{bmatrix}. \tag{1}$$

  Note: $Y_i(s) = G_{ij}(s)U_j(s)$, when $U_k(s) = 0 \ \forall k \neq j$.

- Each $G_{ij}(s)$ is chosen to be a second-order transfer function, i.e.

$$G_{ij}(s) = \frac{b_{1,ij}s + b_{0,ij}}{s^2 + a_{1,ij}s + a_{0,ij}}. \tag{2}$$

Complete the items described below:

- **Item 1 (1 point)** Compute the output signals y_{11}, y_{12}, y_{21}, y_{22}, defined as follows: the output y_{ij} is the i-th component of the system output obtained by setting $u_j(t) = H(t)$ (unit step) and $u_k(t) \equiv 0 \ \forall k \neq j$.

- **Item 2 (1 point)** Design a low-pass filter to attenuate noise from the signals obtained in **Item 1**.

- **Item 3 (2 point)** Compute the other signals required by the algorithm for continuous-time model identification from sampled data in order to identify the transfer functions $G_{11}(s)$, $G_{12}(s)$, $G_{21}(s)$, $G_{22}(s)$. If necessary, re-tune the low-pass filter in **Item 2** and re-compute the required signals.

- **Item 4 (1 point)** Plot the signals obtained in **Items 1, 2, 3**.

- **Item 5 (3 points)** Identify the transfer functions $G_{11}(s)$, $G_{12}(s)$, $G_{21}(s)$, $G_{22}(s)$ using the algorithm for continuous-time model identification from sampled data.

- **Item 6 (1 point)** Compare the behaviors of the true and identified systems by plotting their respective responses to step input signals. Use a separate graph for each transfer function.

- **Item 7 (1 point)** Compare the behaviors of the true and identified systems by plotting their respective responses to the input signals

$$u_1(t) = u_2(t) = \sin t, \ t \in [0, 15]\text{s}.$$

Use a separate graph for each transfer function.

2.3 Task 3

Assume the identified systems in **Task 2** are as follows:

$$G_{11}(s) = G_{22}(s) = \frac{0.53}{s^2 + 1.05s + 0.005}$$
$$G_{12}(s) = G_{21}(s) = 0. \tag{3}$$

Complete the items described below:

- **Item 1 (2 points)** Choose $x_1 = [y_1, \dot{y}_1]^T \in \mathbb{R}^2$ and find a state space representation of $G_{11}(s)$:

$$\begin{cases} \dot{x}_1 = A_1 x_1 + B_1 u_1 \\ y_1 = C_1 x_1 + D_1 u_1. \end{cases} \tag{4}$$

Analogously, choose $x_2 = [y_2, \dot{y}_2]^T \in \mathbb{R}^2$ and obtain a state space representation of $G_{22}(s)$:

$$\begin{cases} \dot{x}_2 = A_2 x_2 + B_2 u_2 \\ y_2 = C_2 x_2 + D_2 u_2. \end{cases} \tag{5}$$

- **Item 2 (3 points)** Using the state space models found in **Item 1**, compute state feedback controllers $u_1 = -K_1 x_1$ and $u_2 = -K_2 x_2$ such that the closed-loop systems satisfy the following specifications:

  (i) Overshoot is less than 2% [1]

  (ii) The 2% settling time is less than 4s [2]

- **Item 3 (2 points)** Implement the state feedback robot controller in the method `calculate_acceleration` of the `Controller` class in the provided `controller.py` file. *Hint: $\dot{y}_1$ and $\dot{y}_2$ required to evaluate the controller may be approximated using finite differences.*

---

[1] $O.S.\% = 100e^{-\zeta\pi/\sqrt{1-\zeta^2}}$.

[2] $T_s^{2\%} \approx \frac{4}{\zeta\omega_n}$.

- **Item 4 (1 points)** Implement the state feedback robot controller this time by filtering out noise from $\dot{y}_1$ and $\dot{y}_2$ before using them to evaluate u_1 and u_2. Use the following low-pass filter:

$$LPF(s) = \frac{V_f(s)}{V(s)} = \frac{10^3}{(s+10)^3},\tag{6}$$

  where $V(s)$ and $V_f(s)$ represent the unfiltered and filtered $\dot{y}_i$ signals, respectively. *Hint: In order to implement this filter you may proceed as follows.*

  (i) *Obtain a state space representation $(A_f \in \mathbb{R}^{3\times3}, B_f \in \mathbb{R}^{3\times1}, C_f \in \mathbb{R}^{1\times3}, D_f \in \mathbb{R})$ of the filter*

  (ii) *Compute the filtered $\dot{y}_i$ using forward Euler integration of the state space dynamical system representing the filter*

$$x_{f,i}(t_k) = x_{f,i}(t_{k-1}) + (A_f x_{f,i}(t_{k-1}) + B_f \dot{y}_i(t_k))\Delta t$$
$$\dot{y}_{i,\text{filtered}}(t_k) = C_f x_{f,i}(t_k) + D_f \dot{y}_i(t_k),\tag{7}$$

  where $x_{f,i}$ is the state of the i-th filter, $i \in \{1,2\}$, employed to filter out noise from $\dot{y}_1$ and $\dot{y}_2$, respectively.

- **Item 5 (2 points)** Simulate the controlled autonomous mobile robot for a time horizon of 30 seconds and plot the signals y_1, y_2, u_1, u_2 over time. The provided `main.py` file can serve as a starting point.

1 extra point will be given if the LaTeX or Microsoft Word IEEE conference proceedings template available here is used to write the report.

3 Deliverables and grading

The deliverables for each task consist of:

- A report (`.pdf` file, maximum 2 pages, minimum margins 0.75in, minimum font size 11pt) reporting the required plots and explaining the approach
- A Python script (`.py` file) containing the code to solve the task

Important notes

- If the format of a project task submission is not as above, the project task will not be graded.
- The report should contain the explanation of the steps followed to solve each item of a project task, as well as the reference to the part of the code that solves each item, otherwise the corresponding item will not be graded.

4 Logistic details

- The project work is carried out in groups of 2 students. Groups of one are permitted only if an odd number of students are enrolled. Project groups are formed on LEARN.
- Project submissions, one per group, will be handled via LEARN.
- Multiple submissions will be accepted via LEARN before the deadline, but only the most recent one will be available for grading.