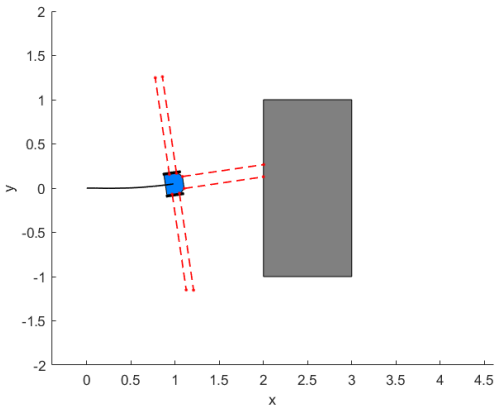
Coursework 1

Coursework Instructions

- A Matlab robot simulator is provided with this coursework and is described below
- Submit a written report (pdf) with a maximum of 5 pages AND a maximum of 2500 words. For each question, describe your solution and reasoning clearly. You can use labelled Matlab plots and mathematical derivations to support your answers.
- Organise the Matlab code for each question in separate folders (Q1, Q2, Q3, Q4, Q5, Q6), and submit everything as a .zip file. Provide clear instructions on which files to run your code, points will be deducted if instructions are not clear or if the code needs to be fixed after submission.

Robot Simulator

A differential drive robot simulator (very similar to Tutorial 2) is provided with this coursework. Run the code with the following Matlab script: `Sim_DifferentialDriveWithObstacles.m`



The Simulation has an obstacle that the robot should avoid. The robot is equipped the following noisy sensors:

- **2 wheel encoders:** 64 ticks for a complete wheel revolution.
- **6 range sensors:** measure distance to obstacle within the range [0.1, 1.1] m. Their positions and measurement directions are displayed by red dotted lines in the simulation plot.

Inside the simulation loop in `Sim_DifferentialDriveWithObstacles.m`, the measurements from these sensors are updated in the variable `y` (first 2 elements for odometry, and last 6 elements for range sensors).

Similarly to tutorial 2:

- The DC motors operate between -6V and +6V
- The robot's initial position is the origin (0,0) and its initial orientation is 0 rad.
- The radius of each robot wheel is unknown (they are both the same)
- The distance between the two wheels is unknown
- There are also some unknown internal parameters that can change with each simulation run.

Part 1 [30 marks total]

Q1 [5 Marks] By moving the robot and reading its measurements, determine the wheel radius and the distance between its wheels. Assume you only have access to odometry and range sensor measurements (i. e. the variable `y`) and the wheel control inputs (variable `u`).

Q2 [15 Marks] Implement a controller (`init.m`, `controller.m`) so that the robot:

1. Moves around the obstacle counterclockwise at a constant distance of 0.5m from its edges for 2 complete laps.
2. Goes back to the original position (0,0) and orientation (0 degrees)

For controlling the robot, assume that you only have access to odometry and range sensor measurements (i. e. the variable `y`) and the wheel control inputs (variable `u`).

For full marks, implement the controller as a single PID (while updating the reference appropriately). The controller should also work if the obstacle was in a slightly different position and orientation (but assume the robot will always find the obstacle if moving straight from its initial position).

Q3 [5 Marks] Implement a controller that performs the same task as Q2, but only using odometry measurements. Assume that the obstacle is always in the same position with corners (2,-1), (3,-1), (3,1), (2,1). Compare the performance of this controller against the one developed in Q2 and comment on the differences.

For full marks, implement the controller as a single PID controller (while updating the reference appropriately).

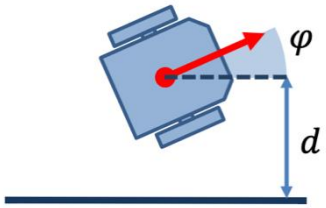
Q4 [5 Marks] Assume that $V_M(t)$ is the average voltage input to the robot and $s_M(t)$ is the speed of the robot at the midpoint between the wheels. Consider that the transfer function modelling their relationship is approximately

$$G(s) = \frac{S_M(s)}{V_M(s)} = \frac{K_s}{\tau_s s + 1}$$

where K_s and τ_s are unknown constants. Estimate the value of K_s using robot measurements (odometry and/or range sensors) and the limit value theorem.

Part 2 [20 marks total]

Consider the simplified scenario of a robot following a wall (similar to Tutorial 3). We will control a differential drive robot to maintain a constant speed of $s_M = 0.2$ m/s while keeping a reference distance $r_d(t)$ to a horizontal wall. The robot is controlled by voltage inputs V_R, V_L to its left and right wheels.



To make this a single-input single-output (SISO) system we're going to keep the average voltage constant

$$V_M = \frac{V_R + V_L}{2}$$

And control just the difference in voltage

$$V_\Delta(t) = V_R - V_L$$

The dynamics of this system can be approximated as

$$\dot{d}(t) \approx s_M \varphi(t)$$

However, in this exercise we do not have direct access to $\varphi(t)$, only to $V_\Delta(t)$, so the system dynamics are more complex.

Assume that the relationship between $V_\Delta(t)$ (as input) and $\dot{\varphi}(t)$ (as output) is given by the following transfer function:

$$G_m(s) = \frac{\dot{\phi}(s)}{V_\Delta(s)} = \frac{0.25}{0.1s + 1}$$

Q5 [10 Marks] Consider that we are controlling the robot distance $d(t)$ with a PD controller with gains $K_d = 10K_p$. Using Matlab's control toolbox, tune the value of K_p such that it maximises the phase margin of the PD controller.

Create a simulation in Simulink of the closed-loop system, when the reference $r_d(t)$ is the following signal:

$$r_d(t) = \begin{cases} 0, & 0 < t < 2.5 \\ t, & 2.5 < t < 5 \\ 2t, & 5 < t < 7.5 \\ 0, & 7.5 < t < 10 \end{cases}$$

Generate a plot that displays all the relevant system variables between 0 and 10 seconds.

Q6 [10 marks] Discretise the transfer function of the PD controller in Q5 with a sampling rate of 0.02s and convert it to a digital filter.
Now, suppose we wanted to deploy this controller in the robot simulator of Part 1 to keep a constant distance to one of the obstacle walls, using feedback from the range sensors:

- How could we compute the distance $d(t)$ from the range sensor measurements?
- The behaviour of this controller in the robot simulator of Part 1 will be very different when compared to Simulink. What are the main reasons for this?