Mobile Robots lab

Robot Control

1 Content of this lab and expected work

The goal of this lab is to control a 2D mobile robot in order to follow a trajectory. Two mobile robots will be considered:

- A 2-0 or unicycle robot, with two actuated, fixed wheels.
- A 1-1 or bicycle-like robot, with two passive fixed wheels and an actuated, steering wheel.

Two control laws will be tested for each robot: static feedback and Lyapunov-based control.

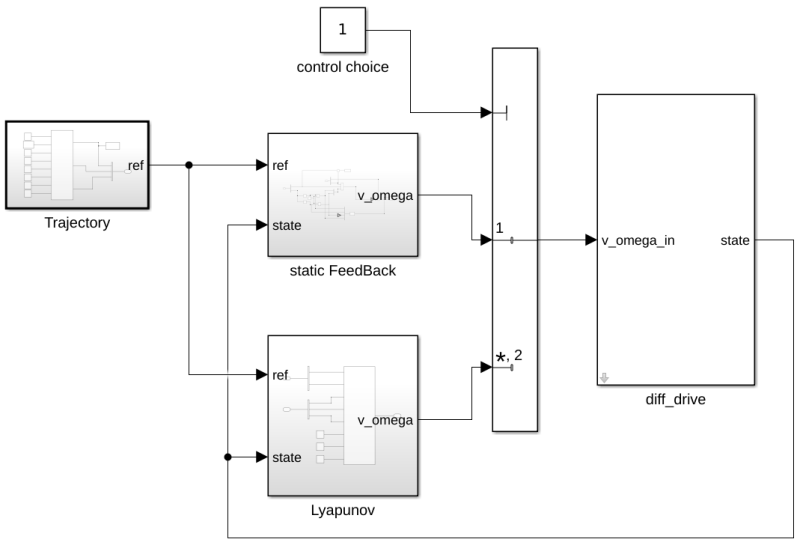
In addition, robustness of the control will be analyzed with regards to several practical issues:

- Bad calibration, ie wrong dimensions of the robot model
- Non linearities, which are typically found with saturation on wheel velocities or angles.

The control laws will be tested in simulation with Matlab/Simulink.
Trajectory generation is partially given and we assume a localization method is here and provides the current posture (x, y, θ) of the robot.

2 Available tools

Two folders are available in the Matlab workspace, called `2-0` and `1-1`. Each folder contains a Matlab script file and a Simulink model. The figure below shows the 2-0 model:



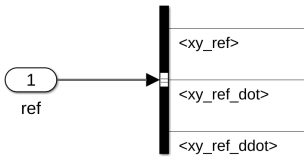
Five main blocks appear:

- `Trajectory`: Generates the trajectory. The `ref` output is a structure containing the desired position, velocity and acceleration.
- `diff_drive`: The model of the robot, takes the control inputs and updates the state of the robot.
- `static FeedBack` / `Lyapunov`: Control blocks, take the reference and state and output the command.
- `Control law switch`: Allows to choose control depending on the value (1 or 2).

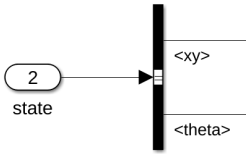
The given script will generate a trajectory from waypoints, then run the simulation and plot the behavior of the robot. The parts to be modified are:

- The content of the two control blocks
- The parameters of the robot model, including saturation and uncertainty
- If needed, the waypoint coordinates used to generate the trajectory, in order to test simpler or more complex trajectories.

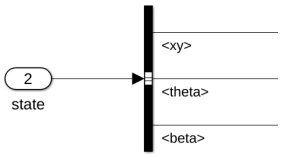
Signals going through the reference and state should be decomposed by using a `Bus Selector` to retrieve the actual values:



Reference to position, velocity and acceleration

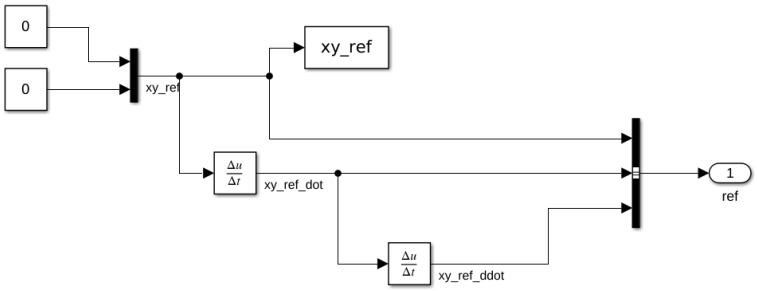


2-0 robot state to (x, y, θ)



1-1 robot state to (x, y, θ, β)

Trajectory generation Initially the trajectory is always $(0, 0)$ as shown in the next figure.



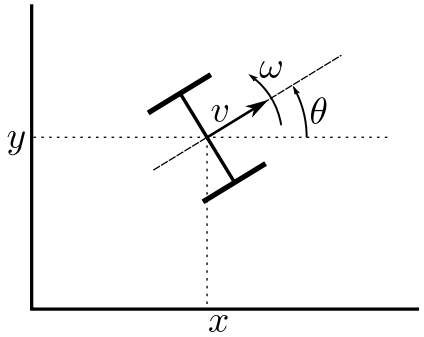
Two parameters a and w are defined in the initial script. They should be used to generate a circular trajectory:

$$\begin{cases} x(t) &= a\cos wt \\ y(t) &= a\sin wt \end{cases}$$

This trajectory will be followed during $t_{\max} = 2\pi/w$ which correspond to doing 3 turns.

3 Differential drive 2-0 robot

The figure below recalls notations and kinematic model of a classical 2-0 mobile robot.



State	(x, y, θ)
Parameters	(base, radius)
Control	$\mathbf{u} = (v, \omega)$
Kinematic model	$\begin{cases} \dot{x} = v\cos\theta \\ \dot{y} = v\sin\theta \\ \dot{\theta} = \omega \end{cases}$

In practice the control input is mapped to wheel velocities through the base (wheel distance) and wheel radius parameters. The corresponding values can be changed by double-clicking the `diff_drive` subsystem as shown in the figure below. Uncertainty on these parameters is initially set to 0.

Non-linearity appears in the 2-0 model when we consider limitation on the wheel velocities. This parameter can also be tuned in the subsystem, it is initially set to a high value in order to reduce its impact.

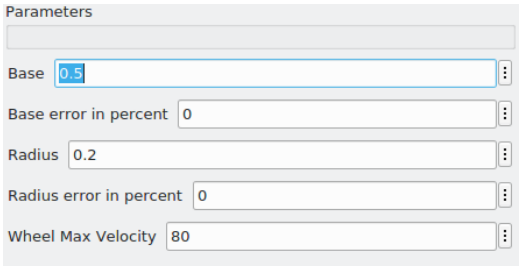


Figure 1: Parameters for the differential drive robot

3.1 Static feedback

For static feedback, we want to control the position $\mathbf{x}_p = (x_p, y_p)$ of a point located at $d > 0$ on the x-axis of the robot. It can be written as:

$$\begin{cases} x_p &= x + d.\cos\theta \\ y_p &= y + d.\sin\theta \end{cases}$$

The first step is to link $\dot{\mathbf{x}}_p$ to the control input: $\dot{\mathbf{x}}_p = \mathbf{K}(\theta).\mathbf{u}$.

Then, a simple Proportional control is used to define the desired velocity of the point:

$$\dot{\mathbf{x}}_p^* = \dot{\mathbf{x}}_r + K_p(\mathbf{x}_r - \mathbf{x}_p)$$

The final control law is thus:

$$\mathbf{u} = \mathbf{K}^{-1}(\theta).\left(\dot{\mathbf{x}}_r + K_p(\mathbf{x}_r - \mathbf{x}_p)\right)$$

Implement this control and test it for various values of d and K_p. Once acceptable values have been found:

- What happens when there are calibration errors?
- What happens when wheel rotation velocity is saturated?

3.2 Lyapunov-based control law

In this approach, the goal is to minimize an energy function that should be equal to 0 when and only when the robot is at the target position and orientation. For 2-0 robots, a classical one is:

$$\mathbf{W} = \frac{1}{2}\left(x_e^2 + y_e^2 + \frac{\theta_e^2}{K_y}\right)$$

where (x_e, y_e, θ_e) is the error in position and heading, corresponding to the target point expressed in the robot frame. They can be obtained easily from the reference values:

$$\begin{cases} x_e &= \cos\theta.(x_r - x) + \sin\theta.(y_r - y) \\ y_e &= -\sin\theta.(x_r - x) + \cos\theta.(y_r - y) \\ \theta_e &= \text{atan2}(\dot{y}_r, \dot{x}_r) - \theta \quad \text{(put back in } [-\pi, \pi]) \end{cases}$$

A candidate control law is thus:

$$\begin{cases} v &= v_r \cos\theta_e + K_x x_e \\ \omega &= \omega_r + K_y y_e v_r \dfrac{\sin\theta_e}{\theta_e} + K_\theta \theta_e \end{cases}$$

Where v_r and ω_r are feedforward control inputs, that can be obtained by:

$$\begin{cases} v_r &= \dot{x}_r \cos\theta + \dot{y}_r \sin\theta \\ \omega_r &= \dfrac{\dot{x}_r \ddot{y}_r - \ddot{x}_r \dot{y}_r}{v_r^2} \quad \text{(or 0 if } v_r = 0) \end{cases}$$

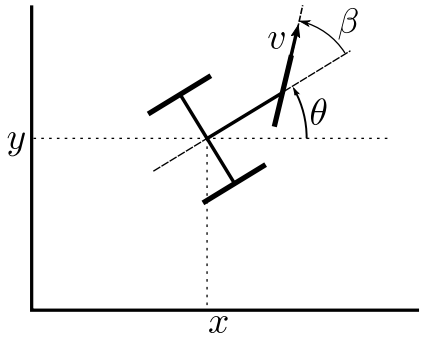
Show that this control makes $\mathbf{W}$ a Lyapunov function, meaning that its derivative is always negative except when the error is null.

Implement this control and test it for various values of (K_x, K_y, K_θ). Once acceptable values have been found:

- What happens when there are calibration errors?
- What happens when wheel rotation velocity is saturated?

4 Bicycle-like 1-1 robot

The figure below recalls notations and kinematic model of a 1-1 mobile robot. Note that is also models classical cars as the two steering wheels angles are linked to a single, virtual centered wheel angle.



State	(x, y, θ, β)
Parameters	(L, radius)
Control	$\mathbf{u} = (v, \dot{\beta})$
Kinematic model	$\begin{cases} \dot{x} = v\cos\theta\cos\beta \\ \dot{y} = v\sin\theta\cos\beta \\ \dot{\theta} = v\sin\beta/L \\ \dot{\beta} = u_2 \end{cases}$

In practice the control input v is mapped to wheel velocity through the wheel radius. The corresponding values can be changed by double-clicking the `bicycle` subsystem as shown in the figure below. Uncertainty on these parameters is initially set to 0. The L parameter is the distance between the origin of the robot (middle of rear axle) and the steering wheel.

Non-linearities appear in the 1-1 model when we consider wheel rotation velocity limit (limits v) and the wheel orientation limit in terms of β and $\dot{\beta}$. These parameter can also be tuned in the subsystem, it is initially set to a high value in order to reduce its impact.

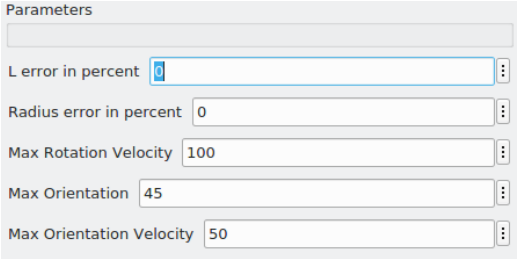


Figure 2: Parameters for the bicycle-like robot

4.1 Static feedback

For static feedback, we want to control the position $\mathbf{x}_p = (x_p, y_p)$ of a point located at $d > 0$ in the direction of the steering wheel. It can be written as:

$$\begin{cases} x_p &= x + L.\cos\theta + d.\cos(\theta+\beta) \\ y_p &= y + L.\sin\theta + d.\sin(\theta+\beta) \end{cases}$$

Follow the same approach as before to get the matrix $\mathbf{K}(\theta, \beta)$ such that: $\dot{\mathbf{x}}_p = \mathbf{K}(\theta, \beta).\mathbf{u}$.

The same Proportional control can then be used to track the trajectory:

$$\mathbf{u} = \mathbf{K}^{-1}(\theta, \beta).(\dot{\mathbf{x}}_r + K_p(\mathbf{x}_r - \mathbf{x}_p))$$

Implement this control and test it for various values of d and K_p. Once acceptable values have been found:

- What happens when there are calibration errors?
- What happens when wheel rotation velocity or wheel orientation is saturated?

4.2 Lyapunov-based control law

The energy function in this case has the same form as in the 2-0 robot. It corresponds to the target position in the steering wheel frame, that can be written as:

$$\begin{cases} x_e &= \cos\theta.(x_r - x - L\cos\theta) + \sin\theta.(y_r - y - L\sin\theta) \\ y_e &= -\sin\theta.(x_r - x - L\cos\theta) + \cos\theta.(y_r - y - L\sin\theta) \\ \theta_e &= \text{atan2}(\dot{y}_r, \dot{x}_r) - \theta - \beta \quad \text{(put back in } [-\pi, \pi]) \end{cases}$$

The control law is the same as before, but the ω component has to be mapped to the $\dot{\beta}$ control input with:

$$\dot{\beta} = \omega - \frac{v}{L}\sin\beta$$

Show that this control makes $\mathbf{W}$ a Lyapunov function.

Implement this control and test it for various values of (K_x, K_y, K_θ). Once acceptable values have been found:

- What happens when there are calibration errors?
- What happens when wheel rotation velocity or wheel orientation is saturated?