MPC Exercise

October 3, 2019

In this lecture you are supposed to apply MPC to make the TurtleBot3 follow a predefined reference (first in simulation and then in the lab). All necessary information to complete this task is given below.

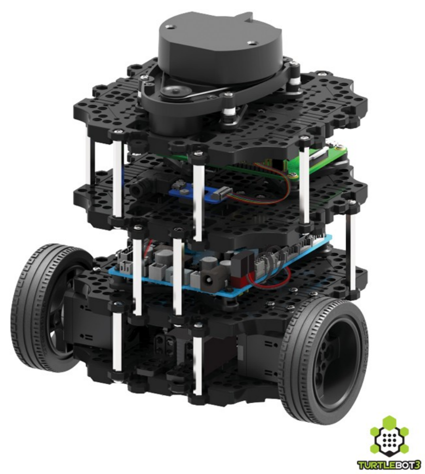


Figure 1: TurtleBot3

The Turtblebot3 has the following Hardware specifications:

Maximum translational velocity	0.22 m/s
Maximum Payload	15 Kg
Weight	1Kg

Table 1: Specifications for the Turtlebot3

The robot is desired to follow a circular trajectory defined as following:

$$ref_x(k) = -0.8 + 0.5\cos(0.2kT + \pi/4) \quad (1a)$$

$$ref_y(k) = -0.4 + 0.5\sin(0.2kT + \pi/4) \quad (1b)$$

Where T is the sample time, and $k \in \mathbb{Z}_{\geq 0}$. However, the robot is not allowed to cross the line $y = -0.6$ while moving on the circular trajectory as shown in the figure below:

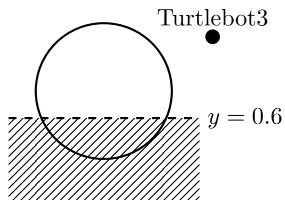


Figure 2: Circular Trajectory of the robot with the unwanted region

The kinematics of the Turtlebot3 can be modeled as following:

$$x(k+1) = x(k) + v(k)\cos(\theta(k))T \tag{2a}$$
$$y(k+1) = y(k) + v(k)\sin(\theta(k))T \tag{2b}$$
$$\theta(k+1) = \theta(k) + \omega(k)T \tag{2c}$$

Where:

x : is the x position of the robot in the inertial frame $\mathcal{I}$.

y : is the y position of the robot in the inertial frame $\mathcal{I}$.

v : is the velocity of the robot in its body frame $\mathcal{B}$.

θ : is the heading angle of the robot (Rotation angle between the inertial frame to the body frame around the z axis.)

ω : is the angular velocity of the heading.

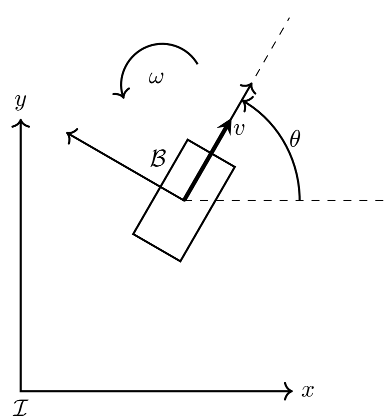


Figure 3: The body frame of the robot in the inertial frame with translational and angular velocity inputs.

The model in (2) is nonlinear and, therefore, a linear MPC cannot be implemented. To deal with this, the MPC will only be relative to the following position model:

$$x(k+1) = x(k) + Tv_x \tag{3a}$$
$$y(k+1) = y(k) + Tv_y \tag{3b}$$
$$\tag{3c}$$

Where:

$$v_x(k) = v(k)\cos(\theta(k))$$
$$v_y(k) = v(k)\sin(\theta(k))$$

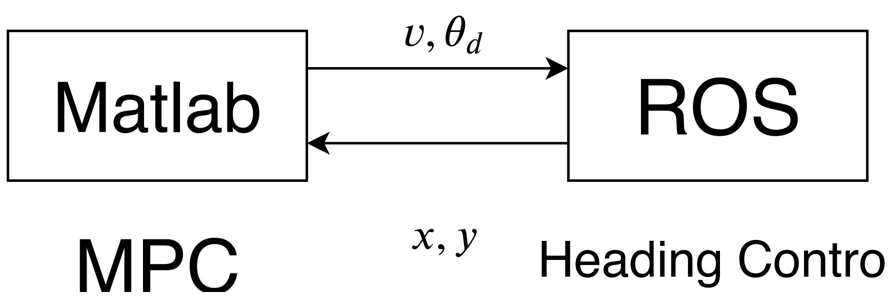
Note that v_x and v_y here are just the component of the velocity vector in the inertial frame. The MPC will then use this model to predict and optimize for the control variables v_x, v_y. After obtaining the control variables v_x and v_y, the velocity in (2) can be recovered as following:

$$v = \sqrt{v_x^2 + v_y^2} \tag{4}$$

and a desired heading angle can be computed from:

$$\theta_d = \text{atan2}(v_y, v_x) \tag{5}$$

Afterwards, a simple P controller running at a higher rate on the robot can control the heading to the desired angle. The following figure shows the structure of the setup:



The MPC will be running at a lower sample time than the simulation and ROS during implementation. The input to the robot from the MPC between its samples is constant. The figure below illustrate the concept:

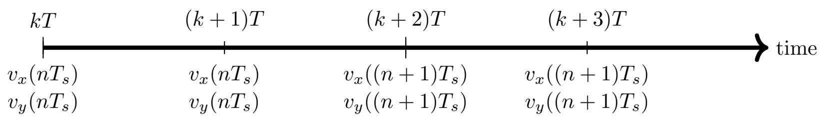


Figure 4: The MPC is running at a sample time T_s lower than the simulation/ROS sample time T. The simulation/ROS apply the same input it has from the MPC until it gets a new one.

Three matlab files are provided:

- `Init`: initialization script to define some parameters.
- `MPCcodeStudent`: the function in which you are supposed to write your MPC code.
- `MPCsimPredictionModelStudent`: simulation script.
- `MPCROS`: implementation script that communicates with a ROS master on the Turtlebot3.