Motion Control RTFM

Raphael Rätz

April 29, 2019

1 Robot Dynamics

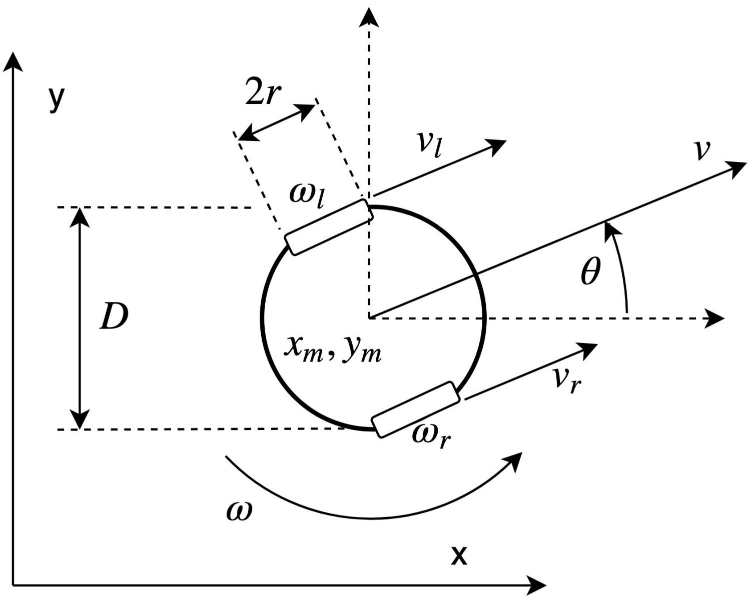


Figure 1: Differential drive robot

The subsequent equations describe the dynamics of a differential drive robot. First of all it is important to understand, that a differential drive robot is a non-holonomic robot. There are certain constraints when moving, namely it impossible to move in a transversal direction. The relation of the wheel velocities and the transversal and rotational velocity of a differential drive robot are given by the matrix T.

$$\begin{bmatrix} v \\ \omega \end{bmatrix} = T \begin{bmatrix} \omega_r \\ \omega_l \end{bmatrix} \quad with \quad T = \begin{bmatrix} \frac{r}{2} & \frac{r}{2} \\ \frac{r}{D} & -\frac{r}{D} \end{bmatrix} \tag{1}$$

The same relation is valid for the accelerations:

$$\begin{bmatrix} \dot{v} \\ \dot{\omega} \end{bmatrix} = T \begin{bmatrix} \dot{\omega}_r \\ \dot{\omega}_l \end{bmatrix} \tag{2}$$

Based on Newtons law of motion, the relation of the transversal force and the torque applied on the robot can be expressed as follows:

$$\begin{bmatrix} F \\ \tau \end{bmatrix} = M \begin{bmatrix} \dot{v} \\ \dot{\omega} \end{bmatrix} \quad with \quad M = \begin{bmatrix} m & 0 \\ 0 & J \end{bmatrix} \tag{3}$$

The translational force in the direction of movement and the torque can as well be expressed as:

$$\begin{bmatrix} F \\ \tau \end{bmatrix} = P \begin{bmatrix} \tau_r \\ \tau_l \end{bmatrix} \quad with \quad P = \begin{bmatrix} \frac{1}{r} & \frac{1}{r} \\ \frac{D}{2r} & -\frac{D}{2r} \end{bmatrix} \tag{4}$$

The equations (4) and (3) are equal:

$$P \begin{bmatrix} \tau_r \\ \tau_l \end{bmatrix} = M \begin{bmatrix} \dot{v} \\ \dot{\omega} \end{bmatrix} \tag{5}$$

And finally, the equations of motion without friction can be represented as:

$$\begin{bmatrix} \dot{\omega}_r \\ \dot{\omega}_l \end{bmatrix} = \Lambda \begin{bmatrix} \tau_r \\ \tau_l \end{bmatrix} \quad with \quad \Lambda = T^{-1} M^{-1} P \tag{6}$$

$$\Lambda = \begin{bmatrix} \frac{1}{mr^2} + \frac{D^2}{4Jr^2} & \frac{1}{mr^2} - \frac{D^2}{4Jr^2} \\ \frac{1}{mr^2} - \frac{D^2}{4Jr^2} & \frac{1}{mr^2} + \frac{D^2}{4Jr^2} \end{bmatrix} \tag{7}$$

2 Kinematic Position Controller

2.1 Kinematics

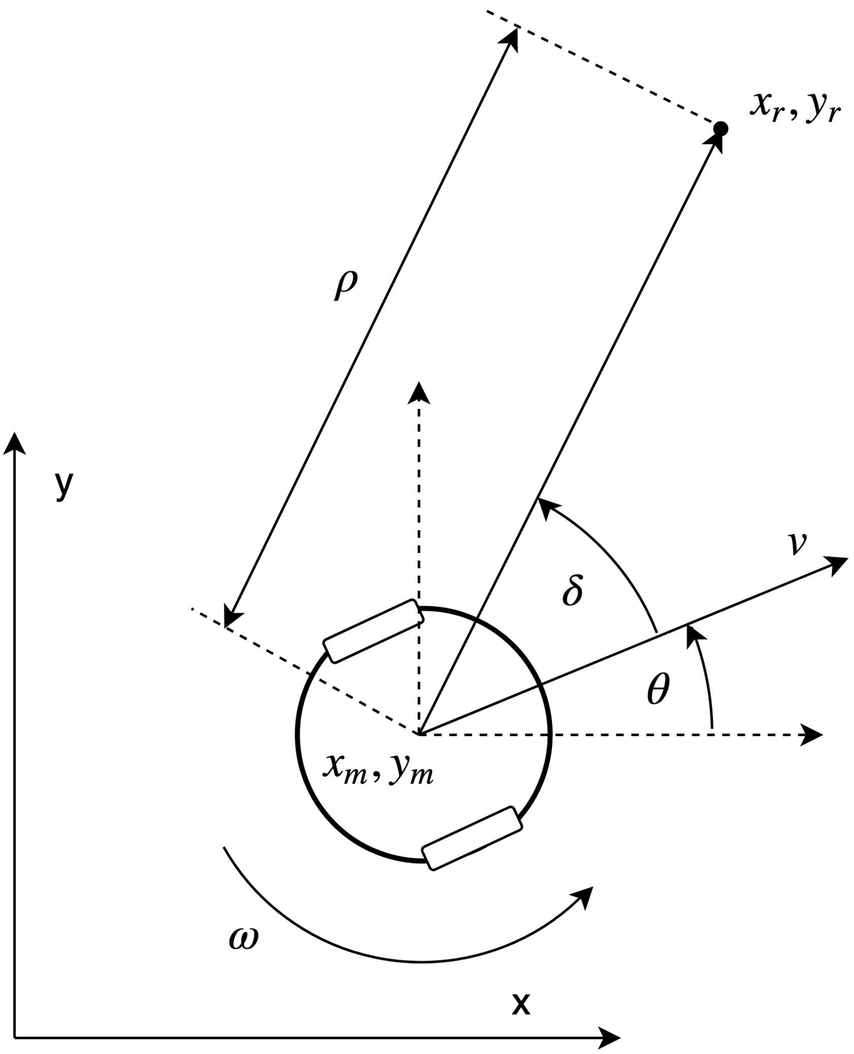


Figure 2: Robot pose and goal position x_r, y_r

Figure 2 shows the measured pose of the robot (x_m, y_m, θ) as well as the goal position (x_r, y_r). The angle θ is measured with respect to the global x-axis. The robot moves with a translational velocity v and an angular velocity ω. The non-holonomic behaviour of the differential drive robot can be represented as follows:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} cos(\theta) & 0 \\ sin(\theta) & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix} \tag{8}$$

The Euklidian distance from the robots position (x_m, y_m) to the reference position (x_r, y_r) is denoted ρ. The angle δ describes the difference between the momentary heading of the robot and the heading of the goal position. The angle δ can be calculated as:

$$\delta = atan2(y_r - y_m, x_r - x_m) - \theta \tag{9}$$

And the distance ρ as follows:

$$\rho = \sqrt{(x_r - x_m)^2 + (y_r - y_m)^2} \tag{10}$$

The dynamics of ρ and δ can be written as:

$$\begin{bmatrix} \dot{\rho} \\ \dot{\delta} \end{bmatrix} = \begin{bmatrix} -vcos(\delta) \\ \frac{v}{r} sin(\delta) - \omega \end{bmatrix} \tag{11}$$

2.2 Lyapunov based Controller

A function V is called a Lyapunov function candidate if it satisfies the requirements:

- $V(x) = 0$ if and only if $x = 0$
- $V(x) > 0$ if and only if $x \neq 0$
- $\dot{V}(x) \leq 0$ for all values $x \neq 0$

The following Lyapunov function candidate with two states x_1 and x_2 has been chosen in this case.

$$V(x) = \frac{1}{2} x^T x = \frac{1}{2} x_1^2 + \frac{1}{2} x_2^2 \tag{12}$$

A dynamic system is said to be globally asymptotically stable if it can be shown that $\dot{V}(x) < 0$ for $x \neq 0$. The corresponding derivative of the chosen $V(x)$ is:

$$\dot{V}(x) = x_1 \dot{x}_1 + x_2 \dot{x}_2 \tag{13}$$

When setting $x_1 = \rho$ and $x_2 = \delta$, the derivative of V takes the following form:

$$\dot{V} = -v\rho cos(\delta) + \delta(\frac{v}{\rho} sin(\delta) - \omega) \tag{14}$$

The left part of this equation can be forced to be negative by defining the translational velocity control law:

$$v = K_\rho cos(\delta)\rho \qquad \text{with} \quad K_\rho > 0 \tag{15}$$

The right term of $\dot{V}$ becomes therefore:

$$\delta(\frac{K_\rho cos(\delta)\rho}{\rho} sin(\delta) - \omega) = \delta\left(K_\rho cos(\delta) sin(\delta) - \omega\right) \tag{16}$$

In order to force the total expression of $\dot{V}$ to be negative, a simple proposition is made. The right term can be forced to be zero by defining the angular velocity as:

$$\omega = K_\rho sin(\delta) cos(\delta) \tag{17}$$

Consequently, the total expression of $\dot{V}$ becomes as follows, which is negative for if $x \neq 0$ and zero if $x = 0$.

$$\dot{V} = -K_\rho cos(\delta)^2 \rho^2 \leq 0 \tag{18}$$

According to Lyapunov, the robot is therefore asymptotically stable if using the control laws (15) and (17) because the derivative of V is globally negative except from the origin where it is zero. The two control laws necessitate only one tuning parameter K_ρ, which makes this controller easily tunable despite its non-linear nature.

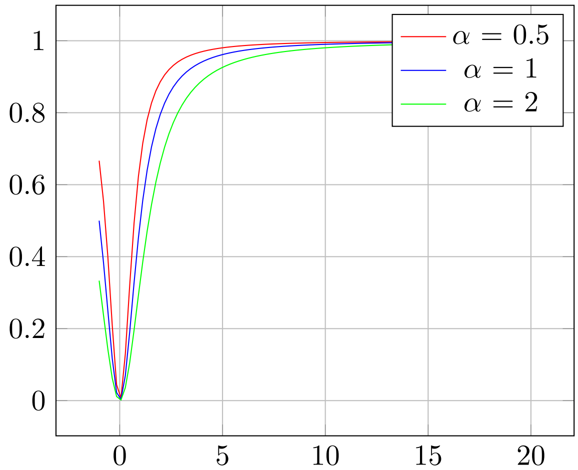
2.3 Practical consideration: damping around origin

Due to the non-holonomic nature of the differential robot, the robot may start rotating strongly around the origin ($x_m \approx x_r$ and $y_m \approx y_r$) in order to converge. Imagine that the robot arrives a few tenth of a millimetre laterally to the goal position. Its only possibility to correct would be to turn at its momentary position and move slightly forward or backwards. If the position is still not perfect, it will continue until it arrives at a position where the error is too small in comparison the the mechanical frictions in the system. Especially the command of ω it is unwanted to be unstable or sensitive to external perturbations. It is very undesired that the robot starts rotating around the origin. It is therefore proposed to dampen or weaken the control law of the angular velocity around the origin. The following function $g(\rho)$ is proposed [1]:

$$g(\rho) = \left(1 - \frac{\alpha}{\rho^2 + \alpha}\right) \tag{19}$$

[1] In the code, "alphaSquared" might be used instead of "alpha". However, "alphaSquared = α"

The characteristic of this function is that its value is approximately 1 for $\rho^2 \gg \alpha$ which guarantees proper functioning of the Lyapunov based control law. However as ρ approaches zero, $g(\rho)$ decreases smoothly to zero.



The function $g(\rho)$ is multiplied with the control law for the angular velocity ω. The final control laws are therefore given by equation (21) and (20). Proximate to the origin, the control law for ω is almost zero, which avoids undesired rotations.

$$\boxed{v = K_\rho cos(\delta)\rho} \tag{20}$$

$$\boxed{\omega = K_\rho g(\rho) sin(\delta) cos(\delta)} \tag{21}$$

3 Motion Planning

3.1 Trapezoidal Motion Planner

In order to guarantee a smooth trajectory, a trapezoidal motion planner is used. It provides trajectories with finite acceleration and deceleration. A trapezoidal profile as it is shown in figure 3 can be calculated based on the desired acceleration a_{max}, the desired velocity v_{max} and the desired distance Δs. The acceleration, the velocity and the covered distance at a given time t are given by the set of equations (22). Note that for these equations, Δs is considered to be positive.

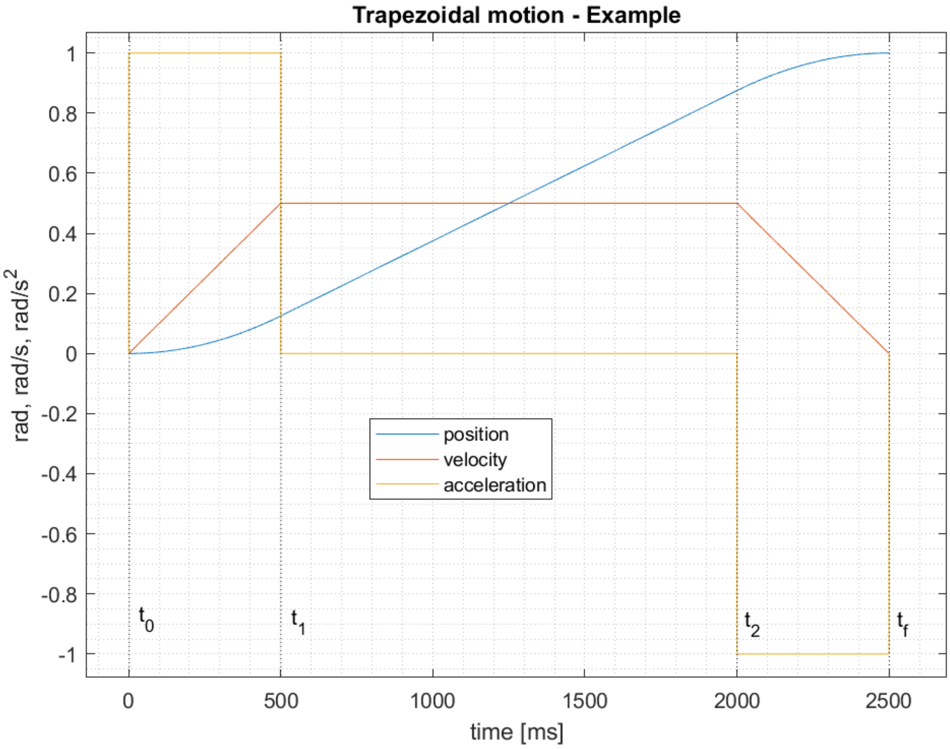


Figure 3: Trapezoidal motion profile

$$a(t) = \begin{cases} a_{max} & \text{for} \quad t < t_1 \\ 0 & \text{for} \quad t_1 \leq t < t_2 \\ -a_{max} & \text{for} \quad t_2 \leq t < t_f \end{cases}$$
$$v(t) = \begin{cases} a_{max}t & \text{for} \quad t < t_1 \\ v_{max} & \text{for} \quad t_1 \leq t < t_2 \\ a_{max}(t_1 + t_2 - t) & \text{for} \quad t_2 \leq t < t_f \end{cases} \tag{22}$$
$$s(t) = \begin{cases} \frac{1}{2}a_{max}t^2 & \text{for} \quad t < t_1 \\ \frac{1}{2}a_{max}t_1^2 + v_{max}(t - t_1) & \text{for} \quad t_1 \leq t < t_2 \\ \frac{1}{2}a_{max}t_1^2 + v_{max}(t_2 - t_1) + \frac{1}{2}a_{max}(t - t_2)^2 & \text{for} \quad t_2 \leq t < t_f \end{cases}$$

The corresponding time instants t_1, t_2 and t_f can be calculated by the equations (23).

$$
\begin{aligned}
t_1 &= \begin{cases} \sqrt{\frac{\Delta s}{a_{max}}} & \text{if} \quad \Delta s \leq \frac{v_{max}^2}{a_{max}} \\ \frac{v_{max}}{a_{max}} & \text{otherwise} \end{cases} \\
t_2 &= \begin{cases} t_1 & \text{if} \quad \Delta s \leq \frac{v_{max}^2}{a_{max}} \\ \frac{\Delta s}{v_{max}} & \text{otherwise} \end{cases} \\
t_f &= \begin{cases} 2t_1 & \text{if} \quad \Delta s \leq \frac{v_{max}^2}{a_{max}} \\ t_1 + t_2 & \text{otherwise} \end{cases}
\end{aligned} \tag{23}
$$

If the acceleration a_0 is not high enough or if the distance Δs is too short, it is possible that the velocity v_{max} can not be reached and therefore $t_1 = t_2$. This kind of motion is known as triangular motion profile and is illustrated in figure 4

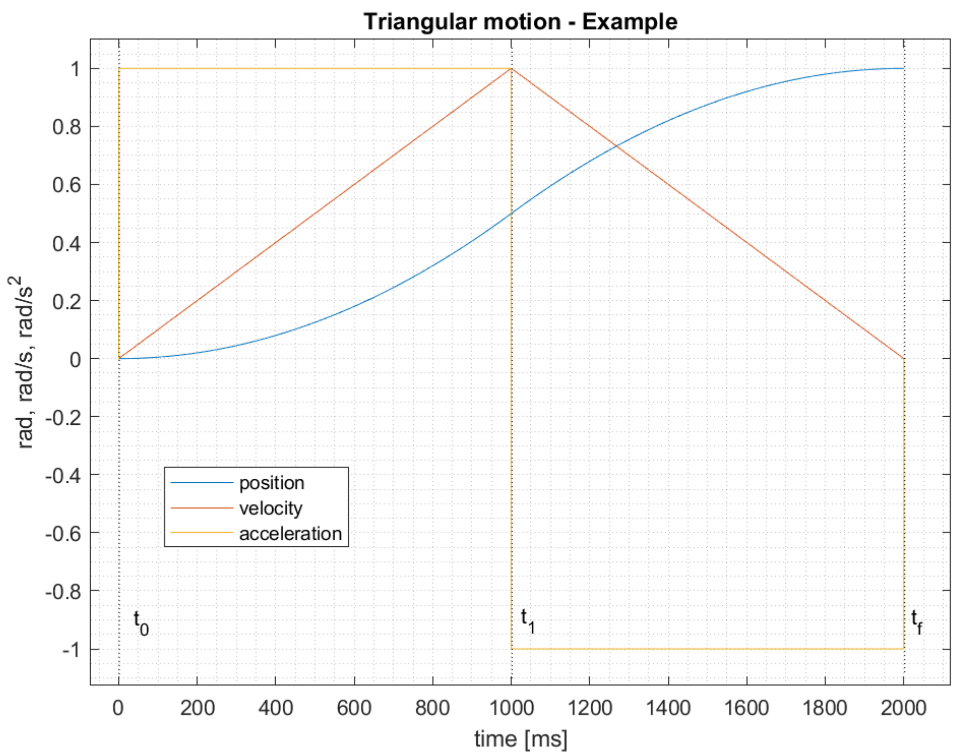


Figure 4: Triangular motion profile

3.2 Bezier Curves

The higher level path planner (Beaglebone) provides a path which consists of several segments of cubic Bezier curves. One cubic Bezier curve is defined as:

$$\mathbf{b}(u) = (1-u)^3\mathbf{p_0} + 3(1-u)^2u\mathbf{p_1} + 3(1-u)u^2\mathbf{p_2} + u^3\mathbf{p_3}$$
$$\text{with} \quad \mathbf{p_n} = \begin{bmatrix} x_n \\ y_n \end{bmatrix} \quad \text{and} \quad 0 \le u \le 1 \tag{24}$$

This equation can be expanded and rearranged to the form:

$$\mathbf{b}(u) = (-\mathbf{p_0} + \mathbf{p_1} - \mathbf{p_2} + \mathbf{p_3})u^3 + (3\mathbf{p_0} - 6\mathbf{p_1} + 3\mathbf{p_2})u^2 + (-3\mathbf{p_0} + 3\mathbf{p_1})u + \mathbf{p_0} \tag{25}$$

The analytical expression for the length of a Bezier curve is rather complicated and consists of several square roots (always having two solutions). Yet, numerical integration can be used in order to calculate the length. One Bezier curve is divided in K segments. The length of each of these smaller segments is approximated by linear segments. The total length L is simply calculated as sum of the lengths of the linear segments.

$$L = \sum_{k=0}^{K-1} \sqrt{(b_x(u_{k+1}) - b_x(u_k))^2 + (b_y(u_{k+1}) - b_y(u_k))^2} \quad \text{with} \quad u_k = \frac{k}{K} \tag{26}$$

4 Current Control

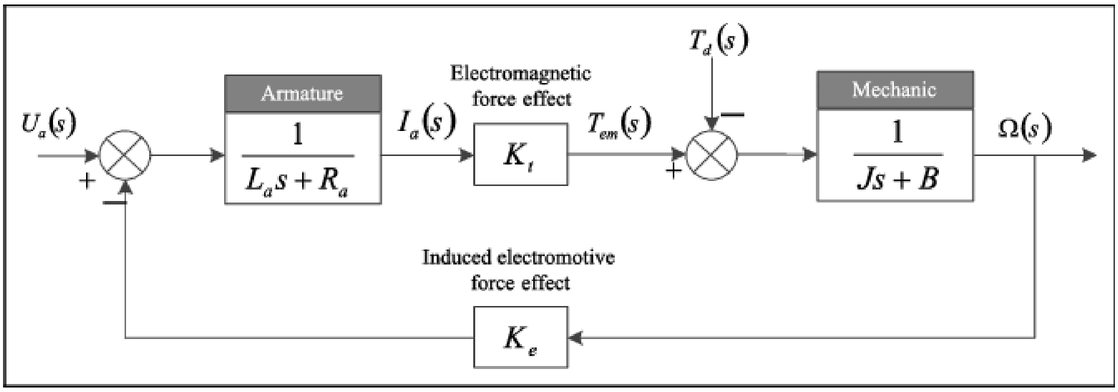


Figure 5: Block schema of DC motor

In order to simplify things, the inertia J contains not only the rotor inertia, but the total inertia opposing to the motor torque (including robot mass, inertia, etc..). The same is true for the friction term b.

Since the motor torque is proportional to the motor current, we have direct control of the torque by using a current controller. The transfer function $G_{iu}(s)$ which describes the behaviour of the motor current in function of the input voltage in the frequency domain can be represented as follows:

$$G_{iu}(s) = \frac{J}{K_t K_e} \cdot \frac{s}{\frac{LJ}{K_t K_e}s^2 + \frac{RJ}{K_t K_e}s + 1} \tag{27}$$

The dynamic behaviour of the motor current can be simplified to (28).

$$G_{iu}(s) \approx \frac{1}{Ls + R} \tag{28}$$

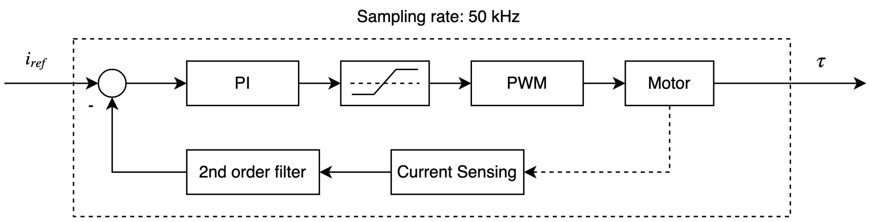


Figure 6: Block schema of the current control

5 Velocity Control

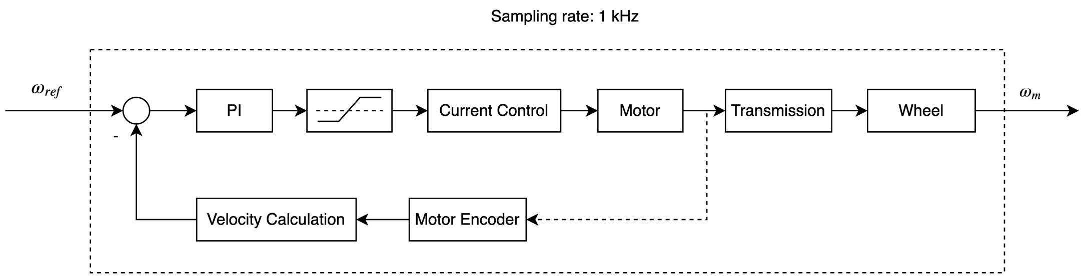


Figure 7: Block schema of the velocity control