Laboration 1: PID-controls

Sensors and Sensing

Michael Floßmann, Tom Olsson

November 18, 2015

List of Figures

Listings

1 Theory and motivation

1.1 PID controller

PID in the name PID-controller is short for *P*roportional-*Integral*-*Derivative*-controller. As this implies, the controlling signal is based on a proportion of the current value, the previous values, and the rate of change of the observed value. The mathemathical formulation of this can be seen in (1).

Let:

$e(t)$	be some error measurement between current state and preferred state
K_p, K_i, K_d	be the respective weights for the proportional, integral and derivate terms
$u(t)$	be the output signal at time t

Then:

$$u(t) = K_p \cdot e(t) + K_i \cdot \int_0^\infty e(t) \cdot dt + K_d \cdot \frac{de}{dt} \tag{1}$$

1.2 Minimum jerk

The minimum jerk equation is an important part of creating smooth control. When a rotating actuator such as a motor starts, both the rotor and the stator will be at rest. The momentum generated by the motor can therefore cause movement in either part. As this can create an unwanted jerk while the rotor accelerates, it is important to accelerate slowly so that the stator remains at rest in relation to the reference frame. This can be achieved by the *minimum jerk equation* shown in (2).

Let:

x_i, x_f	be the initial and final states
t, T	be the elapsed time since the action started, and the preferred total time for the action
$x(t)$	be the estimated state at time t

Then:

$$x(t) = x_i + (x_f - x_i) \cdot \left[10 \left(\frac{t}{T} \right)^3 - 15 \left(\frac{t}{T} \right)^4 + 6 \left(\frac{t}{T} \right)^6 \right] \tag{2}$$

The T parameter has to be estimated. If T is much larger than the actual time that is needed for the trajectory, the velocity will be very low, and if T is too low $x(t)$ will approach infinity unless $\frac{t}{T}$ is clamped to $[0, 1]$. However, this solution is not optimal. Instead, we choose to calculate the derivative $\frac{dx}{dt}$, and this can be seen in **??**.

(3)

2 Implementation

2.1 Hardware

The laboration is performed using an *Arduino Due* microcontroller

2.2 Position controller

Describe the implementation of the position controller

2.3 Velocity controller

Describe the implementation of the velocity controller

3 Verification and results

3.1 PID-tuning

3.2 Results