INTRODUCTION TO THE PID CONTROLLER AND ITS SOFTWARE IMPLEMENTATION — HOMEWORK

PANTELIS SOPASAKIS

Guidelines: Solve the following exercises and send your answers by email to p.sopasakis@gmail.com. Your answers should be submitted in a single PDF file along with a ZIP file with your source code.

Evaluation criteria: Any correct answer is acceptable. You will be evaluated for the correctness of your answers and source code (85%) and the quality and readability of your source code (15%). The relative importance of each question is indicated in brackets.

Deadline: 20 February, 2019, by close of play (17:00)

In the lecture, we presented a *discrete-time approximation* of the PID controller using the following integral approximation

$$\int_{t_{k-1}}^{t_k} e(\tau)\,\mathrm{d}\tau \approx T_s e(t_k) \tag{1a}$$

which is known as the *rectangle rule*. We also used the *backward differences* approximation for the derivative of the error, that is,

$$\frac{\mathrm{d}e(t_k)}{\mathrm{d}t} \approx \frac{e(t_k) - e(t_{k-1})}{T_s}. \tag{1b}$$

(1) [10%] Give a discrete-time approximation of the PID controller using the *backward differences* approximation for the derivative of the error (Equation (1b)) and the *trapezoidal rule* — or *trapezoid rule* — that is,

$$\int_{t_{k-1}}^{t_k} e(\tau)\,\mathrm{d}\tau \approx \frac{T_s}{2}\left(e(t_k) + e(t_{k-1})\right), \tag{2}$$

for the approximation of the integral of the error.

(2) [10%] Write some simple pseudocode for the discrete-time PID controller you derived in question No. 1 using the trapezoidal rule.

(3) [15%] Implement the above discrete-time PID controller in MATLAB. You may build up on the source code found at https://alphaville.github.io/qub/matlab/.

(4) [15%] Compare the discrete-time PID controller implementation you produced in question No. 3 to the one found at https://alphaville.github.io/qub/matlab/. Try out different sampling times. According to your observations, how does the sampling time affect the behaviour of the closed-loop system?

(5) [20%] Suppose that due to a technical glitch, there is a constant bias of $u_{\text{bias}} = 5^\circ$ on the steering angle, that is, the controller commands a control action $u(t)$ to the wheels, but the actual steering angle is $u(t) + u_{\text{bias}}$. Perform simulations to demonstrate that a PD controller cannot eliminate the offset. According to your observations, what is the effect that K_p and K_d have on the offset?

(6) [20%] Experiment with the values of the proportional gain (K_p), the derivative gain (K_d) and the integral gain (K_i) and determine values that lead to a nice, non-oscillatory behaviour with no offset (there is no *single* correct answer). According to your observations, how do K_p, K_d and K_i affect the frequency and amplitude of the oscillations of the error?

(7) [10%] Simulate the controlled system starting from the initial position $p_x(0) = 0\,\mathrm{m}$ and $p_y(0) = 1\,\mathrm{m}$ and initial orientation $\theta(0) = 0\,\mathrm{rad}$ towards the set-point $p_y^{\mathrm{sp}} = 1.5\,\mathrm{m}$ using the PID controller from question No. 6.

Homework for the course "The PID controller and its software implementation." The slides are available at https://alphaville.github.io/qub/pid-101/. The associated MATLAB source code can be found at https://alphaville.github.io/qub/matlab/. For questions you may reach me by email at p.sopasakis@gmail.com or in person in Office No. ... during office hours. Last updated: February 5, 2019.