Robot Programming and Control - Assignment 2

Lorenzo Busellato[1], Giacomo Canella[2], and Michel Valentini[3]

[1]VR442749 - lorenzo.busellato_02@studenti.univr.it
[2]VR481630 - giacomo.canella@studenti.univr.it
[3]VR472456 - michel.valentini@studenti.univr.it

CONTENTS

LIST OF FIGURES

Robot Programming and Control - Assignment 2

I. Introduction

This document presents the results of assignment number 2 for the Robot Programming and Control course. The goals of the assignment are:

- Synthesizing a PD (Proportional-Derivative) and a PID (Proportional-Integral-Derivative) position controller using loop-shaping techniques.
- Converting the controllers to the digital domain by using both MATLAB's Control System Toolbox built-in function *c2d* and Forecast framework's *AnalogFilter* function.
- Implementing both digitalized controllers and comparing their performance against a sweep signal and a step signal.

The employed loop-shaping techniques are described in section II. The methods for controller digitalization are described in section III. The experimental setup and results are presented in section IV. Finally the conclusions are summarized in section V

II. Loop shaping

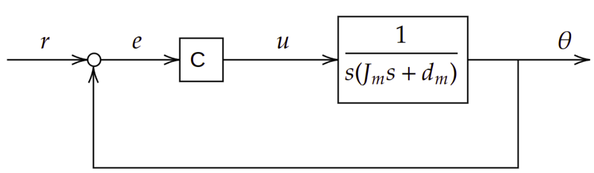

Fig. 1. Block diagram of the controller-plant system

The aim of loop shaping is to design a controller C such that the response of the system to a given reference signal r is acceptable. This essentially means that we add poles and zeros to the controller while keeping some constraints about the stability of the resulting transfer function. The constraints are that the phase margin should be greater than 75° and the bandwidth should be around 5 Hz.

For the PD controller an high frequency (with respect to the frequency of the mechanical pole) zero was added. Then the static gain of the controller was adjusted to achieve the desired bandwidth. Then the frequency of the zero was gradually increased in order to reduce overshoot and improve the step response. These two steps were repeated until the response was acceptable. Finally a pole with a frequency in the decade after the zero was added in order to make the system causal, which is necessary to be able to discretize it with MATLAB's *c2d* function.

The resulting transfer function for the PD controller is:

$$C_{PD} = \frac{4435.4(s + 6.362)}{(s + 602.9)} \tag{1}$$

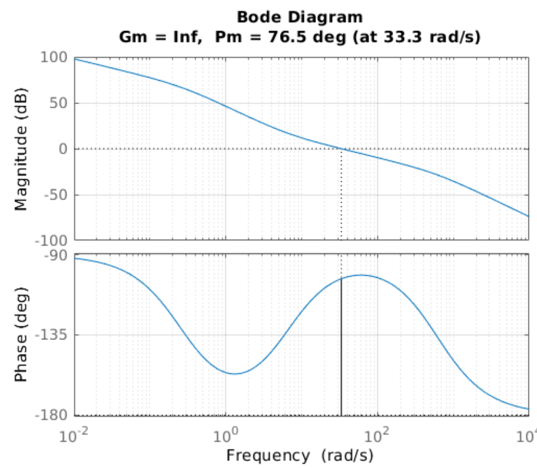

Fig. 2. Bode diagram of the open-loop $C_{PD} \cdot G$

As can be seen in figure 2, the phase margin is above the suggested value of 75°, reducing the possibility of oscillatory behaviour.

The PID controller was designed in a similar way to the PD controller, with the exception that a low frequency zero was added in order to be able to maintain the phase margin within acceptable limits and for the addition of an integrator (i.e. a pole in zero).

The resulting transfer function for the PID controller is:

$$C_{PID} = \frac{3798.4(s + 0.4458)(s + 4.021)}{s(s + 772.6)} \tag{2}$$

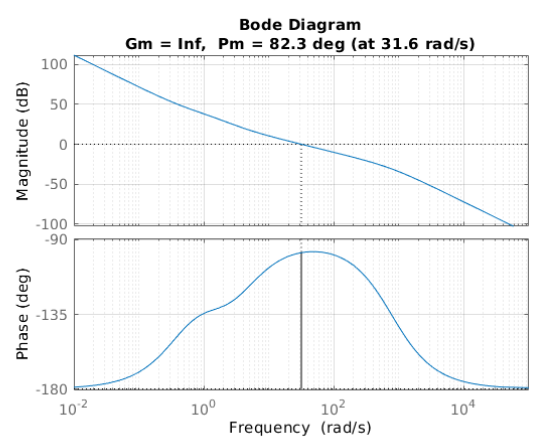

Fig. 3. Bode diagram of the open-loop $C_{PID} \cdot G$

As can be seen in figure 3, the phase margin is above the suggested value of 75°, reducing the possibility of oscillatory behaviour.

III. Controller digitalization

To be implemented on the hardware, the transfer functions obtained from the loop shaping step must first be digitalized by converting them from a continuous domain to a discretized one.

A. c2d

The *c2d* function of MATLAB's Control Systems Toolbox allows to discretize a transfer function. The function takes as arguments the transfer function, the sampling time and the discretization method. In this case a sampling time of $1\,\mathrm{ms}$ and the default discretization method, zero-order hold, were used.

For the PD controller the result of the discretization is:

$$c2d(C_{PD}, 0.001) = \frac{4435.4(z-0.9952)}{(z-0.5472)} \tag{3}$$

For the PID controller the result of the discretization is:

$$c2d(C_{PID}, 0.001) = \frac{3798.4(z-1)(z-0.9974)}{(z-1)(z-0.4618)} \tag{4}$$

Therefore, since:

$$u_k = C(z) \cdot \frac{z^{-1}}{z^{-1}} \cdot e_k$$

where u is the control action, e is the error and z^{-1} represents a delay of one sample, the implementation of the PD controller is:

$$u_k = 4435.4e_k - 4414.1e_{k-1} + 0.5472u_{k-1} \tag{5}$$

while the implementation of the PID controller is:

$$\begin{aligned} u_k = {} & 3798.4e_k - 7586.9e_{k-1} + 3788.5e_{k-2} \\ & + 1.4618u_{k-1} - 0.4618u_{k-2} \end{aligned} \tag{6}$$

B. AnalogFilter

AnalogFilter is a function of Forecast framework that discretizes a transfer function starting from the coefficients of its general form:

$$H(s) = \frac{b_n s^n + b_{n-1}s^{n-1} + \cdots + b_1 s + b_0}{a_m s^m + a_{m-1}s^{m-1} + \cdots + a_1 s + a_0} \tag{7}$$

The discretization is achieved by instantiating an *AnalogFilter* object, whose constructor takes as input the order of the transfer function, which will be 1 for the PD controller and 2 for the PID controller, and the *a* and *b* vectors of coefficients.

The control action is then computed by calling the *process* property of *AnalogFilter*, which takes as input the current error in the position and the sampling time.

IV. Experiments and results

The synthesised controllers were tested on an ESCON testbed with two reference signals, a step and a sweep. The step signal is simply a constant reference of $1\,\mathrm{rad}$, while the sweep signal is built like follows:

$$r(t) = A\sin(2\pi f t) \tag{8}$$

with amplitude $A = \frac{\pi}{16}$ and where f is a frequency which increases with time as follows: $f(t) = t$. For the *AnalogFilter* implementation of the PID controller, the frequency of the reference increases as: $f(t) = 0.2 \cdot t$.

As can be seen in figure 4, the step response of the PD controller is practically the same both in the *AnalogFilter* and *c2d* implementations. The *AnalogFilter* implementation is slightly faster than the *c2d* one, but there is a very small overshoot which is however damped very fastly. For the sweep signal, as seen in figure 5, the responses are again very similar, with both implementations achieving a controller bandwidth of around:

$$f(t \approx 4\,\mathrm{s}) \approx 4\,\mathrm{Hz}$$

For the PID controller step response, as can be seen in figure 6, both implementations reach the target position with no overshoot or instability, but the *AnalogFilter* implementation is much faster than the *c2d* one. In the sweep signal response, as seen in figure 7, the *c2d* implementation seems to accumulate a bigger phase displacement over time. In either case the achieved controller bandwidth is around:

$$f(t \approx 4\,\mathrm{s}) \approx 4\,\mathrm{Hz}$$

which is the same as the PD controller. Therefore, in all implementations the desired controller bandwidth of $5\,\mathrm{Hz}$ is not fully reached, and this is likely due to the current saturation observed in the higher frequencies.

V. Conclusions

Two position controllers, a PD and a PID one, have been designed and implemented on hardware. The transfer functions of the controllers have been discretized using two methods, *AnalogFilter* and MATLAB's *c2d*. Their performance was evaluated with respect to their accuracy in following two types of reference signals, a step signal and a sweep signal, as well as between the two discretization implementations. The PID controller performed better than the PD in the step response, while their performance was about the same in the sweep response. No relevant difference was observed between the discretization with *AnalogFilter* and with *c2d*.

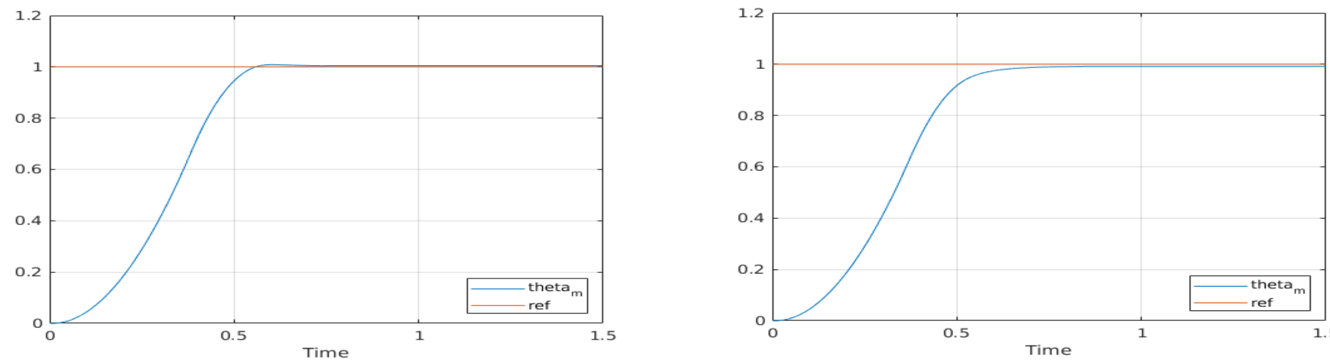

Fig. 4. PD controller response to the step signal - *AnalogFilter* on the left, *c2d* on the right

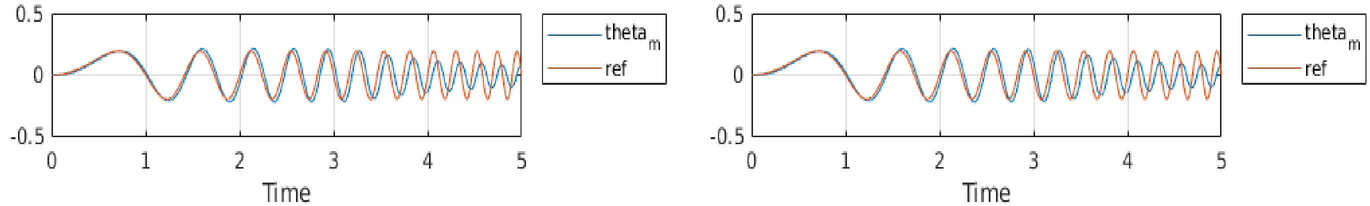

Fig. 5. PD controller response to the sweep signal - *AnalogFilter* on the left, *c2d* on the right

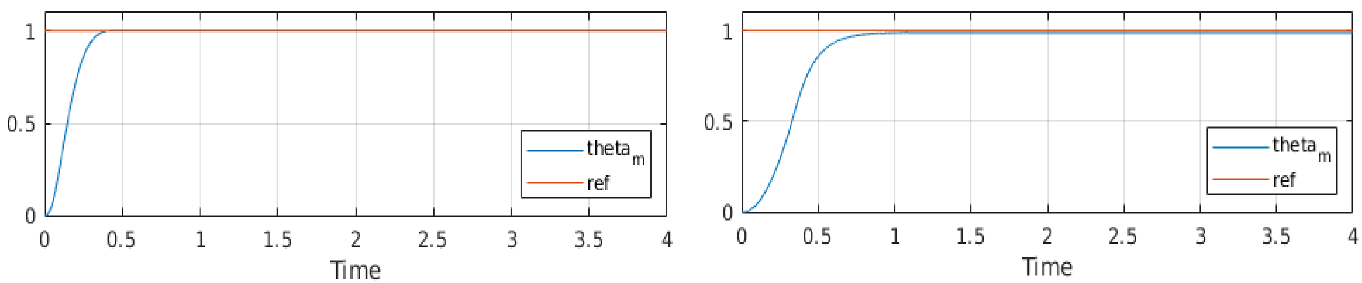

Fig. 6. PID controller response to the step signal - *AnalogFilter* on the left, *c2d* on the right

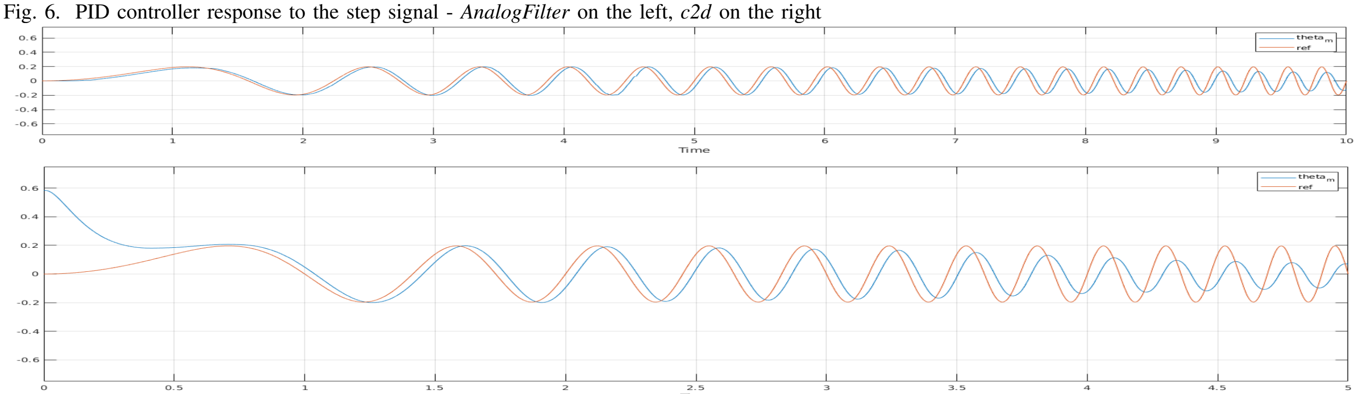

Fig. 7. PID controller response to the sweep signal - *AnalogFilter* on top, *c2d* on the bottom