DC motor control position

Control Theory

Universidade do Minho

Masters in mechatronics engineering
Teacher : Paulo Garrido
2012/2013

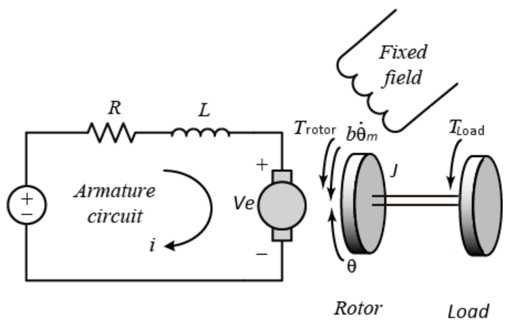
José Gonçalves, student number PG20643
Marco Marques, student number PG22716

Description

DC motors that use feedback control are called DC servomotors. They are known to have precise angular position and have a quick response. This paper will focus on the modeling and position control of a DC motor with permanent magnets. We first develop the differential equations and the Laplace domain transfer function model of the system DC motor/Load. Next we will apply the parameters of the Maxon DC motor 2140.937, identify the parameters of a PID controller using simulation, and make an introduction of the implementation.

1. System modeling of a DC Motor

The electric equivalent circuit of the armature and the free-body diagram of the rotor are shown in the following figure 1. The parameters are describe in the table. This analysis is valid for DC motors with permanent magnets.



R	Resistance of the coil	Ω
L	Inductance of the coil	H
Ve	Back emf (back electromotive force)	V
i(t) = i	Current in the armature	A
Ke	Electromotive force constant	V/ rad/sec
Kt	Motor torque constant	Nm/Amp
θ	Angular position	rad
ω	Angular velocity = dθ/dt = θ´	rad/s
J	Moment of inertia of rotor	kg.m²
b	Motor viscous friction constant	Nm.m
Trotor	Torque	N.m
TLoad	Torque	N.m

Figure 1 - Equivalent circuit of DC Motor (permanent magnets) + Load

1.1 - Differential equations

1 - The permanent magnets in the motor induce the following back emf Ve in the armature: $Ve = Ke . \omega = Ke . \theta'$ (1)

The motor produces the following torque, which is proportional to the motor current *i(t)* $T = Kt . i(t)$ (2)

Assuming there are no electromagnetic losses and no mechanical losses : mechanical power is equal to the electrical power $T . \omega' = Ve . i(t)$ (3)

substituting by equations (1) and (2) => $Kt . i(t) . \omega = Ke . \omega . i(t)$ = > we will consider $Kt = Ke = K$ (4)

2 - From the Kirchhoff´s law voltage, in the armature circuit : $\nu = R\, i(t) + L\, di(t)/dt + Ve$ (5)

3 - From the Newton´s Law $T = T\,rotor + T\,load + T\,viscous = Jm. \theta'' + T\,load + b\,\theta'$ (6)

1.2 - The Laplace domain transfer function model

The main variables of the system are :

ν	input of the system
θ	output of the system
T Load	disturbance of the system

From equations (5) and (1)

$\nu = R\,i(t) + L\,di(t)/dt + Ve = R\,i(t) + L\,di(t)/dt + K\,\omega$

Applying Laplace transform $V(s) = R.\,I(s) + L.s.\,I(s) + K.\Omega(s)$

=> $V(s) - K.\Omega(s) = (Ls + R)\,I(s)$

$$I(s) = \frac{V(s) - K.\Omega(s)}{(Ls+R)} = [\,V(s) - K.\Omega(s)\,]\,.\,\frac{1}{(Ls+R)}$$

From equation (2) $T(s) = K.\,I(s)$

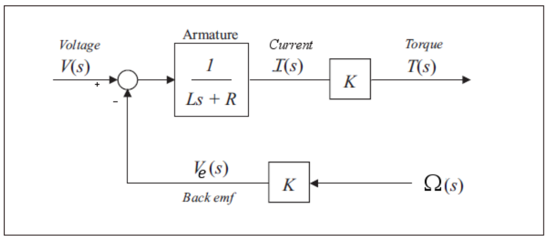


Figure 2

From equation (6) $T = Jm\,.\theta'' + b.\,\theta' + Tload$

Applying Laplace transform

=> $T(s) = Jm.s^2.\,\Theta(s) + B.s.\Theta(s) + Tload(s)$

=> $T(s) - T\,load = s.\,(Jm.s + b)\,.\Theta(s)$

=> $\Theta(s) = (\,T(s) - T\,load\,)\,.\,\frac{1}{J.s+b}\,.\,\frac{1}{s}$

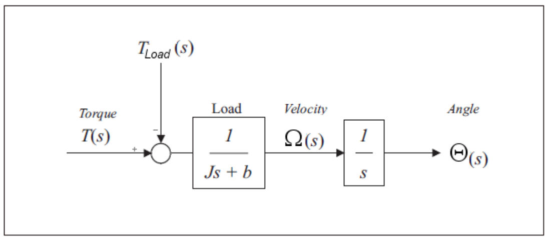


Figure 3

Joining the both previous block diagrams, and assuming from equation (1) $K\,.\,\omega = K\,.\,\theta'$, then from Laplace $K.\Omega(s) = K.s.\Theta(s)$, we get the open-loop equivalent block diagram of the system DC motor/Load of figure 4.

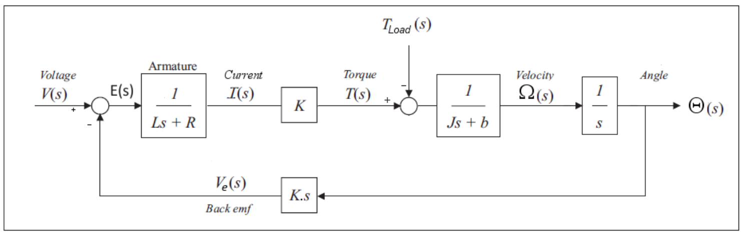


Figure 4 - open loop transfer function of DC motor

If we consider T Load = 0 ,

$$\Theta(s) = \langle \frac{1}{Ls+R}\,.K.\frac{1}{J.s+b}\,.\frac{1}{s}\rangle\,.\,\langle V(s) - K.s\,.\Theta(s)\,\rangle$$

=> $\dfrac{\Theta(s)}{V(s)} = \dfrac{K}{s\,((L.s+R)\,(J.s+b) + K^2)}$ this is the open loop transfer function of DC motor, without load. (7)

L/R is the electrical time constant, and J/b is the mechanical time constant. Usually L/R << Jm /b , so we can neglect (Ls + R). With this consideration, we obtain the simplified block diagram of figure 5.

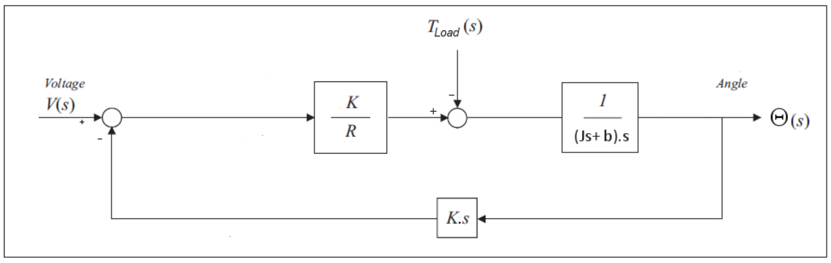


Figure 5 – Simplified open loop transfer function of DC motor

1.3 - Controlled model with feedback

To control the position of the motor, the system must be closed with a feedback, and a controller C(s) has to be added.

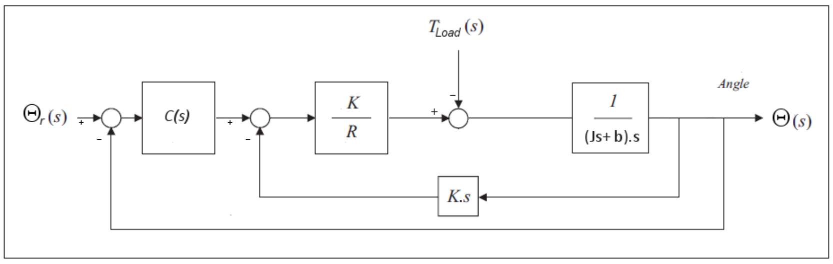


Figure 6 - closed loop transfer function of DC motor

1.4 - Model reduction

The structure of the model have to be reduced and simplified to a better analysis of the model as to capture important characteristics.

In a first step, the TLoad(s) is removed from the minor feedback (in blue in figure 7), to be able further to eliminate this minor feedback, and have a single main feedback from output to input.

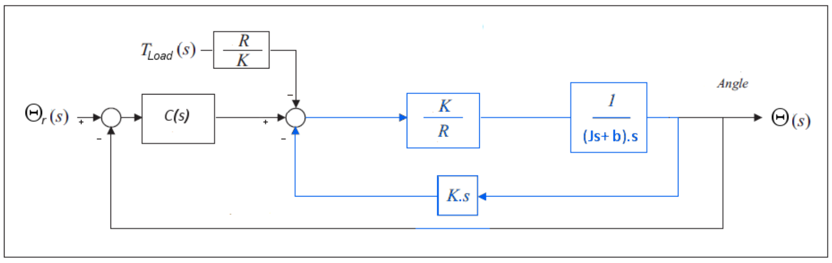


Figure 7

Simplifying the minor feedback transfer function in order to obtain an equivalent open-looptransfer function :

$$\frac{\frac{K}{R(J.s+b).s}}{1+\frac{K}{R(J.s+b).s}.K.s} = \frac{\frac{K}{R(J.s+b).s}}{\frac{R(J.s+b).K^2.s}{R(J.s+b).s}} = \frac{K}{R.J.s^2+(K^2+R.b).s} \tag{8}$$

The "blue" feedback can know be replace by the equivalent open-loop transfer function $\frac{K}{R.J.s^2+(K^2+R.b).s}$

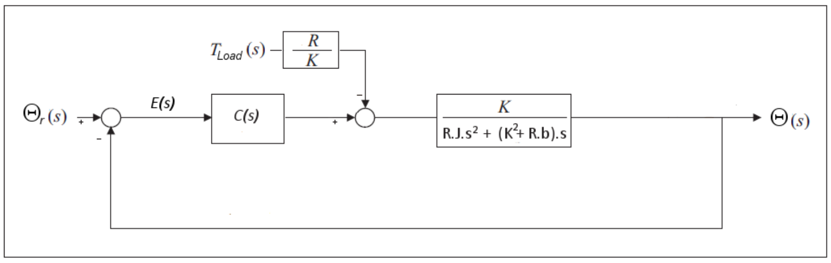


Figure 8 – Simplified closed-loop

2 - Parameters of the DC motor

The motor chosen to make the simulation is the Maxon DC motor 2140.937, with the following characteristics :

Rotor Inertia	Jm = 22.1g.cm² = 2.21E-6 Kg.m²
Terminal Inductance	L = 5.02 mH = 5.02E-3 H
Terminal Resistance	R = 41.5 Ω
Torque constant	Kt = 55.2E-3 N.m/A
Speed constant = 173 rpm/V	=> Ke = 55.2E-3 V/rad/sec , as we can verify, Ke = Kt
Mechanical time constant = 30 ms	Jm/b = 30E-3 => b = 2.21E-6/30E-3 = 7.36E-5 Nm/ (rad./s)

Electrical time constant = L/R = 5.02E-3/41.5 = 0.12 ms, as we can confirm, it can be neglected comparing with the mechanical constant 30 ms.

Replacing the motor characteristics in the block diagram of figure 8 :

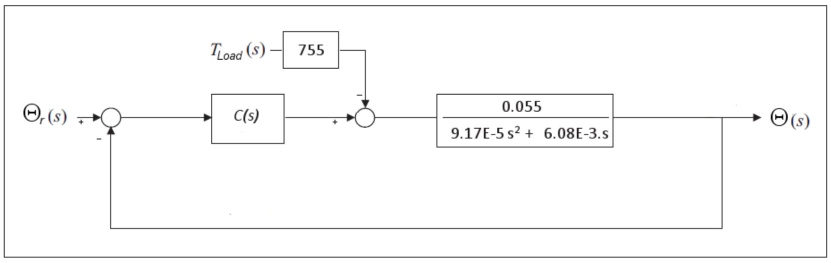


Figure 9

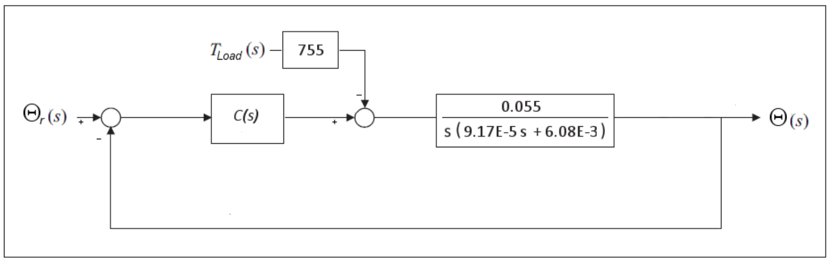


Figure 9a

3 - The project

The control will be provided with a PID controller, this one implemented by software using an Arduino Uno board. It will command and H bridge driver by a PWM signal. In the figure the schema of the system. The PWM is bipolar, 7 bits define the value of the PWM, and the 8º bit define the direction of the rotation. So the Value of the PWM will have a range between -127 and + 127.

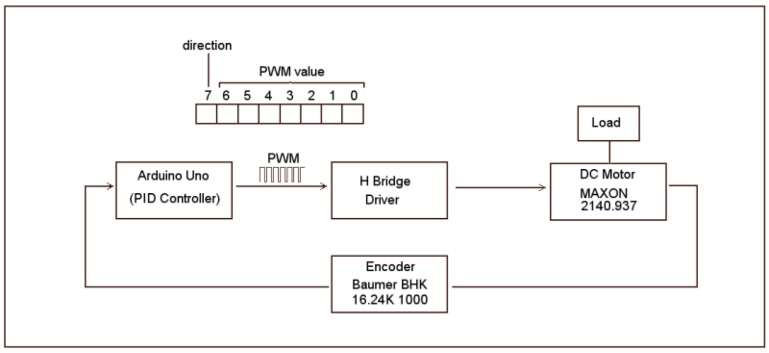


Figure 10

The incremental encoder is a Baumer BHK 16.24K 1000-B6-9, 1000 pulses per revolution, with operation speed max. 12000 rpm, higher than max motor speed.

The max continuous of the motor is 13.5E-3 Nm   (Trotor + Tload + Tviscous), Tload max = (13.5E-3 Nm – Trotor – Tviscous), but to simplify we will consider a Tload of 13.5E-3Nm. The stall torque of the DC motor is 31.9E-3 Nm.

Design requirements

We will want to be able to position the motor very precisely, thus the steady-state error of the motor position should be zero when given a commanded position. We will also want the steady-state error due to a constant disturbance to be minus to 0.01 rad. The other performance requirement is that the motor reaches its final position very quickly without no overshoot. In this case, we want the system to have a settling time of 40 ms for a position of 100 radians. If we simulate the reference input by a 100 units step input, then the motor position output should have:

- Settling time less than 40 milliseconds
- Null Overshoot.
- Steady-state error < 0.01rad with the presence of a step disturbance input with amplitude 13.5E-3.

The function of the PID controller is to add poles and zeros to the original open-loop transfer function, allowing us to reshape the root locus in order to place the closed-loop poles at the desired locations and to get desired response of our system. But first we choose to simulate in Matlab.

4 - Simulation

The following block diagram has been designed with Simulink/Matlab. The controller is PID parallel controller with a transfer function : Kp + Ki/s + Kds .
The values of Kp, Ki, and Kd will be changed in order to get the better response, i.e, quick response minus of 0.5s reach 100 radians, no overshoot, and an error of 0.02%. First, we change the parameter Kp to increase the settling time... saturation limit -127/127

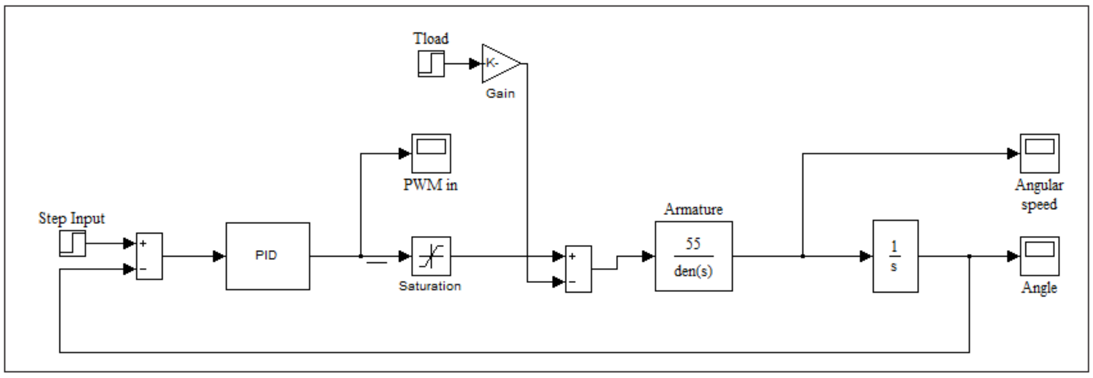


Figure 11

To find the optimize parameters for the PID controller, we must start with set only the parameter Kp (P controller). This parameter will reduce the rise time. The Ki parameter will be added if there is a steady-state error, and the Kd parameter must be added if there is oscillation.

We start with the analysis with no load.
In figure 12 we simulate with no controller (Kp = 1, Ki =0, Kd =0), and we observe a good stability with no steady-state error, but with a rise time too long. In figure 13 we reduce the rise time adding the Kp=120 parameter, but we get know an oscillation. Adding the Kd=1.1 parameter , the oscillation is eliminate as we can see in figure 14. In figure 15 there is a zoom in the steady-state domain, to verify a very low error.

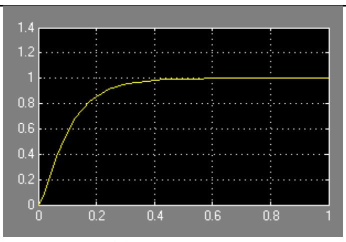


Figure 12 – C(s) = 1

Angle

No-Load
step Input = 1
Kp = 1
Ki = 0
Kd = 0

Settling time 0.6 seconds

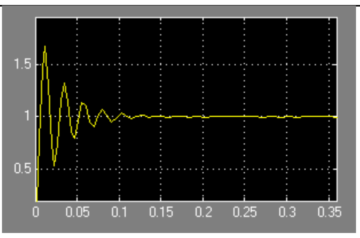


Figure 13 – C(s) = 120

Angle

No-Load
step Input = 1
Kp = 120
Ki = 0
Kd = 0

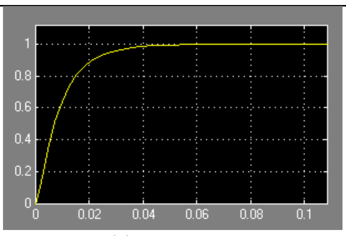


Figure 14 – C(s) = 120+1.1s

Angle

No-Load
step Input = 1
Kp = 120
Ki = 0
Kd = 1.1

Settling time 0.06 seconds

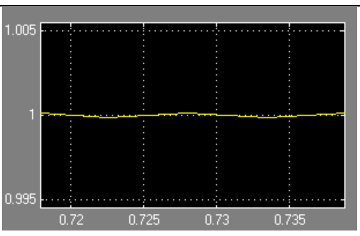


Figure 15 – erro with C(s) = 120+1.1s

Error

No-Load
step Input = 1
Kp = 120
Ki = 0
Kd = 1.1

If we increase the step input from 1 in previous analysis, to 200, we observe a linear rise time because of the controller saturation. In the block diagram of figure 11, a saturation box was inserted with the following parameters : negative saturation -127, positive saturation +127.

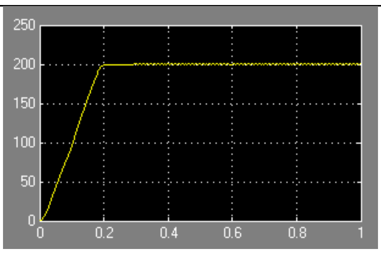
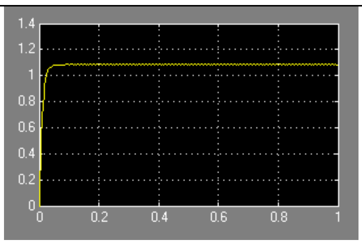


Figure 16

Angle

No-Load
step Input = 200
Kp = 120
Ki = 0
Kd = 1.1

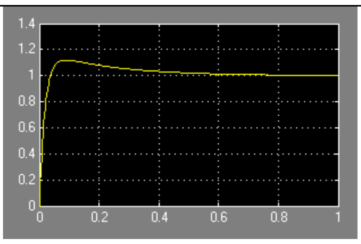
Next the analysis with a load of 13.5E-3 Nm, that´s the maximum continuous torque of our DC motor, and using the same previous parameters for the PD controller (Kp=120, Kd=1.1). We can observe that we have now a steady-state error. If we insert a Ki=500 parameter the steady state error is eliminate but we still have an unwanted response because of the overshoot.



Angle

Load 13.5E-3
step Input = 1
Kp = 120
Ki = 0
Kd = 1.1

Figure 17



Angle

Load 13.5E-3
step Input = 1
Kp = 120
Ki = 500
Kd = 1.1

Figure 18

In figures 19,20 and 21 the response with different amplitude of input step, and with the PD controller (Kp=120, Kd=1.1). The steady-state error seems to be the same in the 3 cases, around 0.1 rad.

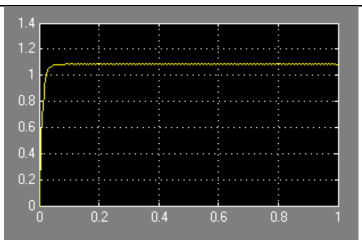


Figure 19 – Step 1

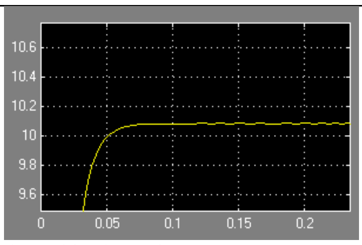


Figure 20 – Step 10

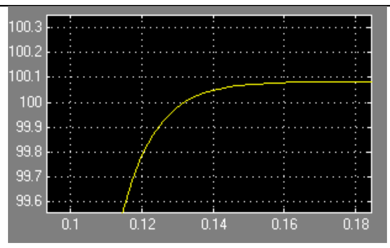


Figure 21 – Step 100

Adding the integral factor Ki/s to the controller, the error is eliminated in steady state but the overshoot is increased. We have to choose if we can have a fixed error of 0.1 rad, or if we can have an overshoot. It depends of the application, but for sure it´s better to have neither of them. Another kind of controller have to be dimensioned, like for example the serial PID controller $Kp . \frac{1+s.Ti}{s.Ti} . \frac{1+s.Td}{1+s.\alpha.Td}$. This controller has the advantage to easily control the position of its poles and zeros, that are easily identified analyzing the previous equation : two poles at positions -α.Td and 0, two zeros at -1/Td and at -1/Ti.

windup

if an integration factor Ki/s is added, the output will continue to increase or decrease as long as there is an error (difference between set point and measurement) until the output reaches its upper or lower limit, i.e., saturation. The result will be an excessive overshooting. The solution is to cut-off the integrator factor Ki/s when saturation is reached.

5 - The control of the project

Applying the formulas in annex A to the closed loopp block diagram of the system in figure 8, we obtain the Servo command and regulation equation (9) that allows to study the stability of the system :

$$\Theta(s) = \frac{C(s).\frac{0.055}{9.17E-5.s^2+6.08E-3.s}}{1+C(s).\frac{0.055}{9.17E-5.s^2+6.08E-3.s}} . \Theta r(s) - \frac{\frac{0.055}{9.17E-5.s^2+6.08E-3.s}}{1+C(s).\frac{0.055}{9.17E-5.s^2+6.08E-3.s}} . 755 . P(s) \qquad (9)$$

As well as the error transfer function :

$$E(s) = \Theta r(s) - P(s) = \frac{1}{1+C(s).\frac{0.055}{9.17E-5.s^2+6.08E-3.s}} . \Theta r(s) - \frac{\frac{0.055}{9.17E-5.s^2+6.08E-3.s}}{1+C(s).\frac{0.055}{9.17E-5.s^2+6.08E-3.s}} . 755 . P(s) \qquad (10)$$

C(s) = PD Controller = Kp + Kd.s = 120 + 1.1s

Error in steaty-state

$$1 + C(s) . \frac{0.055}{9.17E-5.s^2+6.08E-3.s} = 1 + (120+1.1s).\frac{0.055}{9.17E-5.s^2+6.08E-3.s} = 1 + \frac{6.6+0.065s}{9.17E-5.s^2+6.08E-3.s} = \frac{9.17E-5.s^2+6.08E-3.s+6.6+0.065s}{9.17E-5.s^2+6.08E-3.s} = \frac{9.17E-5.s^2+0.071s+6.6}{9.17E-5.s^2+6.08E-3.s}$$

Replacing in equation (10)

$$E(s) = \frac{1}{1 + C(s)\,.\frac{0.055}{9.17\text{E}-5.s^2 + 6.08\text{E}-3.s}}\,.\,\Theta r(s) - \frac{\frac{0.055}{9.17\text{E}-5.s^2 + 6.08\text{E}-3.s}}{1 + C(s).\,\frac{0.055}{9.17\text{E}-5.s^2 + 6.08\text{E}-3.s}}\,.\,755\,.\,P(s)$$

$$= \frac{1}{\frac{9.17\text{E}-5.s^2 + 0.071s + 6.6}{9.17\text{E}-5.s^2 + 6.08\text{E}-3.s}}\,.\,\Theta r(s) - \frac{\frac{0.055}{9.17\text{E}-5.s^2 + 6.08\text{E}-3.s}}{\frac{9.17\text{E}-5.s^2 + 0.071s + 6.6}{9.17\text{E}-5.s^2 + 6.08\text{E}-3.s}}\,.\,755\,.\,P(s) = \frac{9.17\text{E}-5.s^2 + 6.08\text{E}-3.s}{9.17\text{E}-5.s^2 + 0.071s + 6.6}.\,\Theta r(s) - \frac{41.5}{9.17\text{E}-5.s^2 + 0.071s + 6.6}P(s)$$

The error in steady state is equal to $\lim_{s\to 0} s.\,E(s)$

Root Locus

We can verify that all the fractions of equations (9) and (10) have a common denominator $\left(1 + C(s)\,.\,\frac{0.055}{9.17\text{E}-5.s^2 + 6.08\text{E}-3.s}\right)$.

By adding zeros or/and poles via the controller C(s), the root locus can be modified in order to achieve a desired response.
if we replace C(s) by Kc.C´(s) then

$$1 + C(s)\,.\,\frac{0.055}{9.17\text{E}-5.s^2 + 6.08\text{E}-3.s} = 1 + Kc.C'(s)\,.\,\frac{0.055}{9.17\text{E}-5.s^2 + 6.08\text{E}-3.s} \tag{11}$$

The root locus of equation (11) is analysed in next figure 22.

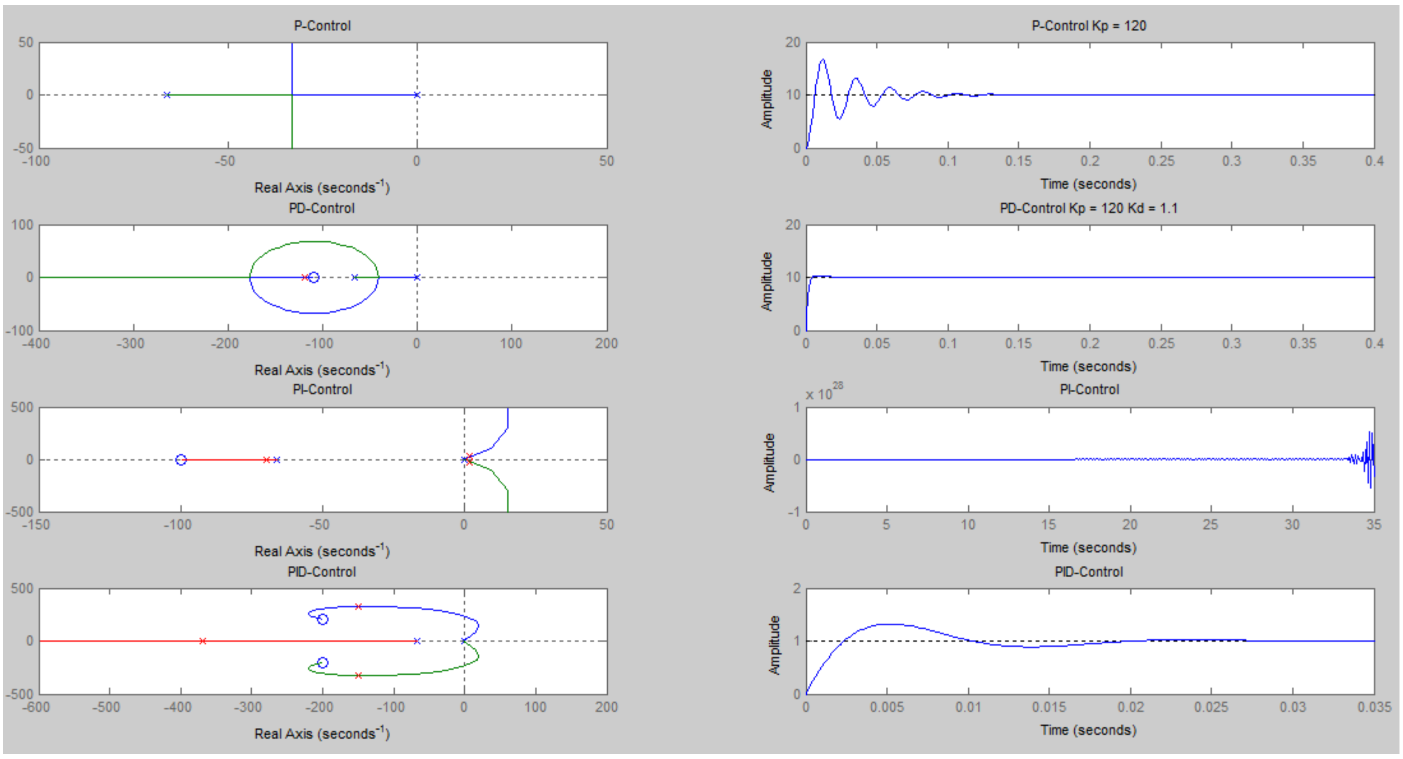


Figure 22 – Root locus and step response analysis, for different PID parameters

Conclusion

The requirements was not reached…
In addition, when a gear is inserted in an actuator system, backlash is experienced on its output due to the coupling between the cogwheels of the gear. This gives rise to nonlinearities that may lead to instability e_ects. For this reason, especially in high precision systems, direct-drive motors are used. Such motors may exert reasonable torques at low speeds, whence they do not need gears to drive the load. However, these motors may not be adopted for high power tasks, since the maximum torque exerted has physical limitations.

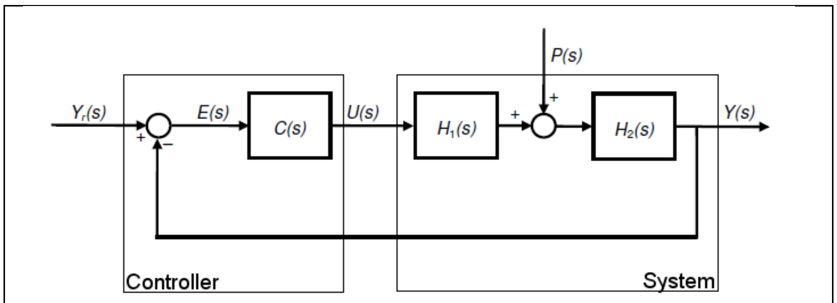
References :

http://ctms.engin.umich.edu/CTMS/index.php?example=MotorPosition&section=SystemModeling

http://ctms.engin.umich.edu/CTMS/index.php?example=MotorSpeed&section=SystemModeling

ANNEX

Unit feedback model



$$Y(s)=\frac{CH(s)}{1+CH(s)}Y_r(s)+\frac{H_2(s)}{1+CH(s)}P(s)$$

$$E(s)=\frac{1}{1+CH(s)}Y_r(s)-\frac{H_2(s)}{1+CH(s)}P(s)$$

Matlab code

```
% Comparison of Root-Locus and step response for different controllers type
% P, PD, PI, PID

% clear matlab memory and close all figures
clear all; close all;

% define motor transfer function, G
L = 5.02E-3; R = 41.5; J = 2.21E-6; B = 7.36E-5; K = 0.055;
numG = K; denG = [ (J*R) (R*B)+(K^2) 0];
G    = tf(numG, denG);

% P-control
numD = [120]; denD = [1]; D  = tf(numD, denD);
subplot(4,2,1); rlocus(series(D,G)); title('P-Control')
cltf = feedback(series(D,G), [1], -1);
hold on; plot(real(roots(cltf.den{:})), imag(roots(cltf.den{:})), 'rx'); hold off;
subplot(4,2,2); step(cltf*10,0.4); title('P-Control Kp = 120');  % amplitude step = 10, time 0.4 s

% PD-control
numD = [1.1 120]; denD = [1];  D  = tf(numD, denD);
subplot(4,2,3); rlocus(series(D,G)); title('PD-Control')
cltf = feedback(series(D,G), [1], -1);
hold on; plot(real(roots(cltf.den{:})), imag(roots(cltf.den{:})), 'rx'); hold off;
subplot(4,2,4); step(cltf*10,0.4); title('PD-Control Kp=120 Ki = 1.1');

% PI-control
numD = [1 100]; denD = [1 0]; D    = tf(numD, denD);
subplot(4,2,5); rlocus(series(D,G)); title('PI-Control')
cltf = feedback(series(D,G), [1], -1);
hold on; plot(real(roots(cltf.den{:})), imag(roots(cltf.den{:})), 'rx'); hold off;
subplot(4,2,6); step(cltf); title('PI-Control');

% PID-control
numD = conv([1 200+200j], [1 200-200j]); denD = [1 0];  D = tf(numD, denD);
subplot(4,2,7); rlocus(series(D,G)); title('PID-Control')
cltf = feedback(series(D,G), [1], -1);
hold on; plot(real(roots(cltf.den{:})), imag(roots(cltf.den{:})), 'rx'); hold off;
subplot(4,2,8); step(cltf); title('PID-Control');
```

PID Arduino Code

```
// calculate a motor speed for the current conditions
int motorSpeed = KP * error;
motorSpeed += KD * (error - lastError);
motorSpeed += KI * (sumError);

// set the last and sumerrors for next loop iteration
lastError = error;
sumError += error;
```