PID controller Implementation for QNET DC Motor

SAARAM ALI CHAUDHRY
BEE 12 C
SEECS 24

344924

AMINA BASHIR
BEE 12 C
SEECS 24

343489

SAAD BAKHTIAR
BEE 12 C
SEECS 24

341150

***Abstract*— The lab helps to understand the principles of PID control based on specific applications. Design a PID controller using MATLAB and analyze its performance in terms of stability, settling time, and steady-state error for a given application. Tune the three different gains of PID controller to get the desired response in each case.**

Keywords— QNET DC Motor , PID, overshoot ,settling time

I. Introduction

This Linear control systems play a crucial role in various engineering applications. The Proportional-Integral-Derivative (PID) controller is one of the most commonly used feedback controllers in linear control systems. Previous lab introduced us with basic design procedure using PIDs and this lab is its continuation, aimed at specific control design (motor speed, position control) using PIDs.

II. OBJECTIVES

A. *Understanding Theory of PID Controller*

First Design a PID controller using MATLAB and analyze its performance in terms of stability, settling time, and steady-state error for a given application.

B. *Calculating the Gain Variables for Controller*

Take a look at how the PID controller works in a closed-loop system using the schematic shown above. The variable (e) represents the tracking error, the difference between the desired output (r) and the actual output (y). This error signal (e) is fed to the PID controller, and the controller computes both the derivative and the integral of this error signal with respect to time. The control signal (u) to the plant is equal to the proportional gain (K_p) times the magnitude of the error plus the integral gain (K_i) times the integral of the error plus the derivative gain (K_d) times the derivative of the error.

C. *Maintaining the Specification for System*

The We will discuss the effect of each of the PID parameters on the dynamics of a closed-loop system and will demonstrate how to use a PID controller to improve a system's performance.

III. Design Problem 1 : Design of proportional controller for motor speed

Using the techniques learnt in lab 10 and the model of QNET DC motor found in lab 3, design a simple proportional controller for the speed control of the DC motor. The controller should meet the following specifications:

1. **%OS < 25%**
2. **Settling time is less than 0.5 second**

A. *MATLAB Code:*

```
clear
clc  num = [25]; den = [1 7.5]; sys= tf(num,
den);
proportional controller
rlocus(sys)
Kp = 0.3;
ctrl = zpk([],[],Kp);
sys_cl = feedback(series(ctrl,sys),1);
sys_param = stepinfo(sys_cl) ;
SSError = abs(1-dcgain(sys_cl)) ;
```

B. *Output:*

```
sys_param =

  struct with fields:

        RiseTime: 0.292934199550913
   TransientTime: 0.521609926045303
    SettlingTime: 0.521609926045303
     SettlingMin: 3.015002471326164
     SettlingMax: 3.333245657733601
       Overshoot: 0
      Undershoot: 0
            Peak: 3.333245657733601
        PeakTime: 1.406111963454989


SSError =

   2.333333333333333
```

Figure 1: Open-loop parameters for DC motor speed control

```
sys_param =

  struct with fields:

        RiseTime: 0.146467099775456
   TransientTime: 0.260804963022653
    SettlingTime: 0.260804963022653
     SettlingMin: 0.452250370698926
     SettlingMax: 0.499986848660040
       Overshoot: 0
      Undershoot: 0
            Peak: 0.499986848660040
        PeakTime: 0.703055981727505


SSError =

   0.500000000000000
```

Figure 2: Proportion-controlled parameters for DC motor speed

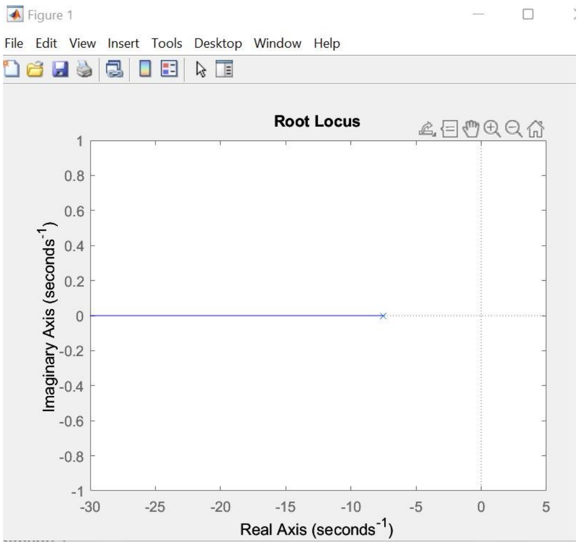


Figure 3: Left-sided root locus indicating system stability

C. Comments:

The overshoot is 0 < 25%.

Settling time is 0.26 sec < 0.5 sec

So the design constraints are met.

IV. Design Problem 3 : Design of PI

Design a PI controller for the speed control of the DC motor. The controller should meet the following specifications:

%OS<25

Settling time is less than 1 seconds

Zero steady state error for step input

A. MATLAB Code:

```
clear clc
num = [25];
den = [1 7.5]; sys= tf(num, den);
%PI controller design
display('Closed Loop System after PI Controller Design');
zero_loc = -15;
compensator = zpk(zero_loc,0,1);
rlocus(series(compensator,sys));
Kp= 1;
Ki = -Kp*zero_loc;
ctrl_p = zpk([],[],Kp);
sys_cl = feedback(series(ctrl,sys),1)
sys_param = stepinfo(sys_cl)
SSError = abs(1-dcgain(sys_cl))
step(sys_cl)
```

B. Output:

```
sys_param =

  struct with fields:

        RiseTime: 0.292934199550913
   TransientTime: 0.521609926045303
    SettlingTime: 0.521609926045303
     SettlingMin: 3.015002471326164
     SettlingMax: 3.333245657733601
       Overshoot: 0
      Undershoot: 0
            Peak: 3.333245657733601
        PeakTime: 1.406111963454989


SSError =

   2.333333333333333
```

Figure 4: Open-loop parameters for DC motor speed control

```
Closed Loop System after PI Controller Design

sys_cl =

      25 (s+15)
  ------------------
  (s^2 + 32.5s + 375)

Continuous-time zero/pole/gain model.


sys_param =

  struct with fields:

        RiseTime: 0.058725397312805
   TransientTime: 0.255119046965671
    SettlingTime: 0.255119046965671
     SettlingMin: 0.914399167802593
     SettlingMax: 1.075174548726860
       Overshoot: 7.517454872686025
      Undershoot: 0
            Peak: 1.075174548726860
        PeakTime: 0.138863593300562


SSError =

   4.440892098500626e-16
```

Figure 5: Proportion-controlled parameters for DC motor speed

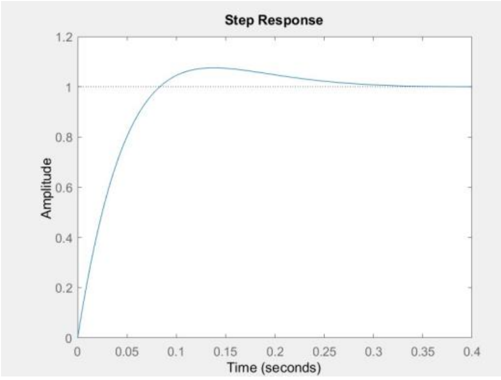


Figure 6: Step response

C. *Comments:*

The %OS = 7.51% (<25%)

Settling time = 0.255sec (<2 sec)

V. Design Problem 6 : Design and Implementation of a PD Controller for Motor Position

Design a PD controller for the position control of the DC motor, to meet the following specifications:

OS < 25%
Settling time is less than 0.5 sec

A. *MATLAB Code:*

```
clear clc
 num = [25]; den = [1 7.5 0]; sys= tf(num, den);

%open loop system properties sys_param = stepinfo(sys)
SSError = abs(1-dcgain(sys))

%PD controller design
display('Closed Loop System after PD Controller
Design'); zero_loc = -15;
compensator = zpk(zero_loc,[],1);
rlocus(series(compensator,sys));
Kd= 1;
Kp = -Kd*zero_loc;
ctrl_p = zpk([],[],Kp);
sys_param = stepinfo(sys_cl);
SSError = abs(1-dcgain(sys_cl)) ;
```

B. *Output:*

```
sys_param =

  struct with fields:

        RiseTime: NaN
   TransientTime: NaN
    SettlingTime: NaN
     SettlingMin: NaN
     SettlingMax: NaN
       Overshoot: NaN
      Undershoot: NaN
            Peak: Inf
        PeakTime: Inf


SSError =

   Inf
```

Figure 7: Open-loop parameters for DC motor speed control

```
Closed Loop System after PD Controller Design

sys_param =

  struct with fields:

        RiseTime: 0.058725397312805
   TransientTime: 0.255119046965670
    SettlingTime: 0.255119046965670
     SettlingMin: 0.914399167802591
     SettlingMax: 1.075174548726858
       Overshoot: 7.517454872685914
      Undershoot: 0
            Peak: 1.075174548726858
        PeakTime: 0.138863593300562


SSError =

   2.220446049250313e-16
```

Figure 8: Proportion-controlled parameters for DC motor speed

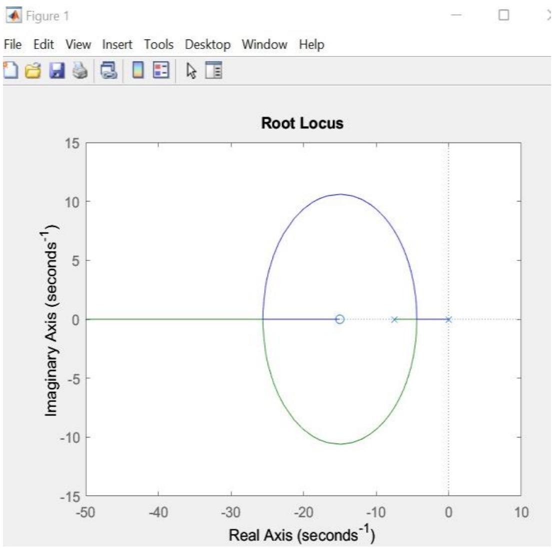


Figure 9: Left-sided root locus indicating system stability

C. *Comments:*

- Overshoot = 7.51 (<25%)
- Settling time = 0.255sec (<0.5sec)

• Although there are no restrictions on steady-state error, still it is very close to zero.

<u>*So the design constraints are strictly achieved by gain adjustment.*</u> Steady state error is very small (4.44e-16) which can be approximated to zero. <u>*So the design constraints are strictly met.*</u>

VI. Design Problem 7 : Design and Implementation of a PID Controller for Motor Position

Note that in the transfer function of motor position vs voltage there is a pole at the origin. Therefore, the system type is 1. Consequently, the closed loop system will have zero steady state error for a step input. If the requirement is to have zero steady state error for step input, then we do not need a PI controller.

If the input is a ramp, then there will be a non-zero steady state error. In that case we may have a PI compensator to increase the steady state error. Design a PID controller to meet the following specifications.

%OS < 25

Settling time is less than 0.5 seconds

Zero steady state error for ramp input

A. MATLAB Code:

```
clear clc
num = [25]; den = [1 7.5 0]; den1 = [1 7.5 0 0];
sys= tf(num, den); sys1= tf(num, den1);

%% PID controller design
display('Closed Loop System after PID Controller Design'); zero_loc_pi = -1; zero_loc_pd = -15;
compensator = zpk([zero_loc_pi, zero_loc_pd],0,1);
rlocus(series(compensator,sys))
Kd= 1;
Kp = -(zero_loc_pd+zero_loc_pi)*Kd; Ki = zero_loc_pd*zero_loc_pi*Kd;
ctrl_p = zpk([],[],Kp);

ramp_mod = tf([1],[1 0]); sys_cl1 = series(sys_cl,ramp_mod);
SSError_Ramp = abs(1-dcgain(sys_cl))
```

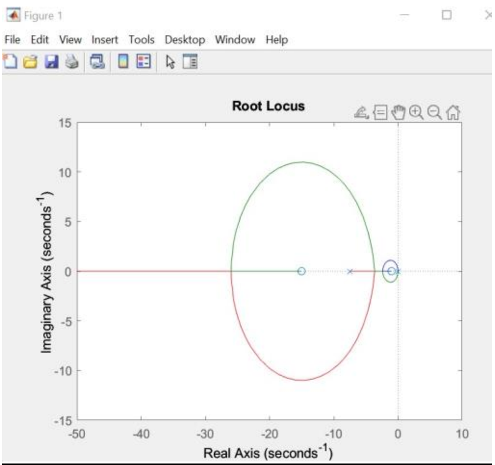
B. Output:

```
sys_param =

  struct with fields:

        RiseTime: NaN
   TransientTime: NaN
    SettlingTime: NaN
     SettlingMin: NaN
     SettlingMax: NaN
       Overshoot: NaN
      Undershoot: NaN
            Peak: Inf
        PeakTime: Inf


SSError =

   Inf
```

<u>*Figure 10: Open-loop parameters for DC motor speed control*</u>

```
sys_param =

  struct with fields:

        RiseTime: 0.055981076240986
   TransientTime: 0.311129475910152
    SettlingTime: 0.311129475910152
     SettlingMin: 0.909477171719281
     SettlingMax: 1.103391783348220
       Overshoot: 10.339178334822051
      Undershoot: 0
            Peak: 1.103391783348220
        PeakTime: 0.137508216319262


SSError_Ramp =

   1.110223024625157e-15
```

<u>*Figure 11: Proportion-controlled parameters for DC motor speed*</u>



<u>*Figure 12: Left-sided root locus indicating system stability*</u>

C. Comments:

%OS = 10.339% (<25%)

Settling time = 0.311 sec (< 0.5 sec)
Steady state error is 1.11e-15, which is very small value and can be approximated as zero *<u>So the design constraints are met using gain adjustment.</u>*

CONCLUSION

This lab was a valuable learning experience for understanding the principles and practical applications of PID controllers. We learned how to implement various types of controllers using MATLAB and obtained specific values for overshoot percentage, settling time, and steady-state error for each type of controller. Through these exercises, we also gained a deeper understanding of the importance of choosing the appropriate controller type for a given system, and the impact of controller parameters on system response. Overall, this lab provided a hands-on opportunity to apply the theoretical concepts of control systems and deepen our understanding of PID control.

ACKNOWLEDGMENT

We acknowledge individuals who provided technical assistance, such as lab technicians or research assistants, or who provided equipment or materials for the lab.

REFERENCES

[1] https://ctms.engin.umich.edu/CTMS/index.php?example=Introduction&section=ControlPID

[2] https://www.youtube.com/watch?v=Jp06lFFvxWc

[3] https://slideplayer.com/slide/18034641/