DC Motor Speed Controller

Jared Woelfel – 01570415 – 12 May 2021

Introduction

The purpose of this project was to test and develop a closed-loop motor system that used a tachnogenerator to provide feedback. Once the open-loop transfer function of the entire system was approximated, both uncompensated and compensated digital systems were designed in MATLAB to allow the user to control the motor setpoint and see the measured RPM. The response characteristics of each system were recorded and compared to demonstrate the benefits of PI compensation.

Solution

The first step of the design process was to approximate the open-loop transfer function of the motor feedback system. This was achieved using MATLAB to cycle the power supply between 6 and 12 volts and recording the step response of the motor feedback system on the oscilloscope as seen in Figure 1. The peak time and overshoot percentage (calculated using Eqn 1) of the step response were used to approximate the transfer function of the open-loop system using Eqn 2 and the results were recorded in Table 1.

In order to better understand the function of the digital control system, a block diagram of the entire system was created and recorded in Figure 2. The system consisted of an RPM setpoint controlled by the user in MATLAB that was converted to a voltage using the tachnogenerator calibration equation. This signal was fed into a summer that subtracted the feedback of the digital multimeter reading and then sent this error signal to a digital controller. The control signal output by the controller was then sent to the power supply that drove the motor which drove the tachnogenerator and supplied a voltage for the digital multimeter to read. The open-loop transfer function approximated in Table 1 represented the system response from the power supply to the digital multimeter.

An uncompensated digital motor controller was designed to allow the user to input a setpoint in RPM. The controller added an offset equal to the setpoint voltage to the error signal to produce the control signal. This prevented the control signal from going to 0V as the error signal diminished. The MATLAB code used to implement this digital control system was recorded in Figure 3.

The uncompensated digital motor controller was then tested to analyze the controller performance. A setpoint of 255 RPM was chosen to test the control system because it was in the center of the motor's operational region according to the tachnogenerator calibration plot. The step response of the system was recorded using the oscilloscope in Figure 4 and the MATLAB plot in Figure 5. The system's response parameters were recorded in Table 2.

A digital PI compensator was chosen for implementation because of its ability to drive the steady state error to zero and improve the transient response of the system while maintaining simplicity in design. In an analog system, PI compensation takes the form of integration, but for digital systems, PI compensation can be implemented by summing the previous control signal and a proportion of the error signal. This concept was used to produce the digital PI compensator seen in the MATLAB code of Figure 6. The gain applied to the error signal was determined using

ultimate cycle tuning where the gain was slowly increased until the system reached the edge of stability and then was reduced by a small margin to retain stability but produce a fast response.

The PI digital motor controller was then tested to analyze the controller performance. A setpoint of 255 RPM was chosen to test the controller. The step response of the system was recorded using the oscilloscope in Figure 7 and the MATLAB plot in Figure 8. The system's response parameters were recorded in Table 3. This controller was clearly more effective. Although both systems had similar rise times, the compensated system had a significantly reduced settling time and percent overshoot as well as no steady-state error. This controller was superior in nearly every way to the uncompensated system.

Finally, the PI digital motor controller was tested using a disturbance simulated by increasing the load on the motor. The motor was allowed to reach its setpoint at 255 RPM. Next, pressure was added to the belt between the motor and tachnogenerator. The pressure was then released. The system's response was recorded using the oscilloscope in Figure 9 and the MATLAB plot in Figure 10. The load was added around measurement index 25 and removed around measurement index 60. The controller had no problem adjusting the power supply voltage to compensate for the load (as seen by the short decrease in RPM followed by a return to setpoint). However, removing the load caused oscillations in RPM because the significant increase in power supply voltage initially required to overcome the load was no longer required once the load was removed. The controller took longer to return to the setpoint after removing the load than when the load was first applied.

Conclusion

One issue with the PI compensated system implemented in MATLAB was that the gain value for feedback was extremely sensitive to small changes and was set near the limit of stability. Therefore, some system trials resulted in the control signal voltage going negative which caused the entire system to fail without warning. The nearly unstable gain value might also have contributed to the major oscillations resulting from removing the motor load. Additionally, to improve the reliability and performance of the system, a faster data acquisition method might be used to read the tachnogenerator voltage because MATLAB could only gather the digital multimeter reading at a rate of several milliseconds and the sampling period was never consistent.

Appendix

Equations:

All equations used came from [1].

$$OS\% = \frac{peak\ voltage - steady\ state\ voltage}{steady\ state\ voltage} * 100 = \frac{4.2V - 3.6V}{3.6V} * 100 = 16.67\% \qquad \text{Equation 1}$$

$$H(s) = \frac{\omega_n^2}{s^2+2\zeta\omega_n s+\omega_n^2}\ , where\ \zeta = \frac{|\ln(OS)|}{\sqrt{\pi^2+\ln^2(OS)}}\ and\ \omega_n = \frac{\pi}{t_p\sqrt{1-\zeta^2}} \rightarrow\ \zeta = \frac{|\ln(0.1667)|}{\sqrt{\pi^2+\ln^2(0.1667)}} = 0.4954, \omega_n = \frac{\pi}{88.5m\sqrt{1-0.4954^2}} =$$
$$40.86, H(s) = \frac{40.86^2}{s^2+2*0.4954*40.86*s+40.86^2} = \frac{1670}{s^2+40.49s+1670} \qquad \text{Equation 2}$$

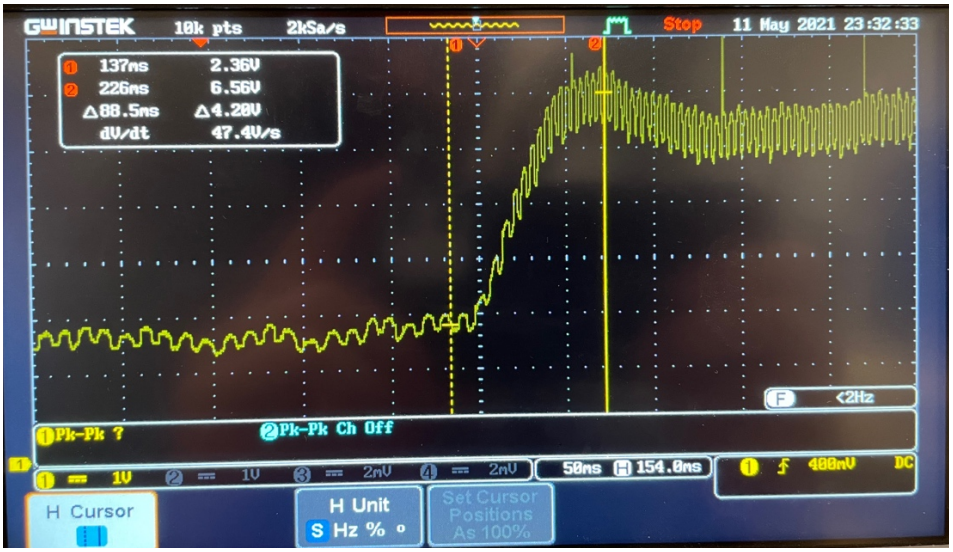

Figure 1: Open-Loop Transfer Function Test

Table 1: Open-Loop Transfer Function Test Results

Peak Time	Percent Overshoot	System Type	Response Type	Open-Loop Transfer Function
88.5ms	16.67%	0	Underdamped	$H(s) = \frac{1670}{s^2 + 40.49s + 1670}$

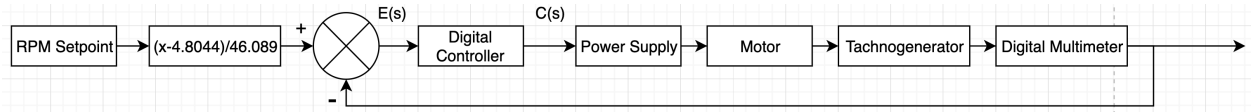

Figure 2: Digital System Block Diagram

```
%% Uncompensated System
clear i
clear RPMSet
clear Vout
clear errorSig
clear VoutSet
clear RPMMeasured
clc

GWPSOutputOn(PSid)
GWPSSetVoltage(PSid,0)
GWPSSetCurrent(PSid,3)
RPMSet = 255
VoutSet = (RPMSet-4.8044)/46.089
i = 1;
while(i < 60)
Vout(i) = GWDMMRead(DMMid);
errorSig(i) = VoutSet - Vout(i) + VoutSet;
GWPSSetVoltage(PSid,errorSig(i));
RPMMeasured(i) = (Vout(i)*46.089)+4.8044
i=i+1;
end

GWPSOutputOff(PSid)

plot(RPMMeasured);xlabel('Measurement Index (i)');ylabel('Measured RPM');
title('Uncompensated System Test for 255 RPM Setpoint')
```

Figure 3: MATLAB Code for Uncompensated Digital Controller

Figure 4: Uncompensated Digital Controller Step Response (Voltage)

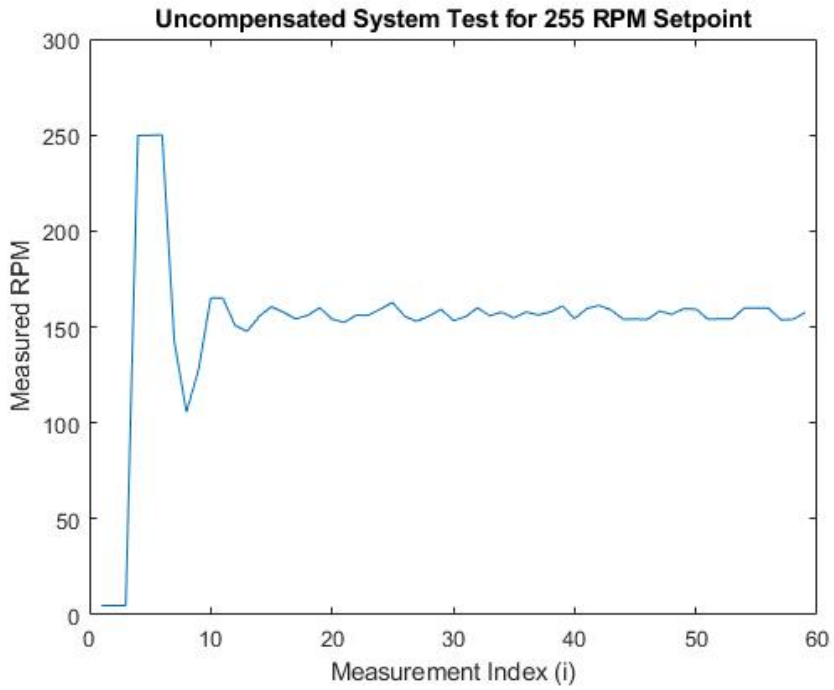

Figure 5: Uncompensated Digital Controller Step Response (RPM)

Table 2: Uncompensated Digital Controller Step Response Parameters

System Type	Response Type	Steady State Error	Rise Time	Percent Overshoot	Settling Time
0	Underdamped	-39%	50ms	100%	1.80s

```
%% Compensated System
clear i
clear RPMSet
clear Vout
clear errorSig
clear VoutSet
clear VPS
clear RPMMeasured
clc

GWPSOutputOn(PSid)
GWPSSetVoltage(PSid,0)
GWPSSetCurrent(PSid,3)
RPMSet = 255
VoutSet = (RPMSet-4.8044)/46.089
i = 2;
VPS(1) = VoutSet;
errorSig(1) = VoutSet;
GWPSSetVoltage(PSid,VoutSet)
while(i < 200)
Vout(i) = GWDMMRead(DMMid);
errorSig(i) = VoutSet - Vout(i);
VPS(i) = (((0.975)*errorSig(i))+VPS(i-1));
GWPSSetVoltage(PSid,VPS(i));
RPMMeasured(i) = (Vout(i)*46.089)+4.8044
i=i+1;
end

GWPSOutputOff(PSid)
plot(RPMMeasured);xlabel('Measurement Index (i)');ylabel('Measured RPM');
title('Compensated System Test for 255 RPM Setpoint')
```

Figure 6: MATLAB Code for PI Compensated Digital Controller

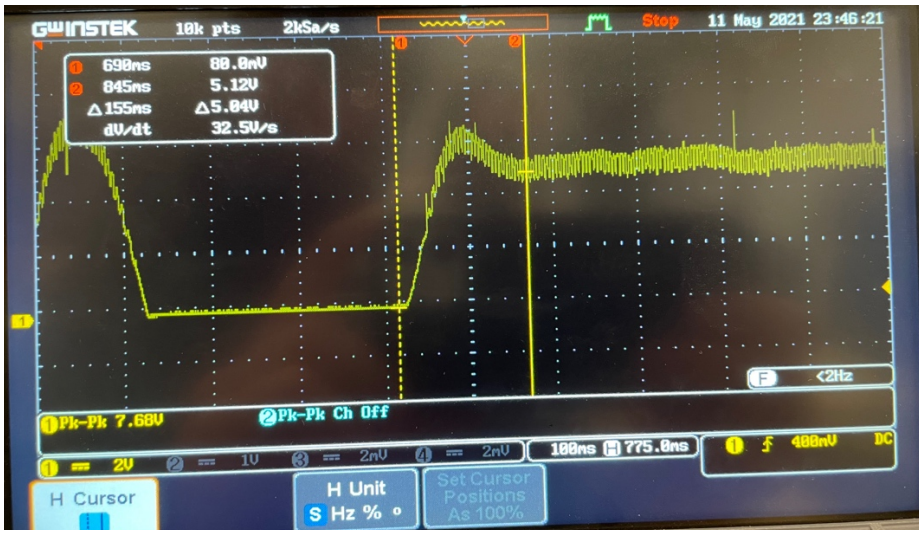

Figure 7: PI Compensated Digital Controller Step Response (Voltage)

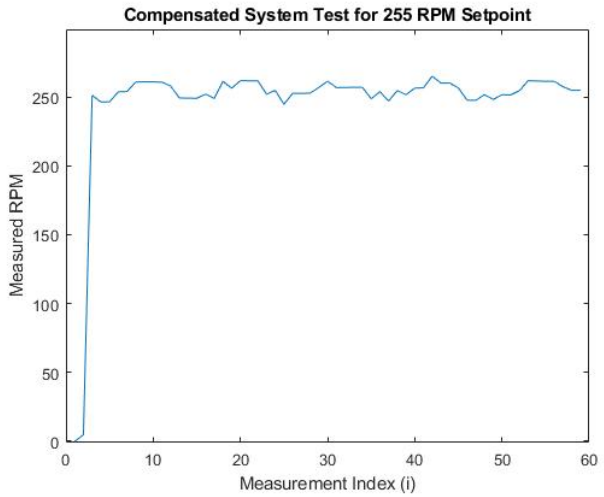

Figure 8: PI Compensated Digital Controller Step Response (RPM)

Table 3: Uncompensated Digital Controller Step Response Parameters

System Type	Response Type	Steady State Error	Rise Time	Percent Overshoot	Settling Time
1	Underdamped	0%	60ms	20%	155ms

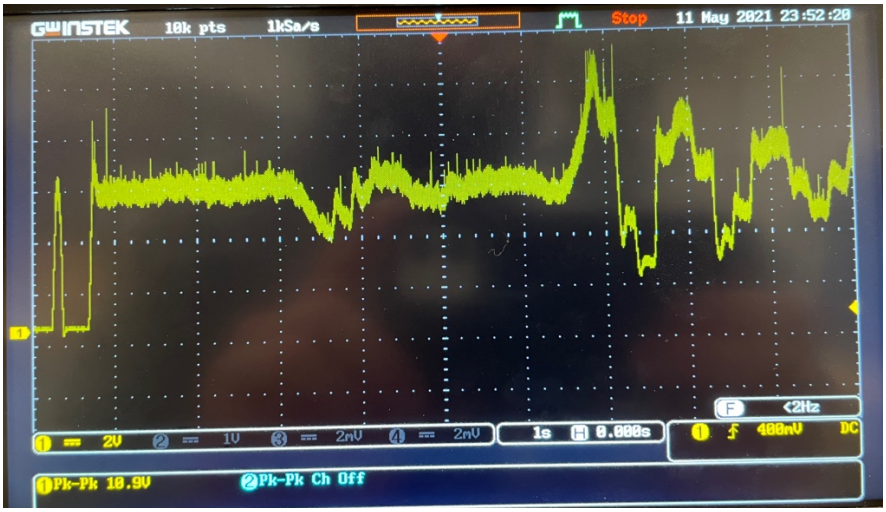

Figure 9: PI Compensated System Disturbance Response (Voltage)

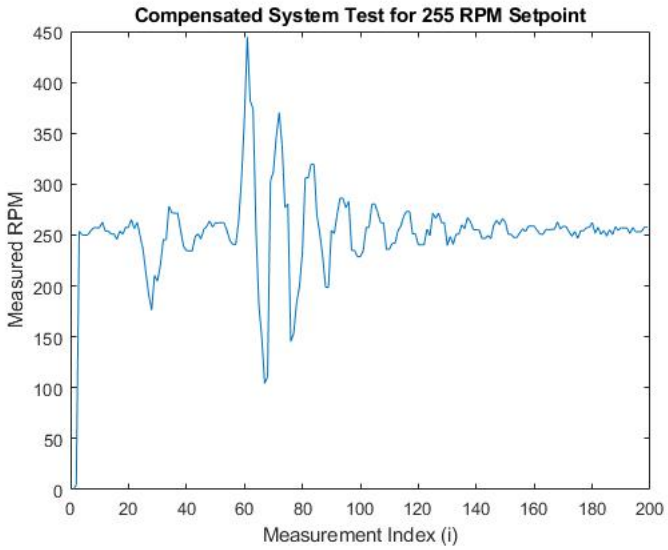

Figure 10: PI Compensated System Disturbance Response (RPM)

Reference

[1] N. S. Nise, *Control Systems Engineering*, 7th ed. Nashville, TN: John Wiley & Sons, 2013.