Laboratory Assignment 5
Digital Velocity and Position control of a D.C. motor

2.737 Mechatronics
Dept. of Mechanical Engineering
Massachusetts Institute of Technology
Cambridge, MA-02139

1 Topics

- Motor modeling
- Proportional, PI velocity control
- Implementation of Anti-Windup
- Proportional position control
- Lead-Lag compensation
- Velocity minor loop / Position major loop

2 Equipment

- PC 486 with ATMIO-16 card
- Oscilloscope
- Function generator
- D.C. motor
- Power amplifier
- Protoboard

3 Introduction

In this laboratory, digital control of a D.C. motor will be studied. Each station is equipped with a D.C. motor (with an integral tachometer to measure angular velocity), a single turn potentiometer to measure angular position and a power amplifier to drive the motor. First, we will develop a model for the motor. This model will then be used to perform velocity and position control experiments on the motor. For the purpose of this lab, controller design will be done using continuous time methods. Discrete time controllers will be obtained from these by using either backward difference or ZOH equivalents.

Initial experiments will focus on velocity control. A proportional controller will be implemented first. Resulting steady state errors are observed and the effect of increasing gain on steady state error is seen. This is followed by implementation of a PI controller to remove steady state errors in response to a step input. Problems with integral windup are observed and an anti-windup scheme is implemented. Next, step and frequency response specifications are met by designing a controller in Matlab and implementing it.

Next, position control of a D.C. motor will be studied. Initially, a proportional controller will be implemented. The effect of increased gain on servo stiffness is observed. Lead-Lag compensators are designed using Matlab and implemented on the actual hardware and the performance studied.

Finally, a velocity minor loop/position major loop controller will be implemented. Proportional gains for both loops are selected based on frequency domain criteria. Note that the structure of the controller is much like that in state feedback control with velocity and position as state variables.

4 D.C. Motor model

Each station has a D.C. motor and a power amplifier. Please follow a few precautions before powering up your setup. The power supply to the power amplifier is from the same Tektronix supply that you used in lab 4. Make sure that the power supply is operated in series as in lab 4 and that the current limit on both supplies (both positive and negative) is set to the maximum. Also, the voltage rails to the power amplifier should not exceed $\pm$ 17 volts. A good practice would be to turn the master voltage knob all the way to zero before switching the power supply on and then slowly bring it up to 15 volts. Note that as in lab 4, exceeding the voltage rails specified could damage the power amplifier. Please exercise caution.

The D.C. motor has a tachometer to measure velocity and a potentiometer to measure angular position. The motor parameters are :

$$
\begin{aligned}
R &= 7.5\Omega \\
L &= 5.55\text{mH} \\
\Rightarrow \tau_e &= \frac{L}{R} = 0.7\text{ms} \\
K_T &= 0.024\text{Nm/A} \\
J &= 1.5 \times 10^{-5}\text{Kgm}^2 \text{ (referred to motor shaft)} \\
K_{tach} &\approx 0.023\text{V/rad/sec} \\
K_{pot} &\approx 0.53\text{V/rad (of motor)}
\end{aligned}
$$

$$
\begin{aligned}
\text{No. of teeth on pinion} &= 36 \\
\text{No. of teeth on driven gear} &= 216 \\
\text{Gearing ratio, N} &= 6 \\
\Rightarrow \tau_m &= \frac{JR}{K_T^2} \approx 200\text{ms} \\
\Rightarrow \frac{\omega_m}{V_m} &= \frac{1}{K_T(\tau_m s + 1)}
\end{aligned} \tag{1}
$$

An appropriate connection of the motor setup is shown in Figure 1. For the purposes of this lab, the reference signal r will be generated by the function generator in some sections. Connect the function generator to AD channel 0, the tachometer signal to AD channel 1 ,and the potentiometer signal to AD channel 2. Also, compute the control output u and send it out on DA channel 0 to the input of the power amplifer.

Please make sure the potentiometer is wired correctly to operate at ± 10 Volts. To do this, we will use the ± 15 Volt power source from the protoboard, and use a pair of resistors to reduce the voltage to ± 10 Volts. The typical value of the potentiometer resistance is about $4.7k\Omega$, the pair of additional resistors can be chosen as $1.2k\Omega$. (Be very careful, don't connect the wiper to any other voltage source or ground, the short circuit between the wiper and ± 10 Volts during wrap-around will damage the potentiometer permanently.)

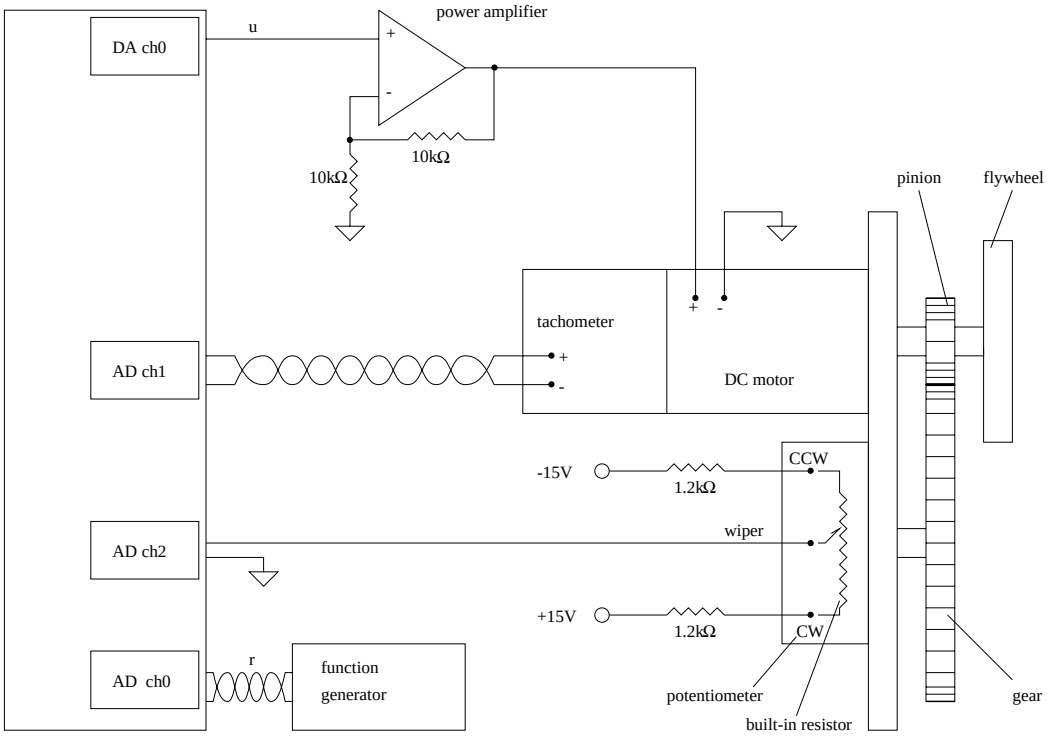

Figure 1: The connection for the DC motor setup

5 Velocity control

We will use the motor model developed in the previous section for implementing velocity control of the D.C. motor. The tachometer output is used to implement feedback. Remember to include the amplifier and tachometer gain when designing the controller. First, we will implement the simplest controller, *i.e.,* Proportional control. We will observe the effects of gain and sampling time on the step response of the motor.

Next, we will implement PI control which removes the steady state errors (to a step input) that were present with a proportional controller. The block diagram of the motor-controller setup implementing a PI controller is given in Figure 2.

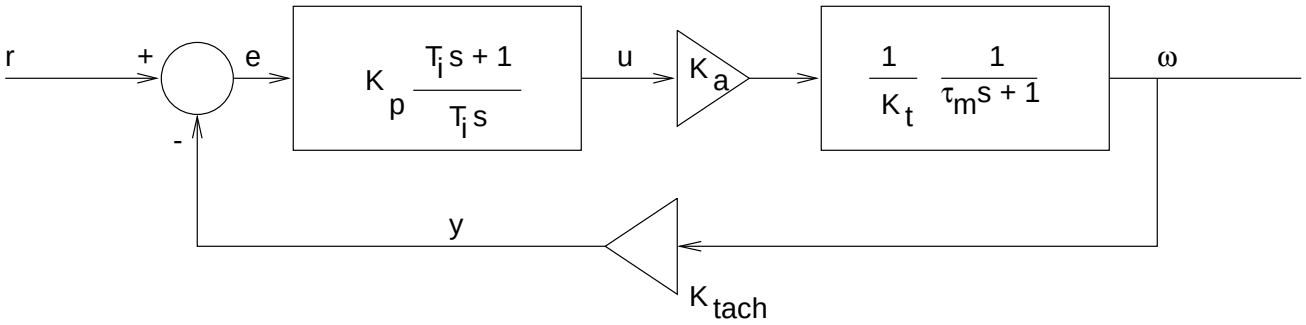

Figure 2: PI velocity control of the DC motor

5.1 Controller Design

A proportional controller is given by

$$D(s) = K_p \tag{2}$$

while a PI controller is given by

$$\begin{aligned} D(s) &= K_p \left(1 + \frac{1}{T_i s}\right) \\ &= K_p \left(\frac{T_i s + 1}{T_i s}\right) \end{aligned} \tag{3}$$

The proportional gain K_p and the integral or reset time T_i are chosen to meet specifications on the response. For the purpose of this lab, we will design controllers based on step and frequency response specifications. Also, we will implement all controllers with a sampling interval of 1 ms which is fast enough for implementation of digital controllers by emulating analog designs for the D.C. motor under consideration.

5.2 Controller Implementation

The command signal for a PI controller using a backward difference scheme is given by

$$u(k) = K_p e(k) + S_2(k-1) + \frac{K_p T_s}{T_i} e(k) \quad (4)$$

where

$$\begin{aligned} K_p &= \text{Proportional gain} \\ T_i &= \text{Reset time} \\ T_s &= \text{Sampling interval} \\ S_2(k) &= S_2(k-1) + Ce(k) \\ C &= \frac{K_p T_s}{T_i} \end{aligned}$$

Collecting terms,

$$u(k) = Ae(k) + S_2(k-1) \quad (5)$$

where

$$A = K_p + \frac{K_p T_s}{T_i}.$$

The values A and C can be precomputed. A code fragment to implement the controller in an interrupt service routine is then given by

```
ad = adin (1);
error = ref - ad;
command = A*error + sum;
daout (0, command);
sum += C*error;
```

The above code fragment implements the basic PI scheme without any anti-windup measures. Note that you can also use the ZOH equivalent to implement the digital controller. To find the ZOH equivalent of the controller, use the `c2dm (num, den, ts, 'zoh')` command in Matlab. This will give you the polynomial to implement the difference equation for your controller.

Note that the presence of an integrator in your control law can lead to *Integrator windup*. This can be noticed if you cause the error to accumulate in spite of saturation of the actuators. The integrating action of the controller keeps accumulating the error and leads to large overshoots. You can observe this by doing PI control of the motor and holding the flywheel with your hand. Be careful not to get yourself hurt while doing this (*i.e.*, *Keep your fingers out of the gear mesh*). When you finally release the flywheel, you should be able to see a rather large overshoot. For the purpose of observing windup, generate the step reference at the computer itself rather than using the signal generator. Look at the program `expt5_1.c` for an example. Note that the size of the overshoot is dependent on the time for which you hold the flywheel stationary ! You can implement anti-windup measures very easily by turning off the integrator when the actuators saturate (in our case - when the control output exceeds the ± 10 volts range). Please add anti-windup to your controller and show us the difference with and without this term.

6 Position control

Position control is in some sense more difficult than velocity control, because you can cause the system to go unstable rather easily due to the presence of the additional integrator in the transfer function. To implement position control, feedback is from the potentiometer. The potentiometer is a one-turn potentiometer with its output varying from -10 to $+10$ volts. There is a wrap around at either end of the range. What happens if you try to run through this wrap-around?

Remember to include the speed reducer gain in your model for the position control problem. Note that the motor angle is -6 times the potentiometer angle, which is what you measure. In your model, please refer all elements to the motor shaft.

6.1 Controller Design

A proportional controller is given by

$$D(s) = K_p \tag{6}$$

and is implemented in the same manner as for velocity control.

Lead lag compensators are designed and implemented to provide better control. A lead compensator is given by

$$D(s) = K_{lead}\frac{s+z_a}{s+p_a}; p_a > z_a \tag{7}$$

A lag compensator is given by

$$D(s) = K_{lag}\frac{s+z_b}{s+p_b}; p_b < z_b \tag{8}$$

A lead-lag compensator is then given by

$$D(s) = K_{lead} \times K_{lag} \times \frac{s^2 + (z_a+z_b)s + z_a z_b}{s^2 + (p_a+p_b)s + p_a p_b} \tag{9}$$

Compensator parameters are chosen to satisfy the system requirements. And the corresponding discrete-time controller can obtained using the methods described earlier. Note that the pole p_b is sometimes placed at $s = 0$ to perform as an integrator.

6.2 Controller Implementation

The controller is implemented in a similar way as for velocity control. Notice the effect of increasing gain on the servo stiffness. Also, notice the effect of the integral term when you implement the lead-lag compensator. Do you see chatter ?

7 State feedback

Having implemented both position and velocity feedback, we can now combine both to implement state feedback. We consider the problem of controlling the position of the motor shaft with both position and velocity feedback as shown in Figure 3. We saw that proportional control by itself led to

very poor performance due to the negligible amount of damping in the system. Lead compensation was used to add damping to the system - adding derivative control has the same effect. In fact, PD control can be seen as an extreme form of lead compensation. However, we observed chatter (on some of the setups at least) when performing lead and lead-lag compensation.

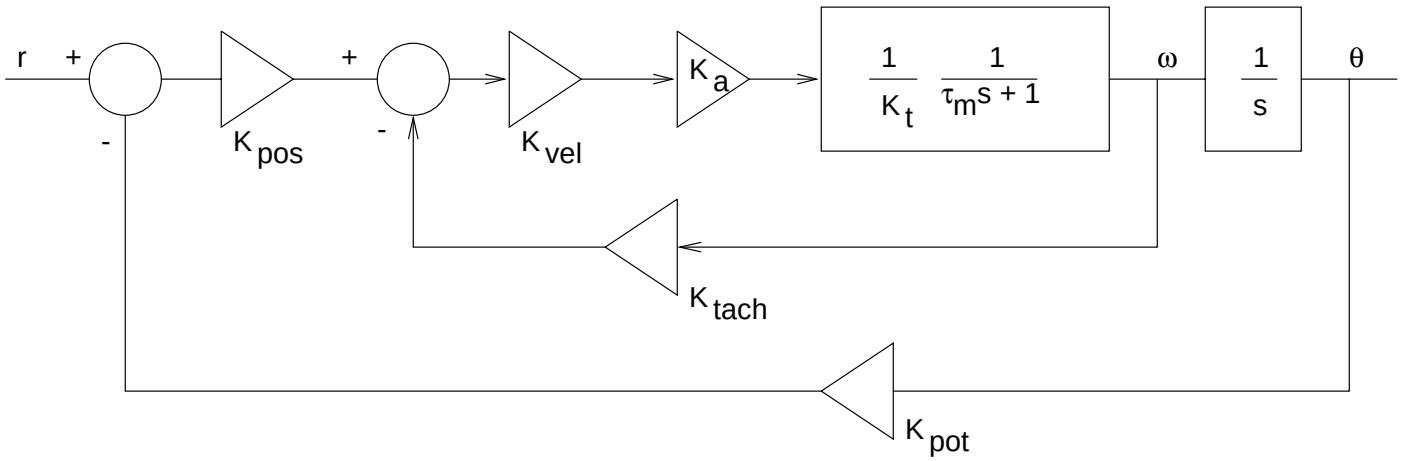

Figure 3: Velocity minor loop Position major loop

With velocity feedback however, we use the tachometer to obtain the derivative of the position, *i.e.,* velocity instead of physically differentiating the output position. Note that differentiation amplifies noise effects. The presence of backlash in the gears leads to chatter. Velocity feedback however, leads to an increased damping ratio without affecting the undamped natural frequency of the system and hence leads to better control performance. A similar controller can also be designed using pole placement techniques (see the Matlab functions `acker` and `place`) for a state variable model.

We will implement this with a velocity minor loop and a position major loop. The velocity minor loop is usually designed with a crossover on the order of 3 to 10 times higher in frequency than the position loop.

8 Laboratory procedure

1. Some programs for use in this lab are in the directory `c:\courses\2.737\lab5`. Change to this directory and run the program `expt5_1.c`. This implements proportional velocity control of the D.C. motor. The program accepts a proportional gain input by the user. Run the program for different value of proportional gain and observe the effects on steady state error. Use a sampling interval of 1 ms. Suggested proportional gain values range from 1 to 20.

2. Design a proportional controller for closed loop system time constant τ = 50 ms in Matlab (Be sure to include the power amplifier gain in your model) and implement it. Collect step response data and compare it with the predicted step response in Matlab. Use the program `expt5_1.c` to implement your controller. Note that the tachometer voltage has a lot of

ripple on it which is not predicted by Matlab. These are not oscillations due to your controller, they are due to the tachometer itself !

3. Design a PI controller for
   $\zeta = 0.5$
   $\omega_n = 30$
   Again, compare with Matlab. Initially write your program and make your hardware connections such that the program generates a reference voltage on the same lines of `expt5_1.c`. It might help to make a copy of the program `expt5_1.c` in your directory and modify it. Hold the flywheel with your hand and notice the large overshoots when you release it. Implement anti-windup measures as suggested and run the program. Again, hold the flywheel with your hand. Notice the greatly improved performance. Explain the difference.

4. Design a PI controller for
   $M_p < 30\%$
   $\omega_n = 50$
   Write your program and make your hardware connections such that the reference is generated by the function generator rather than by the computer. Collect step response data and compare with Matlab predictions. Measure the rise time. Does this match your predictions ?

5. Run the program `expt5_2.c`. This implements proportional position control of the D.C. motor. Use a step reference of 1 volt. Suggested values of gain range from 0.5 to 5 and sample intervals range from 1 to 5 milliseconds. Note the effect of increasing gain on the stiffness of the potentiometer shaft. Also, note the large overshoot that you get with pure proportional control. Remember that the inclusion of another integral in your position transfer function dropped your phase by 90 degrees and increasing gain leads to decreased phase margin leading to increased overshoots ! Note when the servo becomes unstable. Be sure not to confuse wrap-around of the potentiometer with stability of the servo.

6. We see that pure proportional control leads to poor control performance of the motor due to the large overshoots. Hence, lead compensation is called for. First design a proportional controller to obtain 45 degrees of phase margin. Notice that the proportional gain to implement such a controller comes out to be very low. Implementation of the controller reveals that there is virtually no response from the motor. Why ? Consider nonlinearities such as friction, stick-slip etc.

   Now, consider using a lead compensator. Design a lead compensator to obtain a crossover frequency of 30 rad/s and a phase margin of 45 degrees. Implement this controller and record the step response. Compare the performance of this compensator with that of a pure proportional controller.

7. We would like to drive the error to zero in the presence of steady torque disturbances. This can be accomplished by using a lead-lag compensator. Design a lead-lag compensator (set the lag pole at zero to obtain an integrator) to obtain a crossover frequency of 30 rad/s and a phase margin of 45 degrees. Try rotating the potentiometer shaft away from the reference point manually. Notice the restoring torque of the servo increase with time. This is caused by the integrating action of the compensator. Record the step response. Can you see the rather

large overshoots caused by integral windup ? Implement anti-windup measures as discussed for PI velocity control and record step responses. Comment on the differences.

8. Design a velocity minor loop with a crossover of 300 rad/s and a positon major loop with a crossover of 60 rad/s. Find out the gains K_{pos} and K_{vel} to implement the controller. Again compare with Matlab and record step response.