EE 476
DC Motor Control Lab

-

Contents

1

Introduction

This is a set of exercises chosen and modified from and/or in addition to those found in the main lab manual[1] (*Note that throughout what follows we will use the abbreviation MLM*). It is highly recommended–if not altogther necessary–that you read through the main lab manual as you progress through these excercises, including the corresponding Introduction. We try to use the same notation whenever possible and reference it extensively for the additional background, figures, organization, and useful instruction that it provides. We have attempted to create a set of exercises more appropriate to our particular audience.

1.1 Policies

A nominal time table for the completion of each portion of the lab is given in Table 1.1.

Modeling	2 weeks
Speed and Position Control	1 week
Robustness	1 week

Table 1.1: Nominal Schedule

Meetings will still be weekly.

2

Modeling

Look over sections 2.1 - 2.4 in the main lab manual (MLM)[1].

2.1 Pre-Lab

Reference Figure 2.2 on page 18 of your MLM.

Torque Model

Following [2, 3], we will assume the following:

$$T_m = k_t(I_m - I_f)$$

Where I_f is what is often times called zero-load current due to mechanical losses.

$$I_f = I_0 \operatorname{sgn}(\omega_m) + I_1\omega_m + I_2\omega_m^2$$

We are essentially using I_0,I_1, and I_2 to roughly represent sliding, viscous, and airflow drag.

Voltage Model

We will asssume the back-EMF voltage V_{emf} is:

$$V_{emf} = k_m\omega_m(1 + \tau_m\omega_m)$$

where τ_m represents magnetic lags in the motor.

Exercises

Physical Plant Modeling

1. Write out the differential equations for $\dot{I}_m$ and $\dot{\omega}_m$ in terms of I_m, V_m, ω_m, and T_d. Can you write these equations using a linear state equation (i.e. $\dot{x} = Ax + Bu$) for some state x and u? Explain your answer.
2. Create a model in Simulink of the motor using the expressions you have derived such that the parameter values can be easily set and modified. Include the effects of saturation where appropriate.
3. Assume the motor shaft is stationary and examine your differential equation for $\dot{I}_m$. Determine the expression for the motor electrical time constant τ_e. Evaluate it using the nominal parameter values given in Table A.2 of your MLM.
4. Now examine your equation for $\dot{\omega}_m$. For simplicity ignore the nonlinear terms, and determine an expression for the mechanical time constant τ_ω. Based on the available nominal values of Table A.2 of your MLM, describe the magnitude of this quantity as a function of the remaining variable.
5. Use your results to justify neglecting the electrical transient, i.e. assuming $L_m = 0$. Use this assumption to derive a single differential equation for $\dot{\omega}_m$ in terms of ω_m, V_m, and T_d and create a second model in Simulink parallel to your first which ignores the electrical transient.
6. Now, taking this last expression and ignoring all friction terms, saturations, and assuming no magnetic lag, create a linear state space model for the DC motor. You should assume your inputs are the motor terminal voltage V_m and the exogeneous (disturbance) torque T_d. Assume your only measured output is motor angular velocity. Specify your A, B, C, and D matrices.
7. Modify this linear state space model to also output angular position. (Note this is easy enough to do symbolically which you need to show, but there is no need to put much effort into altering your Simulink model as you can simply add an integrator block.)
8. Create a third model in Simulink parallel to the first two representing this linear version. Note that there are default Simulink blocks for linear state space models.
9. Find the transfer function matrix corresponding to your state space model.
10. The above transfer function matrix, if found correctly, can be expressed as two first order systems: $\begin{bmatrix} \frac{K}{\tau s+1} & \frac{K_{T_d}}{\tau_{T_d} s+1} \end{bmatrix}$ Give symbolic expressions for the quantities K, τ, K_{T_d}, and τ_{T_d}, and also evaluate each of them using the nominal parameter values from Table A.2 in your MLM.
11. Create a forth model in Simulink parallel to those previous using only the transfer function matrix you derived. (Note that again there are built in Simulink blocks for this purpose.)

Open Loop System Modeling

12. Use Tables A.2 and A.3, the sampling rate given at the bottom of page 36, and the information about how the outputs are actually obtained and filtered on pages 37 and 121 in your MLM to supplement your model(s) to take account of the sensor and actuator quantization, sampling and bandwidth.

    If any of this is unclear ask your instructor!

2.2 In-Lab

For the following experiments you will make use of MATLAB. Create a directory within your main project/lab directory and call it 'modeling'.

Resistance

Look back at your expression for motor current. Note that if we set $\omega_m = 0$ and "wait" for the transient to settle we have a very nice, simple equation in terms of current, voltage, and resistance. Since the motor is relatively weak you can easily "set" the angular velocity to zero by holding the load stationary with your hand, and observe the current through the display on the software as you apply various DC voltages.

1. First apply 0V and observe the current reading. If it is nonzero then there is a bias you will need to keep track of an subtract out of subsequent current readings. Now, using the offset input, sweep the voltage over a number of values covering the allowable range[1], recording each along with the corresponding current measurement.
2. Enter your collected data as variables in MATLAB: a vector for input voltages, a vector for current readings, and any bias reading. Use the 'save' command to save these variables as 'resistance_data.mat' in your 'modeling' directory.
3. Now write a MATLAB script that will load this data to the workspace and find the optimal estimate for the parameter R_m using least squares. Name it 'estimate_resistance.m'. You should obtain a value at least somewhat close to that given in Table A.2 of your MLM.
4. Add code to compare the optimal estimate obtained using least squares to that obtained by simply averaging as the MLM suggests. Spefically, compare $\|V_m - I_m R_m^{ls}\|_2$ to $\|V_m - I_m R_m^{ave}\|_2$, where V_m and I_m are your vectors of voltage inputs and measured currents respectively, and R_m^{ls} and R_m^{ave} are the estimates using least squares and averaging respectively.
5. Add code to plot the ratio of each voltage input to measured current (V_m/I_m)as dots vs measured current (I_m) along with your best fit constant lines using least squares and averaging. You should see a points clusterd around a flat line at R_m. Clearly label all axes and include a legend.

---

* You can use limited LaTeXcode to improve the formatting of axis labels, plot titles, and legend entries (among other things) in MATLAB. For example, in order to plot $\sin(\theta_0)$, $\cos(\theta_0)$, and $\sin^2(\theta_0) + \cos^2(\theta_0)$ for some range, label the x-axis θ_0 (rad), and include a legend you could use:

```
theta = pi/100:pi/100:pi;
y1 = sin(theta);
y2 = cos(theta);
y3 = y1.^2 + y2.^2;
figure;
plot(theta,y1,'b',theta,y2,'r',theta,y3,'g-.'); grid;
axis([ 0 pi -1.5 1.5 ]);
xlabel('\theta_{0} (rad)');
ylabel('Amplitude');
legend('sin(\theta_{0})','cos(\theta_{0})','sin^{2}(\theta_{0})+cos^{2}(\theta_{0})');
```

Also, although you can save figures easily enough using the GUI, you may want to look at the documentation for the 'saveas' and 'hgexport' commands for alternatives.

[1] To address the possibility of the resistance varying with shaft position–among other things–taking multiple data points at each voltage input with the shaft in different positions is likely a good idea.

Zero-load Current

Looking at our expression for T_m, we see that in steady state conditions $I_f = I_m$. Now, we can use a number of DC input voltages and observe the steady state current[2] and angular velocity at each to identify the drag parameters.

1. Obtain a good amount of such data and save it similarly to before as 'zero_load_current_data.mat'.
2. Write a MATLAB script that assuming $I_f = I_0 \operatorname{sgn}(\omega_m) + I_1\omega_m + I_2\omega_m^2$, will find the best fit quadratic function, i.e. the optimal I_0, I_1, and I_2 using least squares. Call it 'estimate_zero_load_current.m'.
3. Now add to your script another least squares optimization, but this time assume that we are only interested in modeling the friction as viscous. Compare to the I_1 you obtained previously.
4. Add yet another least squares optimization and assume we are assuming only sliding friction. Compare to the I_0 you obtained previously.
5. Add code to plot your collected I_f data as dots vs angular velocity (Ω_m) along with your best fit quadratic function, line, and constant function. Clearly label all axes and include a legend.

Speed/Torque Constant & Magnetic Lag Time Constant

Now we can relate two expressions for V_{emf} to obtain k_m assumed equal to k_t, and τ_m. Again in steady state, we have that $V_{emf} = V_m - R_m I_m = k_m\omega_m + k_m\tau_m\omega_m^2$.

1. Obtain a good number of data points varying input voltage and recording it along with the corresponding angular velocity and current (or use the angualr velocity to calculate a current) and save it similarly to before as 'motor_constant_mag_lag_data.mat'.
2. Now, divide each side of the equation above by ω_m to obtain $\frac{V_m - R_m I_m}{\omega_m} = k_m + k_m\tau_m\omega_m$, and write a MATLAB script that will load your data and obtain the best fit line using least squares, i.e. the parameters k_m and $k_m\tau_m$. Obtain τ_m from these quantities. Call your script 'estimate_motor_constant.m'.
3. Similarly to the Resistance case, add code that will also use an averaging process while assuming $\tau_m = 0$.
4. Again similarly to the Resistance case compare the resulting k_m^{ls} and k_m^{ave}.
5. Finally as before add code to plot the ratio of each back-emf to angular velocity (V_{emf}/Ω_m)as dots vs $k_m^{ls} + k_m^{ls}\tau_m\omega_m$ and k_m^{ave}. Clearly label all axes and include a legend.

Lumped Parameters in Transfer Functions

1. Enter the 'modeling' mode in the software. For now set both K and τ to zero. Excite the system with several step inputs (square waves with very low frequency will do). Think about how to choose the amplitude and offset for your trials and explain your reasoning. Observe the step responses and use them to approximately obtain the transfer function parameters.

[2]Don't forget potential bias issues.

2. Save your test data, both input and output plots, along with the parameters you determine to 'lumped_tf_parameters1.mat'. Also write a MATLAB script to load the data and plot comparisons of your step response data to those MATLAB gives for the same inputs using your parameters. As usual make sure your plots are well labeled and nice looking. Save your script as 'compare_lumped_parameters.m'.
3. Now using various inputs adjust the values of K, and τ in real time until the model response shown and actual response shown match as well as you can get them to. Save the parameters as 'lumped_tf_parameters2.mat'. Add code to produce a couple example plots (both inputs and outputs) from the software showing the 'model response' matching the actual response.
4. How do the τ,K pairs you obtained through each of these methods compare to each other and that you previously obtained using the nominal parameters from Table A.2 in your MLM? Also how do they compare to those you would get using your experimentally identified parameters?

---

* Note that when saving data from the QICii software you should double check that it is actually saving your file. In the past, the save process did not work properly unless the file extension was manually typed on the name to save as. Also note that the data will be saved but it will not be labeled clearly. You must keep this in mind. It is advisable to reformat and rename the data to be clear then save it to the .mat file of your choosing.

Model Validation and System Limitations

This portion is more open ended. Read section 2.6.4 in your MLM and think about what it has to say.

1. How do your various Simulink models compare to the real system?
   (a) Are your models making good predictions? If any of your models is not making as good of predictions as you would expect, double check:
      i. Your equations for correctness.
      ii. That your equations are correctly implemented in Simulink.
      iii. After the above two, if you still are not seeing good predictions, you may need to consider you've made some errors in your parameter identification calculations or that you need more data.
   (b) Do some make better predictions than others generally or for certain inptus/conditions? Would you expect the behavior you see? Explain your reasoning.
2. Provide evidence showing the performance of your modeling and argue why it is good enough.
   (a) Provide plots of experimental data for various types and sizes of inputs along with simulation of the same circumstances, along with some discussion.
   (b) Discuss in general how do you think things like complexity of the plant to be controlled, cost and fesibility of hardware testing, and desired level of performance affect the trade-offs between model accuracy, required effort (i.e. cost) put into modeling, usefulness and necessity for control design, and usefulness and necessity for simulation.
3. Demonstrate your model for your lab instructor and discuss.

3

Speed Control

Read through sections 3.1 - 3.4 in your main lab manual (MLM)[1].

3.1 Pre-Lab

We repeat PI controller equation [3.1] from your MLM here for reference:

$$u(t) = k_p \left(b_{sp} r(t) - y(t)\right) + k_i \int_0^t r(\tau) - y(\tau) d\tau \tag{3.1}$$

The exercises that follow should be useful for you in designing PI speed control in the lab by providing some insight into the approximate relationships between the controller parameters and design goals. If you desire more preparation than these you may do any or all of the exercises from Section 3.5 of your MLM and ask your instructor to compare to the solutions.

Exercises

1. Create a block diagram showing the closed loop system corresponding to using a PI controller as in Equation 3.1. Include both reference and disturbance inputs.
2. Use your transfer function matrix from Exercises 9 and 10 of Section 2.1 to give a transfer function matrix from reference and disturbance inputs to speed output assuming a closed loop with PI control as in Equation 3.1. For purposes of presentation keep the K and τ symbolic. Give some commentary on the way in which the system parameters and the controller parameters enter into the different transfer functions.
3. If done correctly, you should observe that the closed loop transfer function happens to come out with a second order denominator.[1] If we want to be able to design a controller which produces a specific characteristic polynomial of the form $s^2 + 2\zeta\omega_0 + \omega_0^2$, find the appropriate k_p and k_i to do so. (This is very similar to Exereise 3. on page 63 of your MLM.)
4. Give some expression for a rough lower bound on time the system can switch speed between $\pm\omega_{nom}$ using the physical parameters and saturations you've previously determined. Show and explain the

[1] Generally speaking though this is only happening because of how we have chosen to simplify the model. Here things will work out well as our DC motor is simple enough, but this may not always be the case.

steps you use to derive this lower bound. Your lower bound should be a function of 'known' parameters and ω_{nom}.

5. Assume we want to design for a fast response but with no overshoot. We can start by trying for $\zeta = 1$ (i.e. to have a critically damped characteristic polynomial). Even our simple model is not a 'prototype' system however, and we still may have some overshoot depending on the zero in the numerator. We could use $b_{sp} = 0$ to just eliminate the issue, but this would not help our goal of a fast response (See MLM problem 11 on page 67). We can guarantee[2] no overshoot by making the system equivalent to the first order system $\frac{\omega_0}{s+\omega_0}$ through pole zero cancellation. Assuming we use $\zeta = 1$, find an expersson for b_{sp} in terms of the remaining variables τ and ω_0 which accomplishes this.
6. In order to get some better idea how a choice of ω_0 corresponds to a particular settling time, do Exe rcise 12 on page 69 of your MLM.
7. Use your existing transfer functions to derive the transfer function from reference input to error $e(t)$ where $e(t) = r(t) - \omega(t)$. Use this transfer function to find expressions for the steady state error to step references of amplitued A_0 and ramp references of amplitude R_0.

3.2 In-Lab

Integrator Windeup

1. Go through and follow the instruction in Section 3.6.6 of your MLM.

Simulink Model

1. Modify your Simulink model to represent the closed loop system. Use a Discrete PID Controller (2DOF) block from the 'Simulink/Discrete/' library.[3]
2. Run a few experiments using various controller paramters in your model and the QICii software to make sure that your Simulink model is behaving like the real system. If not double check things.

Square Wave Tracking

1. Design a PI controller to achieve the fastest tracking with no overshoot that you can. One tool you may find useful as part of your design process is the automatic tuning capability in Simulink, although it will not automatically adjust the set point weight for you. You may also make use of various heuristic manual tuning methods like the Ziegler-Nichols method (see for example page 90 of your MLM). In any case, you will need to make use of your previous analysis in order to be most efficient in your efforts and to justfiy your final design.
2. A question you want to ask as an engineer after any design is "Is this the best we can do?" Use your previous development and knowledge of the system to make a reasonable argument why the design you end up with cannot be resonably imporoved upon. (At least not without using an entirely different control setup.)
3. Provide plots giving examples of your best perfromance from both the QICii software as well as your model. Discuss any discrepancies.

[2] Insofar as our model is concerned anyway.

[3] `http://www.mathworks.com/help/simulink/slref/pidcontroller2dof.html`

Triangle Wave Tracking

1. Design a PI controller as in Equation 3.1 to achieve the tightest tracking that you can.
2. Use your previous development and knowledge of the system to make a reasonable argument why the design you end up with is the optimal PI for tracking this type of signal. Make note of any trade-offs.
3. Provide plots giving examples of your best perfromance from both the QICii software as well as your model. Discuss any discrepancies.

Reference Bandwidth

1. Use the optimal triangle wave PI controller parameters with a square wave input and vice versa. Make some observations. Among other things, how does the bandwidth of the signal you intend to track affect the amount you can "tighten" your control without running into undesirable behavior?

4

Position Control

Read through sections 5.1 - 5.4 in your main lab manual (MLM)[1].

4.1 Pre-Lab

We repeat the continuous time PID controller equation [5.1] from your MLM here for reference:

$$u(t) = k_p \left(b_{sp} r(t) - y(t)\right) + k_i \int_0^t r(\tau) - y(\tau) d\tau + k_d \left(b_{sd} \frac{d}{dt} r(t) - \frac{d}{dt} y(t)\right) \tag{4.1}$$

The exercises that follow should be useful for you in designing PID position control in the lab. This time we shorten the prelab considerably and assume that your previous work will guide you in doing any additional analysis you may or may not require. If you desire more preparation you may do any or all of the exercises from Section 5.5 of your MLM and ask your instructor to compare to the solutions.

Exercises

1. Use your transfer function matrix from Exercises 9 and 10 of Section 2.1 to give a transfer function matrix from reference and disturbance inputs to position output assuming a closed loop with PID control as in Equation 4.1. For purposes of presentation keep the K and τ symbolic. Give some commentary on the way in which the system parameters and the controller parameters enter into the different transfer functions.
2. Use your transfer functions to derive the transfer functions from reference and disturbance inputs to error $e(t)$ where $e(t) = r(t) - \theta(t)$. Use this transfer function to find expressions for the steady state error to step references of amplitued A_0 and ramp references of amplitude R_0. Comment on the role played by k_i. Specifically what happens to either steady state error if $k_i \to 0$.
3. Read through section 5.5.2 of your MLM.

4.2 In-Lab

Simulink Model

1. Modify your Simulink model to represent the closed loop system. Use a Discrete PID Controller (2DOF) block from the 'Simulink/Discrete/' library.[1]
2. Run a few experiments using various controller paramters in your model and the QICii software to make sure that your Simulink model is behaving like the real system. If not double check things.

Square Wave Tracking

1. Design a PID controller to achieve the best overall tracking of a square wave as given on page 184 of your MLM. Use whatever design methods you prefer.
2. A question you want to ask as an engineer after any design is "Is this the best we can do?" Use your previous development and knowledge of the system to make a reasonable argument why the design you end up with cannot be resonably imporoved upon. (At least not without using an entirely different control setup.)
3. Now vary the input parameters (amplitude/frequencey) and make observations about your controllers performance. Does your tracking performance hold?
4. Provide plots giving examples of your best perfromance from both the QICii software as well as your model. Discuss any discrepancies.

Triangle Wave Tracking

1. Design a PID controller as in Equation 4.1 to achieve the tightest tracking that you can for a triangle wave as given on page 188 of your MLM. Again use whatever design methodology you like.
2. Use your previous development and knowledge of the system to make a reasonable argument why the design you end up with is the optimal PID for tracking this type of signal. Make note of any trade-offs.
3. Now vary the input parameters (amplitude/frequencey) and make observations about your controllers performance. Does your tracking performance hold?
4. Provide plots giving examples of your best perfromance from both the QICii software as well as your model. Discuss any discrepancies.

Response to Disturbance

Go through Section 5.6.5 in your MLM. You are free to choose controller parameters. Do not increment k_i as in the MLM but rather use eitehr 0 or a "good" value. Does the behavior match what you expect?

Reference Bandwidth

1. Use the optimal triangle wave PID controller parameters with a square wave input and vice versa. Also use sinusoidal inputs of low and high frequency with both controllers. Make some observations.

[1] `http://www.mathworks.com/help/simulink/slref/pidcontroller2dof.html`

Among other things, how does the bandwidth of the signal you intend to track affect the amount you can "tighten" your control without running into undesirable behavior? Are you seeing the same kind of trend as with speed control?

Bibliography

[1] K. J. Åström, J. Apkarian, and H. Lacheray, *Quanser Engineering Trainer (QET) Series: USB QICII Laboratory Workbook - DC Motor Control Trainer (DCMCT) - Student Workbook*, Quanser.

[2] M. Drela, "First-order dc electric motor model," Massachusetts Institute of Technology, 2007.

[3] ——, "Second-order dc electric motor model," Massachusetts Institute of Technology, 2006.