

DC Motor Speed Control Mini Project: A Visual MATLAB Design-and-Tuning Report

Control Practice Mini Project

Contents

1	Why This Mini Project?	3
2	Step 0: Minimal MATLAB Setup	3
3	Physical Model and Parameter Choice	3
3.1	Motor Model	3
3.2	Chosen Parameters (Realistic, Design-Feasible)	3
4	Task 1: Derive the State-Space Model (A, B, C, D)	4
5	Task 2: MATLAB ss Model and Open-Loop Step Response	4
5.1	What the Open-Loop Step Should Look Like	4
5.2	Visual interpretation tips	4
6	Task 3: Transfer Function, Poles/Zeros, and DC Gain	5
6.1	Analytical transfer function $v \rightarrow \omega$	5
6.2	Numerical TF, poles, and DC gain	5
7	Task 4: PI Speed Controller Design and Verification	5
7.1	Specifications	5
7.2	Visual tuning logic (what to change if it fails)	6
7.3	Verification	6
7.4	Results table (filled by MATLAB output)	7
8	Task 5: Disturbance Rejection (Load Torque Step)	7
8.1	Disturbance model	7
8.2	Scenario	8
9	Task 6 (Optional): LQI (LQR + Integral) Extension	8
10	Discussion: Trade-offs (Speed vs Effort vs Robustness)	8
11	MATLAB Script (Run Directly)	9
12	Summary	13

1 Why This Mini Project?

This mini project consolidates core Control System Toolbox skills into a complete workflow:

- Derive a DC motor state-space model from physics.
- Simulate open-loop behavior and interpret step-response features.
- Convert to transfer function, inspect poles/zeros, and compute DC gain.
- Design a PI speed controller and verify specifications quantitatively.
- Evaluate disturbance rejection under a load-torque step.
- (Optional) Compare PI with an LQI (LQR + integral) design.

The key emphasis is **read the plots** $\Rightarrow$ **decide what to change** $\Rightarrow$ **predict effects** $\Rightarrow$ **iterate**.

2 Step 0: Minimal MATLAB Setup

Throughout, we use a unity-feedback structure with controller $C(s)$ and plant $G(s)$:

$$L(s) = C(s)G(s), \quad T(s) = \frac{L(s)}{1 + L(s)} = \text{feedback}(L, 1).$$

In MATLAB:

```
1 s = tf('s');           % Laplace variable
2 % G and C will be defined from the motor model and tuning steps
```

3 Physical Model and Parameter Choice

3.1 Motor Model

The DC motor model is:

$$L\dot{i} = -Ri - K_e\omega + v, \quad J\dot{\omega} = K_t i - b\omega,$$

with states $x = [i, \omega]^\top$, input v (armature voltage), and output $y = \omega$ (speed).

3.2 Chosen Parameters (Realistic, Design-Feasible)

We choose the following values (assigned as a reasonable small-motor example):

$$R = 2 \, \Omega, \quad L = 0.05 \, \text{H}, \quad K_t = 0.1 \, \text{N m/A}, \quad K_e = 0.1 \, \text{V/(rad/s)}, \quad J = 0.01 \, \text{kg m}^2, \quad b = 0.001 \, \text{N m s}.$$

Note. A much larger inductance (e.g. $L = 0.5 \, \text{H}$) can yield a very slow electrical mode, making a sub-0.5 s rise-time requirement difficult to achieve with basic PI tuning for the same (J, b) .

4 Task 1: Derive the State-Space Model (A, B, C, D)

Let $x_1 = i$ and $x_2 = \omega$. From the physics:

$$\dot{i} = -\frac{R}{L}i - \frac{K_e}{L}\omega + \frac{1}{L}v, \quad \dot{\omega} = \frac{K_t}{J}i - \frac{b}{J}\omega.$$

Therefore,

$$\dot{x} = Ax + Bv, \quad y = Cx + Dv,$$

with

$$A = \begin{bmatrix} -\frac{R}{L} & -\frac{K_e}{L} \\ \frac{K_t}{J} & -\frac{b}{J} \end{bmatrix}, \quad B = \begin{bmatrix} \frac{1}{L} \\ 0 \end{bmatrix}, \quad C = \begin{bmatrix} 0 & 1 \end{bmatrix}, \quad D = 0.$$

Numerically:

$$A = \begin{bmatrix} -40 & -2 \\ 10 & -0.1 \end{bmatrix}, \quad B = \begin{bmatrix} 20 \\ 0 \end{bmatrix}, \quad C = \begin{bmatrix} 0 & 1 \end{bmatrix}, \quad D = 0.$$

5 Task 2: MATLAB ss Model and Open-Loop Step Response

5.1 What the Open-Loop Step Should Look Like

Open-loop input: a unit step in voltage $v(t) = 1 \cdot u(t)$. The final speed should approach the DC gain times 1 V (computed in Task 3). The response often looks like a *dominant slow real pole* plus a much faster real pole.

Placeholder: Open-loop step response $\omega(t)$ to $v(t) = 1$ V.
Generated by the MATLAB script (exported figure).

Figure 1: DC motor open-loop speed response to unit step voltage.

5.2 Visual interpretation tips

From the open-loop plot:

- If the rise is very slow, the dominant pole is close to the origin (small $|\Re(p)|$).
- If the curve is monotone with no overshoot, poles are real and negative (over-damped).
- If oscillations appear, complex poles exist (less common in this simple motor model).

6 Task 3: Transfer Function, Poles/Zeros, and DC Gain

6.1 Analytical transfer function $v \rightarrow \omega$

Eliminating i in the Laplace domain (zero initial conditions) yields:

$$G_{\omega v}(s) \triangleq \frac{\Omega(s)}{V(s)} = \frac{K_t}{(Ls + R)(Js + b) + K_e K_t}.$$

Expanding the denominator:

$$G_{\omega v}(s) = \frac{K_t}{(LJ)s^2 + (Lb + RJ)s + (Rb + K_e K_t)}.$$

6.2 Numerical TF, poles, and DC gain

With the chosen values:

$$LJ = 5 \times 10^{-4}, \quad Lb + RJ = 0.02005, \quad Rb + K_e K_t = 0.012,$$

so

$$G_{\omega v}(s) = \frac{0.1}{0.0005 s^2 + 0.02005 s + 0.012}.$$

Zeros: none (numerator is constant).

Poles: roots of $0.0005s^2 + 0.02005s + 0.012 = 0$:

$$p_{1,2} \approx -39.49, \quad -0.6077.$$

DC gain:

$$G_{\omega v}(0) = \frac{0.1}{0.012} \approx 8.333 \frac{\text{rad/s}}{\text{V}}.$$

Placeholder: Pole-zero map of $G_{\omega v}(s)$ (or open-loop T).
Generated by the MATLAB script.

Figure 2: Pole-zero map (expect two stable real poles; no zeros).

7 Task 4: PI Speed Controller Design and Verification

7.1 Specifications

We tune a PI controller $C(s) = K_p + \frac{K_i}{s}$ such that the closed-loop speed response to

$$\omega_{\text{ref}}(t) = 100 \cdot u(t) \quad \text{rad/s}$$

satisfies:

- rise time $T_r < 0.5$ s,
- overshoot $< 10\%$,
- steady-state error $e_{ss} < 1\%$ (i.e., < 1 rad/s for a 100 rad/s step).

7.2 Visual tuning logic (what to change if it fails)

Table 1: Closed-loop step-response symptoms and PI tuning actions (DC motor speed loop)

Symptom	Likely cause	Try this
Rise too slow (T_r too large)	Loop bandwidth too low	Increase overall aggressiveness (often increases K_p and/or target crossover).
Overshoot $> 10\%$ / ringing	Phase margin too small	Reduce aggressiveness; increase phase margin target; optionally add derivative/lead (not used here).
Nonzero steady-state error	Insufficient low-frequency gain	Increase K_i (integral action), re-check overshoot.
Large control voltage spikes	Too aggressive gains	Reduce K_p or lower crossover; consider higher R penalty (LQI) or actuator saturation handling.

7.3 Verification

We verify by:

- `stepinfo` for T_r , overshoot, and settling time,
- `dcgain(T)` for steady-state tracking (normalized $e_{ss} = |1 - \text{dcgain}(T)|$),
- `margin(L)` for robustness (phase margin / gain margin),
- plotting control effort $v(t)$ to ensure the design is practically implementable.

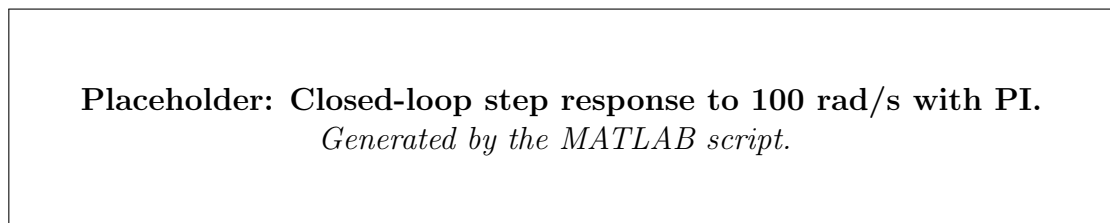


Figure 3: Closed-loop speed response with PI controller (step to 100 rad/s).

Placeholder: Bode/margins of loop $L(s) = C(s)G(s)$ and control signal $v(t)$.

Generated by the MATLAB script.

Figure 4: Robustness and effort: margins of $L(s)$ and control voltage $v(t)$ for a 100 rad/s step.

7.4 Results table (filled by MATLAB output)

After running the script, record the outputs into Table 2. (The script prints these values.)

Table 2: PI closed-loop performance metrics (populate from MATLAB `stepinfo/dcgain/margin`)

Metric	Requirement	Measured (MATLAB)
Rise time T_r (10–90%)	< 0.5 s	[...]
Overshoot (%)	$< 10\%$	[...]
Settling time T_s (2%)	(not specified)	[...]
Normalized $e_{ss} = 1 - \text{dcgain}(T) $	< 0.01	[...]
Phase margin PM	(robustness check)	[...] deg
Gain margin GM	(robustness check)	[...] dB

8 Task 5: Disturbance Rejection (Load Torque Step)

8.1 Disturbance model

Add load torque τ_L in the mechanical equation:

$$J\dot{\omega} = K_t i - b\omega - \tau_L.$$

This is an additional input $d = \tau_L$ with input matrix

$$B_d = \begin{bmatrix} 0 \\ -\frac{1}{J} \end{bmatrix}.$$

Under feedback, the disturbance-to-output map is reduced by the sensitivity function:

$$Y(s) = \underbrace{\frac{G(s)}{1 + C(s)G(s)}}_{\text{complementary shaping}} R(s) + \underbrace{\frac{G_{\omega\tau}(s)}{1 + C(s)G(s)}}_{\text{disturbance shaping}} L(s),$$

so increasing low-frequency loop gain (especially via K_i) typically improves load-torque rejection.

8.2 Scenario

$$\omega_{\text{ref}}(t) = 100 u(t), \quad \tau_L(t) = 0.05 u(t - 1) \text{ N m.}$$

We compare:

- open-loop (feedforward-only voltage that achieves 100 rad/s at DC),
- closed-loop (PI feedback).

Placeholder: Disturbance rejection plot (open-loop vs PI).
Generated by the MATLAB script.

Figure 5: Load-torque step at $t = 1$ s: feedback reduces speed deviation and restores setpoint.

9 Task 6 (Optional): LQI (LQR + Integral) Extension

A standard LQR state-feedback $v = -Kx$ is a regulator around the origin and does not guarantee nonzero step tracking. To compare fairly with PI tracking, we use **LQI** by augmenting an integral state:

$$x_I(t) = \int_0^t (\omega_{\text{ref}}(\tau) - \omega(\tau)) d\tau.$$

Then LQR on the augmented state $[x^\top, x_I]^\top$ produces a controller with integral action (tracking) and explicit effort weighting.

Placeholder: PI vs LQI response and control voltage comparison.
Generated by the MATLAB script (optional section enabled).

Figure 6: Optional comparison: PI vs LQI in speed response and control effort.

10 Discussion: Trade-offs (Speed vs Effort vs Robustness)

- **Faster rise time** usually requires higher bandwidth and larger control effort; check $v(t)$ for peaks and potential saturation.
- **Lower overshoot** correlates with higher phase margin; tuning for $\text{PM} \approx 45^\circ\text{--}60^\circ$ is often a good compromise.

- **Disturbance rejection** improves with higher low-frequency loop gain; integral action is particularly effective for constant/step torque loads.
- **Robustness** is assessed by margins and sensitivity; pushing bandwidth too high can degrade noise tolerance and model-mismatch tolerance.

11 MATLAB Script (Run Directly)

File name: dc_motor_mini_project.m

Required toolbox: Control System Toolbox.

```

1 %% DC Motor Speed Control Mini Project (Run Directly)
2 clear; close all; clc;
3
4 %% Parameters (assigned reasonable values)
5 R = 2;           % Ohm
6 L = 0.05;        % H
7 Kt = 0.1;        % N*m/A
8 Ke = 0.1;        % V/(rad/s)
9 J = 0.01;        % kg*m^2
10 b = 0.001;      % N*m*s
11
12 %% State-space: x = [i; w], input v, output w
13 A = [ -R/L      -Ke/L;
14        Kt/J      -b/J ];
15 Bv = [1/L; 0];   % voltage input
16 C = [0 1];       % output = speed
17 D = 0;
18
19 Smotor = ss(A,Bv,C,D);
20
21 %% Open-loop step response (unit step in voltage)
22 figure('Name','Open-loop step (v->w)');
23 step(Smotor); grid on;
24 title('DC Motor Open-loop Speed Response to 1 V Step');
25 ylabel('\omega (rad/s)');
26
27 % Optional export:
28 % exportgraphics(gcf,'fig_ol_step.png','Resolution',200);
29
30 %% Transfer function, poles/zeros, DC gain
31 Gmotor = tf(Smotor); % v -> w
32 disp('Gmotor(s) ='); disp(Gmotor);
33
34 p = pole(Gmotor);
35 z = zero(Gmotor);
36 Kdc = dcgain(Gmotor);
37
38 fprintf('Poles:\n'); disp(p);
39 fprintf('Zeros:\n'); disp(z);
40 fprintf('DC gain (rad/s)/V = %.6f\n', Kdc);
41

```

```

42 figure('Name','Pole-Zero Map');
43 pzmap(Gmotor); grid on; title('Pole-Zero Map of Gmotor');
44
45 %% PI design using pidtune + simple search to meet specs
46 % Specs: RiseTime < 0.5 s, Overshoot < 10%, steady-state error <
    1% (unit step)
47 opt = pidtuneOptions('DesignFocus','reference-tracking',
    'PhaseMargin',60);
48
49 wCandidates = logspace(0, 2, 40); % 1 to 100 rad/s
50 found = false;
51
52 for w = wCandidates
53     try
54         Cpi = pidtune(Gmotor,'PI',w,opt);
55         Tcl = feedback(Cpi*Gmotor, 1); % r -> w
56         if ~isstable(Tcl), continue; end
57
58         info = stepinfo(Tcl, 'RiseTimeLimits',[0.1 0.9], '
            SettlingTimeThreshold',0.02);
59         ess = abs(1 - dcgain(Tcl)); % normalized steady-state
            error for unit step
60
61         if (info.RiseTime < 0.5) && (info.Overshoot < 10) && (ess
            < 0.01)
62             found = true;
63             break;
64         end
65     catch
66         % ignore and continue
67     end
68 end
69
70 % Fallback PI if not found (rare for this parameter set)
71 if ~found
72     warning('No pidtune candidate met specs; using fallback PI
        gains.');
```

% A reasonable fallback that typically yields fast tracking:

```

73 Cpi = pid(2.5, 15); % Kp=2.5, Ki=15
74 Tcl = feedback(Cpi*Gmotor,1);
75 info = stepinfo(Tcl, 'RiseTimeLimits',[0.1 0.9], '
        SettlingTimeThreshold',0.02);
76 ess = abs(1 - dcgain(Tcl));
77 end
78
79 disp('Chosen PI controller Cpi(s) ='); disp(Cpi);
80
81 %% Closed-loop plots and metrics (100 rad/s step)
82 figure('Name','Closed-loop step (PI), 100 rad/s');
83 step(100*Tcl); grid on;
84 title('Closed-loop Speed with PI (Step to 100 rad/s)');
```

```

86 ylabel('\omega (rad/s)');
87
88 infoPI = stepinfo(Tcl, 'RiseTimeLimits',[0.1 0.9], '
      SettlingTimeThreshold',0.02);
89 essPI = abs(1 - dcgain(Tcl));
90
91 fprintf('\n=== PI VERIFICATION (unit-step metrics) ===\n');
92 fprintf('RiseTime (10-90%) = %.4f s\n', infoPI.RiseTime);
93 fprintf('Overshoot = %.2f %%\n', infoPI.Overshoot);
94 fprintf('SettlingTime (2%) = %.4f s\n', infoPI.SettlingTime);
95 fprintf('Normalized e_ss = %.6g\n', essPI);
96
97 %% Robustness check: margins of L(s)=C(s)G(s)
98 Lloop = minreal(Cpi*Gmotor);
99 [GM, PM, Wcg, Wcp] = margin(Lloop);
100 GMdB = 20*log10(GM);
101
102 fprintf('\n=== LOOP MARGINS ===\n');
103 fprintf('Phase Margin PM = %.2f deg at wcp = %.4f rad/s\n', PM,
      Wcp);
104 fprintf('Gain Margin GM = %.2f dB at wcg = %.4f rad/s\n', GMdB,
      Wcg);
105
106 figure('Name','Bode + margin of loop L(s)');
107 margin(Lloop); grid on; title('Bode/Margins of L(s)=Cpi(s)Gmotor(
      s)');
108
109 %% Control effort v(t) for 100 rad/s step
110 % u = C * e, and e = S * r where S=1/(1+CG)
111 Scl = feedback(1, Lloop);
112 Tu_r = minreal(Cpi * Scl); % U/R
113
114 t = linspace(0, 3, 3001)'; % 0..3 s
115 r = 100*ones(size(t)); % 100 rad/s step
116 u = lsim(Tu_r, r, t);
117
118 figure('Name','Control voltage v(t) for 100 rad/s step');
119 plot(t, u, 'LineWidth', 1.2); grid on;
120 title('Control Voltage v(t) under PI (100 rad/s step)');
121 xlabel('Time (s)'); ylabel('v (V)');
122
123 %% Disturbance rejection: load torque tau_L step at t=1 s
124 % Disturbance enters mechanical equation as -tau_L/J
125 Bd = [0; -1/J];
126 Gd = tf(ss(A, Bd, C, 0)); % tau_L -> w (open-loop)
127
128 Tyd = minreal(Scl * Gd); % closed-loop: y/d = S * (tau->w)
129
130 % Input signals
131 d = 0.05*(t >= 1); % 0 -> 0.05 N*m at t=1s
132

```

```

133 % Closed-loop output under (r,d):  $y = T_{cl}r + T_{yd}d$ 
134 y_cl = lsim([Tcl, Tyd], [r d], t);
135
136 % Open-loop comparison: feedforward voltage to hit 100 rad/s at
    DC + disturbance
137 v_ff = (100/dcgain(Gmotor))*ones(size(t));
138 y_ol = lsim(Gmotor, v_ff, t) + lsim(Gd, d, t);
139
140 figure('Name','Disturbance rejection: open-loop vs PI');
141 plot(t, y_ol, 'LineWidth', 1.0); hold on;
142 plot(t, y_cl, 'LineWidth', 1.4);
143 grid on;
144 title('Load Torque Step at t=1s: Open-loop (FF) vs PI Closed-loop
    ');
145 xlabel('Time (s)'); ylabel('\omega (rad/s)');
146 legend('Open-loop (feedforward only)', 'Closed-loop (PI)', '
    Location', 'best');
147
148 %% Optional: LQI (LQR + Integral) comparison (set true/false)
149 doLQI = true;
150 if doLQI
151     % LQI adds integral state  $x_I = \text{integral}(r - y)$ 
152     Qx = diag([1, 10]); % weights on  $[i; w]$ 
153     Qi = 200; % weight on integral state
154     Q = blkdiag(Qx, Qi);
155     Rlqi = 1; % effort weight
156
157     Klqi = lqi(Smotor, Q, Rlqi); %  $u = -[K_x \ K_i]*[x; x_I]$ 
158     Kx = Klqi(1:2);
159     Ki = Klqi(3);
160
161     % Augmented closed-loop from r to y
162     Aaug = [A, zeros(2,1);
163             -C, 0];
164     Baug = [Bv; 0];
165     Eaug = [zeros(2,1); 1]; % reference enters integrator
        dynamics
166     Caug = [C, 0];
167
168     Kaug = [Kx, Ki];
169     Acl = Aaug - Baug*Kaug;
170
171     SysLQI = ss(Acl, Eaug, Caug, 0);
172
173     figure('Name','PI vs LQI speed response');
174     step(100*Tcl, 100*SysLQI); grid on;
175     title('Speed Step Response: PI vs LQI (100 rad/s)');
176     legend('PI', 'LQI', 'Location', 'best');
177
178 % Compare control effort: simulate PI control u (already
    computed) vs LQI control

```

```

179     [~, ~, xaug] = lsim(SysLQI, r, t);
180     u_lqi = -(xaug * Kaug. ');    % u = -K*[x; xI]
181
182     figure('Name','PI vs LQI control effort');
183     plot(t, u, 'LineWidth', 1.2); hold on;
184     plot(t, u_lqi, 'LineWidth', 1.2);
185     grid on;
186     title('Control Voltage Comparison: PI vs LQI');
187     xlabel('Time (s)'); ylabel('v (V)');
188     legend('PI','LQI','Location','best');
189 end

```

12 Summary

This report completed the full mini project:

- derived (A, B, C, D) from the motor physics and instantiated a numerical model;
- simulated open-loop behavior and interpreted its dominant pole structure;
- obtained $G_{\omega v}(s)$, poles, and DC gain;
- designed a PI controller and verified rise time, overshoot, and steady-state error via `stepinfo/dcgain`;
- evaluated load-torque disturbance rejection and demonstrated feedback benefit;
- optionally compared PI with LQI for tracking and control-effort shaping.