Yağız Gelen 122207048

Emir Eren 122207095

Mehmet Reha Türkeş 123207026

Yağız Mol 122207058

Selin Rabia Kinsiz 122207037

Zehra Füreyya Erdoğan 122207074

MECA311
CONTROL THEORY
SYSTEM IDENTIFICATION AND PID CONTROLLER DESIGN USING ROOT LOCUS

INTRODUCTION

Control systems are essential in engineering applications, influencing areas ranging from industrial production to automotive control. Accurately determining system dynamics and designing effective controllers is a core requirement. This project emphasizes the design and tuning of PID controllers using MATLAB and hands-on experience that reinforces theoretical knowledge from control systems courses. The objective is analysing MATLAB and hand-written results.

SYSTEM ANALYSES BY HAND

1- The given mechanical system was modeled using the standard second order mass sipring damper differential equation:

$$u(t) = m\ddot{x}(t) + c\dot{x}(t) + kx(t)$$

$$u(s) = X(s)(ms^2 + cs + k)$$

Taking the Laplace transform of the equation and assuming zero initial conditions, the transfer function of the system becomes:

$$G(s) = \frac{X(s)}{u(s)} = \frac{1}{(ms^2 + cs + k)}$$

$u(t) = 1 \quad u(s) = \frac{1}{s}$

This form allows direct comparison with the standard second order system model.

2- Stedy state: $x_{ss} \approx 0.02684$

maximum value: $x_{peak} = 0.04174$

peak time: $T_p = 0.524\,s$

3- TF: $G(s) = \frac{1}{ms^2 + cs + k} \longrightarrow G(s) = \frac{1/m}{s^2 + \frac{c}{m}s + \frac{k}{m}}$

By comparing the denominator of the transfer function with the standard second order form

standard 2nd order denominator: $s^2 + 2\zeta\omega_n s + \omega_n^2$

the following relationships are obtained:

$$\rightarrow \omega_n^2 = \frac{k}{m} \rightarrow \omega_n = \sqrt{\frac{k}{m}}$$

$$\rightarrow 2\zeta\omega_n = \frac{c}{m} \rightarrow \zeta = \frac{c}{2\sqrt{km}}$$

4- The damping ratio ζ was calculated using the percent overshoot obtained from the step response

$$\%OS = 100 \cdot \frac{x_{peak} - x_{ss}}{x_{ss}} \longrightarrow \%OS = 100 \cdot \frac{0.04174 - 0.02684}{0.02684} \Longrightarrow \%OS = \%55.54$$

$$\%OS = 100 \cdot e^{\left(-\zeta\pi/\sqrt{1-\zeta^2}\right)} \Rightarrow \zeta = \frac{-\ln(\%OS/100)}{\sqrt{\pi^2 + [\ln(\%OS/100)]^2}}$$

Based on the measured overshoot, the damping ratio was found to be approximately:

$$\zeta \approx 0.184$$

5- Using the peak time relation

$$\left.\begin{aligned} T_p &= \frac{\pi}{\omega_d} \\ \omega_d &= \omega_n\sqrt{1-\zeta^2} \end{aligned}\right\} \omega_n = \frac{\pi}{T_p\sqrt{1-\zeta^2}}$$

The natural frequency was calculated as

$$\omega_n \approx 6.10 \text{ rad/s}$$

6- for a unit mass system ($m=1\,kg$), the spring constant and damping coefficient were obtained as

$$k = m\omega_n^2 \longrightarrow k = (6.10)^2 \approx 37.20 \text{ N/m}$$

$$c = 2\zeta\omega_n m \longrightarrow c = 2\,(0.184)(6.10) \approx 2.24 \text{ Ns/m}$$

(steady state)

$$x_{ss} = \frac{1}{k} = \frac{1}{37.20} \longrightarrow x_{ss} \approx 0.02688 \longrightarrow 0.2684 \text{ (data)} \rightarrow \text{consistent.}$$

7- from both hand calculations and MATLAB results, the following time domain parameters were obtained:

Steady-state displacement: $x_{ss} \approx 0.0268$

Peak time: $T_p \approx 0.524s$

Percent overshoot: $\%OS \approx \%55.54$

Settling time: $T_s \approx \frac{4}{\zeta\omega_n} \Rightarrow T_s = \frac{4}{(0.184)(6.10)} \approx 3.57s$

Settling time (2% criterion): $T_s(2\%) \approx 3.57s$

$$G(s) = \frac{1}{s^2 + 2.24s + 37.20} \longrightarrow (s^2 + 2\zeta\omega_n s + \omega_n)$$

$\omega_n^2 = 37.20 \rightarrow \omega_n = \sqrt{37.20} = 6.10$ rad/s

$2\zeta\omega_n = 2.24 \rightarrow \zeta = \frac{2.24}{2 \times 6.10} = 0.184$

$$G(s) = \frac{1}{\omega_n^2} \times \frac{1}{\left(\frac{s}{\omega_n}\right)^2 + 2\zeta\left(\frac{s}{\omega_n}\right) + 1} = \frac{1}{37.2} \times \frac{1}{\left(\frac{s}{6.10}\right)^2 + 2(0.184)\left(\frac{s}{6.10}\right) + 1}$$

$|G(0)| = \frac{1}{37.2} = 0.02688$

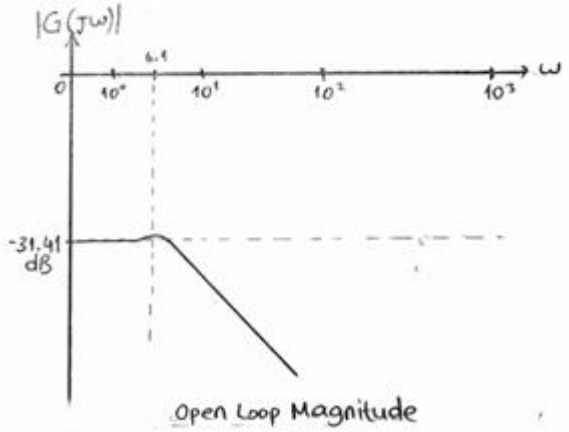
$20\log_{10}|G(0)| = 20\log_{10}(0.02688) = -31.41$ dB

$\omega_b \approx \omega_n = 6.10$ rad/s

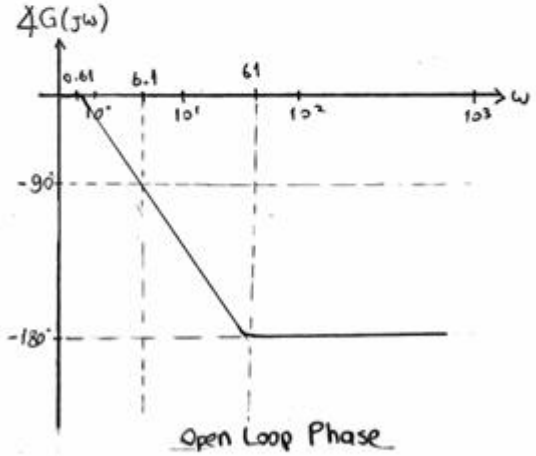
$\omega \ll 6.1 \rightarrow |G(j\omega)| \approx \frac{1}{37.2} = -31.41$ dB

$\omega \gg 6.1 \longrightarrow |G(j\omega)| \approx \frac{1}{\omega^2} \longrightarrow -40$ dB/dec

$M_r \approx \frac{1}{2\zeta\sqrt{1-\zeta^2}} = 2.77 \rightarrow 20\log_{10}(M_r) \approx 8.85$ dB



Open Loop Magnitude



Open Loop Phase

$0.1\omega_n \rightarrow 0.1 \times 6.1 = 0.61$ rad/s
$10\omega_n \longrightarrow 10 \times 6.1 = 61$ rad/s
} $[0.61, 61]$

$\omega \leq 0.61 \rightarrow 0°$
$\omega = 6.1 \longrightarrow -90°$
$\omega \geq 61 \longrightarrow -180°$

8- A PID controller was designed to satisfy the following performance constraints :

$0.14 < T_p < 0.15\,s \qquad \%OS < 12$

- $\zeta_{min} = \frac{-\ln(0.12)}{\sqrt{\pi^2 + [\ln(0.12)]^2}} = 0.559$
- $T_s = \frac{4}{\zeta . \omega_n} = \frac{4}{16.25} \approx 0.246\,s$

A desired damping ratio of $\zeta_d = 0.60$

$\%OS_d = 100 e^{(-0.60\pi/\sqrt{1-(0.60)^2})} \approx \%9.5$

$T_{p,d} = 0.145\,s$

The corresponding desired natural frequency was calculated as

$$\omega_{n,d} = \frac{\pi}{T_{p,d}\sqrt{1-\zeta_d^2}} \approx 27.08\ rad/s$$

$\zeta_d \omega_{n,d} = (0.6)(27.08) = 16.25$

$\alpha = 81.25 \approx 5 \times 16.25$

The dominant closed loop poles were defined as

$$s_{1,2} = -\zeta_d \omega_{n,d} \mp j\omega_d$$

An additional third pole was placed sufficiently far to the left to ensure dominance of the selected poles.

$$s_3 = -\alpha$$

$$[s^2 + 2\zeta_d \omega_{n,d} s + (\omega_{n,d})^2] . (s+\alpha) = s^3 + (2\zeta_d \omega_{n,d} + \alpha)s^2 + (\omega_{n,d}^2 + 2\zeta_d \omega_{n,d}\alpha)s + \omega_{n,d}^2 \alpha$$

$s^3 + (c + K_d)s^2 + (K + K_p)s + K_i = 0$

$c + K_d = 2\zeta_d \omega_{n,d} + \alpha$

$K + K_p = \omega_{n,d}^2 + 2\zeta_d \omega_{n,d}\alpha$

$K_i = \omega_{n,d}^2 \alpha$

$c = 2.24$, $K = 37.20$, $\zeta_d = 0.60$, $\omega_{n,d} = 27.08$, $\alpha = 81.25$

By matching coefficients, the PID controller gains were obtained as:

$K_p \approx 3336.77 \approx 3.34 \times 10^3$

$K_i \approx 59593.23 \approx 5.96 \times 10^4$

$K_d \approx 111.50 \approx 1.12 \times 10^2$

Plant : $G(s) = \frac{1}{s^2 + 2.24s + 37.2}$

PID : $C(s) = K_p + \frac{K_i}{s} + K_d s$

closed loop : $T(s) = \frac{C(s)\,G(s)}{1 + C(s)G(s)}$

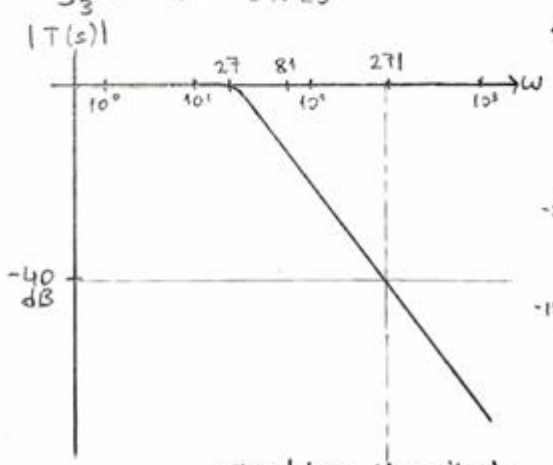
$\omega_{min} = 0.1$ rad/s
$\omega_{max} = 300$ rad/s

$T(0) \approx 1 \rightarrow 20\log_{10}|T| \approx 0\,dB$

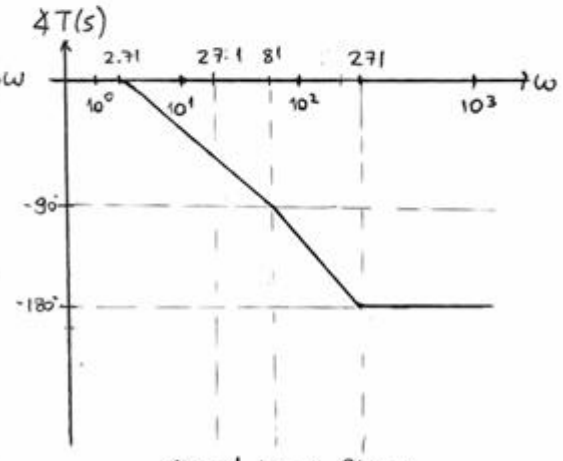
$\omega_{c,cl} \approx 27.1$ rad/s

$\omega < \omega_c \rightarrow 0\,dB$
$\omega \approx \omega_c \rightarrow$ cut off ($\approx -3dB$)
$\omega > \omega_c \rightarrow$ roll off

$s_3 = -K = -81.25$



Closed Loop Magnitude



Closed Loop Phase

9- The open loop frequency response of the system was analyzed using a Bode diagram

$$|G(j\omega)| = \frac{1}{\sqrt{(k-\omega^2)^2 + (c\omega)^2}} \Rightarrow |G(0)| = \frac{1}{k}$$

Since the plant is underdamped, the magnitude is not monotonic. The cutoff frequency is taken on descending side where

$$|G(j\omega_c)| = \frac{|G(0)|}{\sqrt{2}}$$

The cutoff frequency was found as

$\omega_c \approx 9.25$ rad/s $\quad f_c = \frac{\omega_c}{2\pi} \approx 1.47$ Hz

10- for the PID-controller system, the frequency response was analyzed again using a Bode plot

The closed loop cutoff frequency was approximately

$\omega_{c,cl} = \omega_{n,d} \approx 27.1$ rad/s

$f_{c,cl} \approx 4.31$ Hz

11- Comparison of Open-loop and Closed-loop Responses :

The open-loop system exhibits :

- High overshoot
- Significant phase lag
- Poor low-frequency tracking performance

The closed-loop PID controlled system shows :

- Reduced overshoot
- Faster transient response
- Near-zero steady-state error due to the integral action K_i.

$$\%\,error_{T_p} = \left| \frac{0.1430 - 0.145}{0.145} \right| \times 100 = 1.38\%$$

$$\%\,error_{os} = \left| \frac{11.85 - 9.48}{9.48} \right| \times 100 = 25.0\%$$

$$\%\,error_{T_s} = \left| \frac{0.6140 - 0.246}{0.246} \right| \times 100 = 149.6\%$$

The settling time deviation is relatively large. This is expected since the desired settling time is obtained from a second-order approximation, and the actual closed-loop system is third order due to the PID pole placement design and filtering effects.

Smaller errors in peak time and overshoot are expected since these metrics are directly controlled by the dominant pole placement design. In contrast, the settling time is more sensitive to higher order dynamics.

MATLAB ANALYSES

--- SYSTEM IDENTIFICATION ---

wn = 6.100 rad/s

zeta = 0.184

k = 37.2041 N/m

c = 2.2443 Ns/m

Tp = 0.5240 s

Overshoot = 55.54 %

Ts (measured,2%) = 3.2840 s

--- PID CONTROL RESULTS (MANUAL from step response) ---

Kp=190.00 Ki=1450.00 Kd=25.00 Tf=0.00370

Tp = 0.1430 s

Overshoot = 11.85 %

Ts = 0.6140 s

--- FREQUENCY DOMAIN RESULTS ---

Open Loop Cutoff Frequency   = 9.249 rad/s

Closed Loop Cutoff Frequency = 33.996 rad/s

Phase Margin = 72.02 deg

Gain Margin = Inf dB

DONE.

>>

CONCLUSION; RESULTS AND DISCUSSIONS

System is evaluated on MATLAB and with hand write. PID results showed up with different values. Hand calculations gives approximate values of bode diagrams and little terms are always estimated in equation while calculating with hand write. However MATLAB uses the real mathematical model of system. That is the reason Kp, Ki, Kd values are different on MATLAB and hand write. Frequency and gain values select as their approximate values in hand write. That is why gain and frequency values are different on MATLAB and hand write. Hand calculations help to understand the behaviour of the system on the other side MATLAB results are more sensitive and reliable.

APPENDEX

```
clear; clc; close all;

fprintf('============================================================\n');
fprintf('MECA311 PROJECT\n');
fprintf('============================================================\n');

%% ====================== Q1: LOAD DATA ======================
fprintf('\n=== Q1: DATA ACQUISITION ===\n');
load('measurement_time_output.mat');          % measurement data: variables time and output_response

t = time(:);          % convert time to a column vector
y = output_response(:);  % convert output to a column vector

fprintf('Loaded N = %d samples\n', numel(t));
fprintf('t range  = %.4f to %.4f s\n', t(1), t(end));

figure(1); clf;
plot(t, y, 'b', 'LineWidth', 1.6); grid on;
xlabel('Time (s)'); ylabel('Output');
title('Measured Open-Loop Step Response');
hold on;

%% ====================== Q2–Q3: IDENTIFICATION ======================
fprintf('\n=== Q2–Q3: SYSTEM IDENTIFICATION (zeta, wn) ===\n');

N = numel(y);
tailN = max(100, round(0.1*N));          % last 10% (at least 100 samples) for steady-state estimate
x_ss = mean(y(end-tailN+1:end));          % steady-state estimate from tail average
```

```
[y_peak, idxp] = max(y);                % peak value and its index
Tp = t(idxp);                           % peak time

OS_pct = 100*(y_peak - x_ss)/x_ss;      % percent overshoot
OS_dec = OS_pct/100;                    % overshoot ratio for formula

if ~isfinite(OS_dec) || OS_dec <= 0
   error('Invalid overshoot ratio. Check data / x_ss estimate.');
end

% standard 2nd-order relationships: OS -> zeta, Tp -> wn
zeta = -log(OS_dec)/sqrt(pi^2 + (log(OS_dec))^2);
wn   = pi/(Tp*sqrt(1 - zeta^2));

fprintf('x_ss   = %.6f\n', x_ss);
fprintf('y_peak = %.6f\n', y_peak);
fprintf('Tp     = %.6f s\n', Tp);
fprintf('OS     = %.2f %%\n', OS_pct);
fprintf('zeta   = %.6f\n', zeta);
fprintf('wn     = %.6f rad/s\n', wn);

plot(Tp, y_peak, 'ro', 'MarkerFaceColor','r');
yline(x_ss,'g--','x_{ss}');
legend('Measured','Peak','Steady-state','Location','best');

%% ====================== Q4: k and c (m=1) ======================
fprintf('\n=== Q4: PHYSICAL PARAMETERS (m=1 kg) ===\n');

m = 1;                  % mass given in the project
k = m*wn^2;             % wn^2 = k/m -> k = m*wn^2
```

```
c = 2*zeta*wn*m;       % 2*zeta*wn = c/m -> c = 2*zeta*wn*m

fprintf('m = %.2f kg\n', m);
fprintf('k = %.6f N/m\n', k);
fprintf('c = %.6f Ns/m\n', c);

G = tf(1,[m c k]);     % plant: Force -> displacement

%% ==================== Q5: Time-domain specs from measured data (manual) ====================
fprintf('\n=== Q5: TIME-DOMAIN SPECS (MEASURED DATA, MANUAL) ===\n');

Ts_meas = settling_time_2pct(t, y, x_ss);   % 2% settling time from measured data (manual band method)
Ts_approx = 4/(zeta*wn);                    % theoretical approximation for Ts (2% criterion)

% also show theory steady-state for a unit step force: x_ss = 1/k
xss_theory = 1/k;

fprintf('Measured Tp             = %.6f s\n', Tp);
fprintf('Measured OS             = %.2f %%\n', OS_pct);
fprintf('Measured Ts (2%% band)   = %.6f s\n', Ts_meas);
fprintf('Approx Ts = 4/(zeta*wn)  = %.6f s\n', Ts_approx);
fprintf('Theory steady-state (1/k) = %.6f\n', xss_theory);

%% ==================== Q6: PID CONTROLLER ====================
fprintf('\n=== Q6: PID CONTROLLER DESIGN (GAINS GIVEN) ===\n');

% PID gains (use your working values)
Kp = 190.0;
Ki = 1450.0;
Kd = 25.0;
```

```
Tf = 0.0037;          % derivative filter constant

C = pid(Kp, Ki, Kd, Tf); % PID controller (filtered derivative)

L = C*G;              % loop transfer function
Tref = feedback(L, 1);   % closed-loop: reference -> output

% --- manual extraction of Tp/OS/Ts from closed-loop step response ---
t_step = (0:0.001:2).';             % time vector for step analysis
[y_cl_step, t_cl_step] = step(Tref, t_step);

yss_cl = mean(y_cl_step(end-200:end));  % steady-state estimate from last samples
[ymax_cl, idx_cl] = max(y_cl_step);
Tp_cl = t_cl_step(idx_cl);
OS_cl = 100*(ymax_cl - yss_cl)/yss_cl;
Ts_cl = settling_time_2pct(t_cl_step, y_cl_step, yss_cl);

fprintf('PID gains:\n');
fprintf('  Kp=%.3f, Ki=%.3f, Kd=%.3f, Tf=%.6f\n', Kp, Ki, Kd, Tf);

fprintf('Closed-loop (Tref) step metrics (MANUAL from step response):\n');
fprintf('  Tp = %.6f s\n', Tp_cl);
fprintf('  OS = %.3f %%\n', OS_cl);
fprintf('  Ts = %.6f s\n', Ts_cl);
fprintf('  y_ss (estimated) = %.6f\n', yss_cl);

figure(2); clf;
plot(t_cl_step, y_cl_step, 'r', 'LineWidth', 1.6); grid on;
yline(yss_cl,'k--','y_{ss}');
title('Closed-Loop Step Response (T_{ref})');
xlabel('Time (s)'); ylabel('Output');
```

```
%% ==================== ROOT LOCUS PLOT (SAFE LEGEND) ====================
fprintf('\n=== ROOT LOCUS PLOT ===\n');

% target pole marker for visualization only
Tp_target = 0.145;
OS_target = 10;
zeta_d = -log(OS_target/100)/sqrt(pi^2 + (log(OS_target/100))^2);
wn_d   = pi/(Tp_target*sqrt(1-zeta_d^2));
sigma_d = zeta_d*wn_d;
wd_d    = wn_d*sqrt(1-zeta_d^2);
s_target = -sigma_d + 1i*wd_d;

p_cl = pole(Tref);      % actual closed-loop poles

figure(3); clf;
rlocus(L); grid on; hold on;
plot(real(s_target),  imag(s_target), 'rx', 'MarkerSize', 10, 'LineWidth', 2);
plot(real(s_target), -imag(s_target), 'rx', 'MarkerSize', 10, 'LineWidth', 2);
plot(real(p_cl), imag(p_cl), 'ko', 'MarkerSize', 7, 'LineWidth', 1.5);

title('Root Locus of L(s)=C(s)G(s) with Target & Actual Poles');
xlim([-80 10]); ylim([-50 50]);

% dummy handles to avoid legend issues with rlocus objects
hRL = plot(nan,nan,'b-');
hT  = plot(nan,nan,'rx','MarkerSize',10,'LineWidth',2);
hP  = plot(nan,nan,'ko','MarkerSize',7,'LineWidth',1.5);
legend([hRL hT hP], {'Root Locus','Target Poles','Actual CL Poles'}, 'Location','best');

%% ==================== Q7: OPEN-LOOP CUTOFF ====================
```

```
fprintf('\n=== Q7: OPEN-LOOP CUTOFF FREQUENCY ===\n');

w = logspace(-4, 6, 60000);          % wide sweep so cutoff is not missed
[magG, phG] = bode(G, w);
magG = squeeze(magG);
phG  = squeeze(phG);

DC_G = 1/k;                          % manual DC gain for G(s)=1/(ms^2+cs+k)
targetG = DC_G/sqrt(2);              % -3 dB relative to DC

[wc_ol, fc_ol, ph_wc_ol] = cutoff_from_mag_robust(w, magG, phG, targetG);

fprintf('Open-loop DC gain |G(0)|   = %.8f\n', DC_G);
fprintf('Open-loop cutoff wc_ol     = %.6f rad/s\n', wc_ol);
fprintf('Open-loop cutoff fc_ol     = %.6f Hz\n', fc_ol);
fprintf('Phase at wc_ol             = %.2f deg\n', ph_wc_ol);

figure(4); clf;
bode(G); grid on;
title('Bode Plot – Open Loop G(s)');

%% ==================== Q8: CLOSED-LOOP CUTOFF (Tref) ====================
fprintf('\n=== Q8: CLOSED-LOOP CUTOFF FREQUENCY (Tref) ===\n');

[magT, phT] = bode(Tref, w);
magT = squeeze(magT);
phT  = squeeze(phT);

DC_T = 1;                            % with integral action, Tref(0) ~ 1 (manual assumption for tracking)
targetT = DC_T/sqrt(2);              % -3 dB relative to DC
```

```
[wc_cl, fc_cl, ph_wc_cl] = cutoff_from_mag_robust(w, magT, phT, targetT);

if ~isfinite(fc_cl) || fc_cl<=0
   error('Q8: cutoff not found for Tref. Increase sweep or re-check cutoff definition.');
end

fprintf('Closed-loop DC gain |T(0)| = %.8f\n', DC_T);
fprintf('Closed-loop cutoff wc_cl   = %.6f rad/s\n', wc_cl);
fprintf('Closed-loop cutoff fc_cl   = %.6f Hz\n', fc_cl);
fprintf('Phase at wc_cl           = %.2f deg\n', ph_wc_cl);

figure(5); clf;
bode(Tref); grid on;
title('Bode Plot – Closed Loop T_{ref}(s)');

%% ==================== Q9: COMPARISON ====================
fprintf('\n=== Q9: OPEN vs CLOSED LOOP COMPARISON ===\n');

fprintf('Open-loop  fc_ol = %.6f Hz\n', fc_ol);
fprintf('Closed-loop fc_cl = %.6f Hz\n', fc_cl);

if isfinite(fc_ol) && isfinite(fc_cl) && fc_ol>0
   fprintf('Bandwidth ratio fc_cl/fc_ol = %.3f x\n', fc_cl/fc_ol);
end

%% ==================== Q10: GAIN & PHASE MARGINS ====================
fprintf('\n=== Q10: GAIN & PHASE MARGINS (Loop L=C*G) ===\n');

% note: margin() is a toolbox calculation; keep only if your instructor allows it
[gm, pm, wcg, wcp] = margin(L);
```

```
if isinf(gm)
   gm_dB = Inf;
else
   gm_dB = 20*log10(gm);
end

fprintf('Phase Margin (PM)  = %.2f deg\n', pm);
fprintf('Gain Margin  (GM)  = %s dB\n', ternary(isinf(gm_dB),'Inf',num2str(gm_dB)));
fprintf('Gain crossover wcp = %.6f rad/s\n', wcp);
fprintf('Phase crossover wcg= %.6f rad/s\n', wcg);

figure(6); clf;
margin(L); grid on;
title('Margin Plot – Loop L(s)=C(s)G(s)');

%% ===================== Q11: SINUSOIDAL INPUT ANALYSIS ====================
fprintf('\n=== Q11: SINUSOIDAL INPUT ANALYSIS ===\n');

fprintf('u(t) = 5 + sin(2*pi*f*t)\n');
fprintf('Open-loop input -> plant force (G)\n');
fprintf('Closed-loop input -> reference r(t) (Tref)\n');

if ~isfinite(fc_ol) || fc_ol<=0, error('Q11: fc_ol invalid. Check Q7.'); end
if ~isfinite(fc_cl) || fc_cl<=0, error('Q11: fc_cl invalid. Check Q8.'); end

t_sin = (0:0.001:10).';    % time vector for sinusoidal tests (column)

u1    = 5 + sin(2*pi*1*t_sin);
u_fcOl = 5 + sin(2*pi*fc_ol*t_sin);
r1    = 5 + sin(2*pi*1*t_sin);
r_fcCl = 5 + sin(2*pi*fc_cl*t_sin);
```

```
assert(all(isfinite(u1)) && all(isfinite(u_fcOl)) && all(isfinite(r1)) && all(isfinite(r_fcCl)));

y_ol_1Hz = lsim(G,    u1,    t_sin);
y_ol_fc  = lsim(G,    u_fcOl, t_sin);
y_cl_1Hz = lsim(Tref, r1,    t_sin);
y_cl_fc  = lsim(Tref, r_fcCl, t_sin);

amp = @(sig) (max(sig(end-2000:end)) - min(sig(end-2000:end)))/2;

fprintf('Amplitude (steady-state ~ last 2s):\n');
fprintf('  OL @ 1 Hz    : %.6f\n', amp(y_ol_1Hz));
fprintf('  OL @ fc_ol   : %.6f\n', amp(y_ol_fc));
fprintf('  CL @ 1 Hz    : %.6f\n', amp(y_cl_1Hz));
fprintf('  CL @ fc_cl   : %.6f\n', amp(y_cl_fc));

figure(7); clf;

subplot(2,2,1);
plot(t_sin, u1, 'k--', t_sin, y_ol_1Hz, 'b', 'LineWidth', 1.2); grid on;
title('Open Loop: 1 Hz'); xlabel('t (s)'); ylabel('y');
legend('u(t)','y(t)','Location','best'); xlim([8 10]);

subplot(2,2,2);
plot(t_sin, u_fcOl, 'k--', t_sin, y_ol_fc, 'b', 'LineWidth', 1.2); grid on;
title(sprintf('Open Loop: f_{c,ol}=%.3f Hz', fc_ol)); xlabel('t (s)'); ylabel('y');
legend('u(t)','y(t)','Location','best'); xlim([8 10]);

subplot(2,2,3);
plot(t_sin, r1, 'k--', t_sin, y_cl_1Hz, 'r', 'LineWidth', 1.2); grid on;
title('Closed Loop (Tref): 1 Hz'); xlabel('t (s)'); ylabel('y');
```

```
legend('r(t)','y(t)','Location','best'); xlim([8 10]);

subplot(2,2,4);
plot(t_sin, r_fcCl, 'k--', t_sin, y_cl_fc, 'r', 'LineWidth', 1.2); grid on;
title(sprintf('Closed Loop (Tref): f_{c,cl}=%.3f Hz', fc_cl)); xlabel('t (s)'); ylabel('y');
legend('r(t)','y(t)','Location','best'); xlim([8 10]);

%% ===================== FINAL CONSOLE SUMMARY =====================
fprintf('\n============================================================\n');
fprintf('--- FINAL CONSOLE SUMMARY ---\n');
fprintf('============================================================\n');

fprintf('\n--- SYSTEM IDENTIFICATION ---\n');
fprintf('wn  = %.3f rad/s\n', wn);
fprintf('zeta = %.3f\n', zeta);
fprintf('k = %.4f N/m\n', k);
fprintf('c = %.4f Ns/m\n', c);
fprintf('Tp = %.4f s\n', Tp);
fprintf('Overshoot = %.2f %%\n', OS_pct);
fprintf('Ts (measured,2%%) = %.4f s\n', Ts_meas);

fprintf('\n--- PID CONTROL RESULTS (MANUAL from step response) ---\n');
fprintf('Kp=%.2f  Ki=%.2f  Kd=%.2f  Tf=%.5f\n', Kp, Ki, Kd, Tf);
fprintf('Tp = %.4f s\n', Tp_cl);
fprintf('Overshoot = %.2f %%\n', OS_cl);
fprintf('Ts = %.4f s\n', Ts_cl);

fprintf('\n--- FREQUENCY DOMAIN RESULTS ---\n');
fprintf('Open Loop Cutoff Frequency   = %.3f rad/s\n', wc_ol);
fprintf('Closed Loop Cutoff Frequency = %.3f rad/s\n', wc_cl);
fprintf('Phase Margin = %.2f deg\n', pm);
```

```
fprintf('Gain Margin  = %s dB\n', ternary(isinf(gm_dB),'Inf',num2str(gm_dB)));

fprintf('\nDONE.\n');

%% ===================== LOCAL FUNCTIONS ====================
function Ts = settling_time_2pct(tvec, yvec, yss)
% 2% settling time from data:
% Ts is taken as the first time after the last sample outside the ±2% band around yss
  if abs(yss) < 1e-12
    Ts = NaN; return;
  end
  band = 0.02*abs(yss);
  last_out = find(abs(yvec - yss) > band, 1, 'last');
  if isempty(last_out)
    Ts = tvec(1);
  elseif last_out < numel(tvec)
    Ts = tvec(last_out + 1);
  else
    Ts = NaN;
  end
end

function [wc, fc, ph_wc] = cutoff_from_mag_robust(w, mag, ph, targetMag)
% robust cutoff finder (relative to DC):
% - find resonance peak
% - search after the peak for first crossing below targetMag
% - interpolate on log(w) for smoother estimate
  wc = NaN; fc = NaN; ph_wc = NaN;

  if any(~isfinite(mag)) || any(~isfinite(w))
    return;
```

```
end

[~, imax] = max(mag);
startIdx = max(2, imax);

idx_rel = find(mag(startIdx:end) <= targetMag, 1, 'first');
if isempty(idx_rel)
   return;
end
idx = startIdx + idx_rel - 1;

if idx <= 1
   return;
end

w1 = w(idx-1); w2 = w(idx);
m1 = mag(idx-1); m2 = mag(idx);
p1 = ph(idx-1);  p2 = ph(idx);

% if target is not bracketed, fall back to the nearest sample
if (m1-targetMag)*(m2-targetMag) > 0
   wc = w(idx);
   fc = wc/(2*pi);
   ph_wc = ph(idx);
   return;
end

logw = interp1([m1 m2], log10([w1 w2]), targetMag, 'linear', 'extrap');
wc = 10^(logw);
fc = wc/(2*pi);
ph_wc = interp1(log10([w1 w2]), [p1 p2], log10(wc), 'linear', 'extrap');
```

```
end


function out = ternary(cond, a, b)
% small helper: returns a if cond is true, otherwise returns b
    if cond, out = a; else, out = b; end
end
```