Design and Analysis of a Servo-Controlled Robot Cart System With Four Technique

ECE 4115

Date: 26th November, 2024

Name:	Student ID:
Hussein Aljaroudi	2418747

1 Executive Summary

This report focuses on the design, analysis, and simulation of a servo-controlled robot cart system that moves along a path from a starting location near a wall to a defined reference location near a barrier. The project entails multiple steps, including system modeling, performance analysis, and controller design to optimize the system's stability and response. The analysis begins with an uncontrolled system, followed by root locus techniques, proportional-derivative (PD), and proportional-integral-derivative (PID) controller designs. Each method's performance is evaluated in terms of system poles, frequency response, and static error.

The aim is to find the most effective controller design to achieve maximum performance and stability while meeting the required specifications. Using techniques such as root locus, frequency response analysis, and error analysis, the study contrasts the uncontrolled and controlled systems to determine the most suitable controller for the final design. Based on the results, a recommendation will be provided for the optimal control strategy.

This project integrates theoretical concepts and practical tools to solve a real-world problem, emphasizing the importance of control systems in modern engineering applications. By following the structured methodology outlined in this report, the reader will gain insight into the process of analyzing and designing control systems for robotic applications.

2 Introduction

The purpose of this project is to design and analyze a servo-controlled robot cart system capable of moving from a starting position near a wall to a target location near a barrier. The system is modeled as a servo position system, as shown in the block diagram Figure 1, with specific parameters provided for each subsystem.

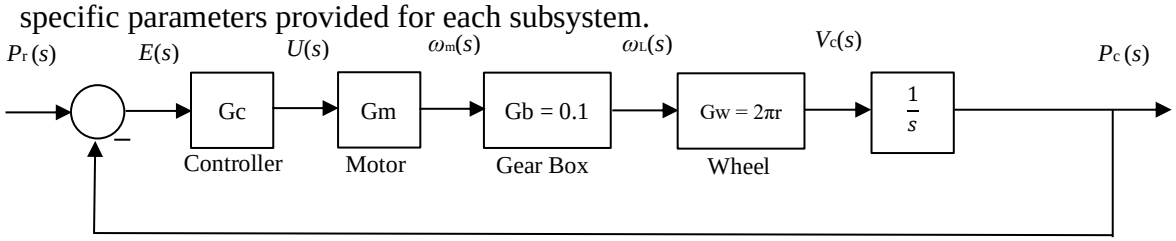

Figure 2: Block diagram of the servo position system

The problem requires analyzing the system performance under various control strategies, including no control, root locus-based control, PD control, and PID control. The goal is to determine the best control technique that ensures the system operates with optimal stability and performance.

The project begins by analyzing the uncontrolled system to understand its open-loop behavior, including system poles, frequency response, and static error. Root locus analysis is then employed to design a proportional gain controller that improves system performance. Finally, PD and PID controllers are designed to further enhance the system's dynamic response and stability. These designs are evaluated and compared to determine the most effective solution.

In this report, detailed mathematical modeling, simulations, and analysis are conducted to justify the design decisions. Figures, tables, and equations are included to support the findings and provide clarity. By the end of the report, the most suitable control method will be recommended based on a comprehensive comparison of all approaches.

3 Body

he primary goal of this project is to design and analyze a robot cart control system that ensures the cart moves accurately and efficiently from its starting position near a wall to a reference location near a barrier. The system is modeled using the block diagram representation (Figure 1), which includes components such as the motor, gearbox, wheel, and controller. The motor and load parameters are provided in **Table 3.1**, while the signals in the block diagram are detailed in **Table 3.2**.

Table 3.1: System specifications.

K_m	70 oz-in/A
K_b	0.0185 V/rpm
B	0.1 oz-in^2/s
R_a	3.50Ω
L_a	0.00075H
J_m	0.01 oz-in^2
J_L	6.74 oz-in^2
r	1.25 (wheel radius)
m	15 lb

Table 3.2: Signals in the block diagram

$P_r(s)$	70 oz-in/A
$E(s)$	0.0185 V/rpm
$U(s)$	0.1 oz-in^2/s
$\omega_m(s)$	3.50Ω
$\omega_L(s)$	0.00075H
$Vc(s)$	0.01 oz-in^2
$P_c(s)$	6.74 oz-in^2

Before analyzing each system, the servo motor model is identified, as shown in **Figure 3.1**. Next, the closed-loop gain of the servo motor is derived to determine the loop gain of the system. The analysis is represented mathematically in **Equation 3.1** and **Equation 3.2**.

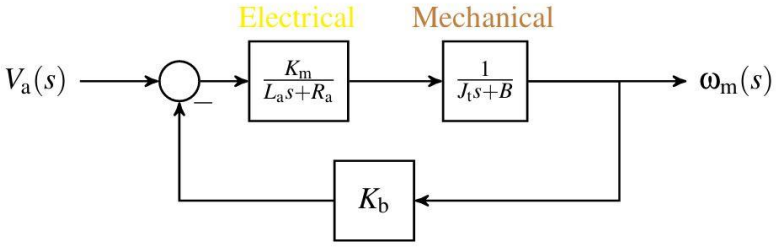

Figure 3.1: Block diagram of the servo-motor system.

The closed-loop gain for the servo-motor

$$\frac{\omega_m(s)}{V_a(s)} = G_m(s) = \frac{K_m}{L_a J_t s^2 + (L_a B + R_a J_t)s + R_a B + K_m K_b} \tag{3.1}$$

The loop-gain for the system using MATLAB

$$L(s) = \frac{54.978}{0.005S^3 + 0.675S^2 + 1.305S} \tag{3.2}$$

3.1 Analyze the basic system without control effort, Gc = 1

The loop-gain for this part will be the equation (3.2) and the system performance is demonstrate in the table 3.1. Note that *ep*, *ev* and *ea* represent static error analysis (position, velocity, and acceleration). For *PM* and *GM* they are the phase and gain margin respectively.

Table 3.3: Measurement of the system without control effort.

Pole1	Pole2	Pole3	ep(∞)	ev(∞)	ea(∞)	3dB BW	GM_{dB}	PM_{dB}
-132.020	0.660+9.050j	0.660-9.050j	0	0	0	14.011 rad/s	10.008dB	8.422°

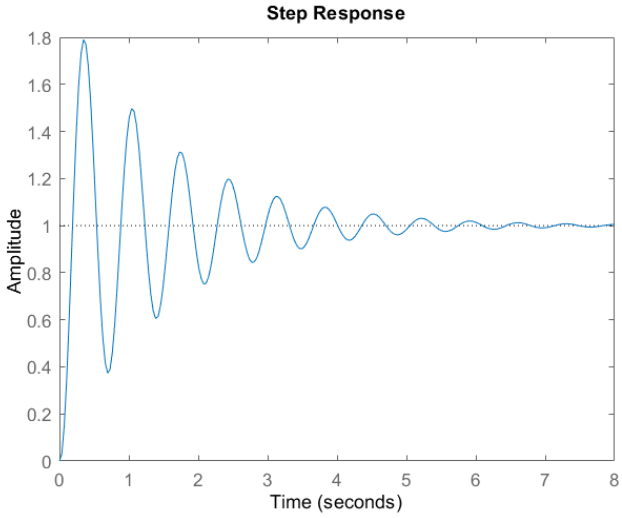

Figure 3.2: Step response of effortless controller.

3.2 Using root locus analysis techniques

The root locus analysis was performed with Gc=*K*, showing how the system poles vary with changes in the gain *K*. The root locus diagram is shown in **Figure 3.3**.

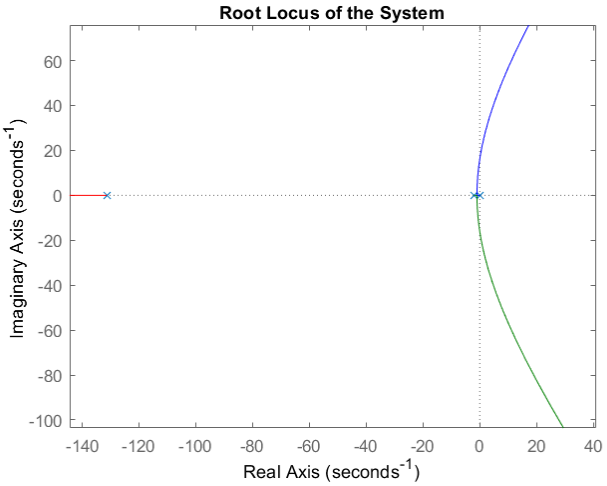

Figure 3.3: Root locus system plot.

To select a gain Gc=K for maximum performance and stability, the MATLAB command sisotool was used. By varying the value of KKK, the output from the step response graphs was analyzed while considering stability and frequency response. As shown in **Figure 3.3**, a gain of K=0.047 was determined to be optimal, achieving an overshoot of 16.9%. The system performance with this gain is summarized in **Table 3.4**

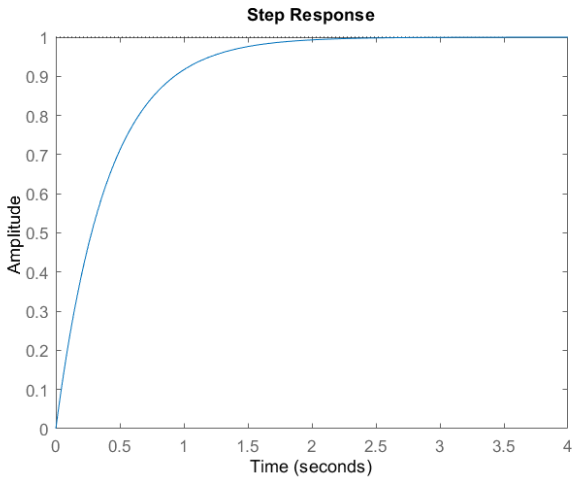

Figure 3.4: Step response of root locus system.

Table 3.4: Root locus system performance.

Pole1	Pole2	Pole3	ep(∞)	ev(∞)	ea(∞)	3dB BW	GM_{dB}	PM_{dB}
-131.420	0.970+1.720j	0.970+1.720j	0	0	0	2.5268 rad/s	36.566dB	50.967°

3.3.1 PD controller systems

Using PD control, the system was further optimized for better performance. The MATLAB pidTuner command was used to determine the proportional (Kp) and derivative (Kd) gains that maximize stability and response. The parameters of the PD controller are shown in **Table 3.5**.

Table 3.5: Paramters of PD controller.

Kd	Kp	Overshoot	Rise Time	Settling Time
0.410	**1.305**	**2.73%**	**0.0457sec**	**0.242 sec**

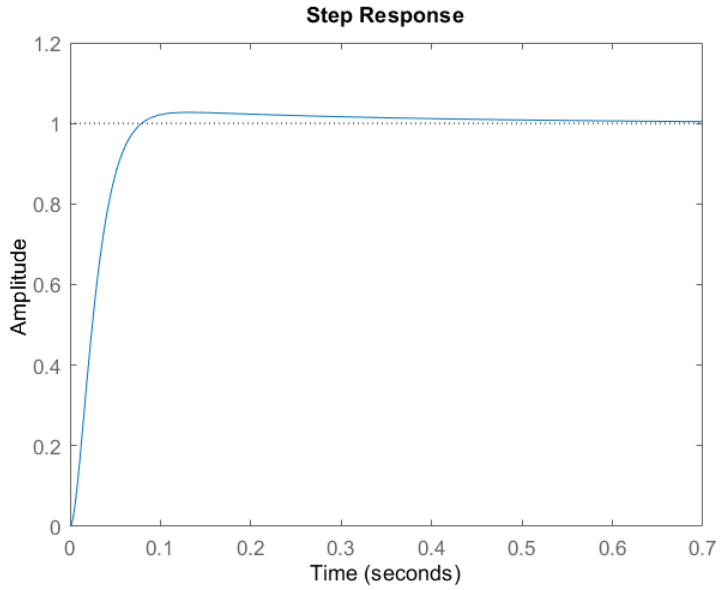

Figure 3.5: Step response plot of the system with PD controller.

Table 3.6 :PID controller systems Parameters

Kd	Kp	Ki	Overshoot	Rise Time	Sattling Time
0.417	**0.853**	**0.356**	**0.329%**	**0.048 sec**	**0.081 sec**

Step response of the PID controller is shown in Figure 3.6

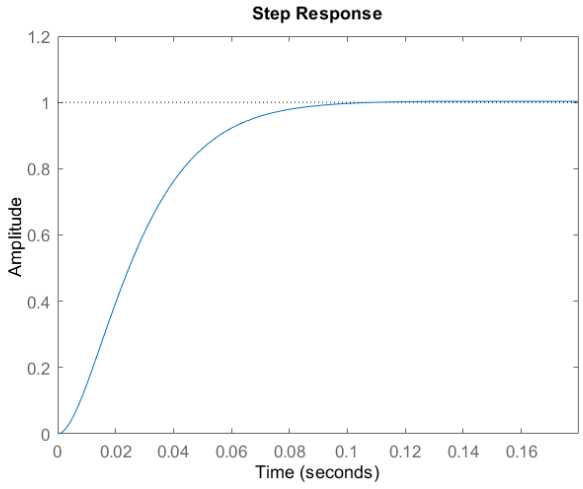

Figure 3.6 Step response plot of the system with PID controller.

3.3.3 Frequency analysis of PID and PD

The frequency response for PD and PID is summarize in the **Table 3.7**.

Table 3.7: Frequency analysis of PID and PD

	Pole1	Pole2	Pole3	Pole4	ep(∞)	ev(∞)	ea(∞)	3dB BW	GM_{dB}	PM_{dB}
PD	-65.030 + 8.038i	-65.0300 - 8.0379i	-3.2882	Na	0	0	0	45.2977 rad/s	∞dB	73.8°
PID	-65.661 +14.426i	-65.662 - 14.426i	-1.424	-0.601	0	0	0	44.7947 rad/s	-∞ dB	75.6°

3.4 Compression between all the systems

Based on the results presented in Figure 3.3, the PID controller, with its optimized maximum gain, provides the fastest step response and achieves the lowest steady-state error when compared to the PD controller. Additionally, when comparing the PID controller to both the Root Locus and Effortless controllers, the PID still outperforms the others in terms of response time. All systems considered are stable and theoretically exhibit zero static error, meaning the cart reaches its final position and settles in just 0.0807 seconds with no static deviation. The gain margin (GM) is -∞ and the phase margin (PM) is 75.6°, which indicates a highly stable system. However, for practical applications, it is generally preferable for a control system to have a gain margin greater than 6 dB and a phase margin greater than 30°[2], ensuring not only stability but also robustness against external disturbances and changes in system parameters.

4 Conclusion

This paper has explored four primary types of controller analysis to develop the most effective controller for a robot cart system. The system moves along a defined path, starting near a wall and reaching a reference location near a barrier. The analysis began with an uncontrolled system, followed by Root Locus analysis, and concluded with PD and PID controller evaluations. MATLAB software was the primary tool for conducting this analysis, enabling the determination of system poles, frequency response, steady-state error, step response, and the optimal gain (K) to ensure system stability. Among the controllers analyzed, the PID controller demonstrated the best performance compared to the PD controller, Root Locus-based controller, and the uncontrolled system. This superior performance is primarily due to the PID controller's ability to combine proportional, integral, and derivative gains, offering greater flexibility in tuning and achieving the desired system performance metrics.

References

[1] R. Provence, “Lecture notes on servo-motor,” Mar. 2024.

[2] 1] N. S. Nise, *Control Systems Engineering*, 6th ed. Hoboken, NJ, USA: Wiley, 2011, pp. 580–600.

A Appendix (MATLAB Code)

```
Km = 70;

Kb = 0.0185;

B = 0.1;

Ra = 3.5;

La = 0.00075;

Jm = 0.01;

JL = 6.74;

Jt = JL + Jm;  % Corrected to include Jm

r = 1.25;

m = 15;

Gb = 1/10;

Gw = 2*pi*r;

IN = tf(1, [1, 0]);  % Define 1/s which is Integrater


% Transfer Functions

EE = tf(Km, [La, B]);  % Electrical transfer function

ME = tf(1, [Jt, B]);   % Mechanical transfer function


% Closed-loop Transfer Function

Gm = feedback(EE * ME, Kb);


Gc = 1;


Ls = Gm*Gw*Gb*Gc*IN;
```

```
TF = feedback(Gm*Gw*Gb*Gc*IN,1);
step(TF);



disp(Ls)



%% b, i
poles_Gc1 = pole(TF);
disp('Poles of the Transfer Function:');
disp(poles_Gc1);

zero_Gc1 = zero(TF);  % Find the zeros
disp('Zeros of the Transfer Function:');
disp(zero_Gc1);

%% b, ii

% Frequency Response Analysis (Bode Plot)
figure;
bode(Ls);
grid on;
title('Bode Plot');

% 3dB Bandwidth
```

```
bw = bandwidth(TF);
disp(['3dB Bandwidth: ', num2str(bw), ' rad/s']);

% Gain and Phase Margins
[Gm, Pm, Wcg, Wcp] = margin(Ls);
disp(['Gain Margin (dB): ', num2str(20*log10(Gm))]);
disp(['Phase Margin (degrees): ', num2str(Pm)]);
disp(['Gain Crossover Frequency (rad/s): ', num2str(Wcg)]);
disp(['Phase Crossover Frequency (rad/s): ', num2str(Wcp)]);

%% b, iii, Static Error Constants
Kp = dcgain(Ls);  % Position error constant
disp(['Position Error Constant (Kp): ', num2str(Kp)]);
ep = 1/(1+Kp);
disp(['Position Error (ep): ', num2str(ep)]);

Kv = dcgain(Ls * tf(1,[1, 0]));  % Velocity error constant
disp(['Velocity Error Constant (Kv): ', num2str(Kv)]);
ev = 1/Kv;
disp(['Velocity Error (ev): ', num2str(ev)]);

Ka = dcgain(Ls * tf(1,[1, 0, 0]));  % Acceleration error constant
disp(['Acceleration Error Constant (Ka): ', num2str(Ka)]);
ea = 1/Ka;
disp(['Acceleration (ea): ', num2str(ea)]);
%% Part 2.a
```

```
% A) Plot the root locus of the system
syms s K;


 rlocus(Ls);
title('Root Locus of the System');

%% B Select a gain, Gc = K, for maximum performance and stability
sisotool(Ls)
Kmax = 0.047; % since it has the best preformence from sisotool
%% Part 2.c system preformnce
Gcr = 0.047;
Lsrkmax = Gm*Gw*Gb*Gcr*IN;
TFrkmax = feedback(Lsrkmax,1);

Poles_R_locus = pole(TFrkmax);
disp('Poles of the Transfer Function K Max Using Root Locus:');
disp(Poles_R_locus);

% 2.c.ii.) Frequency response analysis
% 3dB Bandwidth
bw = bandwidth(TFrkmax);
disp(['3dB BandwidthK Max Using Root Locus: ', num2str(bw), ' rad/s']);

% Gain and Phase Margins
[Gm, Pm, Wcg, Wcp] = margin(Lsrkmax);
```

```
disp(['Gain Margin (dB) K Max Using Root Locus: ', num2str(20*log10(Gm))]);
disp(['Phase Margin (degrees) K Max Using Root Locus: ', num2str(Pm)]);
disp(['Gain Crossover Frequency (rad/s) K Max Using Root Locus: ', num2str(Wcg)]);
disp(['Phase Crossover Frequency (rad/s)K Max Using Root Locus: ', num2str(Wcp)]);

step(TFrkmax)


%% Part 3.a & b
pidTuner(Ls, 'PID') %Using this tool best preformance can be obtined from the step response


%% PD with Maximum K
Kp_PD = 1.300; % Obtined from PD Tuner
Kd_PD = 0.411;

PD = tf([Kd_PD Kp_PD],1); % PD tf
display(PD);

LsPD = PD*Ls;
display(LsPD);
step(feedback(PD*Ls,1));

%% PID with Maximum K
Kp_PID = 0.853; % Obtined from PD Tuner
Kd_PID = 0.417;
```

```
Ki = 0.356;


PID = tf([Kd_PID Kp_PID Ki],[1 0]);
display(PID)


Ls_PID = PID*Ls;
display(Ls_PID);
step(feedback(PID*Ls,1))


%% PD  Controller System Performance
TF_PD = feedback(LsPD,1);
p = pole(TF_PD);
disp('Poles of the Transfer Function:');
disp(p);


% 3dB Bandwidth
bw = bandwidth(TF_PD);
disp(['3dB Bandwidth: ', num2str(bw), ' rad/s']);


% Gain and Phase Margins
[Gm, Pm, Wcg, Wcp] = margin(LsPD);
disp(['Gain Margin (dB): ', num2str(20*log10(Gm))]);
disp(['Phase Margin (degrees): ', num2str(Pm)]);
disp(['Gain Crossover Frequency (rad/s): ', num2str(Wcg)]);
disp(['Phase Crossover Frequency (rad/s): ', num2str(Wcp)]);
```

```
%% 3.c, iii, PD Static Error Constants
Kp = dcgain(LsPD);  % Position error constant
disp(['Position Error Constant (Kp): ', num2str(Kp)]);
ep = 1/(1+Kp);
disp(['Position Error (ep): ', num2str(ep)]);

Kv = dcgain(LsPD * tf(1,[1, 0]));  % Velocity error constant
disp(['Velocity Error Constant (Kv): ', num2str(Kv)]);
ev = 1/Kv;
disp(['Velocity Error (ev): ', num2str(ev)]);

Ka = dcgain(LsPD * tf(1,[1, 0, 0]));  % Acceleration error constant
disp(['Acceleration Error Constant (Ka): ', num2str(Ka)]);
ea = 1/Ka;
disp(['Acceleration (ea): ', num2str(ea)]);

%% PID  Controller System Performance
TFPID = feedback(Ls_PID,1);
p = pole(TFPID);
disp('Poles of the Transfer Function:');
disp(p);

% 3dB Bandwidth
bw = bandwidth(TFPID);
disp(['3dB Bandwidth: ', num2str(bw), ' rad/s']);
```

```
% Gain and Phase Margins
[Gm, Pm, Wcg, Wcp] = margin(Ls_PID);
disp(['Gain Margin (dB): ', num2str(20*log10(Gm))]);
disp(['Phase Margin (degrees): ', num2str(Pm)]);
disp(['Gain Crossover Frequency (rad/s): ', num2str(Wcg)]);
disp(['Phase Crossover Frequency (rad/s): ', num2str(Wcp)]);

%% 3.c, iii, PD Static Error Constants
Kp = dcgain(LsPID);  % Position error constant
disp(['Position Error Constant (Kp): ', num2str(Kp)]);
ep = 1/(1+Kp);
disp(['Position Error (ep): ', num2str(ep)]);

Kv = dcgain(LsPID * tf(1,[1, 0]));  % Velocity error constant
disp(['Velocity Error Constant (Kv): ', num2str(Kv)]);
ev = 1/Kv;
disp(['Velocity Error (ev): ', num2str(ev)]);

Ka = dcgain(LsPID * tf(1,[1, 0, 0]));  % Acceleration error constant
disp(['Acceleration Error Constant (Ka): ', num2str(Ka)]);
ea = 1/Ka;
disp(['Acceleration (ea): ', num2str(ea)]);
```