A Project Report on

State-Space Modeling and Control for Differential Drive Warehouse Robots

Submitted in partial fulfilment of the requirements of ECE 560 Modern Control Theory by

Mukul Bhatia	***MSE Robotics Engineering***
Sajin Shrestha	***MSE Robotics Engineering***
Vighnesh Deolikar	***MSE Electrical Engineering***

Under the Guidance of

Prof. John O'Donnel

Department of Electrical Engineering

University of Michigan - Dearborn

FALL 2024

Abstract

This project presents the development and analysis of a state-space model for autonomous differential drive robots operating within a warehouse environment. The primary goal is to enable dynamic, collision-free navigation for item retrieval tasks, achieved through real-time path planning and control. The state-space model captures essential states of the robot, including position, heading angle, and velocity, with control inputs for linear and angular velocity to govern movement.

Key analyses, including controllability, stability, and observability, confirmed that the robot system is fully controllable and observable, with stable response characteristics. These properties ensure the robot can reach any target position, maintain predictable behavior, and allow for accurate state estimation even when all states are not directly measurable. Based on these results, the project will advance to the design and testing of a controller and observer to enhance the robot's adaptability to real-time changes in the warehouse environment. This approach lays the groundwork for efficient, safe, and scalable warehouse automation.

Contents

Sr.		Title	Page
1		Introduction	
	1.1	Background	4
	1.2	Aim	4
	1.3	Objective	4
2		Design	
	2.1	Physical System	5
	2.2	State Space Model	5
	2.3	Dynamics of Robot	6
	2.4	Feedback Gain	7
	2.5	Stability, Controllability and Observability	8
	2.6	LQR Controller Design	8
	2.7	Observer Design	8
3		Testing of System	
	3.1	Vehicle Dynamics Test	10
	3.2	LQR Controller Test	10
	3.3	Observer Test	10
	3.4	Effect of External Parameters	11
4		Conclusion	
	4.1	Result	12
	4.2	Future Scope	12
5		References	13
6		Appendix	14

List of Figures

Figure No.	Title	Page
Figure 1	Differential Drive Robot in X-Y Plane	5
Figure 2	Test output for Vehicle Dynamics.	10
Figure 3	Test output for LQR Controller.	10
Figure 4	Test output for Observer Design.	11
Figure 5	Robot Path correction with Q compensation.	11
Figure 6	Robot Path deviation due to external parameters.	11

1 Introduction

1.1 Background:

Warehouse automation has become an essential part of modern logistics, driven by the demand for faster, more accurate and efficient handling of goods. Robotic system plays an important role in the warehouse automation where repetitive tasks, minimizing errors and reduced human intervention is required.

Among various Robotic system configuration differential drive robots are popular options in the warehouse automation. The differential drive robots simple and adaptable in controlled environments. The robots have two independently controlled wheels to retrieve and transport various items and products across the warehouse. The differential drive mechanism offers precise control over the robots heading velocity and direction making it ideal for navigating confined spaces of typical warehouse setting.

This project aims to develop a state space model for a differential drive robot with Collison avoidance capabilities, focusing on capturing the robot's position, velocity and heading angle as state variables. The model also provides a framework for simulating and analyzing robot motion and interaction in a warehouse setting laying the ground work for real time control and coordination in multi-robot system.

1.2 Aim:

The primary aim of the project is to develop a state space model for the differential drive robots in warehouse setting. The model will capture states like position, velocity, heading angle etc. of robot to simulate and control its motion effectively.

The state space representation will provide the foundation for the various control aspects of the robot like path navigation, collision avoidance and multi robot operational environment with a future scope of enhancing productivity and operational safety.

1.3 Objective:

1. Designing of state space representation of warehouse robot based on the system requirements.
2. Calculating and optimizing gain for the system to ensure efficient and responsive environment.
3. Analyzing the stability of the system to ensure consistent behavior.
4. Designing of controller based on precise calculation and requirements.
5. Designing observer for the robot to estimate unmeasured states or disturbances.

2 Design

2.1 Physical System:

The differential drive robots operate by controlling the speeds of two independent driven wheels on either side of the chassis of the robot. This type of setup allows for efficient control of the robot to move forward, backward turn left and right in limited spaces.

Modeling a differential drive robot for a warehouse environment where the robot needs to navigate and avoid obstacles to retrieve items. Here, the robots motion is governed by its velocity and heading angle with controls provided by linear acceleration and angular velocity.

A differential drive robot uses two wheels on either side to control forward/ backward movement and turning. Consider the following diagram.

Key Components:

- State Variables:
  x: Position of the robot in the x − direction.
  y: Position of the robot in the y − direction.
  θ: Heading angle (orientation) of the robot.
  v: Linear velocity of the robot.
- Inputs:
  u_1: Linear acceleration
  u_2: Angular Velocity

2.2 State Space Model:

The state space Model of a Differential Drive robot allows to create a mathematical representation which allows to analyze its dynamic behavior in terms of state variables, inputs and outputs. The model represents the robots movement capturing essential aspects like position, velocity, heading angle and trajectory.

Consider the following for the representation of State Space Model for the robot.

Non- Linear State Space Model:

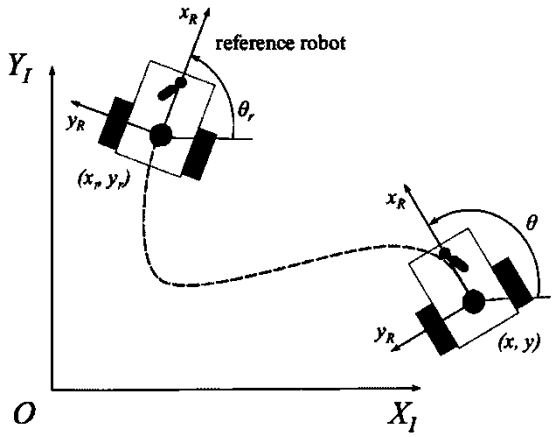

Figure 01: Differential Drive Robot in X-Y Plane

The differential drive robot's dynamics are:

$\dot{x} = v\cos(\theta)$
$\dot{y} = v\sin(\theta)$
$\dot{\theta} = u_2$
$\dot{v} = u_1$

This gives following state vector:

$$x = \begin{bmatrix} x \\ y \\ \theta \\ v \end{bmatrix}$$

The state space representation is:

$$\dot{x} = f(x,u) = \begin{bmatrix} v\cos(\theta) \\ v\sin(\theta) \\ u_2 \\ u_1 \end{bmatrix}$$

Linearized State space model:

$$\dot{x} = Ax + Bu$$

$$\text{Where } A = \begin{bmatrix} 0 & 0 & -v\sin(\theta_0) & \cos(\theta_0) \\ 0 & 0 & v\cos(\theta_0) & \sin(\theta_0) \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} and\ B = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 1 \\ 1 & 0 \end{bmatrix}$$

The above linearized equations will be solved in MALAB, where the simulation will involve applying different controls input u_1 and u_2.

Determining the output equations,

Fot the differential drive robot, the outputs will be:

- Position Cordinates: x, y
- Heading angle: θ
- Linear Velocity: v

For measuring all the state variables the output vector y would be,

$$y = \begin{bmatrix} x \\ y \\ \theta \\ v \end{bmatrix}$$

Assuming $y = x$, all the state variables are directly measurable as output,

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

If there is no direct dependence of the outputs on the inputs u_1 and u_2, then D will be zero matrix:

$$D = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}$$

2.3 Dynamics of Robot:

The control inputs u will be the velocities of the two wheels of the robot, which directly affect its movement.

$$u = \begin{bmatrix} v_R \\ v_L \end{bmatrix}$$

Where,

v_R = Velocity of Right Wheel.
v_L = Velocity of Left Wheel.

The differential drive robot's motion can be described by the following kinematic equations:

- Position Dynamics:

$$\dot{x} = v\cos(\theta)$$
$$\dot{y} = v\sin(\theta)$$

- Heading Dynamics:

$$\dot{\theta} = \frac{v_R - v_L}{d}$$
$$d = distance\ between\ two\ wheels.$$

- Velocity Dynamics:

$$v = \frac{v_R + v_L}{2}$$

The path of the robot can be controlled by varying the speed of the wheels of the robot, consider the following cases:

- For straight Path: Set both wheel velocities equal ($v_R = v_L$) and the robot will move in a straight line.
- For Circular Path: Set the right and left wheel velocities slightly different, but not as extreme as in the spiral case.
- For Spiral Path: Set the right and left wheel velocities slightly different.

2.4 Feedback Gain:

Feedback gain is essential for controlling the robot's position, heading, and velocity to ensure accurate and stable navigation within the warehouse.

Considering following for the desired performance:

- Speed of the response: Ideally a robot should stele to a new position quickly hence should be within 1sec/2sec.
- Damping Ratio: For a good balance between speed and overshoot the damping ration should be around 0.7.
- Natural Frequency: For robots the natural frequency is 3 rad/sec to 5 rad/sec for the desired response. In this case, assuming 5 rad/sec.

Using the second-order system characteristics deriving the poles:

$$Poles = -\zeta w_n \pm j\omega_n\sqrt{1-\zeta^2}$$

$$-\zeta\omega_n = -0.7 \times 5 = -3.5 \ldots . Real\ Part$$

$$w_n\sqrt{1-\zeta^2} = 5\sqrt{1-0.7^2} \approx 5 \times 0.714 = 3.57 \ldots Imaginary\ part$$
$$Poles \approx -3.5 \pm j3.57$$
$$Additional\ real\ poles\ for\ faster\ settling$$
$$Real\ Poles = -5\ (for\ faster\ response)$$

After simulation in MATLAB the following gain matrix K was obtained,

$$\begin{bmatrix} 0.0088 \times 10^{-3} & 4.3998 \times 10^{-3} & 0.8812 \times 10^{-3} & 0.0068 \times 10^{-3} \\ 0.0089 \times 10^{-3} & 4.3867 \times 10^{-3} & 0.8761 \times 10^{-3} & 0.0068 \times 10^{-3} \end{bmatrix}$$

2.5 Stability, Controllability and Observability:

1. Stability: Stability ensures that the robot's state variables position, heading angle and velocity are bounded over time. For the given A matrix the eigenvalues are [0, 0, 0, 0]. This shows that the system is marginally stable.
2. Controllability: Controllability determines whether we can drive the robot to any desired state using available inputs. The given system is controllable as the rank of the matrix is 4.
3. Observability: Observability determines if the system's internal states (position, heading angle, and velocity) can be inferred from the outputs. The given system is observable as the rank of the matrix is 4.

2.6 LQR Controller Design

The objective of the LQR controller design is to minimize the cost function where

$$J = \int_0^\infty (x^T Q x + u^T R u) dt$$

Where,

- Q: Penalizes deviations in states (prioritize position accuracy).

$$Q = diag(q_x, q_y, q_\theta, q_v)$$
$$Q = diag(10,10,1,1)$$

  Position deviations x and y, were assigned higher weights because accurate waypoint tracking is primary objective whereas Heading angle and velocity θ and v respectively were assigned because deviations in these states are less critical as long as the robot is converging to the target position.

- R: Penalizes control effort (avoid excessive actuator use).

$$R = diag(r_w, r_a)$$
$$R = diag(1,1)$$

  Equal weights were assigned to the control inputs because both singular velocity and acceleration are equally critical for smooth and stable control.

The LQR controller minimizes the cost function and the solution to the Algebraic Riccati Equation yieds the P matrix which is used to determine the gain matrix K [6.1].

$$A^T P + PA - PBR^{-1}B^T P + Q = 0$$
$$K = R^{-1}B^T P$$
$$where\ K = \begin{matrix} 3.1106 & 0.5693 & 0.4838 & 2.8204 \\ -0.5693 & 3.1106 & 2.6433 & 0.4838 \end{matrix}$$

The values for Q and R can be verified using the closed loop eigenvalues which are as follows: $-1.4060 \pm 1.3925i$ and $-1.3259 \pm 0.8921i$ which shows that the system is sable and of oscillating nature. Further accuracy can be increased by increasing the Q and R values but will cause the poles to move further from the origin/ Imaginary axis causing faster settling and almost no damping which is not ideal.

2.7 Observer Design

The Observer poles must be placed faster than the controller poles for a quicker response and to ensure the error dynamics $e(t) = x - \hat{x}$ decay rapidly. Therefore the observer poles should have larger negative real part compared to the controller poles from the LQR Design which are $-1.4060 \pm 1.3925i$ and $-1.3259 \pm 0.8921i$.

The observer time constant (τ) should be smaller than the system dynamics so,

$$\tau = \frac{1}{|\mathrm{Re}(\lambda)|}$$
$$as\ highest\ value\ negative\ real\ pole\ in\ controller\ is\ -1.4,$$

$$\tau_c \approx \frac{1}{1.4} \approx 0.71sec$$

$$Therefore, \tau_o < 0.71sec$$

$$assuming\ values\ which\ are\ further\ from\ origin:\ -5, -6, -7, -8$$

$$Hence, \tau_o \approx \frac{1}{5} = 0.2sec$$

$$which\ is\ 3-4\ times\ faster\ than\ the\ controller\ which\ is\ ideal\ for\ design\ consideration.$$

The observer gain matrix L is determined using the pole placement

$$\dot{\hat{x}} = A\hat{x} + Bu + L(y - \hat{y})$$

$$where\ error\ dynamics\ are\ \dot{e} = (A - LC)e\ where\ e = x - \hat{x}.$$

The obtained observer gain matrix is as follows [6.2]:

$$\begin{matrix} 13.5632 & 0.7281 \\ 1.9386 & 12.4368 \\ 19.6970 & 38.5752 \\ 45.1536 & 4.7165 \end{matrix}$$

3 Testing of System

3.1 Vehicle Dynamics Test

The vehicle path can be observed for different wheel speeds using MATLAB function of ode45 function[6.3].

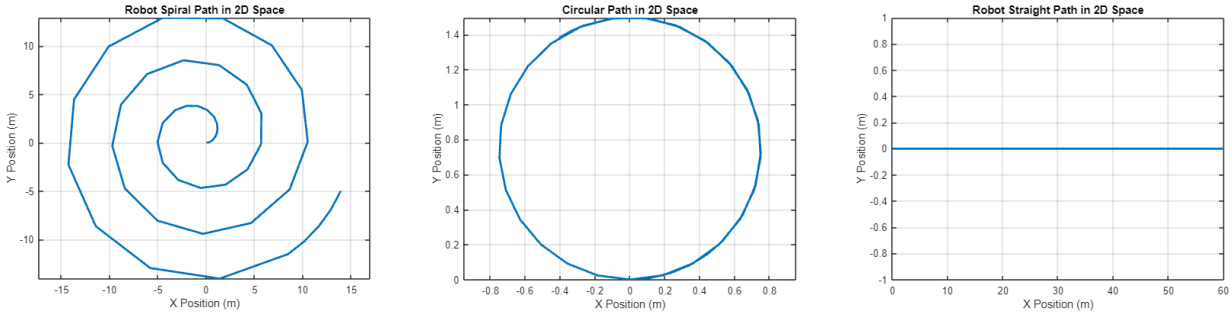

Figure 02: Test output for Vehicle Dynamics.

3.2 LQR Controller Test

The performance of the LQR controller is tested by enabling the robot to follow a set predefined path/waypoints in a 2D space. The waypoints are the predefined target position that the robot follows with the initial state as follows: [0,0,0,1] where the x_0 and y_0 coordinates along with θ_0 is 0 and the velocity v_0 is 1m/sec.

For each waypoint the LQR controller computes u which minimizes the cost function. The state is then propagated over time using MATLAB's ode45 function. The robot's current state and process is repeated after reaching a waypoint. Consider the following output showing LQR controller successfully navigating the robot trough all waypoints[6.4].

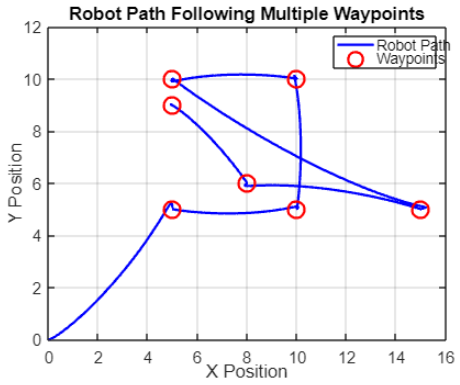

Figure 03: Test output for LQR Controller.

3.3 Observer Test

The performance of the observer in estimating the true states of the system (x, y, θ, v) from measured outputs (x, y) and comparing them against the actual states. MATLAB's ode45 solver is used to simulate the combined system (true states + observer dynamics) over a time span of 10 seconds. The observer successfully estimates the states in real time, allowing the controller to operate effectively using the estimated states instead of the true states.

The estimated states $(\hat{x})$ converge to the true states over time showing the effectiveness of the observer. Due to effective pole placement the observer converges faster and ensures rapid state estimation without overshooting. There is a small deviations in the initial phase are due to the large mismatch between the initial guess and the true state but reduce as the observer corrects itself. Consider the following output for observer design testing[6.5].

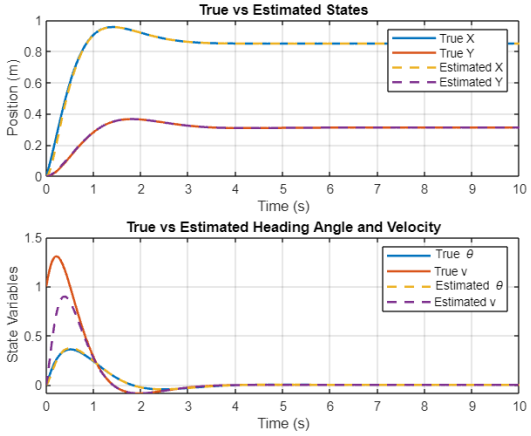

Figure 04: Test output for Observer Design.

3.4 Effect of External Parameters

The Robustness of the Control system is evaluated by subjecting the system to following conditions:

1. Variations in system parameters A and B to simulate modeling inaccuracies.
   A and BBB matrices were modified by adding random perturbations ($\Delta A + \Delta B$) up to 10% of their original values to simulate model uncertainties.
   $$so, A_{sim} = A + \Delta A \; and \; B_{sim} = B + \Delta B$$
2. External disturbances in the form of sinusoidal inputs affecting the system dynamics.
   Time-varying sinusoidal disturbances were applied to the system:
   $$disturbance = [0; 0; 0.1\sin(t)\,; 0.05\cos(t)]$$
3. Measurement noise introduced into the state variables.
   Random noise was added to the system with an amplitude of 0.10.10.1, modeled as Gaussian white noise:
   $$noise = N(0, 0.1^2)$$
4. Varying initial condition to a non-zero value.
   $$Initial\ state: [x_0, y_0, \theta_0, v_0] = [2,3,0.1,0.8]$$

The robot was able to successfully follow the given waypoints despite parameter uncertainties, external disturbances, and noise. However, slight deviations from the ideal trajectory were observed, but by adjusting the Penalizing factor Q to correct the robot trajectory which can be analyzed from the output below[6.6]:

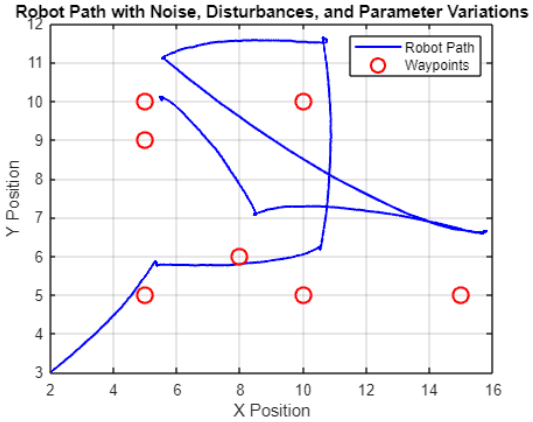

Figure 05: Robot Path deviation due to external parameters.

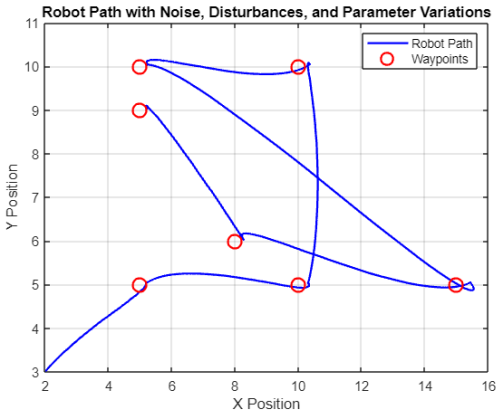

Figure 06: Robot Path correction with Q compensation.

The penalizing factor Q was set to [700,700,1,1] by increasing the penalizing factor from 10 to 700 for x and y coordinated the path deviation reduces significantly. But in real life implementation increasing Q value is not ideal. Hence, different methods like using high precision sensors, feed-forward control, shielding for noise is implemented.

4 Conclusion

4.1 Result

1. State Space Representation:
   Developed a state space representation for the differential drive robot in terms of position, heading angle and velocity which will be used in controller and observer design of the robot.
2. System Dynamics:
   The system dynamics of the robot was determined using MATLAB simulation by varying the velocity of the wheels to understand the robot's motion within the warehouse.
3. Feedback Gain of the system:
   Designed gain matrix for the system using pole placement to achieve the desired characteristics.
4. Controller Design:
   Designed a Linear Quadratic Regulator (LQR) for path tracking. The controller weights were tuned to prioritize accurate position control and reasonable actuator effort.
5. Observer Design:
   Designed a state observer using pole placement to estimate unmeasured states in real time.
6. Robustness Evaluation:
   Tested the system under parameter variations (A and B), external disturbances, and sensor noise.

The robot successfully navigated through all the set waypoints with accuracy with minimum deviation without external parameters and there was a slight deviation observed which can be controlled by compensating Q. The observer also estimated unmeasured states ensuring accurate control input. The robot also handled the noise, disturbance and other external parameters effectively real-world scenarios. The project achieved its primary objective of designing and implementing a robust control and estimation system for a differential-drive robot.

4.2 Future Scope

1. Real-world Implementation: The designed system can be tested on physical robots to validate its practical performance.
2. Adaptive Techniques: Explore adaptive control or observer methods to handle larger uncertainties and disturbances.
3. Extended Applications: Expand the framework for multi-robot coordination or 3D navigation systems.

5 References

1. Control Profile Parameterized Nonlinear Model Predictive Control of wheeled Mobile Robots (https://www.researchgate.net/publication/271702066_Control_Profile_Parameterized_Nonlinear_Model_Predictive_Control_of_wheeled_Mobile_Robots)
2. MATLAB ode 45 command (https://in.mathworks.com/help/matlab/ref/ode45.html) (https://in.mathworks.com/matlabcentral/answers/1938334-problem-with-calculating-forward-dynamics-using-ode45)
3. MATLAB LQR Command (https://in.mathworks.com/help/control/ref/lti.lqr.html)
4. MATLAB observer design (https://in.mathworks.com/matlabcentral/answers/1940104-state-feedback-control-and-observer)
5. S. Morales, J. Magallanes, C. Delgado and R. Canahuire, "LQR Trajectory Tracking Control of an Omnidirectional Wheeled Mobile Robot," 2018 IEEE 2nd Colombian Conference on Robotics and Automation (CCRA), Barranquilla, Colombia, 2018, pp. 1-5, doi: 10.1109/CCRA.2018.8588146. keywords: {Wheels;Mobile robots;Trajectory tracking;Trajectory;PI control;Omnidirectional robot;Trajectory control;LQR controller;PI controller;Cascade control},
6. Zefan Su, Hanchen Yao, Jianwei Peng, Zhelin Liao, Zengwei Wang, Hui Yu, Houde Dai, Tim C. Lueth, LQR-based control strategy for improving human–robot companionship and natural obstacle avoidance, Biomimetic Intelligence and Robotics, Volume 4, Issue 4, 2024, 100185, ISSN 2667-3797, https://doi.org/10.1016/j.birob.2024.100185. (https://www.sciencedirect.com/science/article/pii/S2667379724000433)
7. Autonomous Robots Lab (https://www.autonomousrobotslab.com/lqr-control.html)
8. A. De Luca, D. Schroder and M. Thummel, "An Acceleration-based State Observer for Robot Manipulators with Elastic Joints," Proceedings 2007 IEEE International Conference on Robotics and Automation, Rome, Italy, 2007, pp. 3817-3823, doi: 10.1109/ROBOT.2007.364064. keywords: {Acceleration;Manipulators;Robot sensing systems;Observers;Service robots;Gears;Belts;Shafts;Displays;Trajectory},
9. B. Bona and M. Indri, "Analysis and implementation of observers for robotic manipulators," Proceedings. 1998 IEEE International Conference on Robotics and Automation (Cat. No.98CH36146), Leuven, Belgium, 1998, pp. 3006-3011 vol.4, doi: 10.1109/ROBOT.1998.680887. keywords: {Observers;Position measurement;Robot control;State feedback;Robot sensing systems;Manipulator dynamics;Velocity control;Control systems;Robotics and automation;Service robots},

6 Appendix

For detailed codes please follow the link.

6.1 LQR Controller Design

```
% LQR weighting matrices
Q = diag([10, 10, 1, 1]); % Penalizing deviations in x, y more
R = diag([1, 1]);         % Penalizing control inputs
% Solve Riccati equation for P
P = care(A, B, Q, R);
% Compute gain matrix K manually
K = R \ (B' * P); % Equivalent to inv(R) * B' * P
disp('LQR Gain Matrix (K):');
disp(K);
eig_closed_loop = eig(A - B * K);
disp('Closed-Loop Eigenvalues:');
disp(eig_closed_loop);
```

6.2 Observer Design

```
% Define desired observer poles
observer_poles = [-5, -6, -7, -8]; % Choose faster poles than controller
L = place(A', C', observer_poles)'; % Compute observer gain matrix

disp('Observer Gain Matrix (L):');
disp(L);
```

6.3 Vehicle Dynamics Testing

```
% Function definition for robot dynamics in a circular path
function dxdt = robot_dynamics_circular(t, x, d, v_R, v_L)
    theta = x(3);
    v = x(4);
    % System dynamics
    dxdt = zeros(4,1);  % Initialize the state derivative vector
    dxdt(1) = v * cos(theta);  % x_dot
    dxdt(2) = v * sin(theta);  % y_dot
    dxdt(3) = (v_R - v_L) / d;  % theta_dot (angular velocity)
    dxdt(4) = (v_R + v_L) / 2;  % v_dot (linear velocity)
end
```

6.4 LQR Controller Testing

```
% Simulation parameters
initial_state = [0; 0; 0; 1]; % Initial position, heading angle, velocity
t_span = [0 5]; % Time for each segment
states_all = []; % To store states for the entire path
times_all = []; % To store time data for the entire path
% Loop through each waypoint
current_state = initial_state;
for i = 1:size(waypoints, 1)
    target_state = [waypoints(i, 1); waypoints(i, 2); 0; 0]; % Desired state for each
waypoint
    % Simulate the system for the current segment
```

```
    [t, states] = ode45(@(t, x) controlled_dynamics(t, x, A, B, K, target_state), t_span,
current_state);
    % Update the current state and store results
    current_state = states(end, :)'; % Update to the last state
    states_all = [states_all; states];
    times_all = [times_all; t + (i-1)*t_span(end)];
end
function dxdt = controlled_dynamics(t, x, A, B, K, target_state)
    u = -K * (x - target_state); % Control law
    dxdt = A * x + B * u;        % Dynamics with control
end
```

6.5 Observer Testing

```
% Simulate the system with the observer
initial_state = [0; 0; 0; 1]; % True initial state
initial_estimate = [0; 0; 0; 0]; % Initial guess for state estimation
t_span = [0 10];
[t, states] = ode45(@(t, x) system_with_observer(t, x, A, B, C, L, K, initial_state),
t_span, [initial_state; initial_estimate]);
% Extract true and estimated states
true_states = states(:, 1:4);
estimated_states = states(:, 5:8);
% Observer dynamics function
function dxdt = system_with_observer(t, x, A, B, C, L, K, true_state)
    n = length(true_state); % Number of states
    x_true = x(1:n); % True states
    x_hat = x(n+1:end); % Estimated states
    % Control law
    u = -K * (x_hat - true_state);
    % True dynamics
    dx_true = A * x_true + B * u;
    % Observer dynamics
    y = C * x_true; % True output
    y_hat = C * x_hat; % Estimated output
    dx_hat = A * x_hat + B * u + L * (y - y_hat);
    % Combine
    dxdt = [dx_true; dx_hat];
end
```

6.6 Effect of External Parameters

```
% Robustness: Modify A and B for simulation
delta_A = 0.1 * rand(size(A)); % Small variation in A
delta_B = 0.1 * rand(size(B)); % Small variation in B
A_sim = A + delta_A;
B_sim = B + delta_B;
% Simulation parameters
initial_conditions = [2; 3; 0.1; 0.8]; % Initial conditions
t_span = [0 5]; % Time for each segment
states_all = []; % To store states for the entire path
times_all = []; % To store time data for the entire path
current_state = initial_conditions; % Initialize current state
% Noise parameters
```

```
noise_amplitude = 0.1; % Adjust this for stronger noise

% Dynamics function for LQR-controlled system
function dxdt = controlled_dynamics(t, x, A, B, K, target_state, noise_amplitude)
    % External disturbance
    disturbance = [0; 0; 0.1 * sin(t); 0.05 * cos(t)];
    % Measurement noise
    noise = randn(4, 1) * noise_amplitude;
    % Control law
    u = -K * (x - target_state);
    % Dynamics with control, disturbance, and noise
    dxdt = A * x + B * u + disturbance + noise;
end
```