State-space Modeling And Control For Differential Drive Warehouse Robots

by

Mukul Bhatia | Sajin Shrestha | Vighnesh Deolikar

AIM AND OBJECTIVE

AIM:

- The primary aim of the project is to develop a state space model for the differential drive robots in warehouse setting.

OBJECTIVE:

- Designing of state space representation of warehouse robot.
- Calculating and optimizing gain for the system.
- Analyzing the stability of the system to ensure consistent behavior.
- Designing of controller based on precise calculation and requirements.
- Designing observer for the robot to estimate unmeasured states or disturbances.

STATE SPACE MODEL

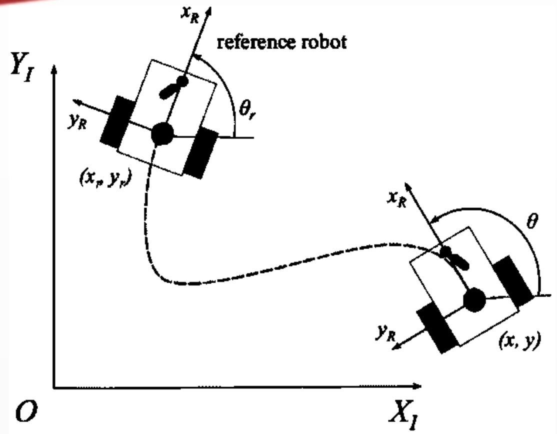

Figure 01: Differential Drive Robot in X-Y Plane

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

Where,

Kinematic Equations are as follows: $\dot{x} = v\cos(\theta)$

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

$$\dot{\theta} = u_2 = \frac{v_R - v_L}{d}$$

$$\dot{v} = u_1 = \frac{v_R + v_L}{2}$$

State Variables are:

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

State-Space Model:

$\dot{x} = Ax + Bu$ and $y = Cx$

Where $A = \begin{bmatrix} 0 & 0 & -v\sin(\theta_0) & \cos(\theta_0) \\ 0 & 0 & v\cos(\theta_0) & \sin(\theta_0) \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, B = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 1 \\ 1 & 0 \end{bmatrix}, C = [I_4], D = [0]$

FEEDBACK GAIN

Considering following for the desired performance:

- Speed of the response: 1sec/2sec.
- Damping Ratio: 0.7.
- Natural Frequency: 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,

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

```
desired_poles = [-3.5 + 3.57j, -3.5 - 3.57j, -5, -5];

K = place(A, B, desired_poles);

disp('Feedback Gain Matrix K:');
Feedback Gain Matrix K:
disp(K);
  1.0e+03 *

    0.0088    4.3998    0.8812    0.0068
     0.89     4.3867    0.8761    0.0068
```

VEHICLE DYNAMICS TESTING

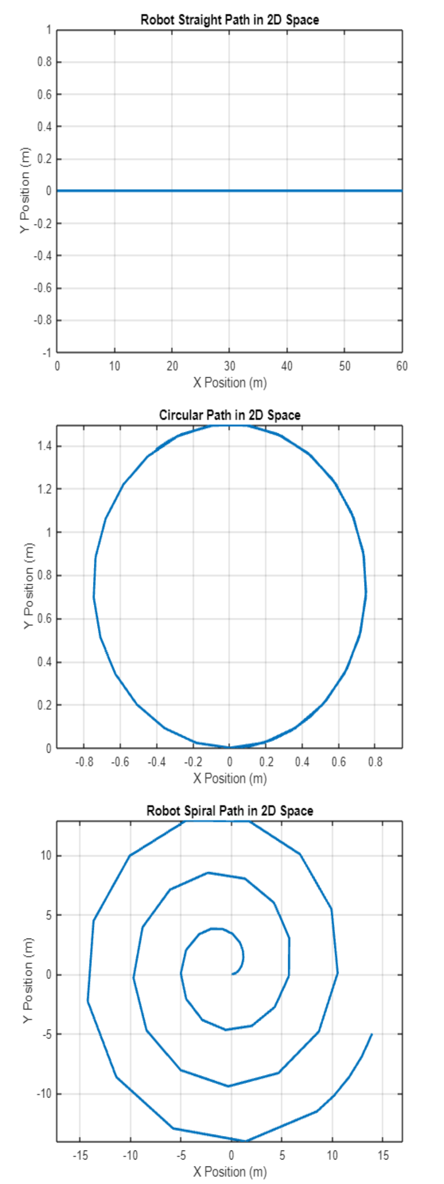

Figure 02: Test output for Vehicle Dynamics.

```
x0 = 0;  % Initial x position
y0 = 0;  % Initial y position
theta0 = 0;  % Initial heading angle (in radians)
v0 = (v_R + v_L) / 2;  % Initial velocity (average of both wheels)
initial_state = [x0; y0; theta0; v0];
```

```
[t, states] = ode45(@(t, x) robot_dynamics_circular(t, x, d, v_R, v_L), t_span, initial_state);
```

```
% 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
```

LQR CONTROLLER DESIGN

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

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

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

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

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

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

Algebraic Riccati Equation yieds the P matrix which is used to determine the gain matrix K.

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

closed loop eigenvalues:

$-1.4060 \pm 1.3925i$ and $-1.3259 \pm 0.8921i$

```
Q = diag([10, 10, 1, 1]);
R = diag([1, 1]);
P = care(A, B, Q, R);
K = R \ (B' * P); % Equivalent to inv(R) * B' * P
disp('LQR Gain Matrix (K):');
LQR Gain Matrix (K):
disp(K);

    3.1106    0.5693    0.4838    2.8204
   -0.5693    3.1106    2.6433    0.4838

eig_closed_loop = eig(A - B * K);
disp('Closed-Loop Eigenvalues:');
Closed-Loop Eigenvalues:
disp(eig_closed_loop);

  -1.4060 + 1.3925i
  -1.4060 - 1.3925i
  -1.3259 + 0.8921i
  -1.3259 - 0.8921i
```

LQR TESTING

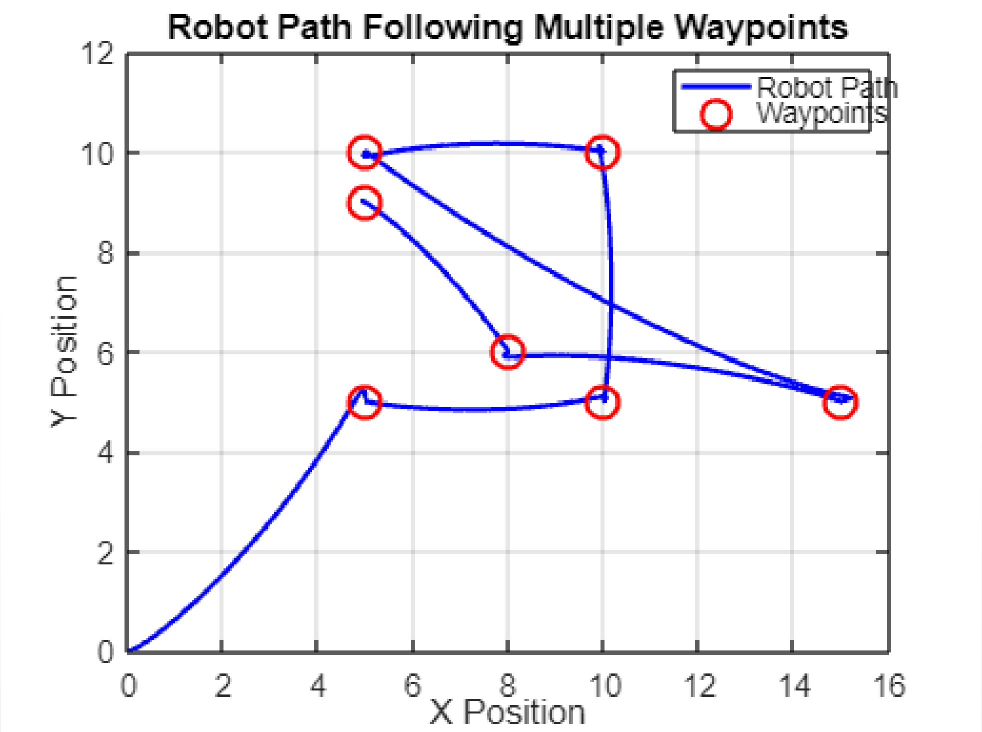

Figure 03: Test output for LQR Controller.

```
% 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
```

EFFECT OF EXTERNAL PARAMETERS

Consider following disturbances:

- Variations in system parameters A and B

$$A_{sim} = A + \Delta A \; and \; B_{sim} = B + \Delta B$$

- External disturbances in the form of sinusoidal inputs.

$$disturbance = [0; 0; 0.1\sin(t)\,; 0.05\cos(t)]$$

- Measurement noise introduced into the state variables.

$$noise = N(0, 0.1^2)$$

- Varying initial condition to a non-zero value.

$$Initial\ state: [x_0, y_0, \theta_0, v_0] = [2,3,0.1,0.8]$$

```
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;

noise_amplitude = 0.1; % Adjust this for stronger noise

function dxdt = controlled_dynamics(t, x, A, B, K,
target_state, noise_amplitude)
    disturbance = [0; 0; 0.1 * sin(t); 0.05 * cos(t)];
    noise = randn(4, 1) * noise_amplitude;
    u = -K * (x - target_state);
    dxdt = A * x + B * u + disturbance + noise;
end
```

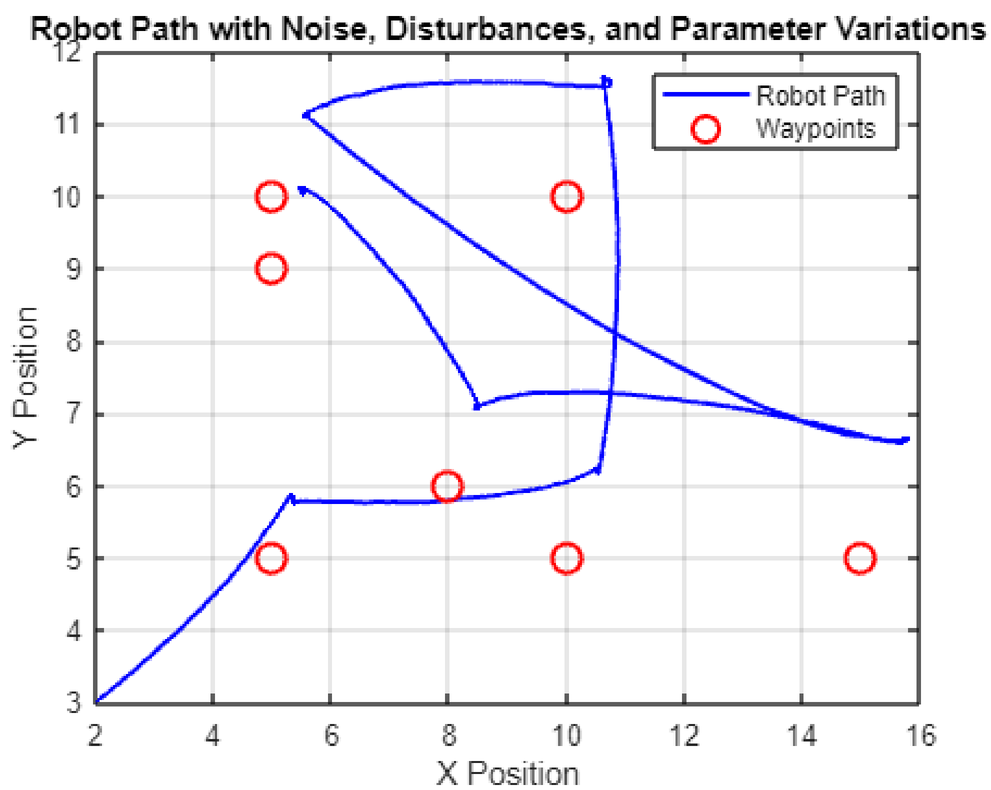

Figure 06: Robot Path correction with Q compensation.

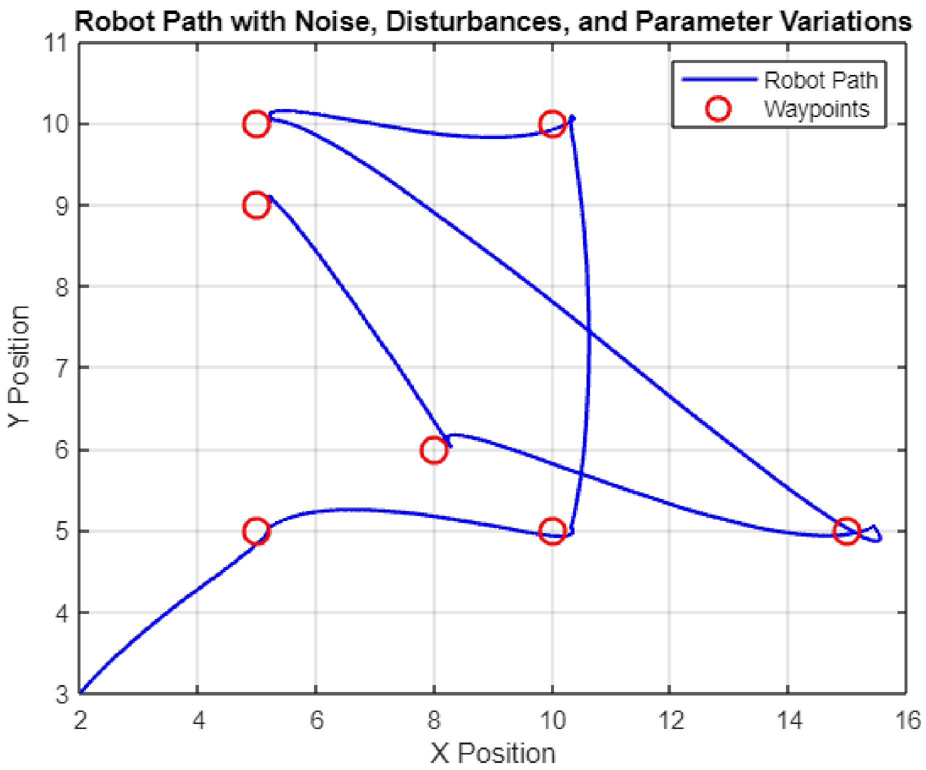

Figure 05: Robot Path deviation due to external parameters.

- 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.

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.
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,

$$\tau = \frac{1}{|\mathrm{Re}(\lambda)|}$$

$as\ highest\ value\ negative\ real\ pole\ in\ controller\ is - 1.4,$

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

$Therefore, \tau_o < 0.71 sec$
$assuming\ values\ which\ are\ further\ from\ origin\colon\ -5, -6, -7, -8$

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

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

OBSERVER GAIN

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

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

The obtained observer gain matrix is as follows:

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

```
% Observability check
O = obsv(A, C);
if rank(O) == size(A, 1)
    disp('System is observable.');
else
    disp('System is not observable.');
end
System is observable.
% 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):');
Observer Gain Matrix (L):
disp(L);
   13.5632    0.7281
    1.9386   12.4368
   19.6970   38.5752
   45.1536    4.7165
```

OBSERVER TESTING

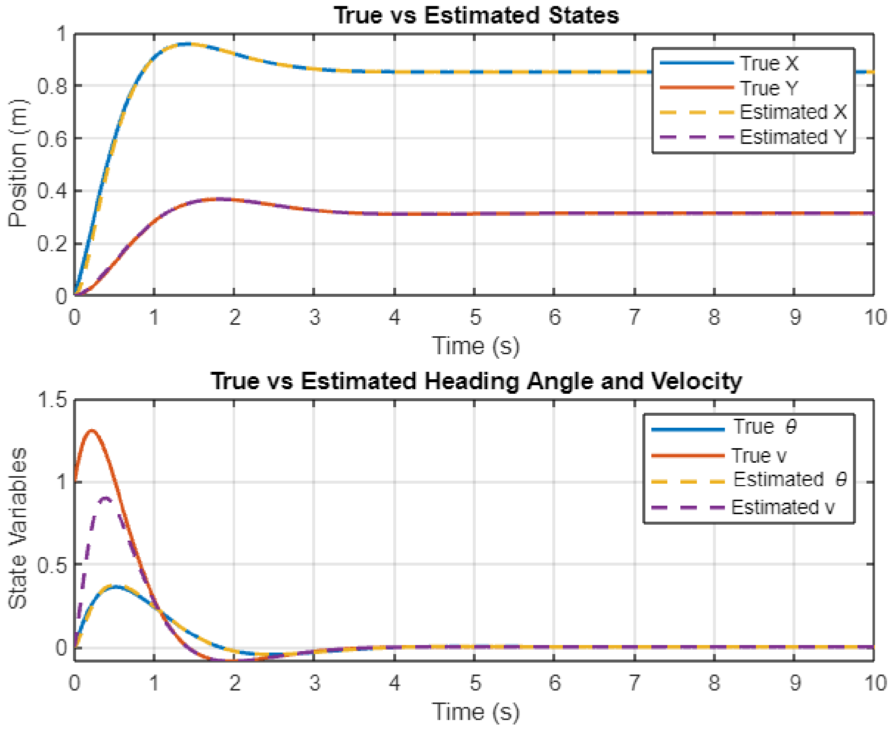

Figure 04: Test output for Observer Design.

```
[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);

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);
    dxdt = [dx_true; dx_hat];
end
```

FUTURE SCOPE

- **Integration with AI Path Planning:** Combine with AI-based algorithms for dynamic environment navigation.
- **Multi-Robot Coordination:** Extend to collaborative tasks with decentralized control.
- **Adaptive Control:** Enhance robustness to handle larger uncertainties.
- **Hardware Implementation:** Test on physical robots with real-world sensors.
- **Sensor Fusion:** Improve state estimation using LIDAR, IMU, or GPS.
- **Real-World Applications:** Deploy for warehouse automation or delivery systems.

CONCLUSION

- **State Space Representation:** Developed a state space representation for the differential drive robot.
- **System Dynamics:** Determined using MATLAB simulation by varying the velocity of the wheels to understand the robot's motion within the warehouse.
- **Feedback Gain of the system:** Designed gain matrix for the system using pole placement.
- **Controller Design:** Designed a Linear Quadratic Regulator (LQR) for path tracking.
- **Observer Design:** Designed a state observer using pole placement to estimate unmeasured states in real time.
- **Robustness Evaluation:** Tested the system under parameter variations (A and B), external disturbances, and sensor noise.

REFERENCES

- 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)
- 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)
- MATLAB LQR Command (https://in.mathworks.com/help/control/ref/lti.lqr.html)
- MATLAB observer design (https://in.mathworks.com/matlabcentral/answers/1940104-state-feedback-control-and-observer)
- 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},
- 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)
- Autonomous Robots Lab (https://www.autonomousrobotslab.com/lqr-control.html)

THANK YOU

Mukul Bhatia | Sajin Shrestha | Vighnesh Deolikar

Q & A