Fault Detection and Isolation Application for SCARA Robot

Using Observer-Based Technique

Presented By: Anirudh Gudi

Introduction to SCARA Robots and the Project Objective

- SCARA Robot: Commonly used in manufacturing for assembly, picking, and placing tasks.
- I have chosen a SCARA robot with three degrees of freedom (2 revolute and 1 prismatic joint).
- Fault detection is crucial for ensuring reliability.
- Objective: Implement fault detection and isolation (FDI) technique for a 3-DOF SCARA robot.
- Approach: Observer-based residual generation to detect faults & Fault isolation using observer at every sensor/actuators

Project Overview

$$M(q)\ddot{q} + V(q,\dot{q}) + G(q) = \tau$$

$$L(q,\dot{q}) = K(q,\dot{q}) - P(q) = \tfrac{1}{2}\dot{q}^T M(q)\dot{q} - P(q).$$

- Calculating system states using the above equations
- Observer design for residual and Observability and Invertibility check.
- Plot residuals for fault detection
- Unknown Input observer design
- Fault Isolation

Equations referred from Book: Robot Manipulator Control Theory and Practice

- Calculating jacobian to linearize the dynamic system
- Initializing from 0 position and 0 torques and force applied

```
x = [q1; q2; d3; dq1; dq2; dd3];
dx = [dq1; dq2; dd3; q_ddot];
```

```
A = jacobian(dx, x);
B = jacobian(dx, [Tau1; Tau2; F3]);
C = eye(6);
D = zeros(6, 3);

init_state = [0; 0; 0; 0; 0; 0];
input_torques = [0; 0; 0];

A = double(subs(A, [x; Tau1; Tau2; F3], [init_state; input_torques]));
B = double(subs(B, [x; Tau1; Tau2; F3], [init_state; input_torques]));
C = double(C);
D = double(D);
```

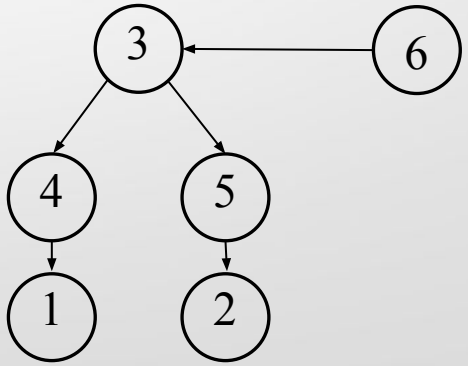
```
init_state = [0; 0; 0; 0; 0; 0]; % Initial q1, q2, d3, dq1, dq2, dd3 = 0
input_torques = [0; 0; 0]; % Zero input torques/forces
```

Fault Detection and Isolation

- Faults can occur in sensors, actuators, or the robot structure.
- Residual generation and analysis help identify faults.
- Observers are designed to estimate system states and detect anomalies.

Observer-Based Residual Generation

- State-space representation of the SCARA robot is derived.
- Residuals are generated by comparing measured and estimated outputs.
- Observer gain is designed ensuring stability and observability.



Graph of the chosen system

Observer-Based Residual Generation

```
%Observability and invertibility check
Ob = obsv(A, C);
if rank(Ob) ~= size(A, 1)
    error('The system is not observable.');
else
    disp('The system is observable.');
end

sys = ss(A, B, C, 0); % State-space system
zeros_sys = tzero(sys);

if any(abs(zeros_sys) >= 1)
    error('The system is not invertible.');
else
    disp('The system is invertible.');
end

% Observer gain L
observer_poles = [-0.5, -1.6, -3.7, -8, -0.2, -10]; % Random pole locations
L = place(A', C', observer_poles)';
```

```
The system is observable.
The system is invertible.
>>
```

Observer-Based Residual Generation

```
% Initial conditions
x = zeros(6, 1); % True state
x_hat = zeros(6, 1); % Estimated state
r = zeros(6, n); % Residuals
u = [0; 0; 0]; % Inputs (no external inputs in this example)

% Applying faults
f = zeros(6, n);
fault_time = T / 2; % Fault applied when halfway through the simulation
for i = 1:6
    f(i, t >= fault_time) = fault_magnitudes(i);
end

% Simulation loop
for i = 1:n

    dx = A*x + B*u + E*f(:, i);
    x = x + dx * dt;

    y = C*x;

    dx_hat = A*x_hat + B*u + L*(y - C*x_hat);
    x_hat = x_hat + dx_hat * dt;

    % Residuals
    r(:, i) = y - C*x_hat;
end
```

```
% Fault magnitudes for [q1, q2, d3, dq1, dq2, dd3]
fault_magnitudes = [1, 0, 0, 0, 0, 0]; % Simulated fault in joint q1
[r, L] = observer_based_residual_generation(params, fault_magnitudes);
```

Repeating this for all joint

Sensor based fault detection

$$x(t+1) = Ax(t) + Bu(t),$$

$$y(t) = \begin{bmatrix} y_1(t) & y_2(t) & \cdots & y_p(t) \end{bmatrix}^{\mathsf{T}} = Cx(t).$$

$$\begin{aligned} \hat{x}(t+1) &= A\hat{x}(t) + Bu(t) + L(y_i(t) - C_i\hat{x}(t)), \\ r(t) &= C\hat{x}(t) - y(t) = C(\hat{x}(t) - x(t)) - f(t). \end{aligned}$$

- Using the observer to place at every sensor to detect and isolate the fault

Equations referred from Book: Lecture Notes

Fault Isolation

- Observers are designed for each sensor to isolate faults.
- Residuals are analyzed to identify faulty components.
- Fault isolation enhances diagnostic accuracy.

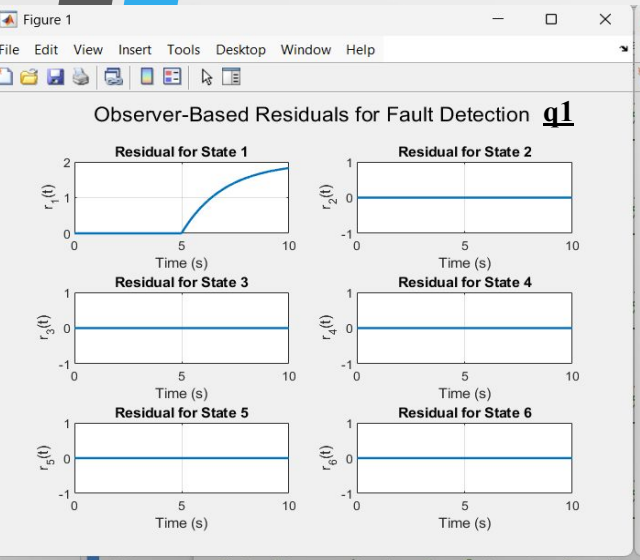
Fault Isolation

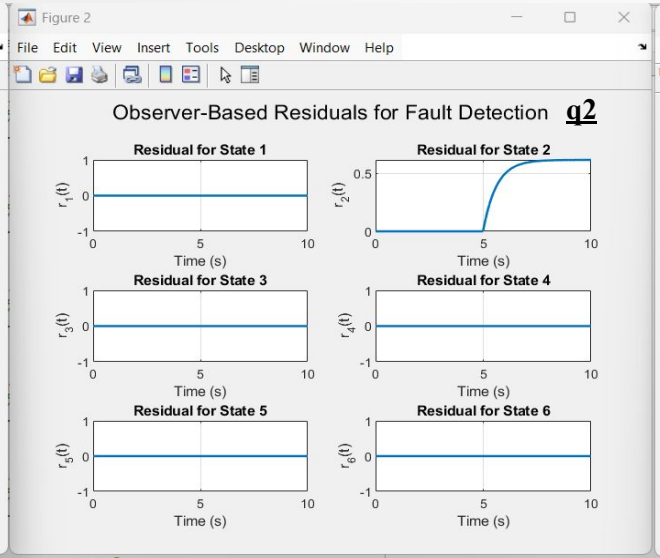
```
T_horizon = 500;    % Time horizon
dt = 0.01;          % Time step
n = size(A, 1);     % Number of states
m = size(B, 2);     % Number of inputs
p = size(C, 1);

for sensor_idx = 1:p

    C_i = C(sensor_idx, :);
    observer_poles = [0.3, -0.5, -0.45, -0.1, 0.5, 0.6]; % Adjust poles dynamically
    L = place(A', C_i', observer_poles)'; % Observer gain

    x_hat = zeros(n, T_horizon);
    x_hat(:, 1) = rand(n, 1); % Initial estimate

    r = zeros(p, T_horizon);

    for t = 1:T_horizon-1

        dx = A*x(:, t) + B*u(:, t) + f(:, t);
        x(:, t+1) = x(:, t) + dx * dt;  % State update

        y(:, t) = C*x(:, t);

        dx_hat = A*x_hat(:, t) + B*u(:, t) + L*(y(sensor_idx, t) - C_i*x_hat(:, t));
        x_hat(:, t+1) = x_hat(:, t) + dx_hat * dt;  % Updating our estimate
        r(:, t+1) = y(:, t+1) - C*x_hat(:, t+1);
    end

    residuals(sensor_idx, :, sensor_idx) = r(sensor_idx, :);
end
```

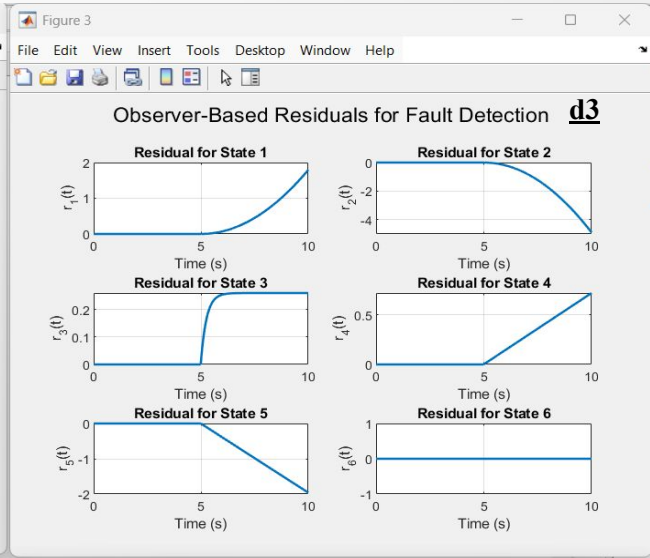
Simulation Results

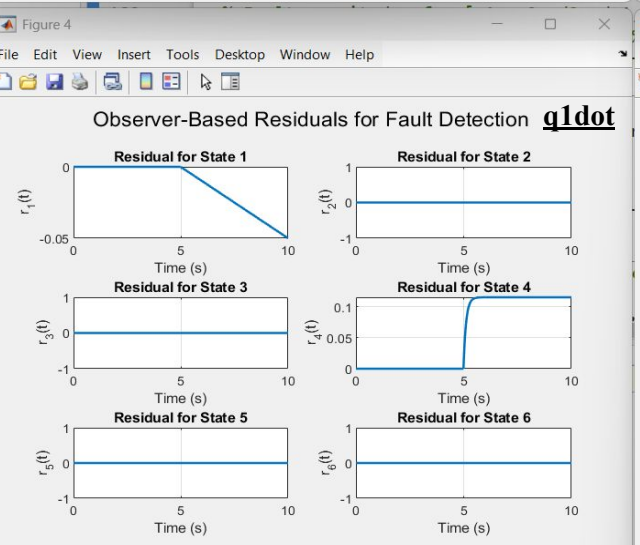
- Observer-based residuals for various fault scenarios.
- UIO-based residuals for fault detection and isolation. In this system, it cannot isolate the fault but with the observer we can only detect the fault.
- Residual plots indicate fault occurrence at every state variables.

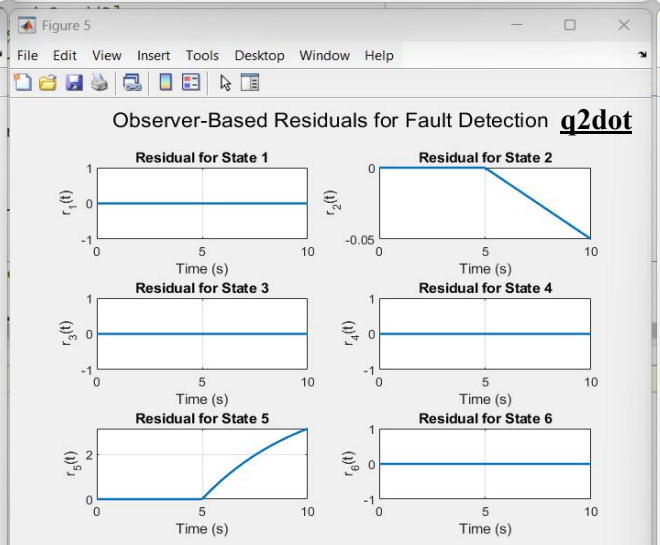
Simulation Results

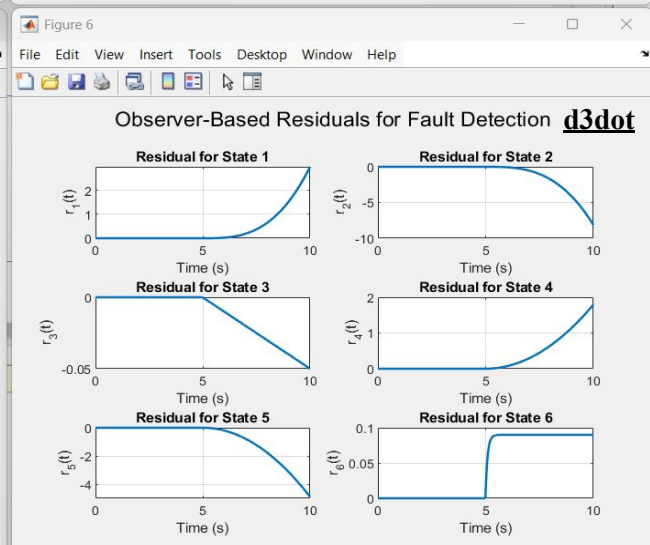












Conclusion

- SCARA robot faults were successfully detected
- Future work includes extending to other robot types and integrating with control systems.