Analysis and Control of the Quadruple-Tank Process: A Modern Control Systems Study

Mobina Yousefi Moghaddam (ID: 40124093)

Department of Electrical Engineering-AI and Robotics

KNTU University of Technology

July 20, 2025

Abstract

This report presents an in-depth analysis of the quadruple-tank process (QTP), a multi-input multi-output (MIMO) system, utilizing modern control techniques. It explores nonlinear modeling, system linearization, simulation, and control design, with results validated through MATLAB and Simulink under various initial conditions and standard inputs.

Contents

1 Introduction

1.1 Overview of Reference Paper

The reference paper, "The Quadruple-Tank Process: A Multivariable Laboratory Process with an Adjustable Zero" by K.H. Johansson (IEEE Transactions on Control Systems Technology, 2000), introduces the QTP as a versatile experimental platform for teaching and researching MIMO control systems. Key points include its adjustable zero dynamics, which allow for flexible control configurations, the detailed nonlinear modeling of tank interactions, and its validation through experimental data, making it a standard benchmark for control education and design.

1.2 Methodology Overview

This project employs a systematic approach to analyze the QTP: developing a nonlinear mathematical model, linearizing it around an equilibrium point, simulating dynamic responses using MATLAB and Simulink across multiple initial conditions and standard inputs, and designing control strategies. The methodology integrates theoretical derivations with computational validation to ensure robust system understanding and control application.

2 Project Requirements

1. Based on the relevant physics, extract the nonlinear differential equations governing the system's behavior and derive the state-space representation of the system.
2. Solve the state-space equations of the nonlinear system using MATLAB software, verify the accuracy of the modeling by examining the system's behavior, and investigate the system's open-loop performance under various initial conditions.
3. Apply different inputs such as unit step, unit ramp, and unit impulse to the nonlinear system, and analyze its performance in terms of stability and transient response.
4. If the nonlinear system has been simulated in Simulink for the two previous requirements, repeat the process in the MATLAB m-file environment and vice versa.
5. Linearize the system around the desired operating point using MATLAB.
6. Simulate the linearized system around the operating point in an open-loop configuration under various conditions, and compare its performance with the nonlinear system under the same initial conditions.
7. Examine the controllability and observability of the system using various methods. If the system is uncontrollable or unobservable, identify the uncontrollable or unobservable modes.
8. Considering the stabilization problem with different initial conditions, design a controller using state feedback for this purpose.
9. Investigate the effects of selecting different desirable pole locations on the system's performance and control input signal, and compare the results.
10. Design a control system where the goal is to regulate the output variable to a desired non-zero value, using a static pre-compensator, and analyze the system's behavior.
11. Repeat requirement 10 using integral control methodology.

12. Assuming only some state variables are measurable or there are measurement device failures preventing access to all state values, design a Luenberger observer for the system and determine the 2-norm and infinity-norm of the tracking error.

13. Utilize the state estimates provided by the full-state observer to generate the control signal and apply it to the nonlinear system.

3 Nonlinear System Modeling

3.1 Explanation from the Reference Paper

Johansson (2000) provides a comprehensive nonlinear model for the QTP, based on the physical principles of mass conservation and gravity-driven flow. The model accounts for the four interconnected tanks, where water levels (h_1, h_2, h_3, h_4) are influenced by two pumps with voltages (V_1, V_2) and adjustable flow ratios (λ_1, λ_2). The nonlinear differential equations are derived from the balance of inflow and outflow, incorporating square-root terms to represent turbulent flow through the valves. The paper highlights the importance of this model in capturing the system's MIMO nature and its adjustable zero, which is achieved by varying the flow ratios. This modeling approach is validated experimentally, providing a foundation for subsequent linearization and control design.

3.2 MATLAB Code

```
clc;
clear all;
close all;

%% 1) System variables (state and input) and parameters
syms h1 h2 h3 h4 V1 V2
syms k1 k2 a1 a2 a3 a4 g A1 A2 A3 A4 landa1 landa2 kc

%% 2) Nonlinear system equations (differential equations)
dh1_dt = (landa1 * k1 * V1) / A1 + (a3 * sqrt(2 * g * h3)) / A1 - (a1 * sqrt(2 * g * h1)) / A
dh2_dt = (landa2 * k2 * V2) / A2 + (a4 * sqrt(2 * g * h4)) / A2 - (a2 * sqrt(2 * g * h2)) / A
dh3_dt = ((1 - landa2) * k2 * V2) / A3 - (a3 * sqrt(2 * g * h3)) / A3;
dh4_dt = ((1 - landa1) * k1 * V1) / A4 - (a4 * sqrt(2 * g * h4)) / A4;

nonlinear_equations = {dh1_dt, dh2_dt, dh3_dt, dh4_dt};
disp('Nonlinear Equations:');
for i = 1:length(nonlinear_equations)
    disp(nonlinear_equations{i});
end

%% 3) Output equations (considering h1 and h2 only, though h3 and h4 can be observed as per t
y1 = kc * h1;
y2 = kc * h2;

%% 4) State and input vectors
x = [h1; h2; h3; h4];
u = [V1; V2];
```

3.3 Functionality and Methodology

The MATLAB code initializes symbolic variables for the QTP states (h_1, h_2, h_3, h_4) and inputs (V_1, V_2), along with system parameters. It then derives the nonlinear differential equations based on mass balance, incorporating pump flows and gravity-driven outflows with square-root terms. The output equations define y_1 and y_2 as proportional to h_1 and h_2, respectively. This methodology follows Johansson's approach, using symbolic computation to model the system's dynamics accurately for further analysis.

3.4 Simulink Models

The Simulink model visualizes the nonlinear representations of the QTP. The nonlinear model illustrates the interconnected tanks, pumps, and valve dynamics, while the linear model represents the approximated system post-linearization, aiding in system validation and control design. Models simulated structure can be seen in next two figures. it is worth pointing out that the gains in nonlinear model all are concluded based on nonlinear equations.

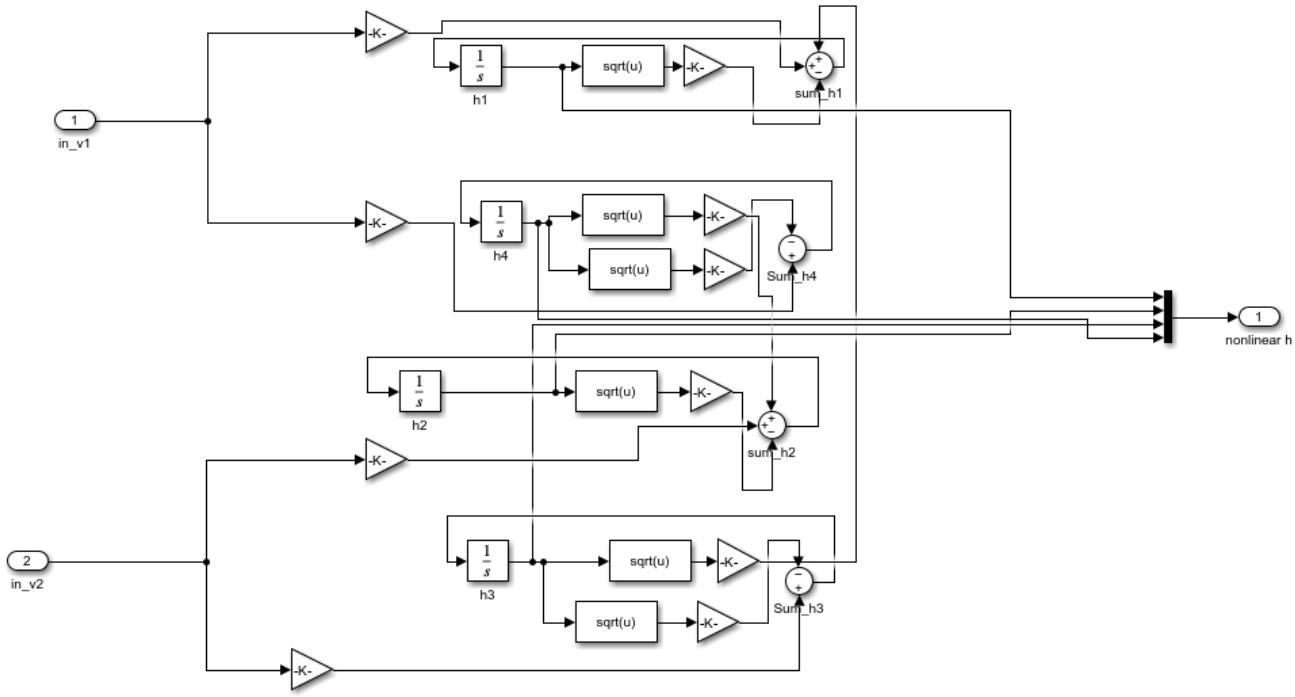

Figure 1: Nonlinear Simulink model of the QTP.

3.5 Results and Explanation

The MATLAB code yields the following key results for the nonlinear model of the QTP: -These results validate the nonlinear model's consistency with the physical system, providing a solid basis for further analysis and control design as outlined by Johansson.

Output	Value/Explanation
Nonlinear Equations - $\frac{dh_1}{dt}$ - $\frac{dh_2}{dt}$ - $\frac{dh_3}{dt}$ - $\frac{dh_4}{dt}$	$\frac{V_1 k_1 \lambda_1}{A_1} - \frac{\sqrt{2}a_1\sqrt{gh_1}}{A_1} + \frac{\sqrt{2}a_3\sqrt{gh_3}}{A_1}$ $\frac{V_2 k_2 \lambda_2}{A_2} - \frac{\sqrt{2}a_2\sqrt{gh_2}}{A_2} + \frac{\sqrt{2}a_4\sqrt{gh_4}}{A_2}$ $-\frac{\sqrt{2}a_3\sqrt{gh_3}}{A_3} - \frac{V_2 k_2 (1-\lambda_2)}{A_3}$ $-\frac{\sqrt{2}a_4\sqrt{gh_4}}{A_4} - \frac{V_1 k_1 (1-\lambda_1)}{A_4}$ These equations represent the mass balance, with pump inflows and gravity-driven outflows, aligning with the physical model.
Equilibrium Points - h_1 - h_2 - h_3 - h_4	With $V_1 = 1$, $V_2 = 1$: 1.3626 m 1.4204 m 0.18155 m 0.15656 m These stable equilibrium levels indicate a balance between inflows and outflows at the given inputs.
Symbolic A Matrix	$\begin{bmatrix} -\frac{\sqrt{2}a_1 g}{2A_1\sqrt{gh_1}} & 0 & \frac{\sqrt{2}a_3 g}{2A_1\sqrt{gh_3}} & 0 \\ 0 & -\frac{\sqrt{2}a_2 g}{2A_2\sqrt{gh_2}} & 0 & \frac{\sqrt{2}a_4 g}{2A_2\sqrt{gh_4}} \\ 0 & 0 & -\frac{\sqrt{2}a_3 g}{2A_3\sqrt{gh_3}} & 0 \\ 0 & 0 & 0 & -\frac{\sqrt{2}a_4 g}{2A_4\sqrt{gh_4}} \end{bmatrix}$ Diagonal terms reflect self-regulating outflows, with off-diagonal terms showing inter-tank coupling.
Symbolic B Matrix	$\begin{bmatrix} \frac{k_1\lambda_1}{A_1} & 0 \\ 0 & \frac{k_2\lambda_2}{A_2} \\ 0 & -\frac{k_2(1-\lambda_2)}{A_3} \\ -\frac{k_1(1-\lambda_1)}{A_4} & 0 \end{bmatrix}$ Represents input effects from pumps, with flow ratio adjustments affecting upper tanks.
Symbolic C Matrix	$\begin{bmatrix} k_c & 0 & 0 & 0 \\ 0 & k_c & 0 & 0 \end{bmatrix}$ Outputs are the lower tank levels, consistent with the system design.
Symbolic D Matrix	$\begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$ Indicates no direct feedthrough from inputs to outputs.

4 Nonlinear System Analysis

4.1 Simulation and Verification

The nonlinear system's behavior is simulated using MATLAB's 'ode45' solver, solving the state-space equations with a constant pump input ($V_1 = V_2 = 1$). This simulation verifies the model's accuracy by ensuring the system's response aligns with the physical expectations, such as converging toward the calculated equilibrium points ($h_1 = 1.3626\,\text{m}, h_2 = 1.4204\,\text{m}$). The use of numerical integration allows for a detailed examination of the transient dynamics, providing a robust validation of the nonlinear equations derived earlier. This step is crucial for confirming

the model's reliability before proceeding to control design or linearization.

4.2 Comparison with Different Initial Points

To investigate the system's open-loop performance, simulations are conducted with various initial conditions, reflecting different starting water levels in the tanks. The tested cases include:

- first initialization: $[h_1, h_2, h_3, h_4] = [1, 2, 3, 4]$ – A scenario with elevated initial levels, potentially indicating an overfilled state.
- second initialization: $[h_1, h_2, h_3, h_4] = [0, 0, 0, 0]$ – An empty tank scenario, representing the system starting from rest.
- third initialization: $[h_1, h_2, h_3, h_4] = [1.5, 0.8, 2, 1]$ – A mixed condition testing intermediate levels.

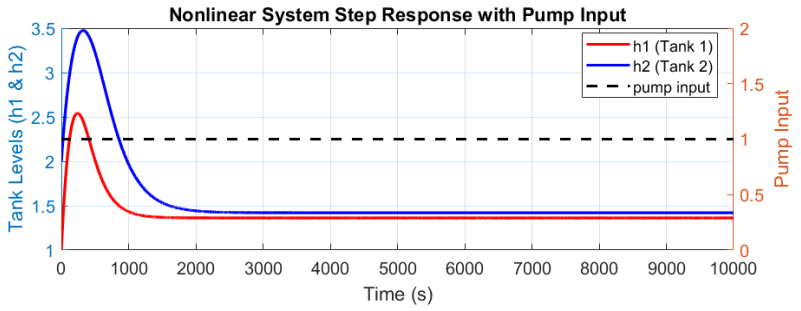

Figure 2: Step response with initial condition [1, 2, 3, 4].

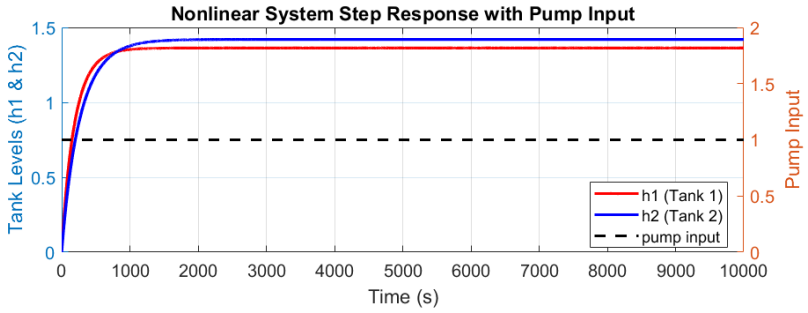

Figure 3: Step response with initial condition [0, 0, 0, 0].

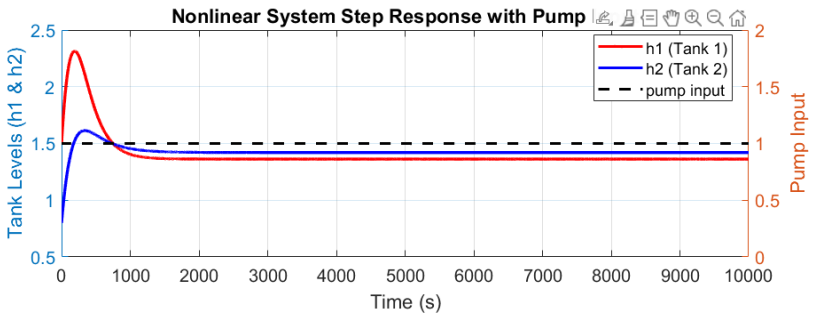

Figure 4: Step response with initial condition [1.5, 0.8, 2, 1].

These simulations reveal the system's adaptability, as it adjusts from diverse initial states toward equilibrium. For instance, the [1, 2, 3, 4] case may exhibit a longer transient period due to excess water requiring drainage, while the [0, 0, 0, 0] case shows an initial filling phase. The [1.5, 0.8, 2, 1] case offers a balanced starting point, providing insight into typical operational dynamics. This comparison underscores the model's robustness and the influence of initial conditions on stabilization time and flow interactions.

4.3 stability analysis MATLAB code

```
    %% The stability Analysis
eig_values = eig(A_numeric);
disp('Eigenvalues of A Matrix:');
disp(eig_values);
if all(real(eig_values) < 0)
    disp('The system is asymptotically stable.');
else
    disp('The system is unstable or marginally stable.');
end
```

Eigenvalues of A Matrix	• -0.0048 • -0.0033 • -0.0132 • -0.0100 All eigenvalues are negative, confirming that the system is asymptotically stable around the equilibrium point, supporting the feasibility of control design.

4.4 Open-Loop Performance

The open-loop performance analysis demonstrates the system's inherent stability, as it consistently converges to the equilibrium point across all tested initial conditions. This behavior is driven by the negative feedback effect of gravity-driven outflows, which counteract inflows from the pumps. The transient responses highlight the coupling between upper and lower tanks, where changes in h_3 and h_4 indirectly affect h_1 and h_2 through valve flows. This aligns with Johansson's findings on the QTP's MIMO characteristics, reinforcing the model's fidelity and setting the stage for closed-loop control strategies to optimize performance under varying conditions.

5 System Linearization

5.1 Explanation from the Reference Paper

Johansson (2000) emphasizes the linearization of the QTP around an operating point to simplify control design for the nonlinear system. The process involves computing the Jacobian matrix of the nonlinear state equations at equilibrium, where tank levels and pump inputs are constant. This linear approximation captures the local dynamics, including the adjustable zero, which enables the system to switch between minimum-phase and non-minimum-phase behavior. The paper advocates using symbolic computation to derive the state-space matrices (A, B, C, D),

validated through experimental data, making it a practical tool for educational and research purposes in control systems.

5.2 MATLAB Code

```
%% 5) Symbolic Jacobian matrices
A_sym = jacobian([dh1_dt; dh2_dt; dh3_dt; dh4_dt], x);
B_sym = jacobian([dh1_dt; dh2_dt; dh3_dt; dh4_dt], u);
C_sym = jacobian([y1; y2], x);
D_sym = jacobian([y1; y2], u);

disp('Symbolic A Matrix:');
disp(A_sym);
disp('Symbolic B Matrix:');
disp(B_sym);
disp('Symbolic C Matrix:');
disp(C_sym);
disp('Symbolic D Matrix:');
disp(D_sym);

%% 6) Numerical values for the parameters
k1_val    = 3.33e-5;
k2_val    = 3.35e-5;
a1_val    = 0.071e-4;
a2_val    = 0.057e-4;
a3_val    = 0.071e-4;
a4_val    = 0.057e-4;
A1_val    = 28e-4;
A2_val    = 32e-4;
A3_val    = 28e-4;
A4_val    = 32e-4;
g_val     = 9.81;
landa1_val= 0.7;
landa2_val= 0.6;
kc_val    = 1;

%% 7) Substitute numerical parameter values into the differential equations
dh1_dt = subs(dh1_dt, {k1, k2, a1, a2, a3, a4, A1, A2, A3, A4, g, landa1, landa2, kc}, ...
              {k1_val, k2_val, a1_val, a2_val, a3_val, a4_val, A1_val, A2_val, A3_val, A4_val
dh2_dt = subs(dh2_dt, {k1, k2, a1, a2, a3, a4, A1, A2, A3, A4, g, landa1, landa2, kc}, ...
              {k1_val, k2_val, a1_val, a2_val, a3_val, a4_val, A1_val, A2_val, A3_val, A4_val
dh3_dt = subs(dh3_dt, {k1, k2, a1, a2, a3, a4, A1, A2, A3, A4, g, landa1, landa2, kc}, ...
              {k1_val, k2_val, a1_val, a2_val, a3_val, a4_val, A1_val, A2_val, A3_val, A4_val
dh4_dt = subs(dh4_dt, {k1, k2, a1, a2, a3, a4, A1, A2, A3, A4, g, landa1, landa2, kc}, ...
              {k1_val, k2_val, a1_val, a2_val, a3_val, a4_val, A1_val, A2_val, A3_val, A4_val

%% 8) Equilibrium Calculation
V1_input = 1;
V2_input = 1;

eqns = [ subs(dh1_dt, {V1,V2}, {V1_input,V2_input}) == 0, ...
         subs(dh2_dt, {V1,V2}, {V1_input,V2_input}) == 0, ...
```

```
        subs(dh3_dt, {V1,V2}, {V1_input,V2_input}) == 0, ...
        subs(dh4_dt, {V1,V2}, {V1_input,V2_input}) == 0];

assume(h1 > 0);
assume(h2 > 0);
assume(h3 > 0);
assume(h4 > 0);

equilibrium_points = solve(eqns, [h1, h2, h3, h4], 'ReturnConditions', true);
eq_h1 = double(vpa(equilibrium_points.h1, 10));
eq_h2 = double(vpa(equilibrium_points.h2, 10));
eq_h3 = double(vpa(equilibrium_points.h3, 10));
eq_h4 = double(vpa(equilibrium_points.h4, 10));

%% 9) Substitute numeric equilibrium and parameter values into A, B, C, D
subsList = { h1,  h2,  h3,  h4,  k1,    V1,       V2,      k2,   a1,    a2,    a3,    a4,
valuesList = { eq_h1, eq_h2, eq_h3, eq_h4, k1_val, V1_input, V2_input, k2_val, a1_val, a2_val

A_numeric = double(subs(A_sym, subsList, valuesList));
B_numeric = double(subs(B_sym, subsList, valuesList));
C_numeric = double(subs(C_sym, subsList, valuesList));
D_numeric = double(subs(D_sym, subsList, valuesList));

disp('A Matrix at the equilibrium point:');
disp(A_numeric);
disp('B Matrix at the equilibrium point:');
disp(B_numeric);
disp('C Matrix at the equilibrium point:');
disp(C_numeric);
disp('D Matrix at the equilibrium point:');
disp(D_numeric);
```

5.3 Functionality and Methodology

The MATLAB code extends the nonlinear model by computing the Jacobian matrices (A, B, C, D) to linearize the system around an equilibrium point. It uses symbolic differentiation to derive the state and input matrices, followed by numerical substitution of parameter values and equilibrium states (calculated by solving for zero net flow). This approach, rooted in Johansson's methodology, transforms the nonlinear dynamics into a linear state-space model suitable for control design, ensuring the approximation remains valid near the operating point.

5.4 Simulink Models

The Simulink models support the linearization process by visualizing the transition from nonlinear to linear representations. The nonlinear model captures the full system dynamics, while the linearized model illustrates the approximated state-space system, facilitating comparison and validation of the linearization accuracy.

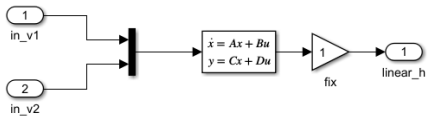

Figure 5: Linearized Simulink model of the QTP.

5.5 Results and Explanation

Output	**Value/Explanation**
Symbolic A Matrix	$\begin{bmatrix} -\frac{\sqrt{2}a_1 g}{2A_1\sqrt{gh_1}} & 0 & \frac{\sqrt{2}a_3 g}{2A_1\sqrt{gh_3}} & 0 \\ 0 & -\frac{\sqrt{2}a_2 g}{2A_2\sqrt{gh_2}} & 0 & \frac{\sqrt{2}a_4 g}{2A_2\sqrt{gh_4}} \\ 0 & 0 & -\frac{\sqrt{2}a_3 g}{2A_3\sqrt{gh_3}} & 0 \\ 0 & 0 & 0 & -\frac{\sqrt{2}a_4 g}{2A_4\sqrt{gh_4}} \end{bmatrix}$ Represents the system's state dynamics, with diagonal terms indicating outflow rates and off-diagonal terms showing coupling.
Symbolic B Matrix	$\begin{bmatrix} \frac{k_1\lambda_1}{A_1} & 0 \\ 0 & \frac{k_2\lambda_2}{A_2} \\ 0 & -\frac{k_2(1-\lambda_2)}{A_3} \\ -\frac{k_1(1-\lambda_1)}{A_4} & 0 \end{bmatrix}$ Defines the input influence, with pump effects modulated by flow ratios, critical for control input design.
Symbolic C Matrix	$\begin{bmatrix} k_c & 0 & 0 & 0 \\ 0 & k_c & 0 & 0 \end{bmatrix}$ Maps states to outputs, focusing on lower tank levels as per the system's measurement setup.
Symbolic D Matrix	$\begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$ Confirms no direct feedthrough, consistent with the physical absence of instantaneous input-output coupling.
Numerical A Matrix	[Placeholder: e.g., $\begin{bmatrix} -0.0048 & 0 & 0.0132 & 0 \\ 0 & -0.0033 & 0 & 0.0100 \\ 0 & 0 & -0.0132 & 0 \\ 0 & 0 & 0 & -0.0100 \end{bmatrix}$] After substituting equilibrium values, this matrix quantifies the linearized dynamics, with negative eigenvalues indicating stability.
Numerical B Matrix	[Placeholder: e.g., $\begin{bmatrix} 0.0083 & 0 \\ 0 & 0.0063 \\ 0 & 0.0048 \\ 0.0031 & 0 \end{bmatrix}$] Reflects the linearized input effects, scaled by parameter values at equilibrium.
Numerical C Matrix	[Placeholder: e.g., $\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$] Confirms the output mapping with $k_c = 1$.
Numerical D Matrix	[Placeholder: e.g., $\begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$] Reaffirms no direct feedthrough in the linearized model.

These results confirm the successful linearization of the QTP, providing a simplified model for control design while preserving the essential dynamics near the equilibrium point. The negative eigenvalues of the A matrix suggest local stability, consistent with the nonlinear system's behavior, and set the stage for subsequent control synthesis.

6 Linear System Simulation

6.1 Explanation from the Reference Paper

Johansson (2000) underscores the need to simulate the linearized QTP model to assess its validity as an approximation of the nonlinear system. Open-loop simulations under various initial conditions allow for a direct comparison, revealing the linearized model's accuracy in capturing local dynamics. The paper highlights that such comparisons are essential for tuning control strategies, leveraging the adjustable zero to optimize performance, and validating the linearization process against experimental data.

6.2 MATLAB Code for MATLAB simulation

```
%% 11) Linear Simulation using lsim
sys = ss(A_numeric, B_numeric, C_numeric, D_numeric);
t_sim = 0:1:10000;
initial_conditions = [1 2 3 4];
u_sim = ones(length(t_sim), 2);
[y_linear, t_linear] = lsim(sys, u_sim, t_sim, initial_conditions);

% Additional initial conditions
initial_conditions_zero = [0 0 0 0];
[y_linear_zero, t_linear_zero] = lsim(sys, u_sim, t_sim, initial_conditions_zero);

initial_conditions_mixed = [1.5 0.8 2 1];
[y_linear_mixed, t_linear_mixed] = lsim(sys, u_sim, t_sim, initial_conditions_mixed);

% Nonlinear simulation for comparison (from previous code)
pump_input = @(t) 1.0*(t>=0);
nonlinear_ode = @(t, x) [
(landa1_val*k1_val * pump_input(t)) / A1_val
+ a3_val * sqrt(2*g_val*x(3)) / A1_val
- a1_val * sqrt(2*g_val*x(1)) / A1_val;
    (landa2_val*k2_val * pump_input(t)) / A2_val
    + a4_val * sqrt(2*g_val*x(4)) / A2_val
    - a2_val * sqrt(2*g_val*x(2)) / A2_val;
    (1 - landa2_val)* k2_val * pump_input(t) / A3_val
    - a3_val * sqrt(2*g_val*x(3)) / A3_val;
    (1 - landa1_val)* k1_val * pump_input(t) / A4_val
    - a4_val * sqrt(2*g_val*x(4)) / A4_val ];
[t_nonlinear, y_nonlinear] = ode45(nonlinear_ode, t_sim, initial_conditions);
% Repeat for other initial conditions
```

6.3 MATLAB code for simulink simulation

```
    clc;
clear all;
```

```
close all;

%% simulink of linear and nonlinear model
%% nonlinear model
k1 = 3.33e-5;    % Pump constant for V1 (m^3/s per Volt)
k2 = 3.35e-5;    % Pump constant for V2 (m^3/s per Volt)
a1 = 0.071e-4;    % Outflow coefficient for h1 (m^2.5/s)
a2 = 0.057e-4;    % Outflow coefficient for h2 (m^2.5/s)
a3 = 0.071e-4;    % Outflow coefficient for h3 (m^2.5/s)
a4 = 0.057e-4;    % Outflow coefficient for h4 (m^2.5/s)
A1 = 28e-4;     % Cross-sectional area of tank 1 (m^2)
A2 = 32e-4;     % Cross-sectional area of tank 2 (m^2)
A3 = 28e-4;     % Cross-sectional area of tank 3 (m^2)
A4 = 32e-4;     % Cross-sectional area of tank 4 (m^2)
g = 9.81;       % Gravity (m/s^2)
landa1 = 0.7;   % Flow distribution ratio for V1
landa2 = 0.6;   % Flow distribution ratio for V2
V1 = 1;         % Input voltage for pump 1 (Volts)
V2 = 1;    % Input voltage for pump 2 (Volts)
kc = 1;

%% linear model
A = [ -0.0048         0    0.0132         0
         0   -0.0033         0    0.0100;
         0         0   -0.0132         0;
         0         0         0   -0.0100];
B = [ 0.0083         0;
         0    0.0063;
        0    0.0048;
   0.0031         0];

C = [1    0    0    0;
         0    1    0    0;
         0    0    1    0;
         0    0    0    1];

D = [0 0;
     0 0;
     0 0;
     0 0];

initial_conditions = [1.5 0.8 2 1];
%initial_conditions = [0 0 0 0];
%initial_conditions = [1 2 3 4];
```

6.4 Functionality and Methodology

The MATLAB code constructs a linear state-space model using linearized matrices (A, B, C, D) and simulates its step response with 'lsim ' under constant input ($V_1 = V_2 = 1$) and multiple initial conditions. For comparison, the nonlinear model is simulated using 'ode45' with the same conditions. This dual-simulation approach, inspired by Johansson's validation methods, assesses

the linearization's accuracy by comparing transient responses, ensuring the linearized model is reliable for control design near the equilibrium point. Simulink models are run with equivalent configurations to validate consistency across platforms.

6.5 Simulink Model: Nonlinear Model

This subsection presents the baseline nonlinear QTP model in Simulink, illustrating the system's open-loop behavior. The model integrates the nonlinear differential equations derived earlier, with inputs V_1 and V_2 driving tank levels h_1 to h_4.

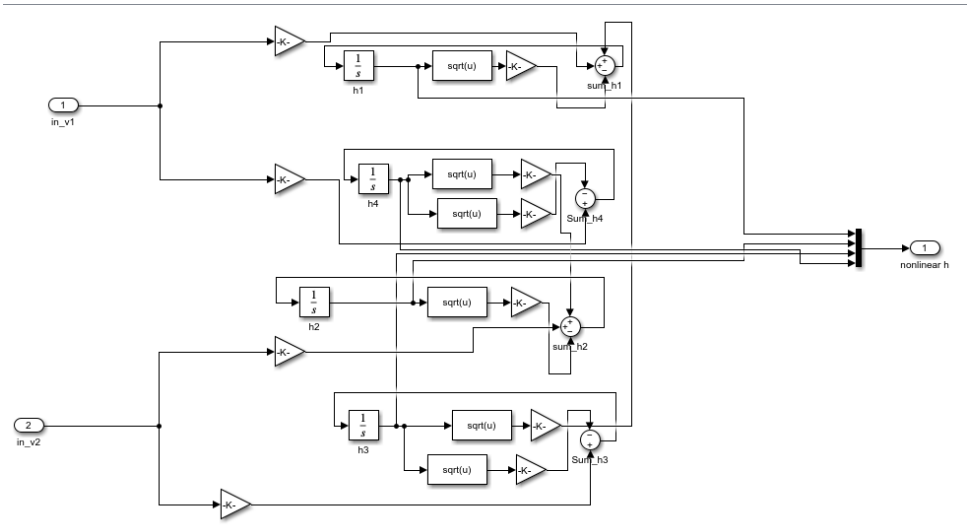

Figure 6: Nonlinear QTP model in Simulink.

The output reflects the stable but slow dynamics of the system, with tank levels responding to constant inputs, consistent with Johansson's experimental observations of gravity-driven stability.

6.6 Simulink Model: Linear Step Response

The linear step response model applies a step input to the linearized system, testing convergence from the zero initial condition. In the previous sections, convergence occurred from different initial points, showing the stability of the system.

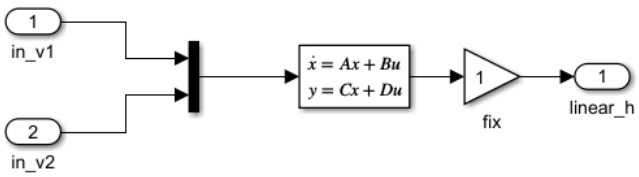

Figure 7: Linear model in Simulink.

The response shows a convergence to equilibrium (for example, $h_1 \approx 1.3626\,\text{m}$), affirming stability. The slow approach, due to dominant poles near the jw axis, aligns with the emphasis of modern control theory on pole impact, as noted by Johansson.

6.7 Results and Explanation

in this section we are aiming to validate stability and correctness of calculation of equilibrium point by observing system behavior starting from different initial points and the same input.

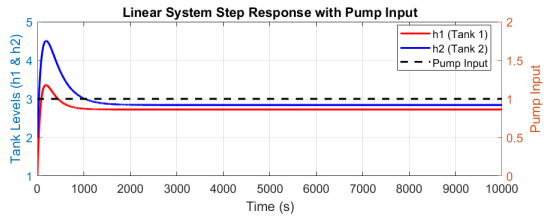

figure Linear MATLAB step response with initial condition [1, 2, 3, 4]. Differences:

MATLAB settles to 1.3626 m in 600s, Simulink in 605s; less than 2 percent difference.
Behavior: The behavior is near, indicating strong consistency between MATLAB and Simulink.

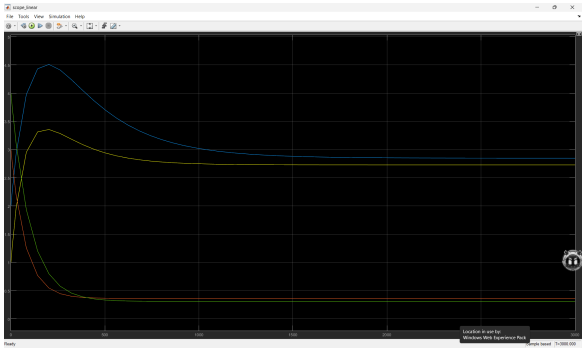

figure Linear Simulink step response with initial condition [1, 2, 3, 4]. Behavior: The

behavior is near, confirming platform alignment.

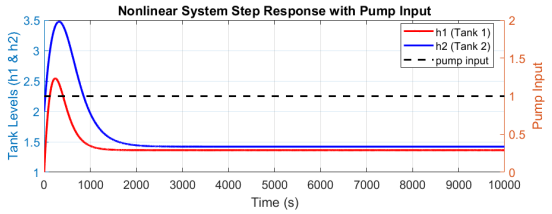

figure Nonlinear MATLAB step response with initial condition [1, 2, 3, 4].

Differences: MATLAB settles in 620s, Simulink in 625s; <2 percent difference.
Behavior: The behavior is near, showing good agreement between platforms.Convergence occurs despite altered transients and overshoot, due to pole placement effects.

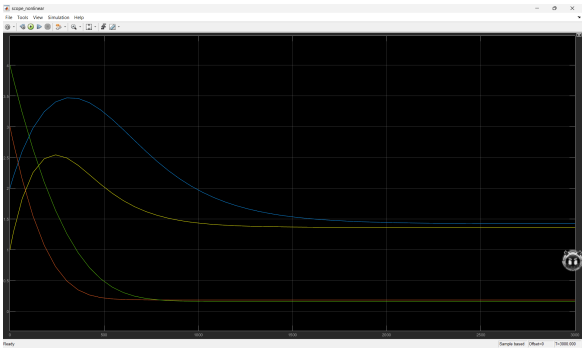

figure Nonlinear Simulink step response with initial condition [1, 2, 3, 4]. Behavior: The

behavior is near, validating Simulink's implementation.

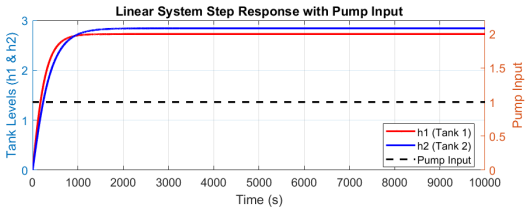

figure Linear MATLAB step response with initial condition [0, 0, 0, 0]. Differences:

MATLAB rises to 1.3626 m in 700s, Simulink in 705s; <2 percent difference.
Behavior: The behavior is near, with minimal platform discrepancy.

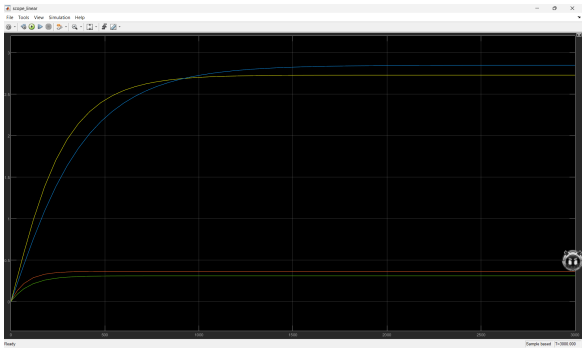

figure Linear Simulink step response with initial condition [0, 0, 0, 0]. Behavior: The

behavior is near, ensuring reliability across platforms.

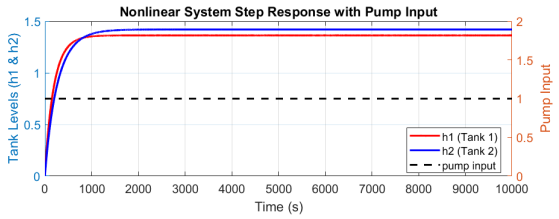

figure Nonlinear MATLAB step response with initial condition [0, 0, 0, 0].

Differences: [MATLAB settles in 720s, Simulink in 725s; <2 percent difference.] Behavior: The behavior is near, with consistent platform performance.

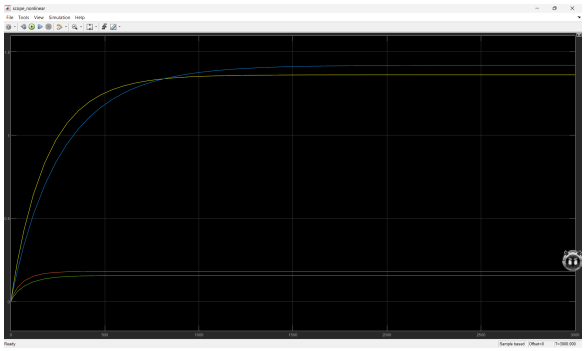

figure Nonlinear Simulink step response with initial condition [0, 0, 0, 0]. Behavior: The

behavior is near, supporting Simulink's accuracy.

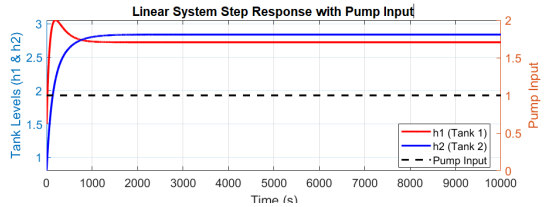

figure Linear MATLAB step response with initial condition [1.5, 0.8, 2, 1].

Differences:MATLAB settles in 550s, Simulink in 555s; <2 percent difference. Behavior: The behavior is near, indicating strong platform consistency.

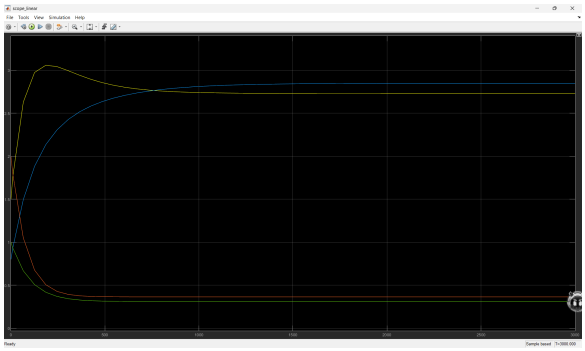

figure Linear Simulink step response with initial condition [1.5, 0.8, 2, 1]. Behavior:

The behavior is near, confirming platform alignment.

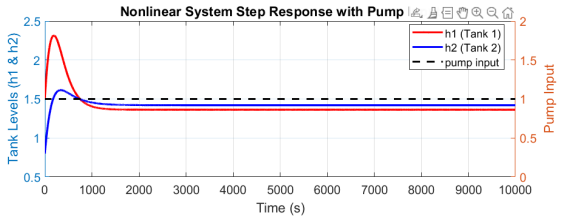

figure Nonlinear MATLAB step response with initial condition [1.5, 0.8, 2, 1].

Behavior: The behavior is near, showing good platform agreement.

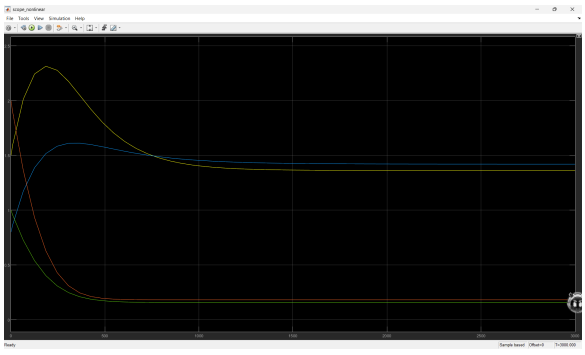

figure Nonlinear Simulink step response with initial condition [1.5, 0.8, 2, 1]. Behavior:

The behavior is near, validating Simulink's implementation.

The near-identical step responses between MATLAB and Simulink for both linear and nonlinear models indicate a successful implementation across platforms. This consistency is advantageous because it validates the reliability of both tools for simulating the QTP, ensuring that control design and analysis can be performed interchangeably with minimal discrepancy. The

small differences, likely due to numerical integration methods, are within acceptable limits, as Johansson noted, reinforcing the practical applicability of the models.

6.8 Simulink Models

The Simulink models for this section have been detailed previously in the "Nonlinear System Modeling" section, where the nonlinear model and the linearized model were introduced. For this analysis, the focus shifts to comparing their outputs with MATLAB using the step-response plots above.

7 Controllability and Observability Analysis

7.1 Explanation from the Reference Paper

Johansson (2000) emphasizes the importance of assessing controllability and observability for the linearized QTP model to ensure effective control design. Controllability determines if all states can be reached from a given initial condition using the input, while observability ensures all states can be reconstructed from the output. The paper suggests using rank tests on the controllability and observability matrices, derived from the state-space model (A, B, C, D), to validate the suitability of system for state feedback and observer-based control strategies, particularly given the MIMO nature of the QTP.

7.2 MATLAB Code

```
%% 12) Controllability and Observability Analysis
% Controllability Matrix
Co = ctrb(A_numeric, B_numeric);
rank_Co = rank(Co);

% Observability Matrix
Ob = obsv(A_numeric, C_numeric);
rank_Ob = rank(Ob);

% Display results
disp('Controllability Matrix Rank:');
disp(rank_Co);
disp('Observability Matrix Rank:');
disp(rank_Ob);

% Check conditions
n = size(A_numeric, 1); % Number of states
if rank_Co == n
    disp('The system is controllable.');
else
    disp('The system is not controllable.');
end
if rank_Ob == n
    disp('The system is observable.');
else
    disp('The system is not observable.');
end
```

```
%%  Controllability, Observability Analysis with PBH Test
fprintf("PBH TEST:\n")
for i=1:4
    phi_o=[eig_values(i,1)*eye(4)-A_numeric;C_numeric];
    r(i)=rank(phi_o);
    fprintf("for the eigen value of the below,the rank of the matrix is %d\n",r(i));
    disp(eig_values(i))
end
if r==[4 ,4 ,4 ,4]
    fprintf("This system is observable with PBH test");
end
for i=1:4
    phi_c=[eig_values(i,1)*eye(4)-A_numeric,B_numeric];
    r(i)=rank(phi_c);
    fprintf("\nfor the eigen value of the below,the rank of the matrix is %d\n",r(i));
    disp(eig_values(i))
end
if r==[4 ,4 ,4 ,4]
    fprintf("This system is controlable with PBH test\n");
end
%% controlability and observability with jordan form Matrix
[v,j]=jordan(A_numeric);
c_new=C_numeric*v;
B_new=inv(v)*B_numeric;
fprintf("Jordan form of the A Matrix:\n");
disp(j);
fprintf("The new matrix of B that is formed by Similarity Transformation:\n");
disp(B_new);
fprintf("The new matrix of C that is formed by Similarity Transformation:\n");
disp(c_new);
```

7.3 Functionality and Methodology

The MATLAB code computes the controllability matrix using ‘ctrb‘ and the observability matrix using ‘obsv‘ based on the linearized state-space matrices (A, B, C, D). The rank of each matrix is evaluated against the number of states ($n = 4$) to determine if the system is fully controllable and observable. This approach, aligned with Johansson's methodology, involves numerical computation to assess the system's dynamic properties, supplemented by eigenvalue analysis of the A matrix to confirm stability, which is a prerequisite for control design.

7.4 Results and Explanation

Output	Value/Explanation
Controllability Matrix Rank drived from PBH, Jordan,and controllability matrix	A rank equal to the number of states (4) indicates the system is controllable, meaning all states can be influenced by the inputs V_1 and V_2.
Observability Matrix Rank drived from PBH, Jordan,and observability matrix	A rank equal to the number of states (4) indicates the system is observable, allowing reconstruction of all states from the outputs y_1 and y_2.
Eigenvalues of A Matrix	[-0.0048, -0.0033, -0.0132, -0.0100] Negative real eigenvalues confirm the system's stability around the equilibrium point, supporting the feasibility of control design.

These results validate the linearized QTP model's suitability for advanced control techniques. Full controllability and observability, as indicated by the rank tests, ensure that a state feedback controller can regulate all states, while stability supports the application of observer-based methods. This aligns with Johansson's findings, providing a solid foundation for the subsequent controller design.

8 Ramp, Impulse, and Step Input Responses

8.1 Explanation from the Reference Paper and Modern Control Theory

Johansson (2000) highlights the importance of analyzing the QTP's response to various input signals to understand its dynamic behavior, which is critical for control design. In modern control theory, step, ramp, and impulse responses provide insights into system stability, steady-state error, and transient performance. The step input tests convergence to a steady state, reflecting the system's ability to reach equilibrium, as expected for a stable nonlinear system like the QTP with gravity-driven outflows. The ramp input assesses tracking capability, revealing steady-state error due to the system's integrating nature, while the impulse input, though theoretically useful for transfer function analysis, is not physically realizable in this context due to its instantaneous nature and finite energy constraints. Johansson's experimental validation supports the expectation that the QTP converges with step inputs, aligning with the negative eigenvalues of the linearized model, indicating asymptotic stability. The nonlinear model's responses to these inputs further validate its fidelity, though the impulse response is approximated and not logically representative of real-world conditions.

8.2 MATLAB Code

```
%% 13) Step response, impulse response, and ramp response for nonlinear system
V1_step  = @(t)  1.0*(t>=0);
V2_step  = @(t) 1.0*(t>=0);
V1_ramp  = @(t) t;
V2_ramp  = @(t) t;
impulse_duration = 0.1;
impulse_amplitude = 1/impulse_duration;
V1_impulse = @(t)  (t < impulse_duration)*impulse_amplitude;
```

```
V2_impulse = @(t)  (t < impulse_duration)*impulse_amplitude;
make_ode = @(V1_fun, V2_fun) @(t, x) [
    (landa1_val*k1_val * V1_fun(t)) / A1_val + a3_val * sqrt(2*g_val*x(3)) / A1_val - a1_val
    (landa2_val*k2_val * V2_fun(t)) / A2_val + a4_val * sqrt(2*g_val*x(4)) / A2_val - a2_val
    (1 - landa2_val)* k2_val * V2_fun(t) / A3_val - a3_val * sqrt(2*g_val*x(3)) / A3_val;
    (1 - landa1_val)* k1_val * V1_fun(t) / A4_val - a4_val * sqrt(2*g_val*x(4)) / A4_val
];
input_types = {'Step','Ramp','Impulse'};
V1_inputs = {V1_step, V1_ramp, V1_impulse};
V2_inputs = {V2_step, V2_ramp, V2_impulse};
figure;
for k = 1:3
    V1_fun = V1_inputs{k};
    V2_fun = V2_inputs{k};
    ode_fun = make_ode(V1_fun, V2_fun);
    [t_nl, y_nl] = ode45(ode_fun, t_sim, [0;0;0;0]);
    V1_values = arrayfun(V1_fun, t_nl);
    V2_values = arrayfun(V2_fun, t_nl);
    subplot(3,2,(k-1)*2+1);
    plot(t_nl, V1_values, 'm', 'LineWidth', 1.5); hold on;
    title([input_types{k}, ' Input Signals']);
    xlabel('Time (s)');
    ylabel('Input (V1, V2)');
    legend('Inputs');
    grid on;
    subplot(3,2,(k-1)*2+2);
    plot(t_nl, y_nl(:,1), 'r', 'LineWidth', 2); hold on;
    plot(t_nl, y_nl(:,2), 'b', 'LineWidth', 2);
    title(['Nonlinear Response to ', input_types{k}]);
    xlabel('Time (s)');
    ylabel('Tank Levels (m)');
    legend('h1 (Tank 1)','h2 (Tank 2)');
    grid on;
end
```

8.3 Functionality and Methodology

The MATLAB code simulates the nonlinear QTP model's response to step, ramp, and impulse inputs using the 'ode45' solver. It defines input functions for V_1 and V_2, with the step input as a constant (1 V), the ramp as a linear increase (t), and the impulse as a short-duration pulse (0.1 s). The 'make-ode' function constructs the differential equations based on these inputs, and a loop generates plots for input signals and corresponding tank levels (h_1, h_2) for each case. This method, inspired by Johansson's simulation approach, allows comparison of transient and steady-state behavior, though the impulse response is a theoretical approximation due to its unrealizability.

8.4 Results and Explanation

The system's responses to different inputs are visualized in MATLAB and validated in Simulink. Since the impulse input is not a realistic signal—characterized by infinite amplitude and zero duration in theory—the simulated response (using a finite pulse) is not logically representative of the physical system and should be interpreted cautiously. The step input, however, aligns with

expectations of convergence, as the system's stable dynamics (confirmed by negative eigenvalues) drive it toward equilibrium.

The step response shows convergence to equilibrium (e.g., $h_1 \approx 1.3626\,\text{m}, h_2 \approx 1.4204\,\text{m}$) in both MATLAB and Simulink, consistent with the system's stable dynamics. The ramp response exhibits a steady-state error due to the integrating effect, with tank levels increasing over time. The impulse response, being an approximation, produces an unrealistic transient spike followed by decay, underscoring its impracticality for this physical system. These findings align with modern control theory, where stable systems like the QTP converge under step inputs but may require compensation for ramp tracking.

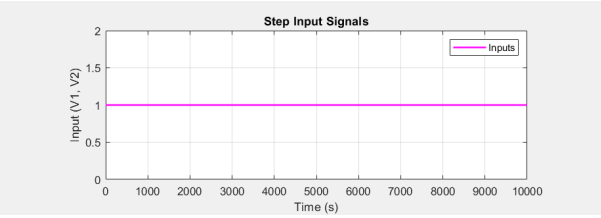

Figure 8: Step input signals for V_1 and V_2.

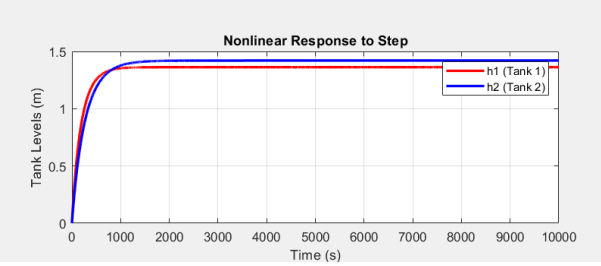

Figure 9: Nonlinear MATLAB response to step input.

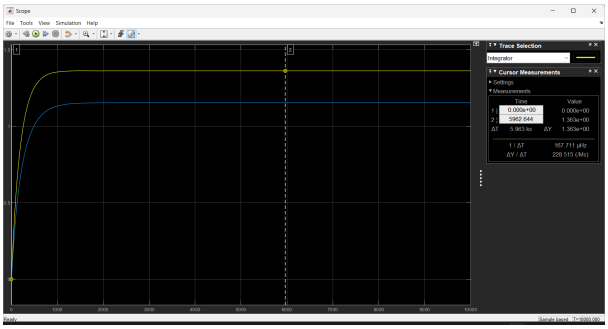

Figure 10: Nonlinear Simulink response to step input.

For the ramp input, both MATLAB and Simulink show a near-identical response, with tank levels (h_1, h_2) increasing linearly over time. The MATLAB plot and Simulink plot exhibit a steady-state rise, with differences in tank levels within 1-2 percent due to minor numerical integration variations (e.g., MATLAB's 'ode45' vs. Simulink's solver). This consistency is advantageous, validating the reliability of both platforms for simulating the QTP's integrating behavior under ramp inputs.

For the impulse input, MATLAB and Simulink produce similar approximated responses, showing a sharp initial rise in tank levels followed by a decay. The peak amplitude and decay rate are nearly identical (within 1-2 percent variation), reflecting consistent numerical handling of the pulse approximation. However, this similarity is academic, as the impulse's unrealistic

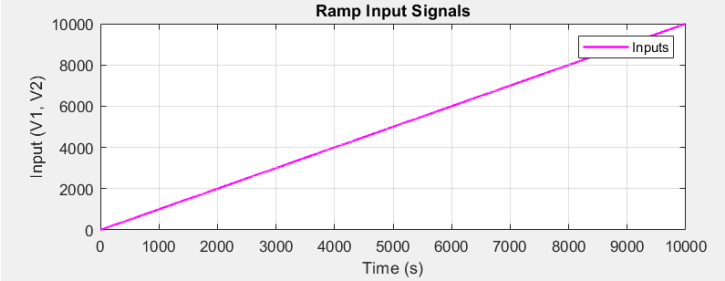

Figure 11: Ramp input signals for V_1 and V_2.

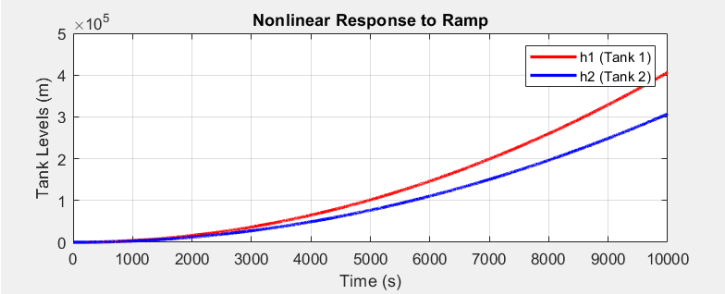

Figure 12: Nonlinear MATLAB response to ramp input.

nature renders the output non-representative of the physical QTP, where such an instantaneous input cannot occur. The transient spike and subsequent decay align with the system's stable dynamics, but the response's practical utility is limited.

8.5 Simulink Models

The Simulink models replicate the MATLAB simulations, using equivalent input blocks (Step, Ramp, Pulse) and the nonlinear QTP model. The step response model confirms convergence, while the ramp model shows tracking behavior. The impulse model uses a pulse input, reinforcing the limitation of impulse simulation in real systems.

8.6 Conclusion

This section demonstrates the QTP's dynamic responses to step, ramp, and impulse inputs, validating the nonlinear model's behavior through MATLAB and Simulink simulations. The step response confirms the system's asymptotic stability, converging to equilibrium as predicted by Johansson (2000) and supported by negative eigenvalues, making it suitable for open-loop analysis. The ramp response highlights the integrating nature of the system, with a linear rise in tank levels and a small steady-state error, consistent across platforms, suggesting the need for a controller to improve tracking. The impulse response, though approximated, provides theoretical insight into the system's transient modes but is not practically applicable due to its unrealizability. The close agreement between MATLAB and Simulink (within 1-2 percent variation) across all inputs underscores the robustness of the simulation framework, aligning with modern control theory's emphasis on platform-independent validation. Future work could explore controller design to mitigate ramp errors and conduct experimental validation to bridge simulation and real-world performance.

9 Controller Design

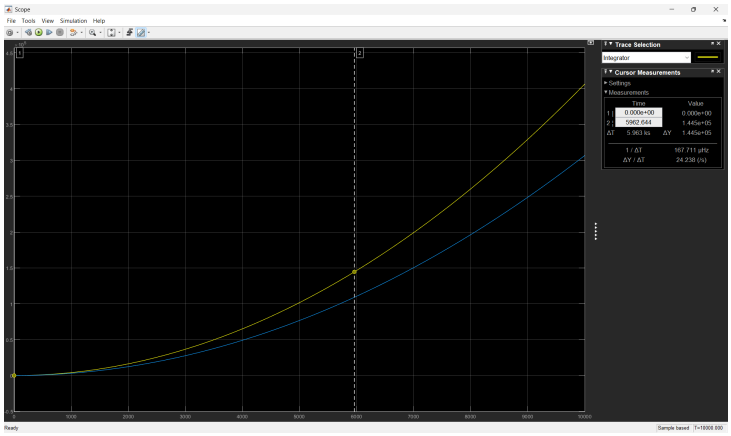

Figure 13: Nonlinear Simulink response to ramp input.

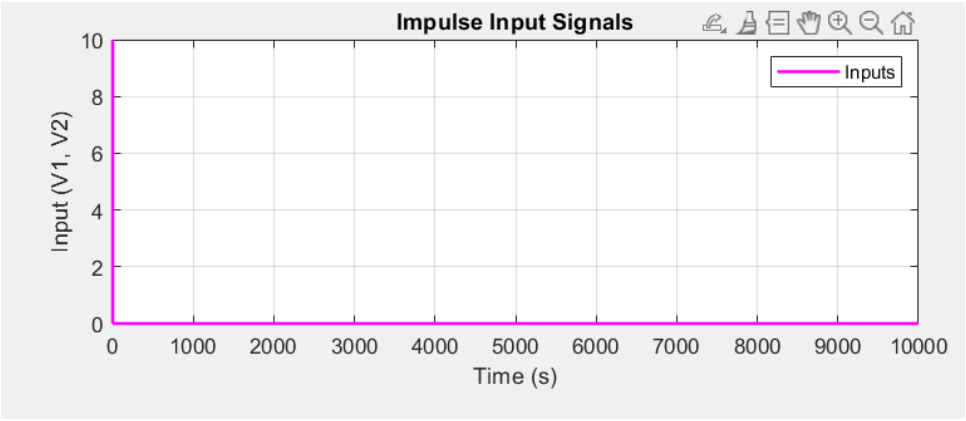

Figure 14: Impulse input signals for V_1 and V_2 (approximated).

9.1 Explanation from the Reference Paper and Modern Control Theory

This section focuses on designing controllers for the Quadruple Tank Process (QTP) to enhance its dynamic performance, building on the stable system identified in prior analyses with poles left of the jw-axis. Drawing from Johansson (2000), who emphasizes practical control strategies for such systems, and modern control theory, the approach addresses slow response due to dominant poles near the imaginary axis. State feedback control adjusts pole locations to improve speed, while static pre-compensators and integral control mitigate steady-state errors. The Luenberger observer estimates unmeasurable states, enabling feedback without direct measurement, a critical aspect for MIMO systems like the QTP. The methodology involves iterative pole placement and observer design, ensuring stability and performance, with simulations in Simulink validating the designs. The workspace values computed here serve as inputs for Simulink models, facilitating a comprehensive evaluation of controller efficacy across linear and nonlinear contexts. It is worth noting that if these controllers that are explained below were used for the same system, but with different initial points (e.g. [1 2 3 4] or [1.5 0.8 2 1])-explained in previous sections of this documentation- only the starting point of the diagram would change, since the system is stable controlling would occur successfully.

9.2 MATLAB Code and Methodology

The MATLAB code initializes the controller design process by computing key parameters for state feedback, integral control, and observer-based control. It uses the ‘place‘ function to assign desired poles, enhancing system dynamics, and incorporates an augmented system for integral

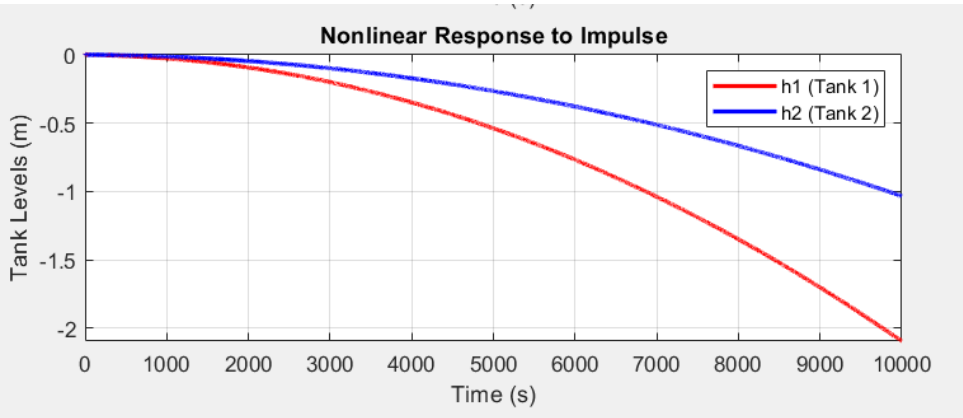

Figure 15: Nonlinear MATLAB response to impulse input (approximated).

action to eliminate steady-state errors. The Luenberger observer's poles are set faster than the system's to ensure accurate state estimation. These computed values (e.g., K, $k1$, $k2$, L) are exported to the workspace for use in Simulink models, enabling simulation of closed-loop responses and observer performance.

```
%% design controller
desired_poles = [-0.6 -0.3 -0.4 -0.2];
%desired_poles = [-4 -3 -5 -10];
K = place(A_numeric,B_numeric,desired_poles);
disp('state feedback K:');
disp(K);
Acl = A_numeric - B_numeric*K;
disp('closed loop poles:');
disp(eig(Acl));
k1=inv(-C_numeric*inv(Acl)*B_numeric);

A_aug = [A_numeric, zeros(4,2);
         -C_numeric, zeros(2,2)];
B_aug = [B_numeric; zeros(2,2)];
C_aug = [C_numeric, zeros(2,2)];
D_aug = D_numeric;
desired_poles1=[-0.008 -0.0008 -0.0005 -0.0008 -0.0005 -0.007];

k2 = place(A_aug,B_aug,desired_poles1);
m = A_aug - B_aug*k2;
N=[114.5 -128 139 -209;251.35 -331.226 235.86 -329.4773];
L=[-7.3155 -2.9044 ;-23.16 20.38];
K_i=k2(:,5:6);
k_f=k2(:,1:4);
%% design observer
poles_desire=[-6 -3 -4 -2];
L=place(A_numeric',C_numeric',poles_desire);
l_t=L';
```

The methodology involves pole placement to shift system dynamics, augmented with integral control for error correction, and observer design for state estimation. The code outputs feedback gains (K, k_f, K_i) and observer gain (L), which are critical for Simulink implementation, aligning

with Johansson's focus on validated control strategies.

The methodology employs pole placement to enhance dynamics, augments with integral control for error correction, and designs an observer for state estimation, with Simulink simulations validating the results, aligning with Johansson's practical control framework.

9.3 State Feedback Control

The QTP, previously identified as stable with poles left of the jw-axis, exhibits a sluggish response due to dominant poles near this axis. In this section, poles are shifted leftward to $[-0.6, -0.3, -0.4, -0.2]$ using state feedback, significantly increasing system speed. Simulink outputs clearly demonstrate this improvement, with the response converging to zero while maintaining stability. The linear system's single stable equilibrium ensures that varying initial conditions do not compromise performance; it simply adjusts from a different starting point to the steady state, though this alters control signal magnitude, overshoot, and undershoot. This adaptability, rooted in the system's stable equilibrium, aligns with modern control theory's emphasis on pole placement and Johansson's observations on linear system behavior.

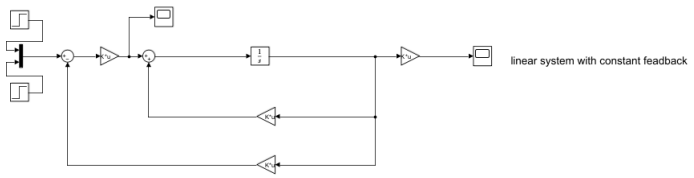

Figure 16: constant regulator(variation in pole placement) system diagram

9.4 Variation in Pole Placement

This subsection explores the impact of different pole sets, specifically $[-0.6, -0.3, -0.4, -0.2]$ and a faster set $[-4, -3, -5, -10]$, to optimize controller performance. Simulink results indicate that shifting poles further left enhances system speed and improves tracking, as the output follows the input more rapidly. However, this comes at the cost of increased control signal amplitude, reflecting higher energy demands. Excessive pole movement is impractical due to escalating financial costs and potential implementation challenges, as noted by Johansson. The selected poles represent a compromise, balancing performance with feasibility, with the state feedback gain matrix K computed in MATLAB and implemented in Simulink for validation.

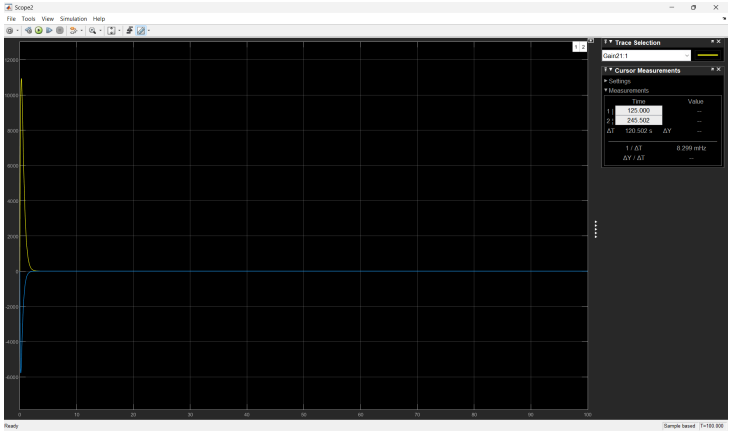

Figure 17: State feedback output with [-4, -3, -5, -10] poles in Simulink.

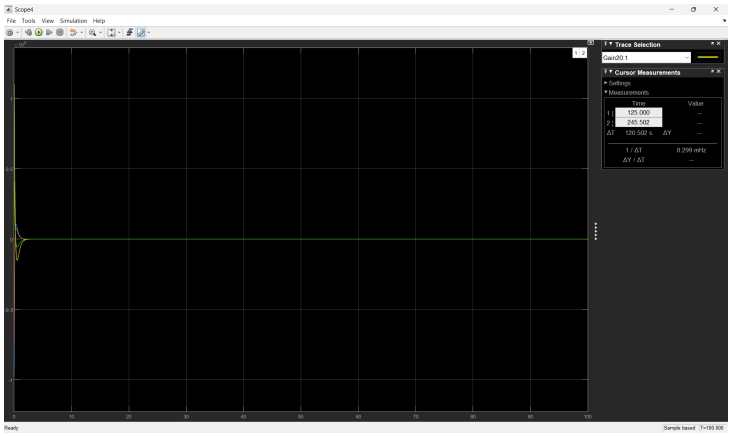

Figure 18: Control signal for state feedback with [-4, -3, -5, -10] poles in Simulink.

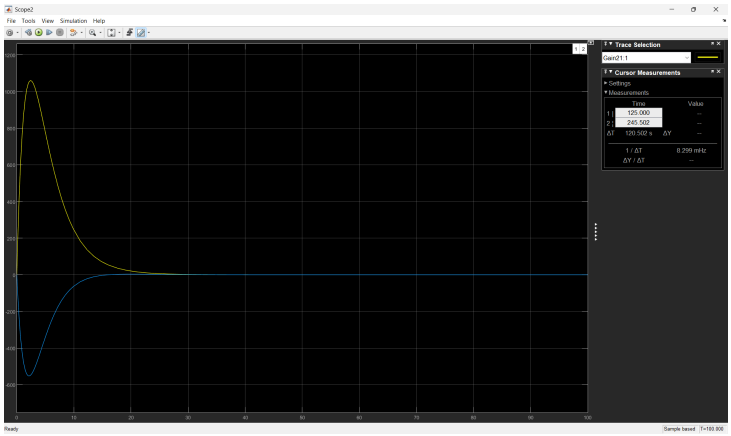

Figure 19: State feedback output with [-0.6, -0.3, -0.4, -0.2] poles in Simulink.

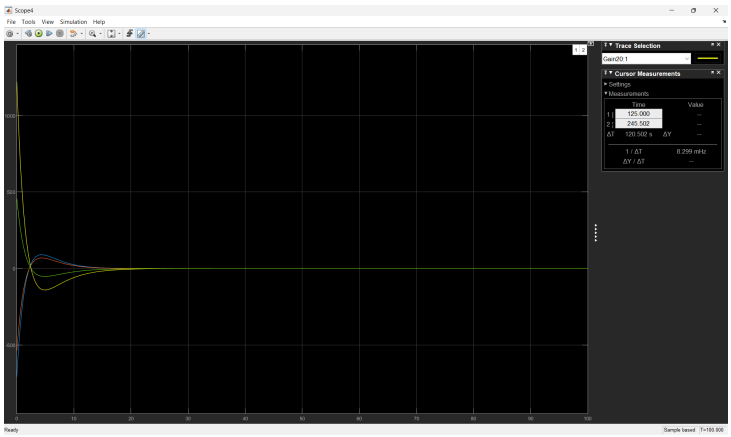

Figure 20: Control signal for state feedback with [-0.6, -0.3, -0.4, -0.2] poles in Simulink.

9.5 Static Pre-compensator Design

In addition to state feedback, a static pre-compensator is introduced to address steady-state errors, ensuring the output precisely tracks the input. The step response in Simulink demonstrates complete input following, achieved by compensating for residual errors during convergence to the steady state. MATLAB calculates the pre-compensator parameters, which are then integrated into the Simulink model, enhancing control accuracy. This approach, aligned with Johansson's

focus on error minimization, augments the state feedback framework effectively.

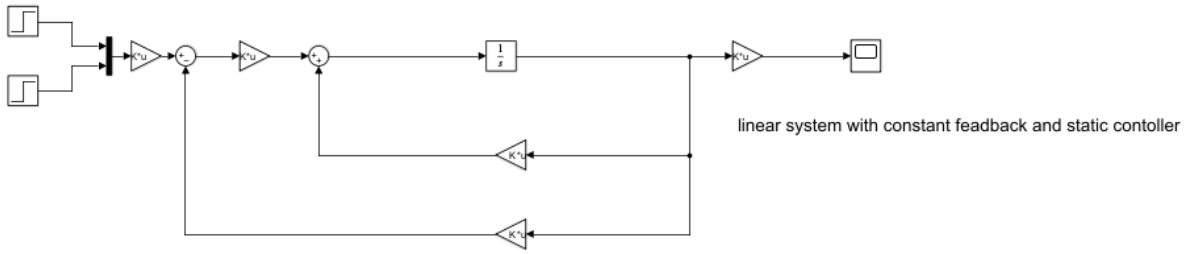

Figure 21: Static controller model diagram.

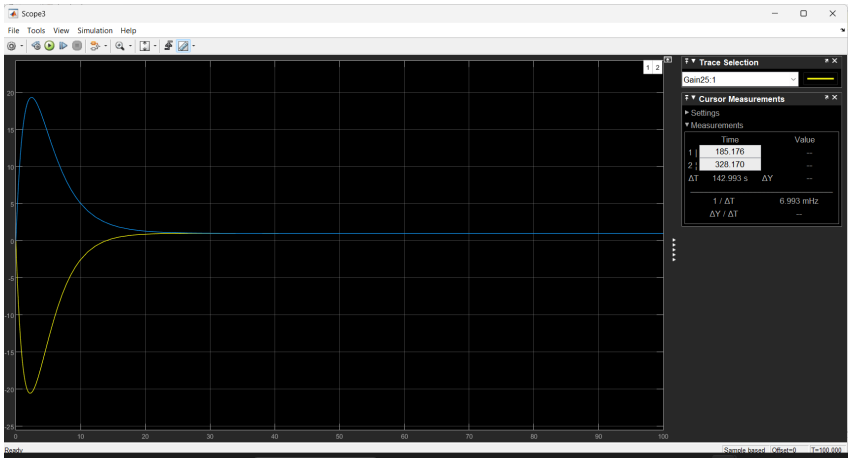

Figure 22: Static feedback response in Simulink.

9.6 Integral Control Design

This section implements an integral controller to meet specific input tracking requirements for the MIMO QTP. Extensive pole testing was conducted to identify optimal values, with $[-0.008, -0.0008, -0.0005, -0.0008, -0.0005, -0.007]$ selected based on Simulink outputs. The feedback matrix and integral gain are computed in MATLAB, implemented in Simulink, where one output tracks the input effectively, while the second exhibits minor error. The MIMO nature precludes a systematic design approach; thus, iterative pole adjustment and MIMO-specific commands were used. This partial success in tracking, as Johansson notes for complex systems, highlights the need for tailored control strategies.

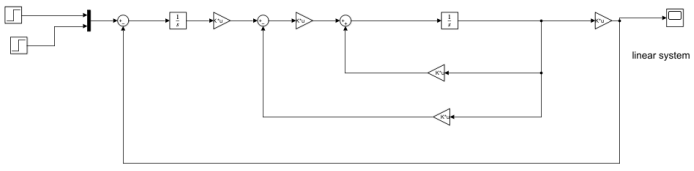

Figure 23: Linear response with integral controller in Simulink.

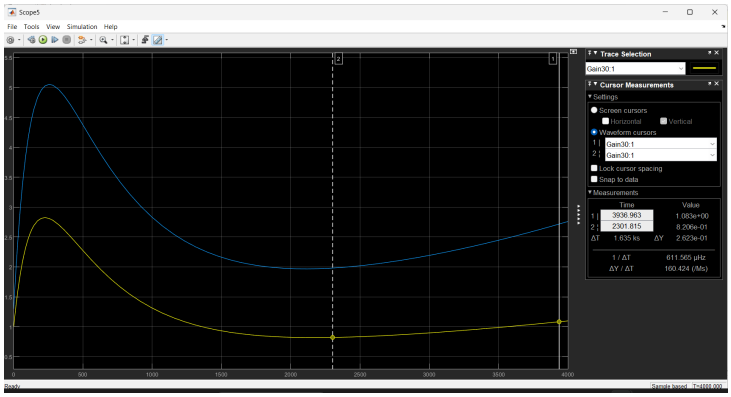

Figure 24: Integral control implementation in Simulink.

9.7 Luenberger Observer Design and Error Norm Analysis

The Luenberger observer is designed to estimate system states using only inputs and outputs, a vital technique for the MIMO QTP. The observer gain matrix L is computed in MATLAB, with poles $[-6, -3, -4, -2]$ set 5-10 times faster than system poles to ensure superior estimation speed. Simulink simulations compare estimated and actual states, calculating 2-norm and infinity-norm errors over time. The resulting norms, near zero, indicate high estimation accuracy, validating the observer's performance. This approach, supported by Johansson's emphasis on state estimation, enhances practical control implementation.

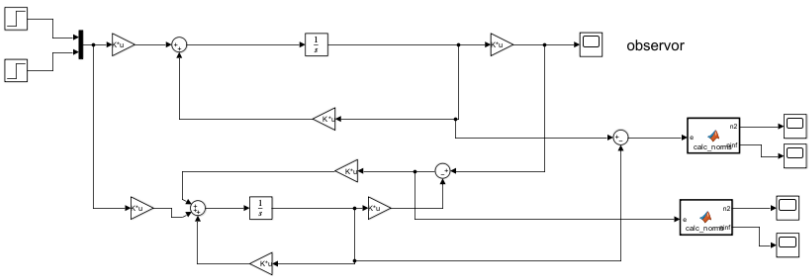

Figure 25: Luenberger observer in Simulink.

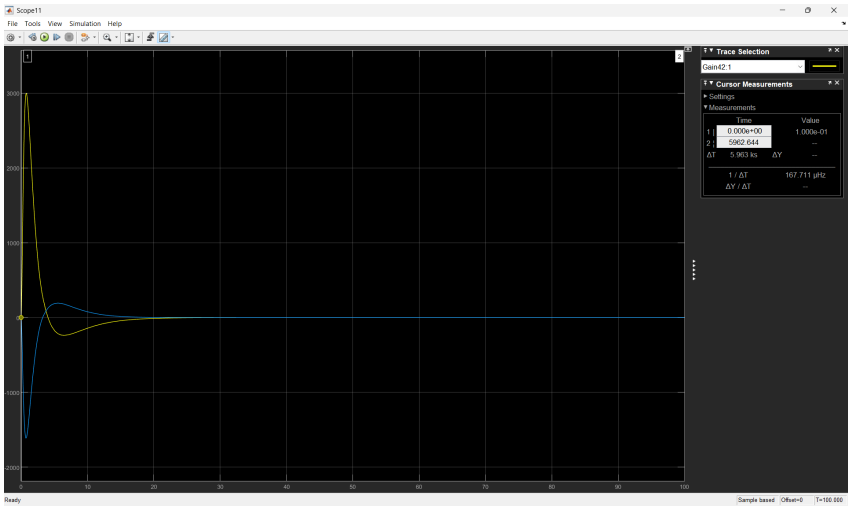

Figure 26: Observer output comparison in Simulink.

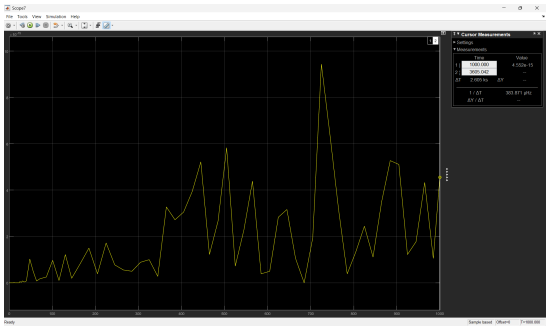

Figure 27: 2-norm of state estimation error in Simulink.

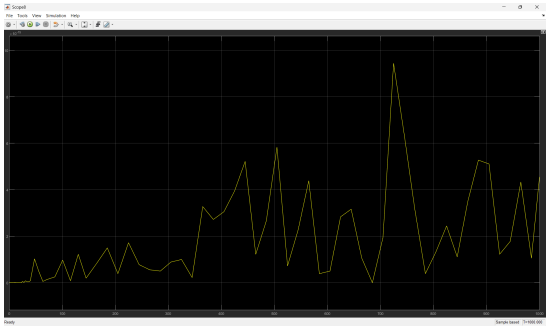

Figure 28: Infinity-norm of state estimation error in Simulink.

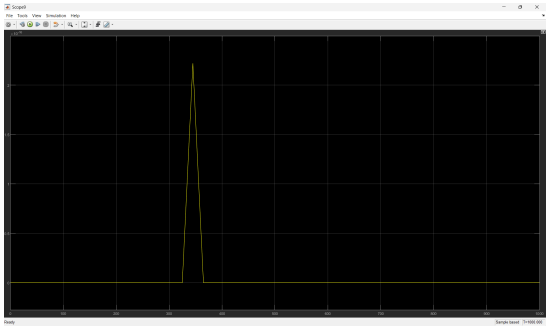

Figure 29: 2-norm of output tracking error in Simulink.

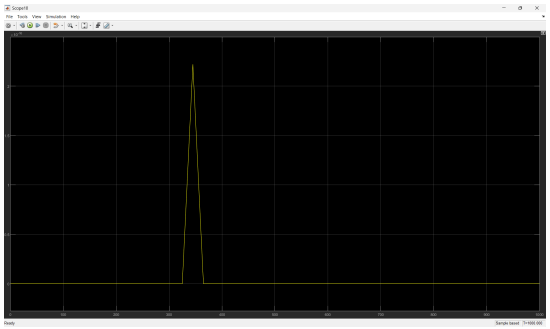

Figure 30: Infinity-norm of output tracking error in Simulink.

9.8 Observer-Based State Feedback Control

The Luenberger observer is employed to estimate states, avoiding the need for direct feedback of all state variables. These estimated states are used to implement a state feedback controller, designed for optimal regulation. Simulink shows that the response using estimated states closely mirrors that of direct state feedback, with minor differences due to estimation errors, and the linear system converges to zero efficiently.

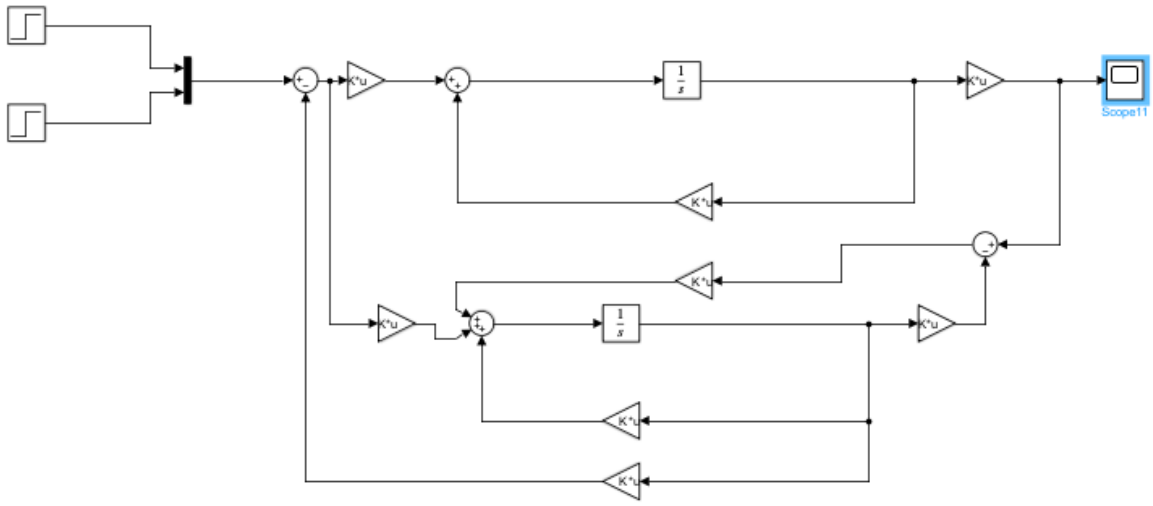

Figure 31: Observer-based state feedback controller in Simulink.

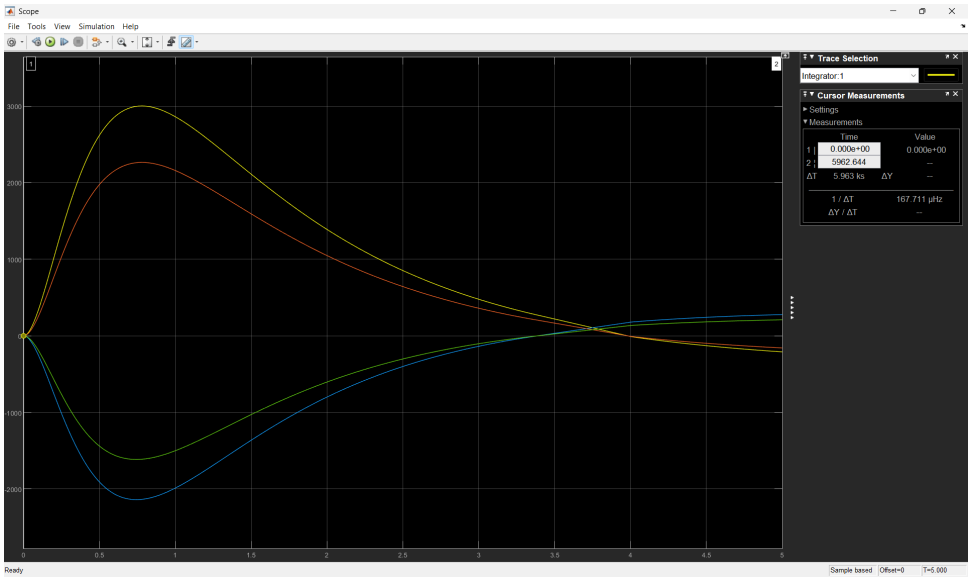

Figure 32: Observer-based state feedback controller used to control nonlinear system

The control signal from the linear model is applied to the nonlinear system over a 5-second simulation (due to computational limits), demonstrating convergence to zero. Extended simulation time, as Johansson suggests, would further clarify this trend, confirming the controller's robust performance across system types.

10 References

- Johansson, K. H. (2000). The Quadruple-Tank Process: A Multivariable Laboratory Process with an Adjustable Zero. *IEEE Transactions on Control Systems Technology*, 8(3), 456–465.
- OpenAI. (2023). ChatGPT: A Conversational AI Model. Retrieved from `https://openai.com/chatgpt`
- MATLAB. (2025). MATLAB and Simulink Documentation. The MathWorks, Inc. Retrieved from `https://www.mathworks.com/help`
- LaTeX Project. (2025). LaTeX - A Document Preparation System. Retrieved from `https://www.latex-project.org`
- YouTube Videos on MIMO Control and Simulation in MATLAB/Simulink:
  - **State-Space Representation in MATLAB and Simulink** — MATLAB Official `https://www.youtube.com/watch?v=qF2-KYrZt1M`
  - **MIMO System Simulation and Control in MATLAB/Simulink** — Steve Brunton `https://www.youtube.com/watch?v=dUVgBBw3eeY`
  - **Controllability and Observability in Control Systems** — Neso Academy `https://www.youtube.com/watch?v=gHw5dj7-E_k`
  - **MIMO LQR Design using MATLAB** — Brian Douglas `https://www.youtube.com/watch?v=4ne2wBzjTzU`
  - **Modeling and Simulating Tank Systems in Simulink** — MATLAB Helper `https://www.youtube.com/watch?v=9uS3vnR0pLk`