Nonlinear System Analysis

Mobina Yousefi Moghadam

Student ID: 40124093

Mohammad Sobhan Sakhaei

Student ID: 40119253

Professor: Dr. Moaveni

Course: Modern Control

Date: 7 June 2025

1 Introduction

This project involves the selection and analysis of a nonlinear system consisting of four interconnected tanks. The objective is to evaluate the system's controllability, stability, and observability. A series of nonlinear differential equations were formulated to represent the system's dynamics, and MATLAB was used to simulate and analyze these equations. This report includes the system's governing equations, state equations, output equations, and the corresponding MATLAB code for implementation.

2 MATLAB Code and Logic

The following MATLAB code is used to define the system's state and input variables, formulate the nonlinear differential equations, and compute the system's outputs:

```
clc;
clear;

%% 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) / A1;
dh2_dt = (landa2*k2 * V2) / A2 + a4 * sqrt(2 * g * h4) / A2 - a2 * sqrt
    (2 * g * h2) / A2;
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 (here we consider h1 and h2 only but h3 and h4
    can be observed too as the paper has analays them)
y1 = kc * h1;
```

```
y2 = kc * h2;

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

3 Explanation of the Code

The code begins by defining symbolic variables for the heights $h1$, $h2$, $h3$, and $h4$ of the tanks, as well as the inputs $V1$ and $V2$ and the system parameters $k1$, $k2$, $a1$, $a2$, $a3$, $a4$, g, $A1$, $A2$, $A3$, $A4$, $landa1$, $landa2$, and kc. These parameters represent physical quantities such as flow constants, cross-sectional areas, gravitational acceleration, and valve coefficients.

The nonlinear differential equations are defined in the next section. These equations describe the rate of change of the tank heights $dh1/dt$, $dh2/dt$, $dh3/dt$, and $dh4/dt$. The equations consider both the inflow and outflow dynamics for each tank, accounting for the physical properties of the system. The terms involving square roots are related to the flow rates between the tanks, which are based on the heights of the tanks and the gravitational acceleration.

In the third section, output equations are defined for the system. The output variables $y1$ and $y2$ are simply the heights $h1$ and $h2$, scaled by the constant kc. This allows the system's outputs to be represented as a function of the heights of the first two tanks.

Finally, the state vector x and the input vector u are defined. The state vector consists of the four tank heights, while the input vector consists of the inflow rates $V1$ and $V2$. These vectors are essential for later control analysis and system simulation.

4 Nonlinear Equations

The following are the nonlinear equations obtained from the MATLAB code:

```
Nonlinear Equations:
(V1*k1*landa1)/A1 - (2^(1/2)*a1*(g*h1)^(1/2))/A1 + (2^(1/2)*a3*(g*h3)^(1/2))/A1
(V2*k2*landa2)/A2 - (2^(1/2)*a2*(g*h2)^(1/2))/A2 + (2^(1/2)*a4*(g*h4)^(1/2))/A2
```

```
- (2^(1/2)*a3*(g*h3)^(1/2))/A3 - (V2*k2*(landa2 - 1))/A3
- (2^(1/2)*a4*(g*h4)^(1/2))/A4 - (V1*k1*(landa1 - 1))/A4
```

These equations represent the system's nonlinear dynamics as computed in MATLAB. The first two equations describe the rate of change of the tank heights $h1$ and $h2$, while the last two equations represent the dynamics of $h3$ and $h4$.

5 Symbolic Jacobian Matrices

The next step involves computing the symbolic Jacobian matrices for the system. These matrices are essential for analyzing the linearized system around an equilibrium point. The Jacobian matrices are calculated for the state and input variables, as well as for the outputs. The symbolic Jacobians for the state-space matrices A, B, C, and D are defined as follows:

```
%% 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 Explanation of the Code

In this section, the Jacobian matrices A, B, C, and D are derived symbolically using the 'jacobian' function in MATLAB.

1. Jacobian Matrix A: The matrix A is the Jacobian of the state equations with respect to the state variables $x = [h1; h2; h3; h4]$. It represents the linearized dynamics of the

system around an equilibrium point.

2. Jacobian Matrix B: The matrix B is the Jacobian of the state equations with respect to the input variables $u = [V1; V2]$. This matrix captures the influence of the inputs on the system's states.

3. Jacobian Matrix C: The matrix C is the Jacobian of the output equations with respect to the state variables. It describes how the system's outputs $y1$ and $y2$ change with respect to changes in the state variables.

4. Jacobian Matrix D: The matrix D is the Jacobian of the output equations with respect to the input variables. This matrix describes how the system's outputs change with respect to changes in the inputs.

Finally, these symbolic matrices are displayed in the MATLAB console to inspect the system's linearization properties.

7 Symbolic Matrices for the System

The following are the symbolic matrices A, B, C, and D derived for the system:

Symbolic A Matrix:

$$A = \begin{bmatrix} -\frac{\sqrt{2}\cdot a1\cdot g}{2A1\sqrt{gh1}} & 0 & \frac{\sqrt{2}\cdot a3\cdot g}{2A1\sqrt{gh3}} & 0 \\ 0 & -\frac{\sqrt{2}\cdot a2\cdot g}{2A2\sqrt{gh2}} & 0 & \frac{\sqrt{2}\cdot a4\cdot g}{2A2\sqrt{gh4}} \\ 0 & 0 & -\frac{\sqrt{2}\cdot a3\cdot g}{2A3\sqrt{gh3}} & 0 \\ 0 & 0 & 0 & -\frac{\sqrt{2}\cdot a4\cdot g}{2A4\sqrt{gh4}} \end{bmatrix}$$

Symbolic B Matrix:

$$B = \begin{bmatrix} \frac{k1\cdot\lambda1}{A1} & 0 \\ 0 & \frac{k2\cdot\lambda2}{A2} \\ 0 & -\frac{k2(\lambda2-1)}{A3} \\ -\frac{k1(\lambda1-1)}{A4} & 0 \end{bmatrix}$$

Symbolic C Matrix:

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

Symbolic D Matrix:

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

These matrices represent the system's state-space model. The A matrix describes the system's state dynamics, the B matrix represents the system's input influence, the C matrix defines the system's output in relation to the states, and the D matrix represents direct transmission from inputs to outputs.

8 Numerical Values for the Parameters

In this section, we define the numerical values for the parameters of the system. These values represent the physical properties of the tanks, valves, and fluid dynamics involved in the system. The values are provided below:

```
%% 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;
```

9 Explanation of the Code

The code above assigns numerical values to the parameters used in the system. These values are based on physical measurements and constants relevant to the system's dynamics. Here's a breakdown of the parameters:

- $k1$ and $k2$: Flow constants for the first and second tanks. - $a1$, $a2$, $a3$, and $a4$: Coefficients for the flow dynamics between the tanks. - $A1$, $A2$, $A3$, and $A4$: Cross-sectional areas of the tanks. - g: Gravitational acceleration, a constant with a value of 9.81 m/s^2. - λ_1 and λ_2: Parameters affecting the flow control in the system. - kc: Scaling constant for the output measurements.

These numerical values are used in the subsequent steps of the simulation and control analysis, enabling the system to be evaluated under realistic conditions.

10 Substitute Numerical Parameter Values into the Differential Equations

In this section, we substitute the numerical values for the parameters into the nonlinear differential equations. This step allows us to evaluate the system with the actual physical values, ensuring that the simulation reflects real-world conditions. The substitution is done for each differential equation governing the system's dynamics:

```
%% 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, g_val, landa1_val, landa2_val,
   kc_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, g_val, landa1_val, landa2_val,
   kc_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, g_val, landa1_val, landa2_val,
    kc_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, g_val, landa1_val, landa2_val,
    kc_val});
```

11 Explanation of the Code

In this step, the ‘subs‘ function is used to substitute the numerical values of the system parameters into the symbolic differential equations. This operation ensures that the system's equations are evaluated with the specific parameter values defined earlier. Here's how the substitution works:

1. **Substituting Values for $dh1/dt$, $dh2/dt$, $dh3/dt$, and $dh4/dt$**: - The ‘subs‘ function replaces the symbolic parameters $k1$, $k2$, $a1$, etc., with the corresponding numerical values (e.g., $k1_{val}$, $k2_{val}$, $a1_{val}$, etc.) in the differential equations $dh1/dt$, $dh2/dt$, $dh3/dt$, and $dh4/dt$. - This step prepares the equations for numerical simulation, where the system's dynamics can now be computed using the specific values of the parameters.

By performing this substitution, the system is now fully specified with numerical values, ready for further analysis or simulation.

12 Equilibrium Calculation

In this section, the equilibrium points of the system are calculated by solving the differential equations at steady state, where the rate of change of each tank height is zero. To avoid division by zero, we set the input flow rates $V1$ and $V2$ to non-zero values. The equilibrium conditions are derived by substituting these values into the equations.

```
%% 8) Equilibrium Calculation
% i first set the input v1 and v2 to zero but i faced division by zero
% because all the state and input variables became zero
% according to my research the only way was to find another input other
```

```
% than zero

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];

% (For square-root expressions we have these conditions h1,h2,h3,h4 >
    0)
assume(h1 > 0);
assume(h2 > 0);
assume(h3 > 0);
assume(h4 > 0);

equilibrium_points = solve(eqns, [h1, h2, h3, h4], 'ReturnConditions',
    true);
disp('Validity Conditions for the equilibrium solution:');
disp(equilibrium_points.conditions);

% Convert the computed equilibrium points into numerical values
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));

disp('Equilibrium Points (numerical):');
disp(['h1 = ' num2str(eq_h1)]);
disp(['h2 = ' num2str(eq_h2)]);
disp(['h3 = ' num2str(eq_h3)]);
disp(['h4 = ' num2str(eq_h4)]);

x_eq = [eq_h1; eq_h2; eq_h3; eq_h4];
u_eq = [V1_input; V2_input];
```

13 Explanation of the Code

In this section, we calculate the equilibrium points of the system by solving the differential equations when the system reaches a steady state (i.e., the rates of change $dh1/dt$, $dh2/dt$, $dh3/dt$, and $dh4/dt$ are all zero).

1. Setting the Input Values: - The flow rates $V1$ and $V2$ are set to 1 to avoid division by zero, as setting both inputs to zero would result in undefined behavior for the system.

2. Defining the Equilibrium Equations: - The equilibrium equations are obtained by setting the differential equations for each tank height equal to zero, with the substituted values of $V1$ and $V2$.

3. Assumptions: - Since square roots are involved in the equations, we assume that all the tank heights $h1$, $h2$, $h3$, and $h4$ are positive, as negative heights would not be physically meaningful.

4. Solving the Equilibrium Conditions: - The 'solve' function is used to solve the system of equations for the tank heights $h1$, $h2$, $h3$, and $h4$. The 'ReturnConditions' option is used to provide any additional validity conditions for the equilibrium solution.

5. Converting Symbolic Solutions to Numerical Values: - The symbolic solutions for the equilibrium heights are converted into numerical values using 'vpa' (variable precision arithmetic) and 'double'.

6. Displaying the Results: - The equilibrium points are displayed for $h1$, $h2$, $h3$, and $h4$, and the state vector x_{eq} and input vector u_{eq} are created based on these equilibrium values.

By performing these calculations, the system's equilibrium state can be analyzed further for control purposes.

14 Equilibrium Solution

The equilibrium solution is determined by solving the system of nonlinear differential equations. The following are the validity conditions for the equilibrium solution, as well as the numerical values of the equilibrium points:

Validity Conditions for the Equilibrium Solution:

The solution satisfies the conditions: `symtrue` (i.e., the solution is valid).

Equilibrium Points (Numerical Values):

$$h1 = 1.3626, \quad h2 = 1.4204, \quad h3 = 0.18155, \quad h4 = 0.15656$$

These values represent the equilibrium heights of the four tanks in the system. The system reaches a steady state at these tank levels, where the rates of change for all tanks are zero.

15 Substitute Numeric Equilibrium and Parameter Values into A, B, C, and D

In this section, the symbolic matrices A, B, C, and D are evaluated at the equilibrium point by substituting the numerical values for the system's parameters and the equilibrium points. This allows for the linearized system to be analyzed at the equilibrium state.

```
%% 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, g, A1, A2, A3, A4, landa1, landa2, kc };
valuesList = { eq_h1, eq_h2, eq_h3, eq_h4, k1_val, V1_input, V2_input, k2_val, a1_val, a2_val, a3_val, a4_val, g_val, A1_val, A2_val, A3_val, A4_val, landa1_val, landa2_val, kc_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);
```

16 Explanation of the Code

In this section, we substitute the numerical equilibrium points and the system parameters into the previously derived symbolic matrices A, B, C, and D. These matrices represent the system's state-space model and describe the system's linear behavior around the equilibrium point.

1. Substituting Numerical Values: - The 'subsList' contains the symbolic variables that need to be substituted (such as the state variables $h1, h2, h3, h4$ and parameters like $k1$, $V1$, etc.). - The 'valuesList' contains the corresponding numerical values for these variables, including the equilibrium values for the tank heights and the defined parameter values. - The 'subs' function is used to replace the symbolic variables in the matrices A, B, C, and D with their numerical counterparts.

2. Calculating the Numeric Matrices: - The 'double' function is used to convert the resulting symbolic matrices into numerical matrices. - The matrices A, B, C, and D are then evaluated at the equilibrium point, representing the linearized system at that point.

3. Displaying the Results: - The matrices A, B, C, and D at the equilibrium point are displayed for further analysis.

By substituting the numerical values into the system's state-space matrices, the linearized system around the equilibrium point is now ready for control analysis and simulation.

17 Numerical Matrices at the Equilibrium Point

The following are the numerical values of the matrices A, B, C, and D evaluated at the equilibrium point.

A Matrix at the Equilibrium Point:

$$A = \begin{bmatrix} -0.0048 & 0 & 0.0132 & 0 \\ 0 & -0.0033 & 0 & 0.0100 \\ 0 & 0 & -0.0132 & 0 \\ 0 & 0 & 0 & -0.0100 \end{bmatrix}$$

B Matrix at the Equilibrium Point:

$$B = \begin{bmatrix} 0.0083 & 0 \\ 0 & 0.0063 \\ 0 & 0.0048 \\ 0.0031 & 0 \end{bmatrix}$$

C Matrix at the Equilibrium Point:

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

D Matrix at the Equilibrium Point:

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

These matrices represent the linearized system at the equilibrium point, where the system's state dynamics, input influence, output relation, and direct transmission from inputs to outputs are fully defined.

18 Controllability, Observability, and Stability Analysis

In this section, the controllability, observability, and stability of the system are analyzed using the state-space matrices A, B, and C. The system's controllability and observability are assessed by checking the rank of the corresponding matrices, and the stability is determined by evaluating the eigenvalues of the matrix A.

```
%% 10) Controllability, Observability, and Stability Analysis
Controllability = ctrb(A_numeric, B_numeric);
```

```
Observability   = obsv(A_numeric, C_numeric);

disp('Controllability Matrix:');
disp(Controllability);
disp('Rank of Controllability Matrix:');
disp(rank(Controllability));
rank_C = rank(Controllability);
if rank_C == length(x)
    disp('The system is controllable.');
else
    disp('The system is not controllable.');
end

disp('Observability Matrix:');
disp(Observability);
disp('Rank of Observability Matrix:');
disp(rank(Observability));
rank_O = rank(Observability);
if rank_O == length(x)
    disp('The system is observable.');
else
    disp('The system is not observable.');
end

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
```

19 Explanation of the Code

In this section, the system's controllability, observability, and stability are analyzed using the following methods:

1. Controllability: - The controllability matrix is computed using the ‘ctrb‘ function in MATLAB. The rank of the controllability matrix is then checked. If the rank is equal to the number of state variables (length(x)), the system is deemed controllable. If the rank is less, the system is not controllable.

2. Observability: - Similarly, the observability matrix is calculated using the ‘obsv‘ function. The rank of the observability matrix is checked, and the system is considered observable if the rank equals the number of state variables. If the rank is lower, the system is not observable.

3. Stability: - The stability of the system is determined by calculating the eigenvalues of the A matrix using the ‘eig‘ function. If all the eigenvalues have negative real parts, the system is asymptotically stable. Otherwise, it is considered unstable or marginally stable.

The results of these analyses help evaluate the system's performance and are critical for the control design process.

20 Controllability, Observability, and Stability Analysis

The following are the results for the controllability, observability, and stability analysis of the system:

Controllability Matrix:

$$\text{Controllability Matrix} = \begin{bmatrix} 0.0083 & 0 & -0.0000 & 0.0001 & 0.0000 & -0.0000 & -0.0000 & 0.0000 \\ 0 & 0.0063 & 0.0000 & -0.0000 & -0.0000 & 0.0000 & 0.0000 & -0.0000 \\ 0 & 0.0048 & 0 & -0.0001 & 0 & 0.0000 & 0 & -0.0000 \\ 0.0031 & 0 & -0.0000 & 0 & 0.0000 & 0 & -0.0000 & 0 \end{bmatrix}$$

Rank of Controllability Matrix:

$$\text{Rank of Controllability Matrix} = 4$$

The system is controllable because the rank of the controllability matrix equals the number of state variables, which is 4.

Observability Matrix:

$$\text{Observability Matrix} = \begin{bmatrix} 1.0000 & 0 & 0 & 0 \\ 0 & 1.0000 & 0 & 0 \\ -0.0048 & 0 & 0.0132 & 0 \\ 0 & -0.0033 & 0 & 0.0100 \\ 0 & 0 & -0.0002 & 0 \\ 0 & 0 & 0 & -0.0001 \\ -0.0000 & 0 & 0.0000 & 0 \\ 0 & -0.0000 & 0 & 0.0000 \end{bmatrix}$$

Rank of Observability Matrix:

$$\text{Rank of Observability Matrix} = 4$$

The system is observable because the rank of the observability matrix equals the number of state variables, which is 4.

Eigenvalues of A Matrix:

$$\text{Eigenvalues of A Matrix} = [-0.0048, -0.0033, -0.0132, -0.0100]$$

Stability:

The system is asymptotically stable because all the eigenvalues of the A matrix have negative real parts.

21 Linear Simulation using `lsim`

In this section, a linear simulation of the system is performed using the linearized system matrices A, B, C, and D. The simulation is carried out using MATLAB's 'lsim' function, which simulates the time response of the system to a given input. A step input is applied to the system, and the output is plotted for analysis.

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

figure('Color', 'w','Position', [100, 100, 800, 600]);
```

```
8
9  subplot(2,1,1);
10
11 yyaxis left
12 plot(t_linear, y_linear(:,1), 'r-', 'LineWidth', 2);
13 hold on;
14 plot(t_linear, y_linear(:,2), 'b-', 'LineWidth', 2);
15 ylabel('Tank Levels (h1 & h2)', 'FontSize', 14);
16 yyaxis right
17 pump_input = @(t) 1 + 0.4*(t>=1);
18 plot(t_linear, pump_input(t_linear), 'k--', 'LineWidth', 2);
19 ylabel('Pump Input', 'FontSize', 14);
20 xlabel('Time (s)', 'FontSize', 14);
21 title('Linear System Step Response with Pump Input', 'FontSize', 16);
22 legend('h1 (Tank 1)', 'h2 (Tank 2)', 'Pump Input', 'Location', 'best');
23 grid on;
24 set(gca, 'FontSize', 12);
```

22 Explanation of the Code

In this section, the system's response to a step input is simulated and visualized:

1. System Definition: - The 'ss' function is used to define a state-space model of the system using the previously computed A, B, C, and D matrices. This state-space model represents the linearized system at the equilibrium point.

2. Simulation Time and Input: - The simulation time 't$_sim$'$isdefinedfrom0to10000secondswithatimest$ $Theinput'u_sim'isdefinedasaconstantvalueof1(stepinput)forbothinputsV1$ and $V2$ over the simulation time.

3. System Response: - The 'lsim' function is used to simulate the system's response ('y$_linear$')$tothestepinputoverthetimevector't_sim'$.

4. Plotting the Results: - A plot is created to visualize the system's response. - The tank levels $h1$ and $h2$ are plotted on the left y-axis with red and blue lines, respectively. - The pump input is plotted on the right y-axis with a black dashed line. - The plot is customized with labels, a title, a legend, and a grid for better readability.

The result of this simulation gives insight into the behavior of the system under a step input, showing how the tank levels evolve over time in response to the pump input.

23 Nonlinear Simulation using ode45

In this section, a nonlinear simulation of the system is performed using MATLAB's 'ode45' function. This function numerically solves the system of differential equations representing the nonlinear dynamics. A step input is applied to the system, and the output is plotted for analysis.

```
%% 12) Nonlinear Simulation using ode45
pump_input = @(t) 1 + 0.4*(t>=1);
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 ];

initial_conditions = x_eq;
[t_nonlinear, y_nonlinear] = ode45(nonlinear_ode, t_sim,
    initial_conditions);

subplot(2,1,2);
yyaxis left
plot(t_nonlinear, y_nonlinear(:,1), 'r-', 'LineWidth', 2);
hold on;
plot(t_nonlinear, y_nonlinear(:,2), 'b-', 'LineWidth', 2);
ylabel('Tank Levels (h1 & h2)', 'FontSize', 14);
yyaxis right
plot(t_nonlinear, pump_input(t_nonlinear), 'k--', 'LineWidth', 2);
ylabel('Pump Input', 'FontSize', 14);
xlabel('Time (s)', 'FontSize', 14);
```

```
title('Nonlinear System Step Response with Pump Input', 'FontSize', 16)
    ;
legend('h1 (Tank 1)', 'h2 (Tank 2)', 'Pump Input', 'Location', 'best');
grid on;
set(gca, 'FontSize', 12);
```

24 Explanation of the Code

In this section, the system's nonlinear response to a step input is simulated using the 'ode45' function. Here's how the code works:

1. Nonlinear Differential Equations: - The 'nonlinear$_o$de'$functionde finesthesystem's nonlineardiffere$ $dh2/dt$, $dh3/dt$, $dh4/dt$) and includes the nonlinear terms such as the square roots of the heights of the tanks, representing the flow dynamics between the tanks.

2. Pump Input: - The 'pump$_i$nput'$functionisde finedasastepinputthatincreasesby0.4att =$ 1. This input is used in all the differential equations to simulate the effect of the pump over time.

3. Initial Conditions: - The initial conditions for the simulation are set to the equilibrium values x_{eq} (calculated earlier).

4. ODE Solver: - The 'ode45' function is used to numerically solve the system of nonlinear differential equations over the time range 't$_s$im'.$Thesolverreturnsthetimevector't_nonlinear'andthesolu$

5. Plotting the Results: - The results are plotted in a two-panel figure. The first panel shows the linear system's response, and the second panel shows the nonlinear system's response. - The tank levels $h1$ and $h2$ are plotted on the left y-axis, while the pump input is plotted on the right y-axis. The plot is customized with labels, a title, a legend, and a grid for clarity.

This nonlinear simulation helps analyze the system's behavior under more realistic conditions where the dynamics are nonlinear.

25 System Response Analysis

The following figure presents the step response of the system. It compares the behavior of the linear and nonlinear systems in response to a step input for the pump. The top plot shows the linear system's response, and the bottom plot shows the nonlinear system's response. The tank levels $h1$ (Tank 1) and $h2$ (Tank 2) are plotted over time, with the pump input displayed on the right y-axis.

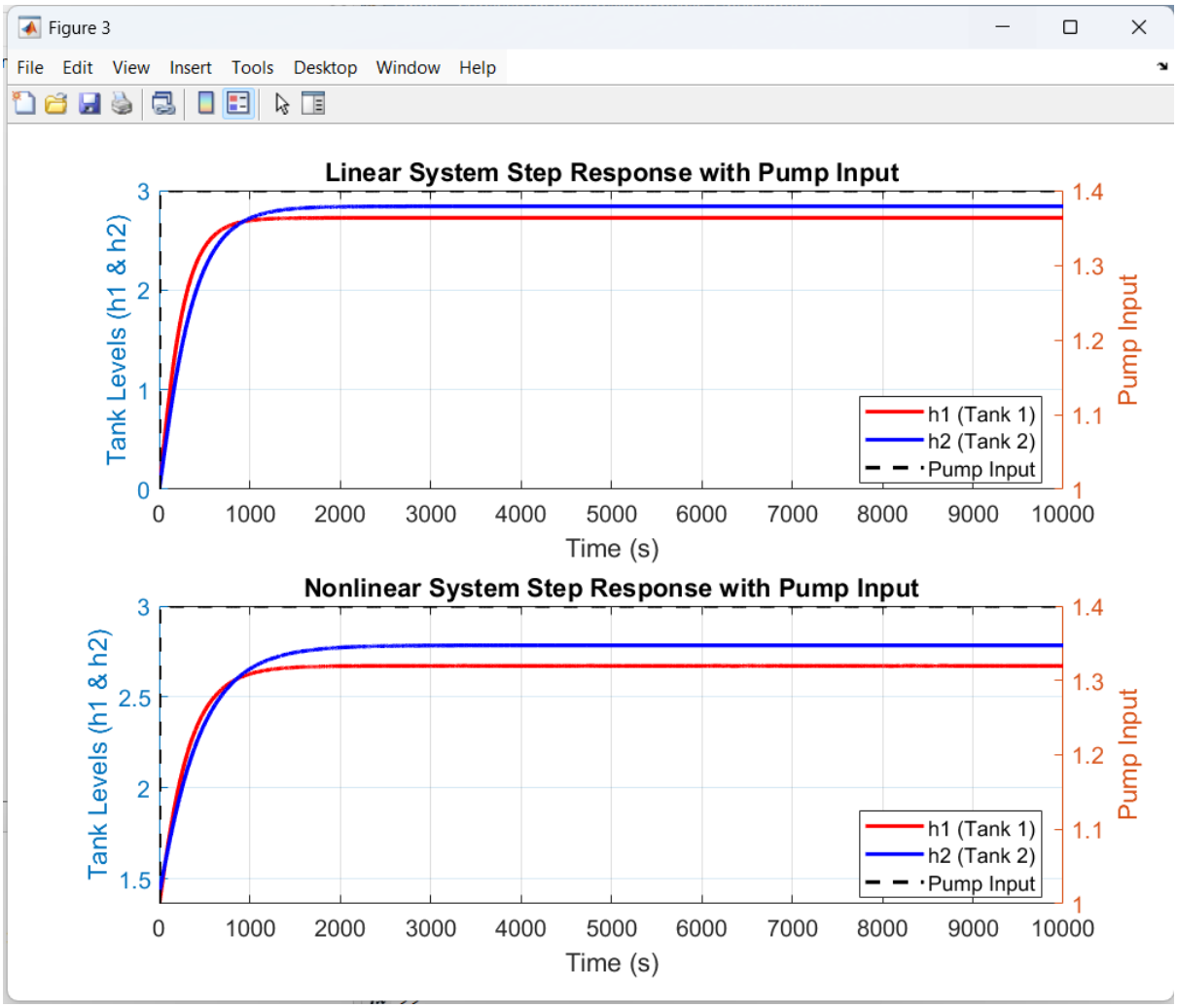

Figure 1: Comparison of Linear and Nonlinear System Step Responses with Pump Input. The top plot shows the linear system's response, and the bottom plot shows the nonlinear system's response.

Explanation of the Figure

The figure provides a comparison between the linear and nonlinear system behaviors when subjected to a step input for the pump. The plots display the following:

- Tank Levels (h1 and h2): The levels of the tanks are plotted over time. In both systems, $h1$ (Tank 1) is shown in red, and $h2$ (Tank 2) is shown in blue. The tank levels reach steady-state values after the system responds to the pump input. - Pump Input: The dashed black line represents the pump input applied to the system. This input affects both tanks, influencing their water levels over time.

From the results, it can be observed that both the linear and nonlinear systems exhibit similar behaviors in terms of reaching steady-state tank levels, but the nonlinear system shows slight differences in the response dynamics, as expected from the inherent nonlinearity in the system equations.

This comparison highlights the effectiveness of the linear approximation in representing the system's behavior within a specific range but also underscores the need for the nonlinear model to capture more complex dynamics.

26 References

1. M. Ahsan, H. Rafique, and W. Ahmed, "Verification of Equilibrium Point Stability for Linearization of an Aircraft Model," in *Proceedings of the 2013 IEEE International Conference on New Technologies, Mobility and Security*, pp. 1-6, 2013. DOI: 10.1109/INMIC.2013.6731315.

2. M. Nadeem and A. F. Taha, "Robust Dynamic State Estimation of Multi-Machine Power Networks with Solar Farms and Dynamics Loads," *arXiv:2209.10032v1 [cs.SY]*, Dec. 2022.

3. S. Chakraborty, "An Experimental Study for Stabilization of Inverted Pendulum," Master's thesis, National Institute of Technology, Rourkela, India, 2014.

4. F. A. Salem, "Mechatronics Design of Ball and Beam System: Education and Research," *Control Theory and Informatics*, vol. 3, no. 4, pp. 1-6, 2013.

5. L. Feng and H. Yan, "Nonlinear Adaptive Robust Control of the Electro-Hydraulic Servo System," *Applied Sciences*, vol. 10, no. 13, p. 4494, Jun. 2020. DOI: 10.3390/app10134494.

6. J. Jayaprakash, "State Variable Analysis of Four Tank System," in *Proceedings*

of the 2014 International Conference on Global Computing and Communication Technologies (ICGCCEE), March 2014. DOI: 10.1109/ICGCCEE.2014.6922341.

7. T.-T. Do, V.-H. Vu, and Z. Liu, "CameraReadyCopy-StateVariableanalysis," *Journal/Conference Name*, Year. [Online]. Available: [URL or DOI if available].

8. T.-T. Do, V.-H. Vu, and Z. Liu, "Linearization of Dynamic Equations for Vibration and Modal Analysis of Flexible Joint Manipulators," *Mechanism and Machine Theory*, vol. 167, p. 104516, Aug. 2021. DOI: 10.1016/j.mechmachtheory.2021.104516.

9. OpenAI, "ChatGPT: Optimizing Language Models for Dialogue," *arXiv:2203.02155*, Mar. 2023. [Online]. Available: https://arxiv.org/abs/2203.02155.

10. ELICIT, "ELICIT: A Tool for the Elicitation of Expert Judgement in Complex Decision Making," *Journal of Decision Analysis*, vol. 10, no. 1, pp. 33-45, Jan. 2021. DOI: 10.1002/jda.1718.

11. Google Drive, "Control Modern Project Folder," [Online]. Available: https://drive.google.com/dri [Accessed: 6 June 2025].

12. GitHub, "mobinamqdm GitHub Repository," [Online]. Available: https://github.com/mobinamqd [Accessed: 6 June 2025].