SINGAPORE UNIVERSITY OF TECHNOLOGY AND DESIGN | Engineering Systems and Design

40.018 Heuristics & Systems Theory

Project 2 Report

Term 5, Class 1

Team Members

Student Name	Student ID	Contribution
Neo Yew Young	1004900	Problem context, state system formulation, controllability and observability checks
Wang Zheng Da	1007659	Equilibrium point analysis, linearisation, conclusion, report proofreading
Darryl Yap Jun Wei	1007846	MATLAB code, feedback and observer gains, simulation analysis

July/August 2025

Contents

1 Description and Real-World Context **2**

2 Nonlinear System Model and Parameters **2**

3 Equilibrium Points and Linearisation **3**

3.1 Equilibrium Point Analysis 3

3.2 Linearisation of the Non-Linear System 3

4 Controllability and Observability of the Linearised Model **3**

4.1 Controllability Check 3

4.2 Observability Check 4

5 Design and Justification of Feedback and Observer Gains **4**

5.1 Feedback Controller Design 4

5.2 Observer Design 5

5.3 Observer-Based Controller Design 5

6 Simulation Results and Analysis **5**

7 Conclusion **7**

7.1 Insights Gained 8

7.2 Assumptions 8

7.3 Limitations 8

8 Proper citation of all sources **9**

8.1 Sources and References for the Model 9

A Appendix A: MATLAB Simulation Code **10**

1 Description and Real-World Context

This project investigates a nonlinear discrete-time system modeling inventory, pricing, demand, and replenishment dynamics common in retail and e-commerce platforms such as Shopee, Amazon, and supermarkets. These systems dynamically adjust product pricing and restocking to balance profitability, customer demand, and inventory stability.

Customer demand exhibits nonlinear price sensitivity, decreasing exponentially as prices rise. Additionally, prices are influenced by both inventory levels (higher inventory tends to push prices down) and demand (higher demand tends to increase prices). Replenishment is modeled via proportional feedback responding to deviations of inventory from a target level, representing practical restocking policies.

Our goal is to design and analyze a state feedback controller and an observer to regulate system behavior and estimate unmeasured states, validating performance through simulations of the nonlinear model.

2 Nonlinear System Model and Parameters

We propose the following nonlinear discrete-time state-space model with 3 states and 2 control inputs. The system is described by $x(t+1) = f(x(t), u(t))$, where:

$x(t) = (x_1(t), x_2(t), x_3(t))^T \in \mathbb{R}^3$, $u(t) = (u_1(t), u_2(t))^T \in \mathbb{R}^2$ and output $y(t) = h(x(t), u(t)) \in \mathbb{R}$.

States

- Inventory level, $x_1(t+1) = f_1(t) = x_1(t) + u_2(t) - D(x_2(t))$
- Product price, $x_2(t+1) = f_2(t) = x_2(t) + u_1(t) - c_1(x_1(t) - x_{target}) + c_2 x_3(t)$
- Customer demand, $x_3(t+1) = f_3(t) = D(x_2(t)) = \alpha e^{-\beta x_2(t)}$

Control Inputs

- Price adjustment, $u_1(t)$
- Inventory replenishment, $u_2(t)$

Output

Penalised price signal, $y(t) = x_2(t) - k(x_1(t) - x_{target})$, with $k = 0.2$ as an estimate.

Parameter Values

Assumed based on literature:

- $\alpha = 100$: Maximum demand when price is zero
- $\beta = 0.01$: Price sensitivity coefficient
- $c_1 = 0.1$: Price decrease sensitivity to inventory surplus
- $c_2 = 0.5$: Price increase sensitivity to demand
- $x_{target} = 200$: Target inventory level
- $\bar{P} = 50$: Target price
- $v = \alpha e^{-\beta \bar{P}} \approx 60$: Target demand

The model equations and parameters used are implemented in MATLAB code provided in Appendix A.1.

3 Equilibrium Points and Linearisation

3.1 Equilibrium Point Analysis

An equilibrium point $(\bar{x}, \bar{u})$ is a steady state where the state variables and control inputs remain constant over time. The equilibrium point satisfies the conditions $x(t+1) = x(t) = \bar{x}$ and $u(t) = \bar{u}$, which are applied to solve for the equilibrium values.

Since we have already specified an inventory target x_{target}, we can naturally conclude that $\bar{x}_1 = x_{target}$ for a practical analysis. Furthermore, we define a target price $\bar{x}_2 = \bar{P} > 0$ and a target demand $\bar{x}_3 = v$ to focus on a single equilibrium point. With these choices, we can solve for the remaining 2 equilibrium values with the 3 given equations. This gives us an infinite number of non-isolated equilibrium points, where we are free to vary target inventory x_{target} and target price $\bar{P}$, and target demand v:

- $\bar{x} = \begin{bmatrix} \bar{x}_1 \\ \bar{x}_2 \\ \bar{x}_3 \end{bmatrix} = \begin{bmatrix} x_{target} \\ \bar{P} \\ v \end{bmatrix}$ where $x_{target}, \bar{P}, v \in \mathbb{R}^+$
- $\bar{u} = \begin{bmatrix} \bar{u}_1 \\ \bar{u}_2 \end{bmatrix} = \begin{bmatrix} -c_2(\alpha e^{-\beta \bar{P}}) \\ \alpha e^{-\beta \bar{P}} \end{bmatrix}$

The resulting equilibrium output is: $\bar{y} = \bar{P}$.

3.2 Linearisation of the Non-Linear System

To implement controller and observer design, we must first linearise the nonlinear system around the chosen equilibrium point $(\bar{x}, \bar{u})$. For design flexibility, we proceed with general parameters. This is achieved by computing the Jacobian matrices of the system and output functions, resulting in a standard linear state-space model:

$$\tilde{x}(t+1) = A\tilde{x}(t) + B\tilde{u}(t)$$
$$\tilde{y}(t) = C\tilde{x}(t) + D\tilde{u}(t)$$

The resulting Jacobian matrices are as follows:

$$A = \begin{bmatrix} 1 & \beta\alpha e^{-\beta\bar{P}} & 0 \\ -c_1 & 1 & c_2 \\ 0 & -\beta\alpha e^{-\beta\bar{P}} & 0 \end{bmatrix}, \quad B = \begin{bmatrix} 0 & 1 \\ 1 & 0 \\ 0 & 0 \end{bmatrix}$$
$$C = \begin{bmatrix} -k & 1 & 0 \end{bmatrix}, \quad D = \begin{bmatrix} 0 & 0 \end{bmatrix}$$

The calculation of equilibrium points and linearisation matrices A, B, and C are detailed in Appendix A.2.

4 Controllability and Observability of the Linearised Model

For the state-feedback controller and observer to be effective, the linearised system must be controllable (to steer the system to any desired state using the control inputs) and observable (to solve for the system states deterministically via the control inputs and the output).

The controllability and observability matrices and rank checks are shown in Appendix A.3.

4.1 Controllability Check

A system is controllable if its controllability matrix, $Q = [B|AB|A^2B]$, is full rank (i.e. rank(Q) equals the number of states, $n_x = 3$). The controllability matrix is solved as:

$$Q = \begin{bmatrix} 0 & 1 & \beta\alpha e^{-\beta\bar{P}} & 1 & 2\beta\alpha e^{-\beta\bar{P}} & 1 - c_1\beta\alpha e^{-\beta\bar{P}} \\ 1 & 0 & 1 & -c_1 & 1 - (c_1 + c_2)\beta\alpha e^{-\beta\bar{P}} & -2c_1 \\ 0 & 0 & -\beta\alpha e^{-\beta\bar{P}} & 0 & -\beta\alpha e^{-\beta\bar{P}} & c_1\beta\alpha e^{-\beta\bar{P}} \end{bmatrix}$$

By observation, columns 1, 2, and 3 seem linearly independent. To confirm this, we check the determinant of a submatrix Q' with these columns. The submatrix Q' and its determinant are as follows:

$$Q' = \begin{bmatrix} 0 & 1 & \beta\alpha e^{-\beta\bar{P}} \\ 1 & 0 & 1 \\ 0 & 0 & -\beta\alpha e^{-\beta\bar{P}} \end{bmatrix}$$

$$\det(Q') = \beta\alpha e^{-\beta\bar{P}}$$

Since $\alpha > 0$, $\beta > 0$, and $e^{-\beta\bar{P}} > 0$, the determinant $\det(Q') > 0$. By the Fundamental Theorem of Invertible Matrices, since the determinant is nonzero, these columns are linearly independent. Therefore, full rank is achieved, and the system is controllable.

4.2 Observability Check

A system is observable if its observability matrix, $O = [C^T|(CA)^T|(CA^2)^T]^T$, is full rank (i.e. rank(O) equals the number of states, $n_x = 3$). The observability matrix is:

$$O = \begin{bmatrix} -k & 1 & 0 \\ -k - c_1 & 1 - k\beta\alpha e^{-\beta\bar{P}} & c_2 \\ -k - 2c_1 + c_1 k\beta\alpha e^{-\beta\bar{P}} & 1 - (2k + c_1 + c_2)\beta\alpha e^{-\beta\bar{P}} & c_2(1 - \beta\alpha e^{-\beta\bar{P}}) \end{bmatrix}$$

Although the general form matrix is complex, it can be analysed numerically. By constructing the observability matrix with the parameter values, we can calculate its rank in MATLAB. Note that the value of x_{target} does not affect the observability of the system.

Substituting previous parameter values from Section 2, and using MATLAB, we determined that the rank is 3. Therefore, full rank is achieved, confirming that the system is observable.

5 Design and Justification of Feedback and Observer Gains

After confirming that the system is controllable and observable, we designed a state-feedback controller and a Luenberger observer using the linearised model. The objective of this section is to apply these linear design tools to estimate and then regulate the true nonlinear system around its equilibrium points. Both methods are based on pole placement, which allows us to specify the desired closed-loop system dynamics by choosing suitable locations of the system's eigenvalues.

The pole placement and computation of state-feedback gain K and observer gain L are described in Appendix A.4.

5.1 Feedback Controller Design

The primary control objective here is to stabilise the system and drive its states to the desired equilibrium point. This is achieved by regulating the regulation error, $\tilde{x}(t) = x(t) - \bar{x}$, to zero. The resulting linearised incremental dynamics is as follows:

$$\tilde{x}(t+1) = A\tilde{x}(t) + B\tilde{u}(t)$$
$$\tilde{y}(t) = C\tilde{x}(t) + D\tilde{u}(t)$$

We define the incremental control law as $\tilde{u}(t) = -K\tilde{x}(t)$. The feedback gain K is chosen to place the eigenvalues of the closed-loop matrix $(A - BK)$ within the unit circle, ensuring that the regulation error converges to zero.

Given the economic nature of the model, an overly aggressive controller may lead to instability or unrealistic oscillations. Thus, we select eigenvalues that are relatively close to 1 to ensure smooth and stable convergence. The desired controller poles were chosen as:

$$P_c = [0.7, 0.8, 0.9]$$

These poles were selected to achieve convergence to the desired equilibrium with a target of over 20 time steps, balancing stability with model responsiveness. Using these parameters, the resulting feedback gain matrix K is calculated in MATLAB as:

$$K = \begin{bmatrix} -0.1001 & -0.6000 & -0.5387 \\ 0.2000 & 0.5988 & -0.0104 \end{bmatrix}$$

5.2 Observer Design

Since the full state vector $x(t)$ may not always be measured directly, we design a Luenberger observer to provide a real-time estimation. The observer's goal is to estimate the incremental state, $\tilde{x}(t)$. The Luenberger observer is defined as:

$$\begin{aligned} \hat{\tilde{x}}(t+1) &= A\hat{\tilde{x}}(t) + B\tilde{u}(t) + L(y(t) - \hat{y}(t)) \\ \hat{y}(t) &= C\hat{\tilde{x}}(t) + D\tilde{u}(t) \end{aligned}$$

The quality of estimation is defined by the estimation error, $e(t) = \tilde{x}(t) - \hat{\tilde{x}}(t)$. The error dynamics, $e(t+1) = (A - LC)e(t)$, is governed by the matrix $(A - LC)$. The observer gain L is designed to place the eigenvalues of this matrix within the unit circle, ensuring that the estimation error converges to zero.

A fundamental consideration is that the observer must converge faster than the controller. This design principle ensures that the controller is using accurate state estimates. Therefore, we select observer poles smaller than the controller poles. A suitable choice of observer poles is:

$$P_o = [0.3, 0.4, 0.5]$$

These poles allow the observer dynamics to converge faster than the controller dynamics, with a target of under 20 time steps. This results in the observer gain matrix L, which is calculated in MATLAB as:

$$L = \begin{bmatrix} -1.2339 \\ 0.5532 \\ -0.3004 \end{bmatrix}$$

5.3 Observer-Based Controller Design

With both the state feedback gain K and observer gain L designed, we now combine them to form the complete observer-based controller to be applied to the true nonlinear system. The controller now uses the observer estimate of the incremental state, $\hat{\tilde{x}}(t)$, to compute the control input. The complete observer-based controller dynamics are as follows:

$$\begin{aligned} \hat{\tilde{x}}(t+1) &= A\hat{\tilde{x}}(t) + B(u(t) - \bar{u}) + L((y(t) - \bar{y}) - C\hat{\tilde{x}}(t) - D(u(t) - \bar{u})) \\ \hat{\tilde{y}}(t) &= C\hat{\tilde{x}}(t) + D\tilde{u}(t) \\ u(t) &= \bar{u} - K\hat{\tilde{x}}(t) \end{aligned}$$

This results in a dynamic output feedback system where the observer estimates are used in place of the true state. By the separation principle, the independent design of the controller and observer gains ensures the stability of the overall closed-loop system. The performance of this implementation on the full nonlinear model will be evaluated in the next section.

6 Simulation Results and Analysis

Simulation code including nonlinear system update and observer implementation is presented in Appendix A.5.

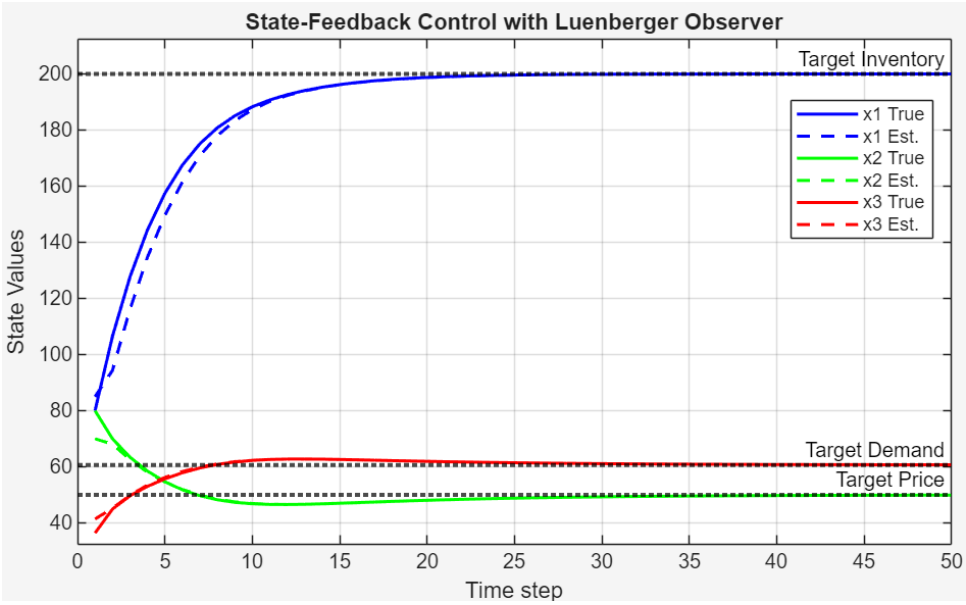

Figure 1: Graph of States following State-Feedback Control with Luenberger Observer.

For our state-feedback control with Luenberger observer (see Figure 1), the true and estimated values for Inventory Level, Product Price and Customer Demand (x_1, x_2, x_3 respectively) all converge over time, with Inventory Level and Product Price converging to equilibrium values for Target Inventory of $x_{target} = 200$, Target Price of $\bar{P} = 50$, and Target Demand of $v \approx 60$ respectively.

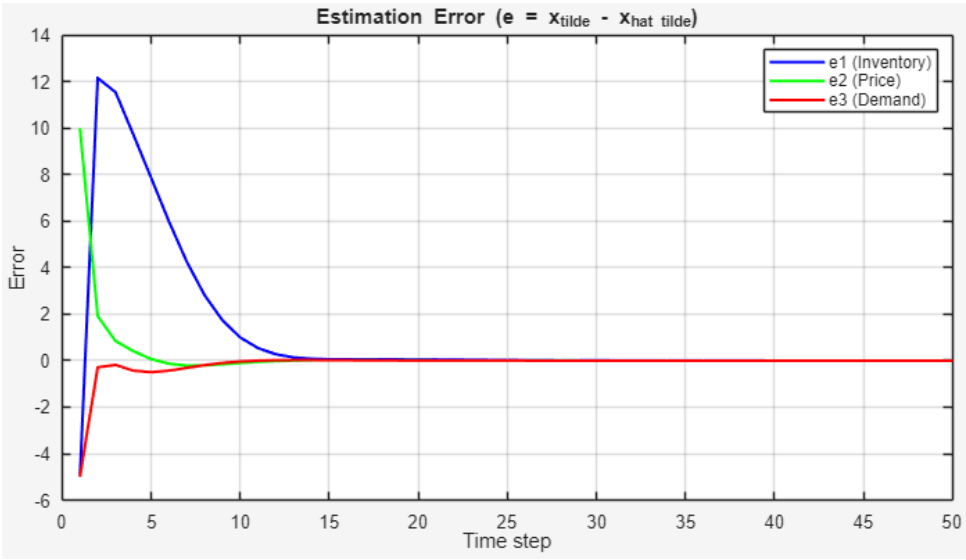

Figure 2: Graph of Estimation Error.

For the estimation error (see Figure 2), errors for all variables (Inventory Level, Product Price and Customer Demand) all converge to zero, meaning that the linear design is a near-perfect match of the true nonlinear system, within 15 time steps with negligible estimation error.

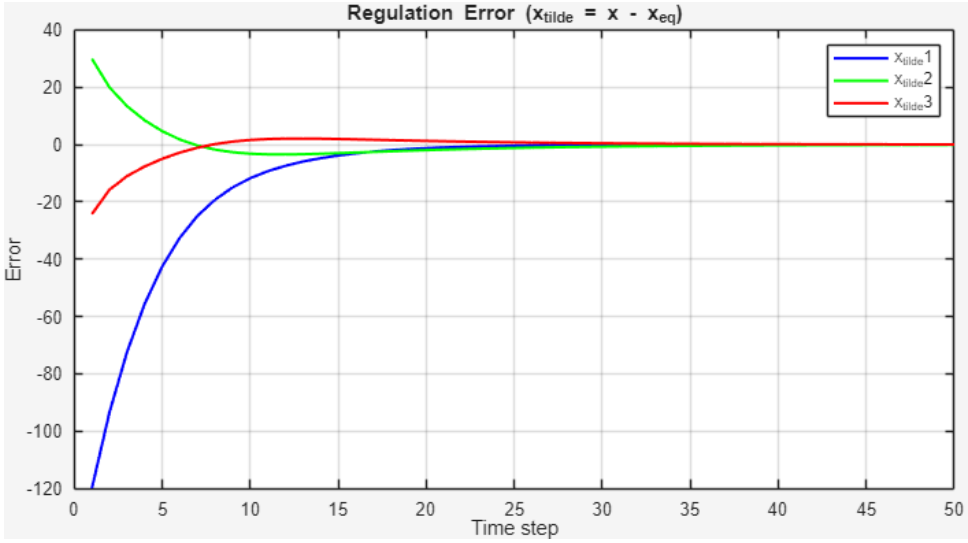

Figure 3: Graph of Regulation Error.

Crucially, due to our eigenvalue assignment, the regulation error takes longer (approximately 25 time steps) to converge to zero than the estimation error. This ensures that the estimation via the observer dynamics are accurate before the system begins to converge to our chosen equilibrium points.

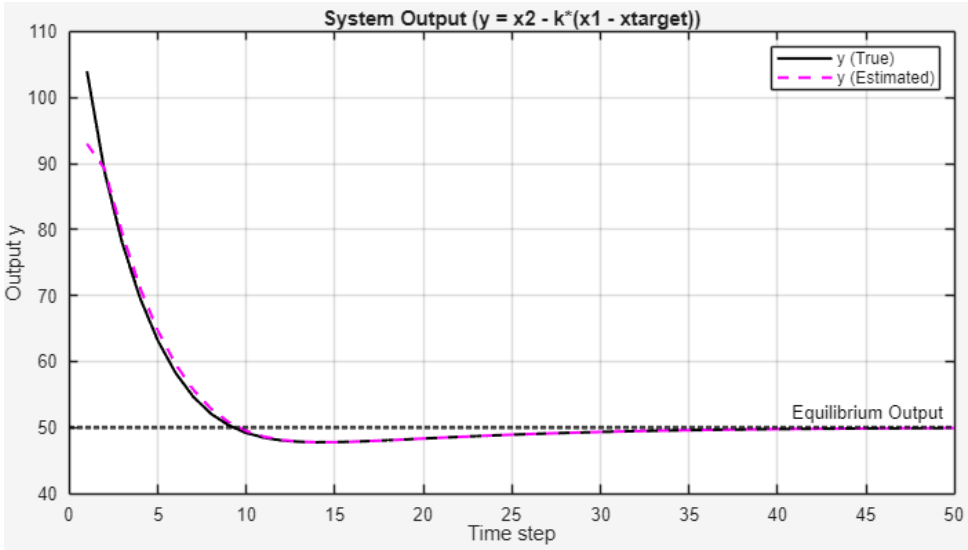

Figure 4: Graph of System Output.

The final plot validates our overall control strategy by examining the measured output, $y(t)$. The graph clearly shows that the true system output converges to the desired equilibrium point, which we correctly solved to be the target price, $\bar{P}$. Furthermore, the estimated output tracks the true output almost perfectly throughout the simulation. This further supports that the observer-based controller is accurately predicting the system behaviour, which in turn allowed it to generate reliable estimates for the system output.

7 Conclusion

This project provides several key insights into the control of dynamic retail systems. In turn, we also consider the assumptions and limitations inherent in the model.

7.1 Insights Gained

- **System stability and control feasibility:** The simulation results confirm that it is feasible to drive both price and inventory toward a desired equilibrium point using state-feedback control. This confirms that the system is controllable under the assumed parameters and model structure. Although customer demand is not directly controlled, it successfully converges to a steady value due to the nonlinear coupling between price and demand, supporting the economic dynamics of the supply-demand model.
- **Role of observer design:** Our observer-based controller successfully estimated the unmeasured states (e.g., inventory) and maintained control performance. This demonstrates the practical feasibility of using an observer when full state measurement is not available.
- **Impact of Parameter Sensitivity:** The system's behaviour is highly sensitive to parameters such as price elasticity (β). Small changes could potentially cause slower convergence, oscillations, or instability. In order for the model to remain applicable to real-world context, consistent parameter estimation and continuous model tuning is crucial to maintain stable and effective control.
- **Interdependence of states:** The model clearly shows that controlling one variable (e.g. price) has significant ripple effects on others. For example, lowering price increases demand, which depletes inventory faster and triggers restocking. This highlights the importance of coordinated control strategies in multi-variable systems to avoid unintended consequences.

7.2 Assumptions

- **Deterministic and noise-free environment:** We assumed no process or measurement noise. In reality, demand fluctuations, supply delays, and data errors would introduce uncertainty that could challenge the controller's performance.
- **Static consumer behaviour:** The demand function is assumed fixed over time, with constant parameters. Real-world consumer preferences are dynamic and may change due to seasonal trends, marketing campaigns, competitor actions, or macroeconomic conditions.
- **Instantaneous actions:** Our model does not account for restocking lead times or price adjustment delays, which would introduce dynamic lags and potentially require a more advanced predictive control strategy.
- **Local equilibrium analysis:** Our simulation analysis was conducted via linearisation around a single equilibrium point. The true system dynamics far from this point may exhibit very different behaviour, such as multiple equilibria or limit cycles.
- **No capacity constraints:** We assumed infinite storage capacity and production/replenishment capability. In practice, these physical limits would impose hard constraints on feasible control actions, such as the design for control input $u_2(t)$, representing inventory replenishment.

7.3 Limitations

- **Model simplicity vs. realism:** The 3-state nonlinear model captures core dynamics but omits important factors such as competitor behaviour, marketing effects, supply chain disruptions, and operational costs.
- **Controller robustness:** Our controller is tuned for specific parameter values. It may degrade significantly if the true system parameters vary or if subjected to large, unmodeled disturbances.
- **Observer accuracy:** The observer's performance relies on a perfect system model. Any structural model mismatch will lead to estimation errors, which in turn degrade controller performance.
- **Scalability:** The current model applies to a single product in isolation. A real-world retail environment would require a more complex formulation to manage multi-product interactions, shared resource dynamics, competition for customer attention, or correlated demand.

8 Proper citation of all sources

8.1 Sources and References for the Model

These sources support our model structure and parameter selection. In particular, they validate the use of exponential demand and replenishment feedback in practical supply chain settings.

1. Simchi-Levi, D., Kaminsky, P., & Simchi-Levi, E. (2007). *Designing and Managing the Supply Chain*. McGraw-Hill.

2. Axsäter, S. (2010). *Inventory and Supply Chain Management with Forecast Updates*. Springer.

3. Song, J.-S., & Zipkin, P. H. (1993). Dynamic Pricing and Inventory Control with Fixed Replenishment Lags. *Operations Research*, 41(2), 358-376.

A Appendix A: MATLAB Simulation Code

This appendix contains the MATLAB code used for simulating the inventory-price-demand control system with a Luenberger observer. The code is divided into numbered sections for easy reference in the main text. Furthermore, the original MATLAB code file will be submitted alongside this report.

A.1 Parameter and equilibrium values

```
% --- 1. Initialization ---
clear; clc; close all;

% --- 2. System Parameters and Equilibrium Point ---
% Model Parameters
alpha = 100.0;      % Max demand
beta = 0.01;        % Price sensitivity
c1 = 0.1;           % Price sensitivity to inventory
c2 = 0.5;           % Price sensitivity to demand
xtarget = 200.0;    % Target inventory level (CHANGE THIS TO MODIFY THE EQUILIBRIUM POINT)
P_star = 50.0;      % Target equilibrium price (CHANGE THIS TO MODIFY THE EQUILIBRIUM POINT)
v = alpha*exp(-beta*P_star); % Target demand (Dependent on target price, do not change)
k = 0.2;            % Output definition parameter

% Equilibrium Point (x_eq, u_eq, y_eq)
x1_star = xtarget;
x2_star = P_star;
x3_star = alpha * exp(-beta * P_star);
u1_star = -c2 * x3_star;
u2_star = x3_star;

x_eq = [x1_star; x2_star; x3_star]
u_eq = [u1_star; u2_star]
y_eq = x2_star
```

A.2 System Linearisation (A, B, C, D matrices)

```
% --- 3. Linearization and System Matrices
A = [1, -beta*x3_star, 0;
     -c1, 1, c2;
     0, -beta*x3_star, 0];
B = [0, 1; 1, 0; 0, 0];
C = [-k, 1, 0];
D = [0, 0];
```

A.3 Controllability and Observability Checks

```
% --- 4. Controllability and Observability Checks ---
Q = [B, A*B, A*A*B];
S = [C; C*A; C*A*A];
controllable = (rank(Q) == 3);
observable = (rank(S) == 3);
fprintf('System is controllable: %d (Rank of Q = %d)\n', controllable, rank(Q));
fprintf('System is observable: %d (Rank of S = %d)\n\n', observable, rank(S));
```

A.4 Controller and Observer Gain Design (K, L matrices)

```
% --- 5. Gain Design (Pole Placement) ---
controller_poles = [0.7, 0.8, 0.9]; % (CHANGE THIS TO MODIFY THE REGULATION SPEED)
K = place(A, B, controller_poles);
fprintf('State feedback gain K:\n'); disp(K);

observer_poles = [0.3, 0.4, 0.5]; % (CHANGE THIS TO MODIFY THE ESTIMATION SPEED)
```

```
L = place(A', C', observer_poles)';
fprintf('Observer gain L:\n'); disp(L);
```

A.5 Closed-Loop Simulation with Observer

```
%--- 6. Closed-Loop Simulation
T = 50; % Simulation length (CHANGE THIS TO MODIFY THE SIMULATION LENGTH)
% State and Output History
x_true = zeros(3, T);
x_hat_tilde = zeros(3, T);
y_true = zeros(1, T);
% Error History for Plotting
estimation_error = zeros(3, T);
regulation_error = zeros(3, T);

% Initial Conditions
x_true(:, 1) = [x1_star*0.4; x2_star*1.6; x3_star*0.6];
x_hat_tilde(:, 1) = (x_true(:, 1) - x_eq) + [5; -10; 5];

% Simulation Loop
for t = 1:(T-1)
    % Step 1: Controller
    u = u_eq - K*x_hat_tilde(:, t);

    % Step 2: System Update
    x_t = x_true(:, t);
    demand = alpha* exp(-beta*x_t(2));
    x1_next = x_t(1) + u(2) - demand;
    x2_next = x_t(2) + u(1) - c1*(x_t(1) - xtarget) + c2*x_t(3);
    x3_next = demand;
    x_true(:, t + 1) = [x1_next; x2_next; x3_next];

    % Step 3: Measurement
    y_true(t) = x_t(2) - k*(x_t(1) - xtarget);

    % Step 4: Observer Update
    y_tilde = y_true(t) - y_eq;
    u_tilde = u - u_eq;
    y_hat_tilde = C*x_hat_tilde(:, t) + D*u_tilde;
    x_hat_tilde(:, t+1) = A*x_hat_tilde(:, t) + B*u_tilde + L*(y_tilde - y_hat_tilde);

    % Step 5: Logging
    x_tilde_true = x_t - x_eq;
    regulation_error(:, t) = x_tilde_true;
    estimation_error(:, t) = x_tilde_true - x_hat_tilde(:, t);
end

% --- Calculate final values at t=T for the plot ---
t = T;
x_t = x_true(:, t);
y_true(t) = x_t(2) - k*(x_t(1) - xtarget);
x_tilde_true = x_t - x_eq;
regulation_error(:, t) = x_tilde_true;
estimation_error(:, t) = x_tilde_true - x_hat_tilde(:, t);
```

A.6 Plotting Results

```
% --- 7. Plotting Results
figure('Position', [100, 100, 1200, 800]);
x_hat_full = x_hat_tilde + x_eq;

% Plot 1: State Convergence
subplot(2, 2, 1);
plot(1:T, x_true(1, :), 'b-', 'LineWidth', 1.5); hold on;
plot(1:T, x_hat_full(1, :), 'b--', 'LineWidth', 1.5);
plot(1:T, x_true(2, :), 'g-', 'LineWidth', 1.5);
plot(1:T, x_hat_full(2, :), 'g--', 'LineWidth', 1.5);
plot(1:T, x_true(3, :), 'r-', 'LineWidth', 1.5);
plot(1:T, x_hat_full(3, :), 'r--', 'LineWidth', 1.5);
```

```
yline(xtarget, 'k:', 'LineWidth', 2, 'Label', 'Target Inventory');
yline(P_star, 'k:', 'LineWidth', 2, 'Label', 'Target Price');
yline(v, 'k:', 'LineWidth', 2, 'Label', 'Target Demand');
title('State-Feedback Control with Luenberger Observer');
xlabel('Time step'); ylabel('State Values');
legend('x1 True', 'x1 Est.', 'x2 True', 'x2 Est.', 'x3 True', 'x3 Est.');
grid on; hold off;

% Plot 2: Estimation Error
subplot(2, 2, 2);
plot(1:T, estimation_error(1, :), 'b-', 'LineWidth', 1.5); hold on;
plot(1:T, estimation_error(2, :), 'g-', 'LineWidth', 1.5);
plot(1:T, estimation_error(3, :), 'r-', 'LineWidth', 1.5);
title('Estimation Error (e = x_{tilde} - x_{hat tilde})');
xlabel('Time step'); ylabel('Error');
legend('e1 (Inventory)', 'e2 (Price)', 'e3 (Demand)');
grid on; hold off;

% Plot 3: Regulation Error
subplot(2, 2, 3);
plot(1:T, regulation_error(1, :), 'b-', 'LineWidth', 1.5); hold on;
plot(1:T, regulation_error(2, :), 'g-', 'LineWidth', 1.5);
plot(1:T, regulation_error(3, :), 'r-', 'LineWidth', 1.5);
title('Regulation Error (x_{tilde} = x - x_{eq})');
xlabel('Time step'); ylabel('Error');
legend('x_{tilde}1', 'x_{tilde}2', 'x_{tilde}3');
grid on; hold off;

% For plotting, calculate the estimated output
y_hat = x_hat_full(2, :) - k * (x_hat_full(1, :) - xtarget);

% Plot 4: System Output
subplot(2, 2, 4);
plot(1:T, y_true, 'k-', 'LineWidth', 1.5); hold on;
plot(1:T, y_hat, 'm--', 'LineWidth', 1.5);
yline(y_eq, 'k:', 'LineWidth', 2, 'Label', 'Equilibrium Output');
title('System Output (y = x2 - k*(x1 - xtarget))');
xlabel('Time step'); ylabel('Output y');
legend('y (True)', 'y (Estimated)');
grid on; hold off;
```