서 울 대 학 교
SEOUL NATIONAL UNIVERSITY

1. State Feedback, Tracking and Robust Controller

Consider a CT-LTI System:

$$\dot{x} = \begin{bmatrix} 1 & -2 & 1 \\ 1 & 1 & 3 \\ -1 & 4 & 0 \end{bmatrix} x + \begin{bmatrix} 1 \\ -1 \\ 0 \end{bmatrix} u$$

$$y = \begin{bmatrix} 1 & 0 & 2 \end{bmatrix} x$$

(1) Design state feedback controller and state observer

a) Design the state feedback system that have $-5, -5+3j, -5-3j$ as its eigenvalues.

To design a state feedback system with desired eigenvalues, two distinct approaches are commonly employed: one involving the utilization of the "place" function and another without it. The process begins by establishing the state space equation, enabling the computation of eigenvalues and the characteristic polynomial equation for the original system.

By formulating the system's dynamics using matrices A, B, C, and D, the eigenvalues can be determined from the A matrix, providing valuable insights into the system's stability characteristics. Additionally, the characteristic polynomial equation is derived from these eigenvalues, offering a mathematical representation of the system's behavior.

```
%Define State Space Equation System
A = [1 -2 1; 1  1 3; -1  4 0];
B = [1; -1; 0];
C = [1 0 2];
D = 0;
sys = ss(A, B, C, D);
eigenvals = eig(A);
char_poly = poly(eigenvals);
```

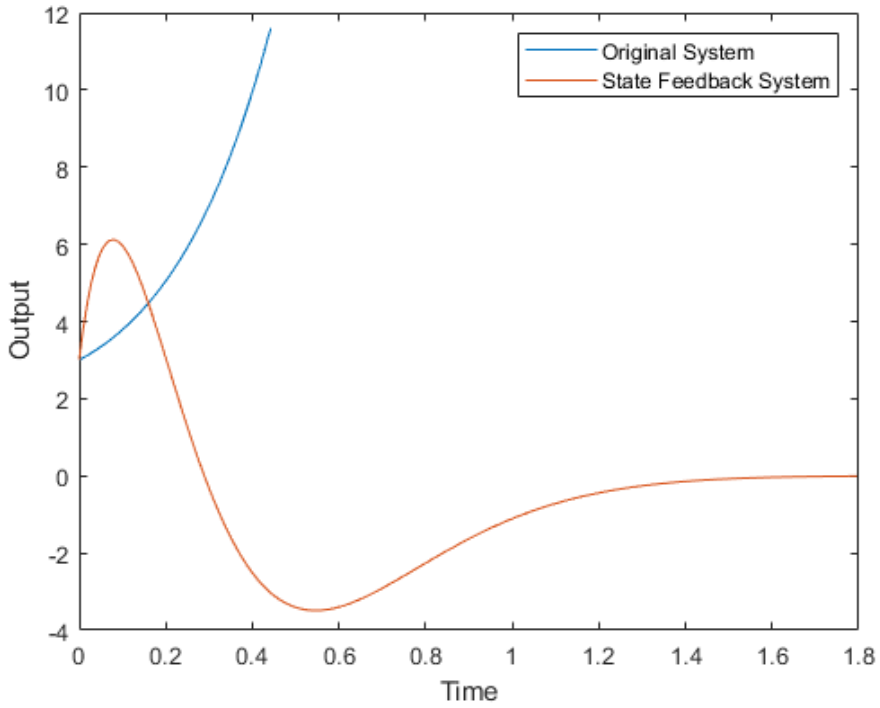
In this section, the computed eigenvalues of the system are identified as -2.0791, 0.1215, and 3.9576, leading to a characteristic polynomial equation of $1 * s^3 - 2 * s^2 - 8 * s + 1$. To design the appropriate gain K for the state feedback controller, it is necessary to compute the transformation matrix P, as mentioned in the previous section. Additionally, the transformed gain $\bar{K}$ is computed to transition from the original eigenvalues to the desired eigenvalues. Finally, the final gain K can be calculated using $\bar{K}$ and P. This section code:

```
% Compute State Feedback Gain K w/o 'place Funtion
C_ = [B A*B A*A*B];
C_bar = [1 -2 -8; 0 1 -2; 0 0 1];
P = inv(C_*C_bar);
K_bar = [17 92 169];
K = K_bar*P;
```

For compute state feedback gain K, MATLAB provide the function `place` to compute:

```
% Compute State Feedback Gain K w 'place Funtion
desired_poles=[-5 -5+3i -5-3i];
K_placed = place(A, B, desired_poles);
```

After performing the computational analysis, the determined state feedback gain, denoted as K, is found to be -17.5570, -34.5570, and -35.7342. This selected gain value effectively transforms the eigenvalues of the original system to the desired values, thereby achieving the desired system stability. The following result presents a comparative analysis of the step response between the original system and the state feedback controller system. The eigenvalues of the original system exhibit a positive portion, indicating instability. Conversely, the state feedback controller system successfully achieves stability by ensuring all eigenvalues reside in the negative domain, as evidenced by its stable step response.



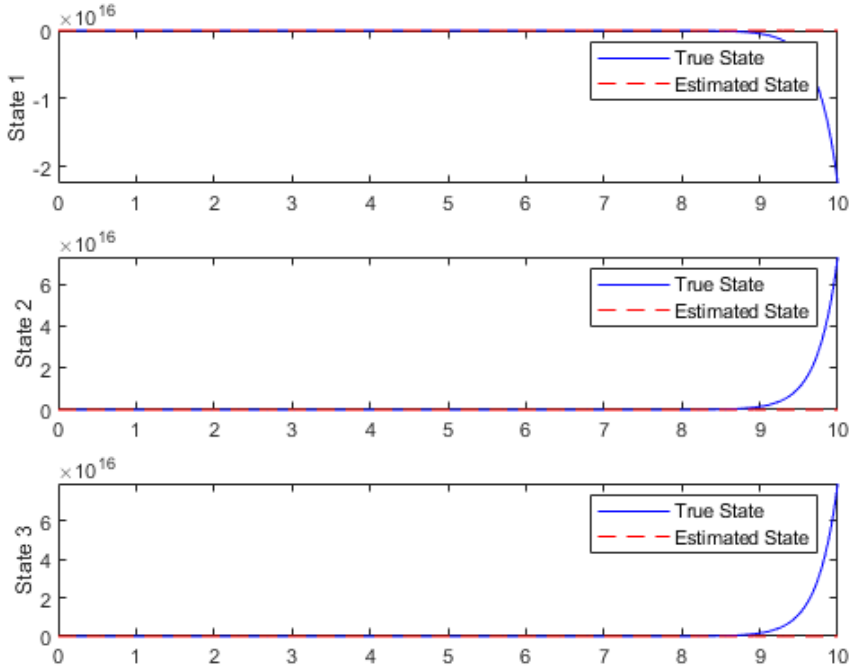
b) Design three state estimators: open-loop observer, two Luenberger observers that each have following sets of eigen values, $-1, -2, -3$ and $-0.01, -0.02, -0.03$. Plot the output responses from the initial condition $x(0) = [1 \quad 1 \quad 1]^T, \hat{x}(0) = [0 \quad 0 \quad 0]^T$ and discuss about them.

For the **open-loop observer**, this means that the observer using the same state space equation system to tracking the original system. Because there has some situations that the states of the system are very expansive or hard to measure. In textbook, it mentioned that there are two disadvantages in using an open-loop estimator. First, the initial state must be computed and set each time we use the estimator. This is very inconvenient and may be expansive to measure. Second, and more seriously, if the matrix A has eigenvalues with positive real parts (the system is unstable) then even for a very small difference between $x(t_0)$ and $\hat{x}(t_0)$ for some t_0, which may be caused by disturbance or imperfect estimation of the initial state, the difference between $x(t)$ and $\hat{x}(t)$ will grow with time.

```
% Mainly for the State Feedback and Track Controller
A = [1 -2 1; 1  1 3; -1  4 0];
B = [1; -1; 0];
C = [1 0 2];
D = 0;
desired_poles=[-5 -5+3i -5-3i];
K_placed = place(A, B, desired_poles);

% Define the initial conditions
x0 = [1; 1; 1];         % Initial state
x_estimate0 = [0; 0; 0];% Initial estimated state
u = 0;                  % Input

% Define the simulation parameters
dt = 0.01;              % Time step
t_end = 10;             % Simulation end time
```

```
% Simulate the system and observer dynamics
t = 0:dt:t_end;
x = zeros(3, length(t));
y = zeros(3, length(t));

x_estimate = zeros(3, length(t));
y_estimate = zeros(3, length(t));
x(:, 1) = x0;
x_estimate(:, 1) = x_estimate0;
for i = 2:length(t)
    % Update the state using the system equations
    x(:, i) = x(:, i-1) + dt * (A*x(:, i-1) + B*u);
    % Update the estimated state using the observer equations
    x_hat(:, i) = x_hat(:, i-1) + dt * (A*x_hat(:, i-1) + B*u);
end
```

As the result of this section code, the figure has shown three states of the original system and observer with different initial state. Because the matrix A of the original system has the real parts, and the small different from the initial states. The observer can not detect real states and track it successfully.



For the **closed-loop observer**, we shall modify the estimator making the output $y(t) = \boldsymbol{cx}(t)$ compared with $\boldsymbol{c\hat{x}}(t)$. And the difference passing through an $n \times 1$ constant gain vector L. And I compute the gain L with the similar method of gain K. As the condition, the eigen values set by the closed-loop observer.

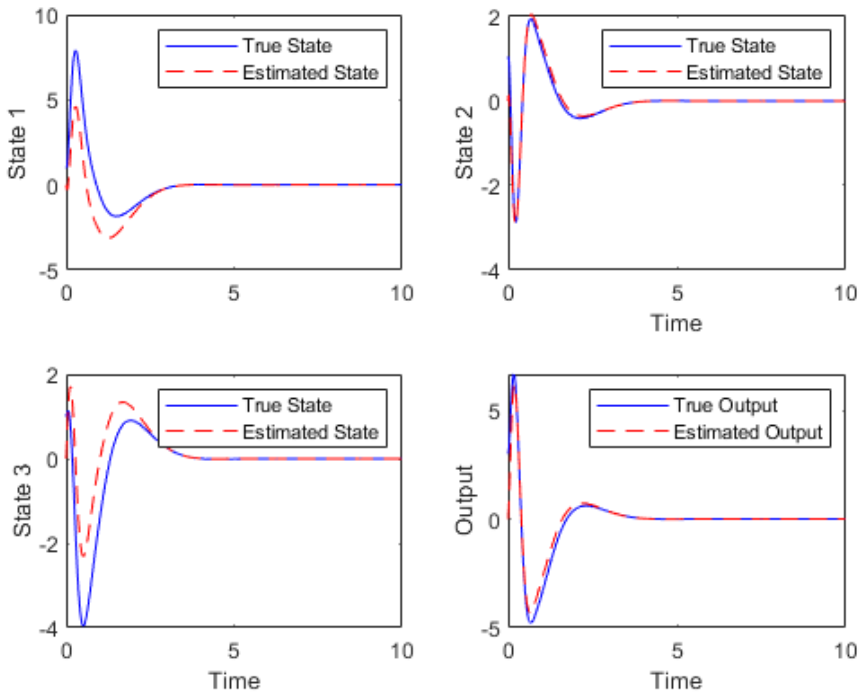
```
% Compute Observer Gain K w/o 'place Funtion
A = [1 -2 1; 1  1 3; -1  4 0];
B = [1; -1; 0];
C = [1 0 2];
D = 0;
% Compute State Feedback Gain L w/o place Function
C_ = [C' A'*C' A'*A'*C'];
C_bar = [1 -2 -8; 0 1 -2; 0 0 1];
P = inv(C_*C_bar);
L_bar = [8 19 5];
L = (L_bar*P)';
% Compute State Feedback Gain L and K w 'place Funtion
desired_poles = [-0.01 -0.02 -0.03];
L_placed = place(A', C', desired_poles)';
```

```
desired_poles=[-5 -5+3i -5-3i];
K_placed = place(A, B, desired_poles);
```

After check the L and L_{placed}, I use the gain and design the closed-loop observer. Then check the output response of different conditions.

```
% Define the initial conditions
x0 = [1; 1; 1];         % Initial state
x_estimate0 = [0; 0; 0];     % Initial estimated state
u = 0;                  % Input

% Define the simulation parameters
dt = 0.01;              % Time step
t_end = 10;             % Simulation end time

% Simulate the system and observer dynamics
t = 0:dt:t_end;
x = zeros(3, length(t));
y = zeros(1, length(t));
x_estimate = zeros(3, length(t));
y_estimate = zeros(1, length(t));

x(:, 1) = x0;
y(:, 1) = C*x0;
x_estimate(:, 1) = x_hat0;
y_estimate(:, 1) = C*x_estimate0;

for i = 2:length(t)
    u = -K_placed * x_hat(:, i-1);
    x(:, i) = x(:, i-1) + dt * (A*x(:, i-1) + B*u);
    y(i) = C*x(:, i);
    x_estimate(:, i) = x_hat(:, i-1) + dt * ((A - L*C)*x_hat(:, i-1) + B*u + L*y(i));
    y_estimate(i) = C*x_estimate(:, i);
end
```

Then the result has shown that the closed-loop observer can track the real states successfully and the output response is correct.



(2) Design the reference tracking controller so that $\lim_{t\to\infty} y(t) = 1$. Plot the output response from the initial condition $x(0) = [1 \quad 1 \quad 1]^T$.

Consider the reference tracking problem: $y(t) \to r(t)$. For compute the satisfy conditions:

$$0 = (AN_x + BN_u)r$$
$$r = (CN_x + DN_u)r$$

And for the computation,

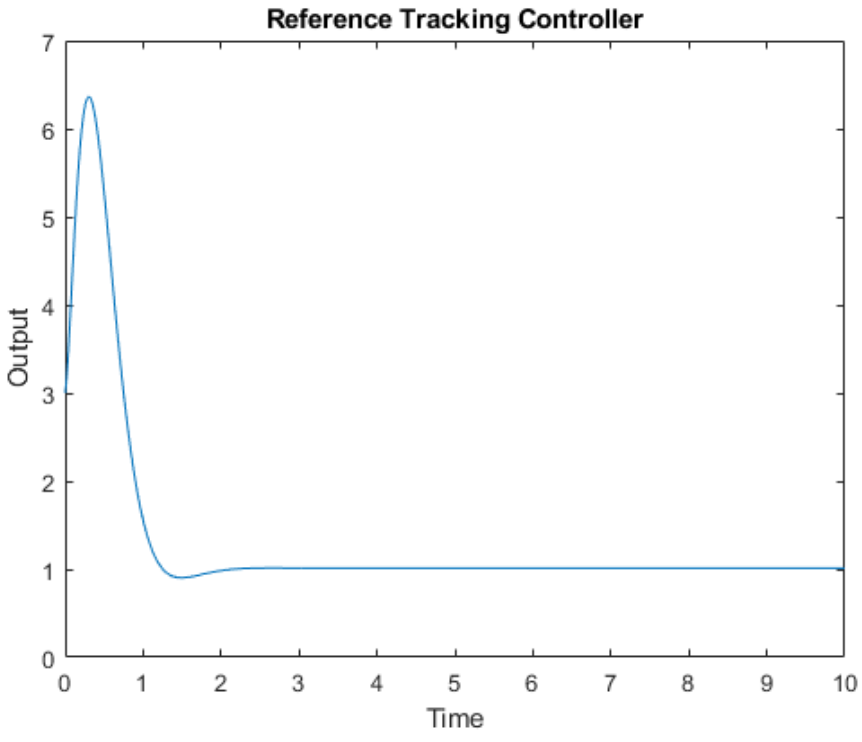
$$N_x = [8 \quad 2 \quad -3.5]; N_u = -0.5$$

Then for the reference controller, we can incorporate feedforward action into the feedback control, using the input

$$u = -K * x + (K * N_x + N_u) * r$$

From the result, the figure has shown the output response can successfully track the input reference signal.

```
dt = 0.01;              % Time step
t_end = 10;
K_Ref = [17 35 35];
R = 1;

t = 0:dt:t_end;
x = zeros(3, length(t));
y = zeros(1, length(t));

x(:, 1) = x0;
y(:, 1) = C*x0;
for i = 2:length(t)
    x(:, i) = x(:, i-1) + dt * (A*x(:, i-1) + B*(K_Ref*x(:, i-1)-85*R));
    y(:, i) = C*x(:, i);
end
```



(3) Design the state feedback controller and state observer for the system, and plot $x(t), \hat{x}(t)$.

Consider sinusoidal disturbance $d = \xi sin(\omega t)$ with unknown magnitude ξ and frequency ω:

$$\dot{x} = Ax + B(u + d)$$
$$y = Cx$$

where d is a disturbance, only know to satisfy differential equation noise model:

$$\dot{\xi} = A_d\xi,\ d = C_d\xi$$

And for the sinusoidal signal,

$$A_d = \begin{bmatrix} 0 & -\omega \\ \omega & 0 \end{bmatrix}, C_d = [1 \quad 0]$$

Thus the total(augmented) system dynamics is,

$$\begin{pmatrix} \dot{x} \\ \dot{\xi} \end{pmatrix} = \begin{bmatrix} A & BC_d \\ 0 & A_d \end{bmatrix} \begin{pmatrix} x \\ \xi \end{pmatrix} + \begin{bmatrix} B \\ 0 \end{bmatrix} u$$
$$y = [C \quad 0] \begin{pmatrix} x \\ \xi \end{pmatrix}$$

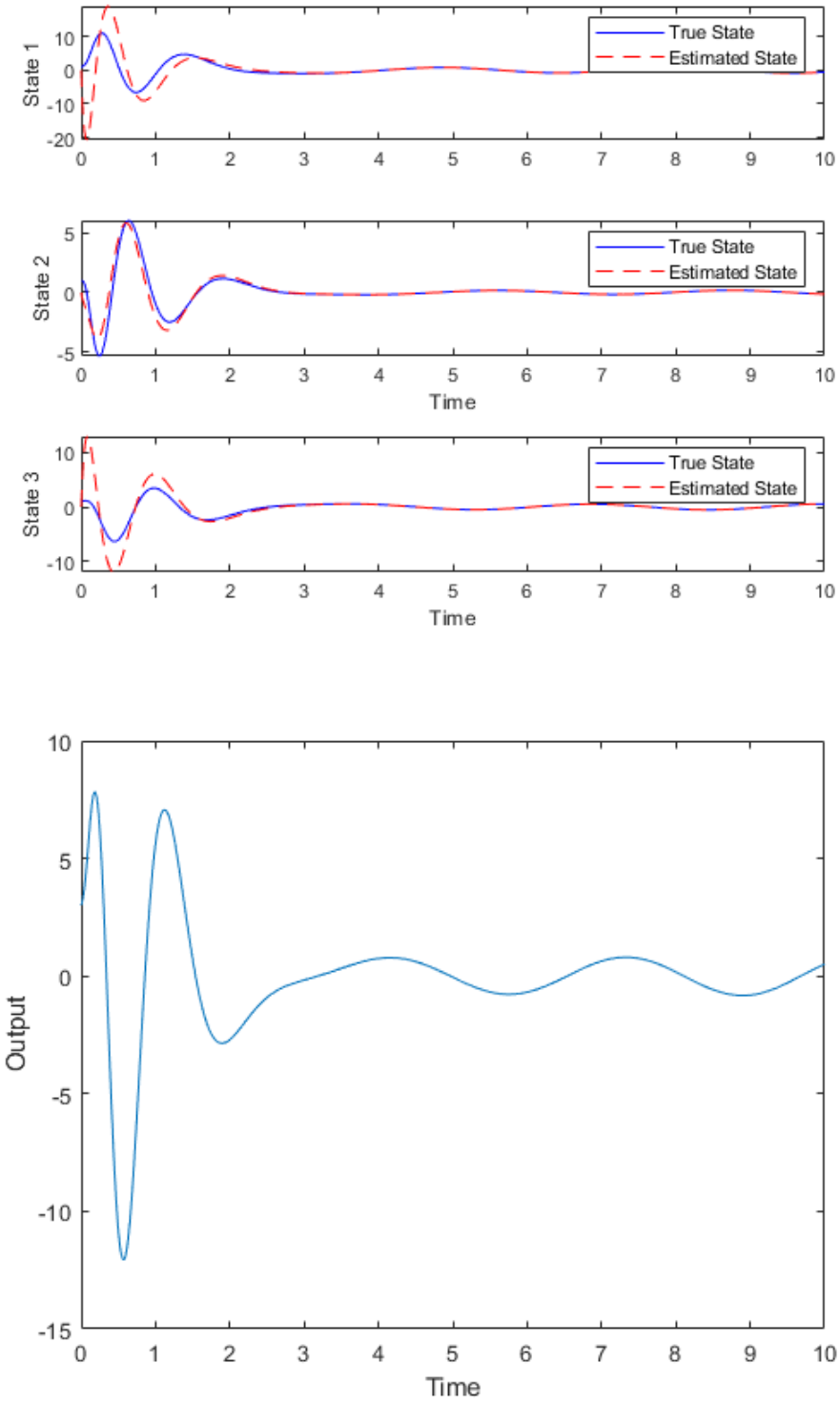
I design a state observer for this system is,

$$\begin{pmatrix} \dot{\hat{x}} \\ \dot{\hat{\xi}} \end{pmatrix} = \begin{bmatrix} A & BC_d \\ 0 & A_d \end{bmatrix} \begin{pmatrix} \hat{x} \\ \hat{\xi} \end{pmatrix} + \begin{bmatrix} B \\ 0 \end{bmatrix} u + \begin{bmatrix} L_1 \\ L_2 \end{bmatrix} (y - C\hat{x})$$

The code is,

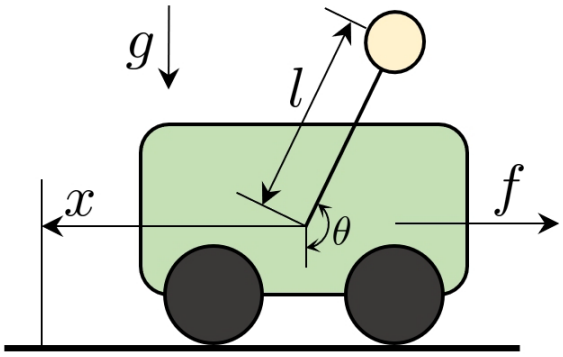
```
A = [1 -2 1; 1  1 3; -1  4 0];
B = [1; -1; 0];
C = [1 0 2];
D = 0;
Ad = [0, -2; 2, 0];
Cd = [1, 0];
A_aug = [A B*Cd; zeros(2,3) Ad];
B_aug = [B; zeros(2,1)];
C_aug = [C zeros(1,2)];
% Compute State Feedback Gain L and K w 'place Funtion
desired_poles = [-4 -4+0.5i -4-0.5i -4+2i -4-2i];
L_placed = place(A_aug', C_aug', desired_poles)';
desired_poles=[-5 -5+3i -5-3i];
K_placed = place(A, B, desired_poles);

R = 1;
dt = 0.01
t = 0:dt:10;
x = zeros(5, length(t));
x_estimate = zeros(5, length(t));
y = zeros(1, length(t));
y_estimate = zeros(1, length(t));

% Define the initial conditions
x(:, 1) = [1; 1; 1; 5; 5];
x_estimate(:,1) = [0; 0; 0; 5; 5];
y(:, 1) = C_aug*x(:, 1) ;
y_estimate(:,1) = C_aug*x_estimate(:,1);

for i = 2:length(t)
    u = -K_placed * x_estimate(1:3, i-1);
```

```
    x(:, i) = x(:, i-1) + dt * (A_aug*x(:,i-1)+ B_aug*u);
    y(:, i) = C_aug*x(:, i);
    x_estimate(:, i) = x_estimate(:, i-1) + dt * (A_aug*x_estimate(:, i-1) + B_aug*u +
L_placed*(y(i-1)-C_aug*x_estimate(:,i-1)));
end
```



2. Linear Quadratic Regulator (LQR)

Consider a following cart-pole system with mass of the cart and the particle on the massless pole m_c and m_p respectively.



The dynamics of the system can be written as:

$$H(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) = Bu$$

where $q := \begin{bmatrix} x & \theta \end{bmatrix}^T$, and the above matrices are:

$$H(q) = \begin{bmatrix} m_c + m_p & m_p l cos(\theta) \\ m_p l cos(\theta) & m_p l^2 \end{bmatrix},$$

$$C(q,\dot{(q)}) = \begin{bmatrix} 0 & -m_p l \dot{\theta} sin(\theta) \\ 0 & 0 \end{bmatrix},$$

$$G(q) = \begin{bmatrix} 0 \\ m_p g l sin(\theta) \end{bmatrix},$$

$$B = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

Let $m_c = m_p = l = 1$, $g = 10$ for gravity. Then the linearized dynamics can be achieved around the fixed point of interest with state $\begin{bmatrix} q & \dot{q} \end{bmatrix}^T$:

$$\bar{A} = \begin{bmatrix} 0 & I \\ -H^{-1}\frac{\partial G}{\partial q} & -H^{-1}C \end{bmatrix}\Bigg|_{x=\bar{x},u=\bar{u}}, \bar{B} = \begin{bmatrix} 0 \\ H^{-1}B \end{bmatrix}\Bigg|_{x=\bar{x},u=\bar{u}}$$

(1) Design a LQR controller to stabilize the system around the fixed point

Before design the LQR controller, I would like to describe some derivation of this inverted pendulum dynamics equations. Using the Lagrangian Method, $l = T - V$ where T is the kinematic energy and V is the potential energy of the system.

> $T = T_c + T_p$ where the first term is the kinematic energy of the cart and the second term is that of the pole.

As the Cartesian coordinate of the cart and pole that can be written as $\begin{bmatrix} x \\ 0 \end{bmatrix}$ and $\begin{bmatrix} x + L * sin(\theta) \\ L * cos(\theta) \end{bmatrix}$, the kinematic energy can be written as: $T_c = \frac{1}{2} M \dot{x}^2$ and $T_p = \frac{1}{2} m[(x + L * \dot{sin}(\theta))^2 + (L * \dot{cos}(\theta))^2]$.

The potential energy is $V = mgLcos(\theta)$, Thus the total energy is,

$$l = T - V = \frac{1}{2} M \dot{x}^2 + \frac{1}{2} m[(x + L * \dot{sin}(\theta))^2 + (L * \dot{cos}(\theta))^2] - mgLcos(\theta)$$

The generalized coordinates are selected as $\Omega = \begin{bmatrix} x \\ \theta \end{bmatrix}$, so the Lagrangian equations are:

$$\frac{d}{dt}\frac{\partial l}{\partial \dot{x}} - \frac{\partial l}{\partial x} = f$$

$$\frac{d}{dt}\frac{\partial l}{\partial \dot{\theta}} - \frac{\partial l}{\partial \theta} = 0$$

This yields,

$$(M+m)\ddot{x} + mL\ddot{\theta}cos(\theta) - mL\dot{\theta}^2 sin(\theta) = f$$
$$mL\ddot{x}cos(\theta) + mL^2\ddot{\theta} - mgLsin(\theta) = 0$$

Then define the system state with $X = \begin{bmatrix} x \\ \dot{x} \\ \theta \\ \dot{\theta} \end{bmatrix}$, and from the above equation,

$$\dot{X} = \begin{bmatrix} \dot{x} \\ \ddot{x} \\ \dot{\theta} \\ \ddot{\theta} \end{bmatrix} = \begin{bmatrix} \dot{x} \\ \frac{f+msin(\theta)(L\dot{\theta}^2 - gcos(\theta))}{M+msin^2\theta} \\ \dot{\theta} \\ \frac{-fcos(\theta) - mL\dot{\theta}^2 sin(\theta)cos(\theta) + (M+m)gsin(\theta)}{L(M+msin^2(\theta))} \end{bmatrix}$$

And the system equation need to be linearized at upward position, the equilibrium is $X = \begin{bmatrix} x \\ \dot{x} \\ \theta \\ \dot{\theta} \end{bmatrix} \approx \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$

I consider the external force as the input u of the system, and the system can be linearized by ignoring the high order terms. Thus the linearized system is,

$$\dot{X} = \begin{bmatrix} \dot{x} \\ \ddot{x} \\ \dot{\theta} \\ \ddot{\theta} \end{bmatrix} = \begin{bmatrix} \dot{x} \\ \frac{f+msin(\theta)(L\dot{\theta}^2 - gcos(\theta))}{M+msin^2\theta} \\ \dot{\theta} \\ \frac{-fcos(\theta) - mL\dot{\theta}^2 sin(\theta)cos(\theta) + (M+m)gsin(\theta)}{L(M+msin^2(\theta))} \end{bmatrix} = \begin{bmatrix} \dot{x} \\ \frac{f-m\theta g}{M} \\ \dot{\theta} \\ \frac{-f+(M+m)g\theta}{LM} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & -\frac{mg}{M} & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & \frac{(M+m)g}{LM} & 0 \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \\ \theta \\ \dot{\theta} \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{M} \\ 0 \\ -\frac{1}{LM} \end{bmatrix} f$$

$$y = \begin{bmatrix} x \\ \theta \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \\ \theta \\ \dot{\theta} \end{bmatrix}$$

To design the Linear Quadratic Regulator, it utilizes a cost function: $J = \int_0^\infty (x^T Q x + u^T R u) dt$, and to find the optimal control low which means that using feedback gain K can minimize the cost function. In my situation, I choose the matrix Q with identity matrix and R is 1.

(2) Show your result using MATLAB from the initial condition $q = [0 \quad 0.01]^T$, $\dot{q} = [0 \quad 0]^T$ with $Q = I$, $R = 1$.

Start with system dynamic equation,

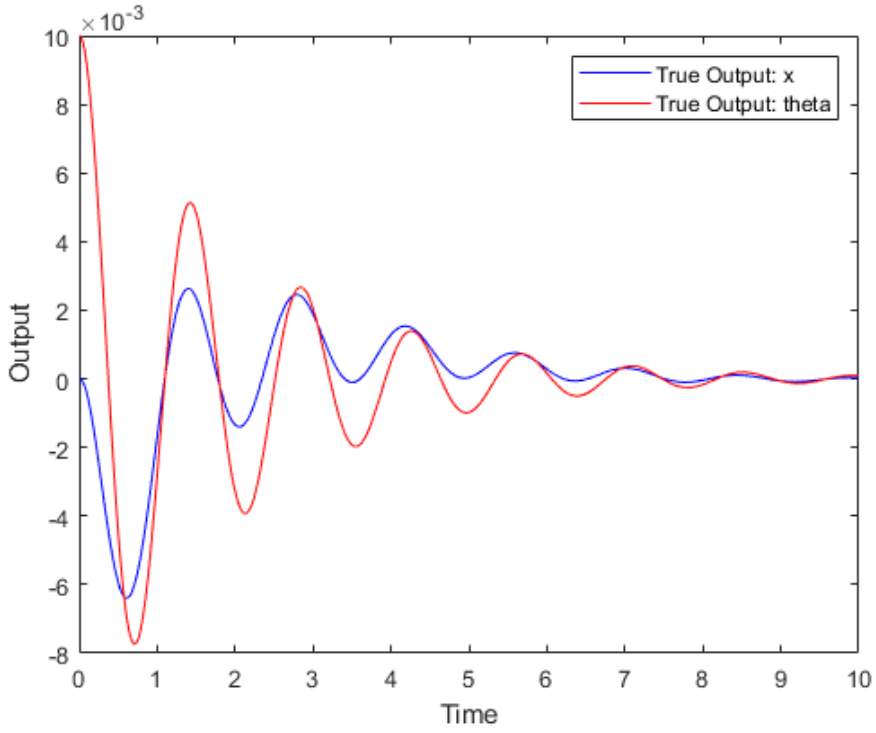
```
%% Parameter Definition
mc = 1;
mp = 1;
l = 1;
g = 10;
%% System Dynamics Equation
A = [0 0 1 0; 0 0 0 1; 0 -mp*g/mc 0 0; 0 -(mc+mp)*g/(l*mc) 0 0];
B = [0; 0; 1/mc; 1/(l*mc)];
C = [1 0 0 0; 0 1 0 0];
D = 0;
```

Then compute state feedback gain K using `lqr` function,

```
Q = diag([1,1,1,1]);
R = 1;
K = lqr(A, B, Q, R);
% Define the initial conditions
x0 = [0; 0.01; 0; 0];   % Initial state
u = 0;                  % Initial Input
```

Then we can construct the response and plot it.

```
% Simulate the system and observer dynamics
t = 0:dt:t_end;
x = zeros(4, length(t));
y = zeros(2, length(t));

x(:, 1) = x0;
y(:, 1) = C*x0;


for i = 2:length(t)
    u = -K * x(:, i-1);
    x(:, i) = x(:, i-1) + dt * (A*x(:, i-1) + B*u);
    y(:, i) = C*x(:, i);
end

plot_ = true;
if plot_
    figure;
    % subplot(2, 1, 1);
    % plot(t, x(1, :), 'b', t, x(2,:), 'r');
    % ylabel('State');
    % legend('True State: x', 'True Output: theta');

    % subplot(2, 1, 2);
    plot(t, y(1,:), 'b', t, y(2,:), 'r');
    xlabel('Time');
    ylabel('Output');
    legend('True Output: x', 'True Output: theta');
end
```



3. Kalman Filter

Consider the following linear system with process noise and measurement noise

$$x(k+1) = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix} x(k) + \begin{bmatrix} 0.3 \\ 0 \\ -0.3 \end{bmatrix} u(k) + w(k)$$
$$y = [1.9 \quad 1.3 \quad 1]x(k) + v(k)$$

where the initial state $x(1) = [0 \quad 0 \quad 0]^T$, and the noise are both zero mean white Gaussian noise

$$w \in N\left(0, \begin{bmatrix} 0.004 & 0.002 & 0.001 \\ 0.002 & 0.004 & 0.000 \\ 0.001 & 0.000 & 0.001 \end{bmatrix}\right), v \in N(0, 0.7)$$

Conduct Kalman Filter to estimate the state under a control sequence as follows,

$$u(k) = \begin{cases} 1 & if\ k \leq 100 \\ -1 & else \end{cases}$$

plot the three components of state and their estimations by function of time. *Write code from scratch (do not use MATLAB function* `kalman`*).*

For solving this problem, we need to define the discrete-time linear system with above information at first, and initialize some states and estimation states. Then the process noise and measurement noise need to be generated by Gaussian noise, including 2-dimensional noise and 1-dimensional noise.

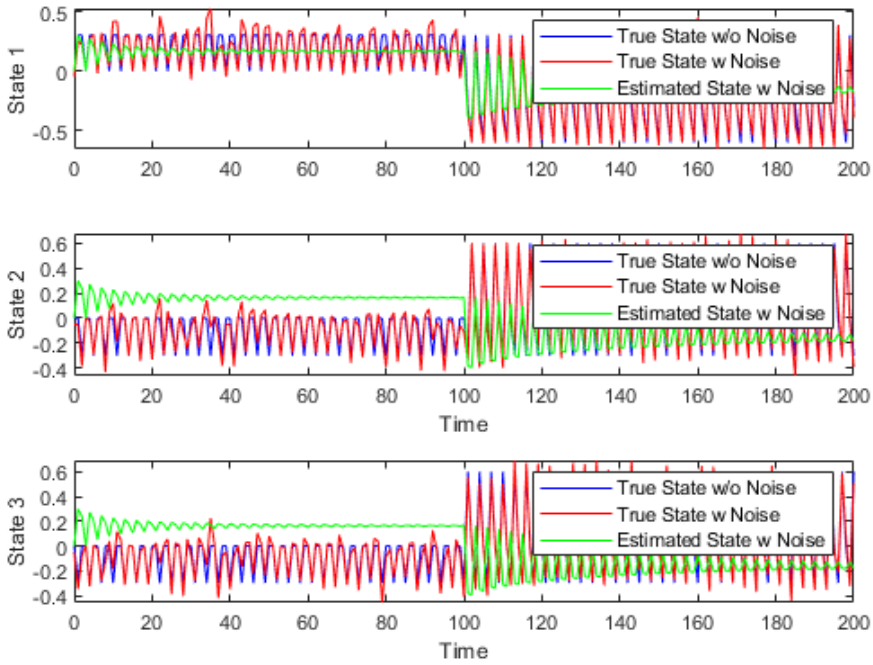
```
%% Parameter Definition
A = [0 1 0; 0 0 1; 1 0 0];
B = [0.3; 0; -0.3];
C = [1.9 1.3 1];
D = 0;
% Define the initial conditions
x0 = [0; 0; 0];         % Initial state
u = 0;                  % Input

% Define the simulation parameters
```

```
dt = 1;                  % 1：Discrete Time
t_end = 200;             % Simulation end time

% Simulate the system and observer dynamics
t = 0:dt:t_end;
x = zeros(3, length(t));
y = zeros(1, length(t));
x_true = zeros(3, length(t));
y_true = zeros(1, length(t));
x_estimate = zeros(3, length(t));
y_estimate = zeros(1, length(t));
% Parameters
meanValue = 0;  % Mean value
stdDeviation = 0.7;  % Standard deviation
measurementGaussianNoise = meanValue + stdDeviation * randn(t_end+1, 1);
meanVector = [0; 0; 0];
covarianceMatrix = [0.004 0.002 0.001; 0.002 0.004 0.000; 0.001 0.000 0.001];
processGaussianNoise = meanVector + chol(covarianceMatrix)' * randn(3, t_end+1);

x(:, 1) = x0;
x_true(:, 1) = x0 + processGaussianNoise(1);
y(:, 1) = C*x0;
y_true(:, 1) = C*x0 + measurementGaussianNoise(1);
x_estimate(:, 1) = [0;0;0];
y_estimate(:, 1) = C*x_estimate(:, 1);
```

Then, Kalman Filter need the gain K and matrix P to state feedback and update matrix P in one step.

```
Q = covarianceMatrix; % Process Noise Covariance
R = stdDeviation; % Measurement Noise Covariance
P = eye(3);

for i = 1:length(t)-1
    if(i<=100)
        u = 1;
    else
        u = -1;
    end

    % Predict the next state
    x_predicted = A*x_estimate(:,i) + B*u;
    P_predicted = A*P*A'+Q;
    % Update the state estimate based on the measurement
    K = P_predicted * C' / (C*P_predicted*C' + R);
    x_estimate(:, i+1) = x_predicted + K * (y_true(:, i) - C * x_predicted);
    P = (eye(3) - K * C) * P_predicted;
    % True state update
    x(:, i+1) = (A*x(:, i) + B*u);
    x_true(:, i+1) = (A*x(:, i) + B*u + processGaussianNoise(i));
    % Simulate measurement
    y(:, i) = C*x(:, i);
    y_true(:, i) = C*x(:, i) + measurementGaussianNoise(i);
    % Store the estimated state
end
```

Result:



MingshanHe

淡泊名利，矢志不渝；

上下求索，苦心孤诣；

百折不挠，夜以继日；

宁静致远，知行合一；