ME599 Homework 5 - Akshat Dubey

Problem 1

We have an aerial vehicle with the state $\boldsymbol{x}$ and control input u:

$$\boldsymbol{x} = \begin{bmatrix} x \\ y \\ z \\ \dot{x} \\ \dot{y} \\ \dot{z} \\ \phi \\ \theta \\ \psi \\ \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \in \mathbb{R}^{12}, \quad u = \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} \in \mathbb{R}^4 \quad n = 12, \quad m = 4$$

The dynamics are given by the function $F =$ `aerialVehSim.p` in discrete time for a given time step Ts:

$$\boldsymbol{x}_{k+1} = F(\boldsymbol{x}_k, u_k, Ts)$$

Problem 1.a

We want to design a discrete-time LQR controller for the system that drives the state to 0, with the cost function

$$J = \sum_{k=0}^{\infty} \left(\boldsymbol{x}_k^T Q \boldsymbol{x}_k + u_k^T R u_k \right)$$

where

$$\begin{aligned} Q &= 10 I_{12} \\ R &= I_4 \\ \boldsymbol{x}_{x+1} &= F(\boldsymbol{x}_k, u_k, Ts) \\ Ts &= 0.1 \\ \boldsymbol{x}_0 &= \begin{bmatrix} 1 \\ -1 \\ 0 \\ 0.5 \\ -1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \end{aligned}$$

We note that the function F is obscured, and is thus we dont know the matrices A and B for the system where

$$\boldsymbol{x}_{k+1} = A\boldsymbol{x}_k + Bu_k$$

Hence, we shall use Reinforcement Learning based LQR for this problem.

First, we define the Value function V, which is the recursive optimal cost-to-go function under the optimal control policy K_{opt} such that $u_k = -K_{opt}\boldsymbol{x}_k$:

$$\begin{aligned}
V(\boldsymbol{x}_k) &= \boldsymbol{x}_k^T Q \boldsymbol{x}_k + u_k^T R u_k + V(\boldsymbol{x}_{k+1}) \\
&= \sum_{t=k}^{\infty} \left(\boldsymbol{x}_t^T Q \boldsymbol{x}_t + u_t^T R u_t\right) \text{ (value from time } k \text{ to } \infty) \\
&= \sum_{t=k}^{\infty} \left(\boldsymbol{x}_t^T Q \boldsymbol{x}_t + (-K_{opt}\boldsymbol{x}_t)^T R(-K_{opt}\boldsymbol{x}_t)\right) \text{ (substituting } u_t = -K_{opt}\boldsymbol{x}_t) \\
&= \sum_{t=k}^{\infty} \left(\boldsymbol{x}_t^T Q \boldsymbol{x}_t + \boldsymbol{x}_t^T K_{opt}^T R K_{opt} \boldsymbol{x}_t\right) \\
&= \sum_{t=k}^{\infty} \left(\boldsymbol{x}_t^T (Q + K_{opt}^T R K_{opt}) \boldsymbol{x}_t\right) \\
&= \sum_{t=k}^{\infty} \left(\boldsymbol{x}_t^T S \boldsymbol{x}_t\right) \text{ (where } S = Q + K_{opt}^T R K_{opt})
\end{aligned}$$

We know that $\boldsymbol{x}_{k+1} = (A - BK_{opt})\boldsymbol{x}_k$, so we can write the value function in terms of P, which is some combination of A, B, S and K_{opt},

$$V(\boldsymbol{x}_k) = \boldsymbol{x}_k^T P \boldsymbol{x}_k$$

Then, we construct the Q-function, which seeks to converge to the value function as we perform reinforcement learning. At timestep k, the Q-function is defined as the cost of being in the state $\boldsymbol{x}_k$ and taking action u_k, and then following the optimal policy $u_{k+1} = -K_{opt}\boldsymbol{x}_{k+1}$ from then onwards:

$$\mathcal{Q}(\boldsymbol{x}_k, u_k) = \boldsymbol{x}_k^T Q \boldsymbol{x}_k + u_k^T R u_k + V(\boldsymbol{x}_{k+1})$$

Using the dynamics and the definition of the value function, we can rewrite the Q function as:

$$\begin{aligned}
\mathcal{Q}(\boldsymbol{x}_k, u_k) &= \boldsymbol{x}_k^T Q \boldsymbol{x}_k + u_k^T R u_k + V(\boldsymbol{x}_{k+1}) \\
&= \boldsymbol{x}_k^T Q \boldsymbol{x}_k + u_k^T R u_k + \boldsymbol{x}_{k+1}^T P \boldsymbol{x}_{k+1} \\
&= \boldsymbol{x}_k^T Q \boldsymbol{x}_k + u_k^T R u_k + (F(\boldsymbol{x}_k, u_k, Ts))^T P(F(\boldsymbol{x}_k, u_k, Ts)) \\
&= \boldsymbol{x}_k^T Q \boldsymbol{x}_k + u_k^T R u_k + (A\boldsymbol{x}_k + Bu_k)^T P(A\boldsymbol{x}_k + Bu_k) \\
&= \boldsymbol{x}_k^T Q \boldsymbol{x}_k + u_k^T R u_k + (\boldsymbol{x}_k^T A^T P A \boldsymbol{x}_k + \boldsymbol{x}_k^T A^T P B u_k + u_k^T B^T P A \boldsymbol{x}_k + u_k^T B^T P B u_k) \\
&= \boldsymbol{x}_k^T (Q + A^T P A) \boldsymbol{x}_k + u_k^T (R + B^T P B) u_k + \boldsymbol{x}_k^T A^T P B u_k + u_k^T B^T P A \boldsymbol{x}_k
\end{aligned}$$

Defining

$$\boldsymbol{z}_k = \begin{bmatrix} \boldsymbol{x}_k \\ u_k \end{bmatrix}, \quad \Theta = \begin{bmatrix} Q + A^T P A & A^T P B \\ B^T P A & R + B^T P B \end{bmatrix}$$

Then,

$$\mathcal{Q}(\boldsymbol{x}_k, u_k) = \boldsymbol{z}_k^T \Theta \boldsymbol{z}_k$$

And the optimal policy can be obtained by minimizing over the converged Q function:

$$\pi^*(\boldsymbol{x}_k) = \arg\min_{u_k} \mathcal{Q}(\boldsymbol{x}_k, u_k)$$

Which can be found by setting the derivative of $\mathcal{Q}$ with respect to u_k to zero:

$$\begin{aligned}
\frac{\partial}{\partial u_k}\left(\boldsymbol{x}_k^T(Q+A^TPA)\boldsymbol{x}_k + u_k^T(R+B^TPB)u_k + \boldsymbol{x}_k^TA^TPBu_k + u_k^TB^TPA\boldsymbol{x}_k\right) &= 0 \\
\frac{\partial}{\partial u_k}\left(u_k^T(R+B^TPB)u_k + \boldsymbol{x}_k^TA^TPBu_k + u_k^TB^TPA\boldsymbol{x}_k\right) &= 0 \\
\frac{\partial}{\partial u_k}\left(u_k^T(R+B^TPB)u_k + 2u_k^TB^TPA\boldsymbol{x}_k\right) &= 0 \\
2(R+B^TPB)u_k + 2B^TPA\boldsymbol{x}_k &= 0 \\
u_k &= -(R+B^TPB)^{-1}(B^TPA\boldsymbol{x}_k)
\end{aligned}$$

Which gives us the optimal policy:

$$\begin{aligned}
\pi^*(\boldsymbol{x}_k) &= -(R+B^TPB)^{-1}(B^TPA\boldsymbol{x}_k) \\
\implies u_k^* &= -K_{opt}\boldsymbol{x}_k \\
\text{where } K_{opt} &= (R+B^TPB)^{-1}(B^TPA)
\end{aligned}$$

Note that K_{opt} is a function of the last row of Θ, so as long as we can converge to an estimate Θ we can find K_{opt}. We don't need to find the individual matrices A and B!

$$\Theta = \begin{bmatrix} \Theta_{xx} & \Theta_{xu} \\ \Theta_{ux} & \Theta_{uu} \end{bmatrix}, \text{then, } K_{opt} = \Theta_{uu}^{-1}\Theta_{ux}$$

Initially, P will contain some $K \neq K_{opt}$, but after many iterations, once the Q function has converged, the Q function with the optimal policy $u_k = -K_{opt}\boldsymbol{x}_k$ should also converge to the function V

$$\mathcal{Q}(\boldsymbol{x}_k, -K_{opt}\boldsymbol{x}_k) = V(\boldsymbol{x}_k)$$

Using the definition of the Q function after convergence,

$$\begin{aligned}
\mathcal{Q}(\boldsymbol{x}_k, u_k) &= \boldsymbol{x}_k^TQ\boldsymbol{x}_k + u_k^TRu_k + V(\boldsymbol{x}_{k+1}) \\
&= \boldsymbol{x}_k^TQ\boldsymbol{x}_k + u_k^TRu_k + \mathcal{Q}(\boldsymbol{x}_{k+1}, -K_{opt}\boldsymbol{x}_{k+1}) \\
\text{Let } \boldsymbol{z}^* &= \begin{bmatrix} \boldsymbol{x}_k \\ u_k^* \end{bmatrix} = \begin{bmatrix} \boldsymbol{x}_k \\ -K_{opt}\boldsymbol{x}_k \end{bmatrix} \text{ then,} \\
&= \boldsymbol{z}_k^T \begin{bmatrix} Q & 0 \\ 0 & R \end{bmatrix} \boldsymbol{z}_k + \boldsymbol{z}_{k+1}^{*T}\Theta\boldsymbol{z}_{k+1}^* \\
\text{defining } \bar{Q} &= \begin{bmatrix} Q & 0 \\ 0 & R \end{bmatrix} \\
&= \boldsymbol{z}_k^T\bar{Q}\boldsymbol{z}_k + \boldsymbol{z}_{k+1}^{*T}\Theta\boldsymbol{z}_{k+1}^*
\end{aligned}$$

Hence

$$\mathcal{Q}(\boldsymbol{x}_k, u_k) = \boldsymbol{z}_k^T\Theta\boldsymbol{z}_k = \boldsymbol{z}_k^T\bar{Q}\boldsymbol{z}_k + \boldsymbol{z}_{k+1}^{*T}\Theta\boldsymbol{z}_{k+1}^*$$

Now, we just need to find a way to iteratively update Θ and K so that the Q function converges to the value function and K converges to K_{opt}. We can do this by using an off-policy algorithm. First, we will need to generate some data using a sequence of control inputs $\{u_k\}_{k=0}^{N-1} = \{u_0 \ldots u_{N-1}\}$ where $N \geq (m+1)L - 1$. We generate this by starting at x_0, sequentially applying random control inputs and capture the next state. This sequence is said to be persistently exciting of order L if the Hankel matrix $H_L(\{u_k\}_{k=0}^{N-1})$ is has full row rank mL. Then Hankel matrix is defined as:

$$H_L(\{u_k\}_{k=0}^{N-1}) = \begin{bmatrix} u_0 & u_1 & \dots & u_{N-L} \\ u_1 & u_2 & \dots & u_{N-L+1} \\ \vdots & \vdots & \ddots & \vdots \\ u_{L-1} & u_L & \dots & u_{N-1} \end{bmatrix} \in \mathbb{R}^{mL\times(N-L+1)}$$

We need a sequence of order $n+1$, which means

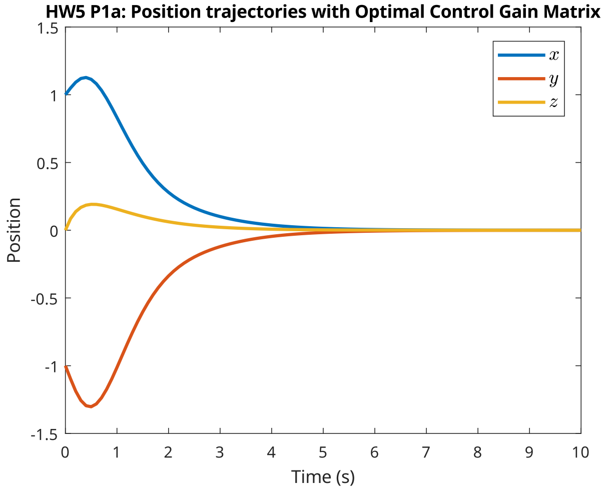
$$\begin{aligned} L &= n+1 = 12+1 = 13 \\ N &\geq (m+1)L - 1 \\ N &\geq (m+1)(n+1) - 1 \\ N &\geq m(n+1) + n \\ N &\geq 4(12+1) + 12 \\ \implies N &\geq 64 \\ \implies mL &= 4 \cdot 13 = 52 \end{aligned}$$

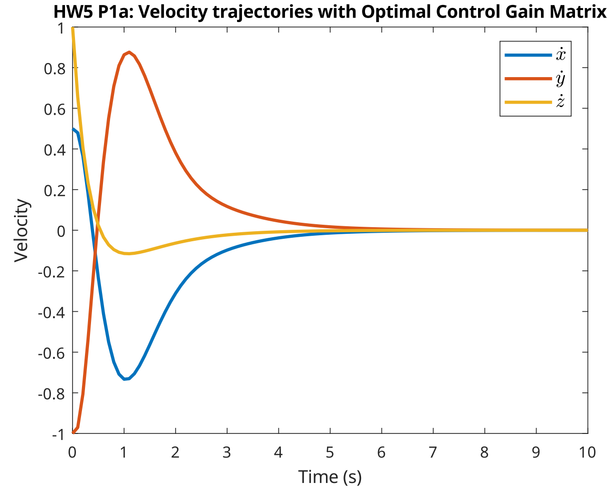
From the N total samples, we select $l = n+m = 16$ samples of $\boldsymbol{x}_{k_j}, u_{k_j}, \boldsymbol{x}_{k_j+1}$ such that the control input and state pairs $\boldsymbol{z}_{k_j} = \begin{bmatrix} \boldsymbol{x}_{k_j} \\ u_{k_j} \end{bmatrix}$,where $j \in \{1 \dots l\}$ are linearly independent.

Then:

1. Initialize $i = 0$, $K_i = 0$
2. Initialize $\boldsymbol{z}^*_{i,k_j+1} = \begin{bmatrix} \boldsymbol{x}_{k_j+1} \\ -K_i \boldsymbol{x}_{k_j+1} \end{bmatrix}$ for $j = 1 \dots l$.
3. Using all l samples, define the set of l equations
   - $\boldsymbol{z}_{k_j}^T \Theta_{i+i} \boldsymbol{z}_{k_j} = \boldsymbol{z}_{k_j}^T \bar{Q} \boldsymbol{z}_{k_j} + \boldsymbol{z}^{*T}_{i,k_j+1} \Theta_{i+1} \boldsymbol{z}^*_{i,k_j+1}$
   - Stack $\boldsymbol{z}_{k_j}$ and $\boldsymbol{z}^*_{i,k_j+1}$ columnwise into Z and Z^* where
     - $Z = \begin{bmatrix} \boldsymbol{z}_{k_1} & \boldsymbol{z}_{k_2} & \dots & \boldsymbol{z}_{k_l} \end{bmatrix}$
     - $Z^* = \begin{bmatrix} \boldsymbol{z}^*_{k_1+1} & \boldsymbol{z}^*_{k_2+1} & \dots & \boldsymbol{z}^*_{k_l+1} \end{bmatrix}$
   - Form the equation $Z^T \Theta_{i+1} Z = Z^T \bar{Q} Z + Z^{*T} \Theta_{i+1} Z^*$
     $\implies Z^{*T} \Theta_{i+1} Z^* - Z^T \Theta_{i+1} Z + Z^T \bar{Q} Z = 0$
4. Using MATLAB's `dlyap` function, we can solve the above equation for Θ_{i+1}
   - `dlyap`$(A, Q, [], E)$ solves the Lyapunov equation $AXA^T - EXE^T + Q = 0$ for X.
   - We have $A = Z^{*T}$, $Q = Z^T \bar{Q} Z$, $E = Z^T$ and $X = \Theta_{i+1}$.
   - Hence, we use `dlyap`$(Z^{*T}, Z^T \bar{Q} Z, [], Z^T)$ to solve for Θ_{i+1}.
   - Note: `dlyap`$(Z^T, -Z^T \bar{Q} Z, [], Z^{*T})$ is equivalent.
5. Update $K_{i+1} = \Theta^{-1}_{i+1,uu} \Theta_{i+1,ux}$.
6. If $||K_{i+1} - K_i|| < \epsilon$, then stop. Else, set $i = i+1$ and repeat from step 2.

After iteratively updating K and Θ, we can use the final K as K_{opt} and propagate the system in time using the dynamics F and the control input $u_k = -K_{opt} \boldsymbol{x}_k$.





Problem 1.b

In the previous part, the K_{opt} we converged to minimized the cost function such that the state $\boldsymbol{x}$ converged to 0. In this part, we want to minimize the cost function such that the state $\boldsymbol{x}$ converges to a desired state $\bar{\boldsymbol{x}}$. The x, y, z of desired state $\bar{\boldsymbol{x}}$ are given as:

$$x = 0.1sin\frac{t}{2}$$
$$y = 0.1cos\frac{t}{2}$$
$$z = 0.1t$$

$\bar{\boldsymbol{x}}_k$ at each timestep k can be computed by just taking the discrete timesteps that we are evaluating the cost function at.

The control penalty remains the same and state penalty matrix Q now only penalizes the position error, hence it can be written as:

$$Q = \begin{bmatrix} 10I_3 & 0 \\ 0 & 0_9 \end{bmatrix}$$

The goal is to follow a reference trajectory, so we now start from the first point in the reference trajectory

$$\boldsymbol{x}_0 = \begin{bmatrix} 0 \\ 0.1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

We can now use the same reinforcement learning based LQR algorithm as before, but with the updated Q. Once we have converged to K_{opt}, it will drive the positions to 0. To follow the reference trajectory, we can input the difference between the state and the reference trajectory so that $\boldsymbol{x}_k - \bar{\boldsymbol{x}}_k$ converges to 0. Hence, at each timestep k, we can compute the control input as:

$$u_k = -K_{opt}(\boldsymbol{x}_k - \bar{\boldsymbol{x}}_k)$$

With these changes, the reference trajectory is followed as shown below:

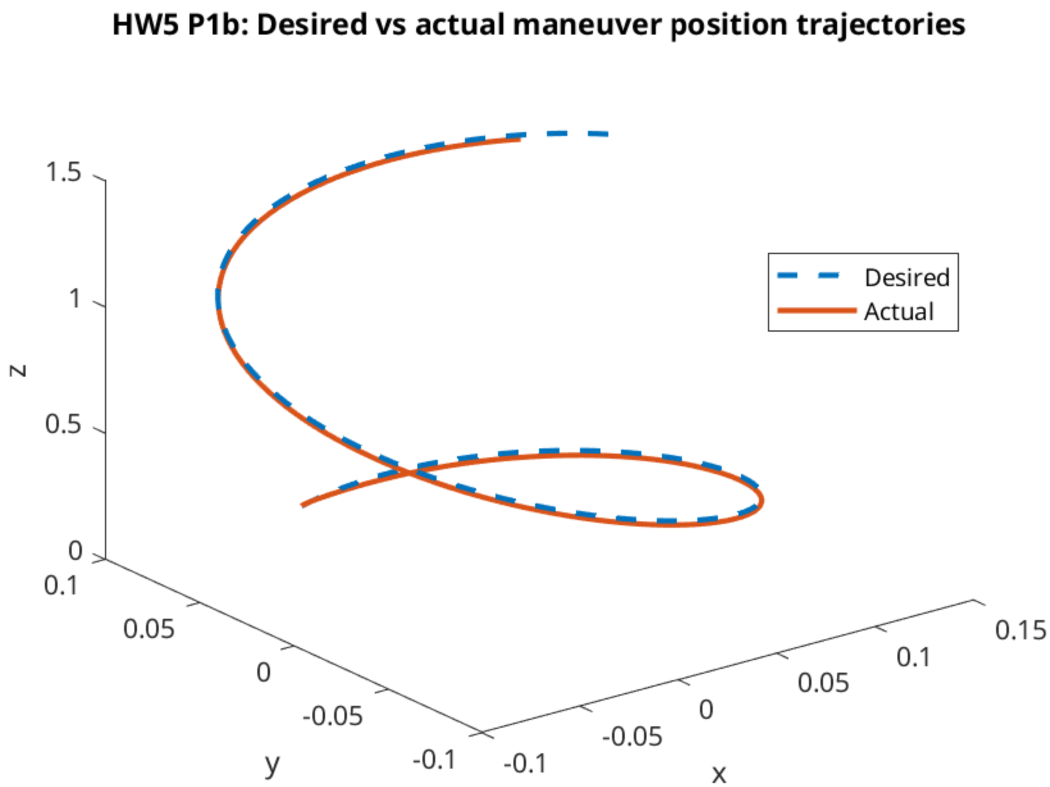


Figure 1: Desired and actual maneuver trajectory

Table of Contents

ME599 HW5 Problem 1

```
clc; clear; close all;

% setup
n = 12; % number of states
m = 4; % number of inputs
Ts = 0.1; % sample time

L = n+1;
extra = 0;
N = (m+1)*L-1 + extra; % total number of samples
mL = m*L; % required rank of Hankel matrix
R = eye(m);

% generate us and Hankel matrix
u = randn(m, N)*5; % input
% u = repmat(0:N-1, m, 1); % test input

HL = zeros(mL, N-L+1);
for i = 1:L
    rowstart = (i-1)*m+1;
    rowend = i*m;
    HL(rowstart:rowend, :) = u(:, i:N-L+i);
end

% check rank of Hankel matrix
if rank(HL) < mL
    error('Hankel matrix is not full rank');
else
    fprintf('Hankel matrix is full rank\n');
end

% generate all N samples
z = zeros(n+m, N);
zopt_kp1 = zeros(n+m, N);
x_k = randn(n,1);
for i = 1:N
    z(1:n, i) = x_k;
    z(n+1:end, i) = u(:, i);
    x_kp1 = aerialVehSim(x_k, u(:, i), Ts);
    zopt_kp1(1:n, i) = x_kp1;
    x_k = x_kp1;
end
```

```
% select l samples
l = n+m;
Z = z(:, 1:l);
Zopt = zopt_kp1(:, 1:l);

% check rank of Z
if rank(Z) < l
    error('Z matrix is not full rank');
else
    fprintf('Z matrix is full rank\n');
end
```

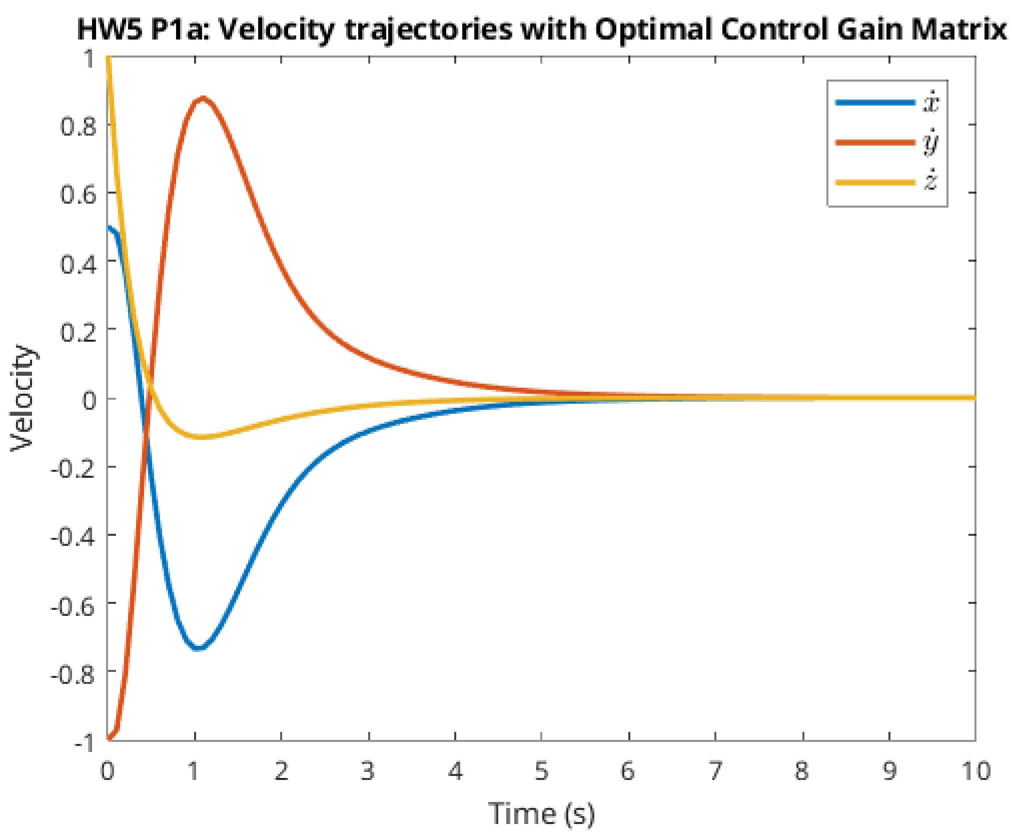
Hankel matrix is full rank
Z matrix is full rank

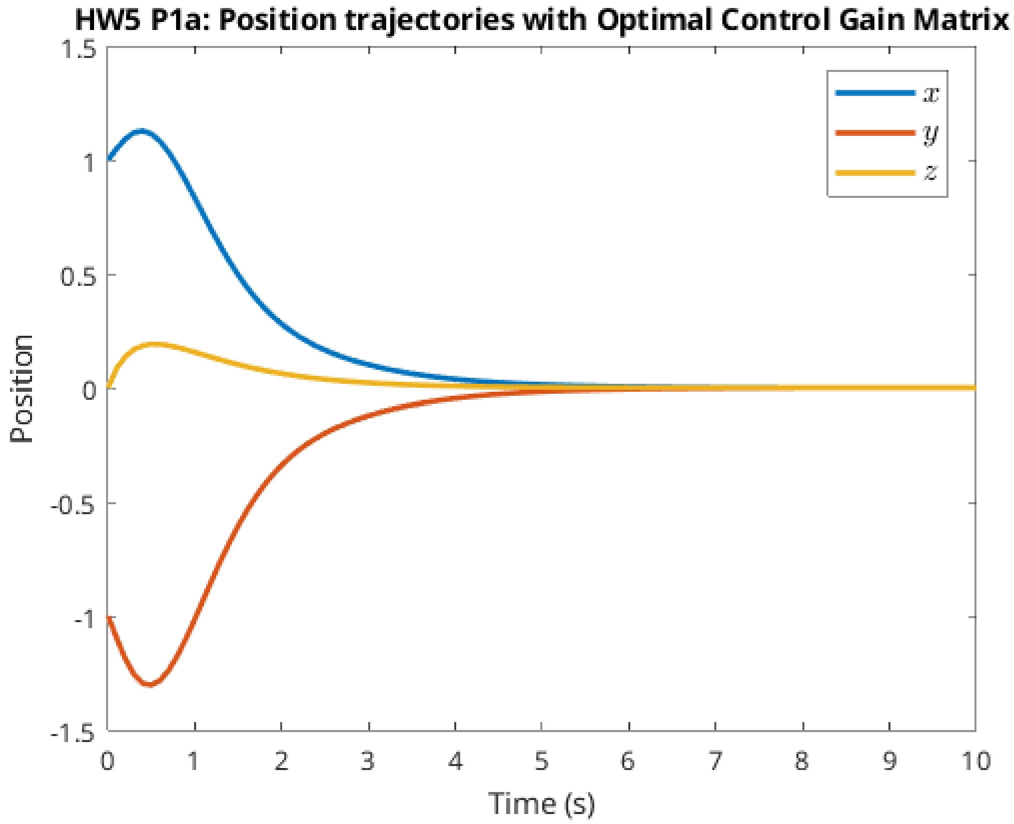
part a

```
Q = 10*eye(n);
Qbar = blkdiag(Q, R);
Kopt = RLLQR(n, m, Qbar, Z, Zopt);

% simulate using the optimal control gain matrix
Tend = 10; % simulation time
tsteps = 0:Ts:Tend;
x_all = zeros(n, length(tsteps));
x_k = [1; -1; 0; 0.5; -1; 1; 0; 0; 0; 0; 0; 0];

for i=1:length(tsteps)
    x_all(:, i) = x_k;
    u_k = -Kopt * x_k;
    x_kp1 = aerialVehSim(x_k, u_k, Ts);
    x_k = x_kp1;
end

% plot results
fig = figure;
plot(tsteps, x_all(1, :), 'LineWidth', 2, 'DisplayName', "$x$");
hold on;
plot(tsteps, x_all(2, :), 'LineWidth', 2, 'DisplayName', "$y$");
plot(tsteps, x_all(3, :), 'LineWidth', 2, 'DisplayName', "$z$");
ylabel("Position");
xlabel("Time (s)");
legend('Interpreter', 'latex', 'FontSize', 12, 'Location', 'best');
title("HW5 P1a: Position trajectories with Optimal Control Gain Matrix");
saveas(fig, "figs/hw5p1a_pos.svg");

fig = figure;
plot(tsteps, x_all(4, :), 'LineWidth', 2, 'DisplayName', "$\dot{x}$");
hold on;
plot(tsteps, x_all(5, :), 'LineWidth', 2, 'DisplayName', "$\dot{y}$");
plot(tsteps, x_all(6, :), 'LineWidth', 2, 'DisplayName', "$\dot{z}$");
ylabel("Velocity");
xlabel("Time (s)");
```

```
title("HW5 P1a: Velocity trajectories with Optimal Control Gain Matrix");
legend('Interpreter', 'latex', 'FontSize', 12, 'Location', 'best');
saveas(fig, "figs/hw5p1a_vel.svg");
```

```
Iteration 1: norm(K_i - K_ip1) = 4110.075767
Iteration 2: norm(K_i - K_ip1) = 4460.324909
Iteration 3: norm(K_i - K_ip1) = 2184.181812
Iteration 4: norm(K_i - K_ip1) = 24.802127
Iteration 5: norm(K_i - K_ip1) = 12.415983
Iteration 6: norm(K_i - K_ip1) = 6.127382
Iteration 7: norm(K_i - K_ip1) = 2.826830
Iteration 8: norm(K_i - K_ip1) = 1.012937
Iteration 9: norm(K_i - K_ip1) = 0.170946
Iteration 10: norm(K_i - K_ip1) = 0.005106
Iteration 11: norm(K_i - K_ip1) = 0.000005
Iteration 12: norm(K_i - K_ip1) = 0.000000
Converged after 12 iterations
```





part b

only penalize position

```
Q = blkdiag(10*eye(3), zeros(n-3, n-3));
Qbar = blkdiag(Q, R);
Kopt = RLLQR(n, m, Qbar, Z, Zopt);

% simulate using the optimal control gain matrix
Tend = 15; % simulation time
tsteps = 0:Ts:Tend;
x_all = zeros(n, length(tsteps));
x_ref = zeros(n, length(tsteps));
x_ref(1:3, :) = [
    0.1*sin(tsteps/2);
    0.1*cos(tsteps/2);
    0.1*tsteps;
    ];
x_k = x_ref(:, 1);

for i=1:length(tsteps)
    x_all(:, i) = x_k;
    u_k = -Kopt * (x_k - x_ref(:, i));
    x_kp1 = aerialVehSim(x_k, u_k, Ts);
    x_k = x_kp1;
end
```

```
% plot results
fig = figure;
plot3(x_ref(1, :), x_ref(2, :), x_ref(3, :), '--', 'LineWidth', 2,
'DisplayName', "Desired");
hold on;
plot3(x_all(1, :), x_all(2, :), x_all(3, :), 'LineWidth', 2, 'DisplayName',
"Actual");
xlabel("x");
ylabel("y");
zlabel("z");
title("HW5 P1b: Desired vs actual maneuver position trajectories");
legend('Location', 'best');
saveas(fig, "figs/hw5p1b_pos.svg");

function Kopt = RLLQR(n, m, Qbar, Z, Zopt)
% iterate and find theta matrix
K_i = zeros(m, n);

for i=1:1000
    Zopt(n+1:end, :) = -K_i * Zopt(1:n, :);
    theta_ip1 = dlyap(Z', -Z'*Qbar*Z, [], Zopt');
    % theta_ip1 = dlyap(Zopt', Z'*Qbar*Z, [], Z'); % alternative
    K_ip1 = inv(theta_ip1(n+1:end, n+1:end)) * theta_ip1(n+1:end, 1:n);

    % check convergence
    fprintf('Iteration %d: norm(K_i - K_ip1) = %f\n', i, norm(K_i - K_ip1));
    if norm(K_ip1 - K_i) < 1e-6
        fprintf('Converged after %d iterations\n', i);
        break;
    end

    K_i = K_ip1;
end
Kopt = K_i;
end
```

```
Iteration 1: norm(K_i - K_ip1) = 4117.966843
Iteration 2: norm(K_i - K_ip1) = 4468.186825
Iteration 3: norm(K_i - K_ip1) = 2186.600516
Iteration 4: norm(K_i - K_ip1) = 24.822961
Iteration 5: norm(K_i - K_ip1) = 12.445177
Iteration 6: norm(K_i - K_ip1) = 6.166353
Iteration 7: norm(K_i - K_ip1) = 2.858747
Iteration 8: norm(K_i - K_ip1) = 1.048394
Iteration 9: norm(K_i - K_ip1) = 0.186601
Iteration 10: norm(K_i - K_ip1) = 0.056217
Iteration 11: norm(K_i - K_ip1) = 0.034742
Iteration 12: norm(K_i - K_ip1) = 0.022563
Iteration 13: norm(K_i - K_ip1) = 0.015195
Iteration 14: norm(K_i - K_ip1) = 0.010467
Iteration 15: norm(K_i - K_ip1) = 0.007302
Iteration 16: norm(K_i - K_ip1) = 0.005125
Iteration 17: norm(K_i - K_ip1) = 0.003602
```

```
Iteration 18: norm(K_i - K_ip1) = 0.002494
Iteration 19: norm(K_i - K_ip1) = 0.001785
Iteration 20: norm(K_i - K_ip1) = 0.001150
Iteration 21: norm(K_i - K_ip1) = 0.000008
Iteration 22: norm(K_i - K_ip1) = 0.000465
Iteration 23: norm(K_i - K_ip1) = 0.000280
Iteration 24: norm(K_i - K_ip1) = 0.000112
Iteration 25: norm(K_i - K_ip1) = 0.001089
Iteration 26: norm(K_i - K_ip1) = 0.000370
Iteration 27: norm(K_i - K_ip1) = 0.000289
Iteration 28: norm(K_i - K_ip1) = 0.000149
Iteration 29: norm(K_i - K_ip1) = 0.000125
Iteration 30: norm(K_i - K_ip1) = 0.000304
Iteration 31: norm(K_i - K_ip1) = 0.000223
Iteration 32: norm(K_i - K_ip1) = 0.000122
Iteration 33: norm(K_i - K_ip1) = 0.000069
Iteration 34: norm(K_i - K_ip1) = 0.000334
Iteration 35: norm(K_i - K_ip1) = 0.000175
Iteration 36: norm(K_i - K_ip1) = 0.000288
Iteration 37: norm(K_i - K_ip1) = 0.000134
Iteration 38: norm(K_i - K_ip1) = 0.000290
Iteration 39: norm(K_i - K_ip1) = 0.000121
Iteration 40: norm(K_i - K_ip1) = 0.000453
Iteration 41: norm(K_i - K_ip1) = 0.001085
Iteration 42: norm(K_i - K_ip1) = 0.000525
Iteration 43: norm(K_i - K_ip1) = 0.000373
Iteration 44: norm(K_i - K_ip1) = 0.000297
Iteration 45: norm(K_i - K_ip1) = 0.000389
Iteration 46: norm(K_i - K_ip1) = 0.000489
Iteration 47: norm(K_i - K_ip1) = 0.000448
Iteration 48: norm(K_i - K_ip1) = 0.000409
Iteration 49: norm(K_i - K_ip1) = 0.000199
Iteration 50: norm(K_i - K_ip1) = 0.000664
Iteration 51: norm(K_i - K_ip1) = 0.000439
Iteration 52: norm(K_i - K_ip1) = 0.000429
Iteration 53: norm(K_i - K_ip1) = 0.001470
Iteration 54: norm(K_i - K_ip1) = 0.000522
Iteration 55: norm(K_i - K_ip1) = 0.000317
Iteration 56: norm(K_i - K_ip1) = 0.000381
Iteration 57: norm(K_i - K_ip1) = 0.000883
Iteration 58: norm(K_i - K_ip1) = 0.000212
Iteration 59: norm(K_i - K_ip1) = 0.000468
Iteration 60: norm(K_i - K_ip1) = 0.000281
Iteration 61: norm(K_i - K_ip1) = 0.000121
Iteration 62: norm(K_i - K_ip1) = 0.000200
Iteration 63: norm(K_i - K_ip1) = 0.000161
Iteration 64: norm(K_i - K_ip1) = 0.000038
Iteration 65: norm(K_i - K_ip1) = 0.000330
Iteration 66: norm(K_i - K_ip1) = 0.000424
Iteration 67: norm(K_i - K_ip1) = 0.000308
Iteration 68: norm(K_i - K_ip1) = 0.000136
Iteration 69: norm(K_i - K_ip1) = 0.000142
Iteration 70: norm(K_i - K_ip1) = 0.000636
Iteration 71: norm(K_i - K_ip1) = 0.000394
```

```
Iteration 72: norm(K_i - K_ip1) = 0.000217
Iteration 73: norm(K_i - K_ip1) = 0.000212
Iteration 74: norm(K_i - K_ip1) = 0.000271
Iteration 75: norm(K_i - K_ip1) = 0.000538
Iteration 76: norm(K_i - K_ip1) = 0.000863
Iteration 77: norm(K_i - K_ip1) = 0.000317
Iteration 78: norm(K_i - K_ip1) = 0.000049
Iteration 79: norm(K_i - K_ip1) = 0.000562
Iteration 80: norm(K_i - K_ip1) = 0.000537
Iteration 81: norm(K_i - K_ip1) = 0.000104
Iteration 82: norm(K_i - K_ip1) = 0.000153
Iteration 83: norm(K_i - K_ip1) = 0.000077
Iteration 84: norm(K_i - K_ip1) = 0.000334
Iteration 85: norm(K_i - K_ip1) = 0.000369
Iteration 86: norm(K_i - K_ip1) = 0.000502
Iteration 87: norm(K_i - K_ip1) = 0.001160
Iteration 88: norm(K_i - K_ip1) = 0.000681
Iteration 89: norm(K_i - K_ip1) = 0.000602
Iteration 90: norm(K_i - K_ip1) = 0.000103
Iteration 91: norm(K_i - K_ip1) = 0.001273
Iteration 92: norm(K_i - K_ip1) = 0.000518
Iteration 93: norm(K_i - K_ip1) = 0.000363
Iteration 94: norm(K_i - K_ip1) = 0.000244
Iteration 95: norm(K_i - K_ip1) = 0.000119
Iteration 96: norm(K_i - K_ip1) = 0.000063
Iteration 97: norm(K_i - K_ip1) = 0.000192
Iteration 98: norm(K_i - K_ip1) = 0.000407
Iteration 99: norm(K_i - K_ip1) = 0.000379
Iteration 100: norm(K_i - K_ip1) = 0.000338
Iteration 101: norm(K_i - K_ip1) = 0.000201
Iteration 102: norm(K_i - K_ip1) = 0.000138
Iteration 103: norm(K_i - K_ip1) = 0.000022
Iteration 104: norm(K_i - K_ip1) = 0.000191
Iteration 105: norm(K_i - K_ip1) = 0.000266
Iteration 106: norm(K_i - K_ip1) = 0.000389
Iteration 107: norm(K_i - K_ip1) = 0.000561
Iteration 108: norm(K_i - K_ip1) = 0.000018
Iteration 109: norm(K_i - K_ip1) = 0.000655
Iteration 110: norm(K_i - K_ip1) = 0.000522
Iteration 111: norm(K_i - K_ip1) = 0.000009
Iteration 112: norm(K_i - K_ip1) = 0.000571
Iteration 113: norm(K_i - K_ip1) = 0.000559
Iteration 114: norm(K_i - K_ip1) = 0.000088
Iteration 115: norm(K_i - K_ip1) = 0.000288
Iteration 116: norm(K_i - K_ip1) = 0.000476
Iteration 117: norm(K_i - K_ip1) = 0.000028
Iteration 118: norm(K_i - K_ip1) = 0.000007
Iteration 119: norm(K_i - K_ip1) = 0.000517
Iteration 120: norm(K_i - K_ip1) = 0.000281
Iteration 121: norm(K_i - K_ip1) = 0.000318
Iteration 122: norm(K_i - K_ip1) = 0.000643
Iteration 123: norm(K_i - K_ip1) = 0.000159
Iteration 124: norm(K_i - K_ip1) = 0.000036
Iteration 125: norm(K_i - K_ip1) = 0.000106
```

```
Iteration 126: norm(K_i - K_ip1) = 0.000119
Iteration 127: norm(K_i - K_ip1) = 0.000170
Iteration 128: norm(K_i - K_ip1) = 0.000379
Iteration 129: norm(K_i - K_ip1) = 0.000448
Iteration 130: norm(K_i - K_ip1) = 0.000142
Iteration 131: norm(K_i - K_ip1) = 0.000378
Iteration 132: norm(K_i - K_ip1) = 0.000412
Iteration 133: norm(K_i - K_ip1) = 0.000583
Iteration 134: norm(K_i - K_ip1) = 0.000918
Iteration 135: norm(K_i - K_ip1) = 0.000190
Iteration 136: norm(K_i - K_ip1) = 0.000246
Iteration 137: norm(K_i - K_ip1) = 0.000212
Iteration 138: norm(K_i - K_ip1) = 0.000844
Iteration 139: norm(K_i - K_ip1) = 0.000311
Iteration 140: norm(K_i - K_ip1) = 0.000122
Iteration 141: norm(K_i - K_ip1) = 0.000146
Iteration 142: norm(K_i - K_ip1) = 0.000051
Iteration 143: norm(K_i - K_ip1) = 0.000185
Iteration 144: norm(K_i - K_ip1) = 0.000344
Iteration 145: norm(K_i - K_ip1) = 0.000152
Iteration 146: norm(K_i - K_ip1) = 0.000286
Iteration 147: norm(K_i - K_ip1) = 0.000114
Iteration 148: norm(K_i - K_ip1) = 0.000068
Iteration 149: norm(K_i - K_ip1) = 0.000263
Iteration 150: norm(K_i - K_ip1) = 0.000059
Iteration 151: norm(K_i - K_ip1) = 0.000268
Iteration 152: norm(K_i - K_ip1) = 0.001036
Iteration 153: norm(K_i - K_ip1) = 0.000566
Iteration 154: norm(K_i - K_ip1) = 0.000087
Iteration 155: norm(K_i - K_ip1) = 0.000281
Iteration 156: norm(K_i - K_ip1) = 0.000171
Iteration 157: norm(K_i - K_ip1) = 0.000035
Iteration 158: norm(K_i - K_ip1) = 0.000279
Iteration 159: norm(K_i - K_ip1) = 0.000285
Iteration 160: norm(K_i - K_ip1) = 0.000265
Iteration 161: norm(K_i - K_ip1) = 0.000502
Iteration 162: norm(K_i - K_ip1) = 0.000050
Iteration 163: norm(K_i - K_ip1) = 0.000151
Iteration 164: norm(K_i - K_ip1) = 0.000219
Iteration 165: norm(K_i - K_ip1) = 0.000160
Iteration 166: norm(K_i - K_ip1) = 0.000015
Iteration 167: norm(K_i - K_ip1) = 0.000249
Iteration 168: norm(K_i - K_ip1) = 0.000470
Iteration 169: norm(K_i - K_ip1) = 0.000405
Iteration 170: norm(K_i - K_ip1) = 0.000375
Iteration 171: norm(K_i - K_ip1) = 0.000028
Iteration 172: norm(K_i - K_ip1) = 0.000112
Iteration 173: norm(K_i - K_ip1) = 0.000225
Iteration 174: norm(K_i - K_ip1) = 0.000082
Iteration 175: norm(K_i - K_ip1) = 0.000143
Iteration 176: norm(K_i - K_ip1) = 0.000190
Iteration 177: norm(K_i - K_ip1) = 0.000169
Iteration 178: norm(K_i - K_ip1) = 0.000078
Iteration 179: norm(K_i - K_ip1) = 0.000070
```

```
Iteration 180: norm(K_i - K_ip1) = 0.000368
Iteration 181: norm(K_i - K_ip1) = 0.000595
Iteration 182: norm(K_i - K_ip1) = 0.000480
Iteration 183: norm(K_i - K_ip1) = 0.000438
Iteration 184: norm(K_i - K_ip1) = 0.000017
Iteration 185: norm(K_i - K_ip1) = 0.000459
Iteration 186: norm(K_i - K_ip1) = 0.000168
Iteration 187: norm(K_i - K_ip1) = 0.000046
Iteration 188: norm(K_i - K_ip1) = 0.000371
Iteration 189: norm(K_i - K_ip1) = 0.000328
Iteration 190: norm(K_i - K_ip1) = 0.000227
Iteration 191: norm(K_i - K_ip1) = 0.000330
Iteration 192: norm(K_i - K_ip1) = 0.001007
Iteration 193: norm(K_i - K_ip1) = 0.000448
Iteration 194: norm(K_i - K_ip1) = 0.000257
Iteration 195: norm(K_i - K_ip1) = 0.000261
Iteration 196: norm(K_i - K_ip1) = 0.000080
Iteration 197: norm(K_i - K_ip1) = 0.000054
Iteration 198: norm(K_i - K_ip1) = 0.000072
Iteration 199: norm(K_i - K_ip1) = 0.000029
Iteration 200: norm(K_i - K_ip1) = 0.000054
Iteration 201: norm(K_i - K_ip1) = 0.000146
Iteration 202: norm(K_i - K_ip1) = 0.000155
Iteration 203: norm(K_i - K_ip1) = 0.000026
Iteration 204: norm(K_i - K_ip1) = 0.000282
Iteration 205: norm(K_i - K_ip1) = 0.000049
Iteration 206: norm(K_i - K_ip1) = 0.000554
Iteration 207: norm(K_i - K_ip1) = 0.000060
Iteration 208: norm(K_i - K_ip1) = 0.000045
Iteration 209: norm(K_i - K_ip1) = 0.000140
Iteration 210: norm(K_i - K_ip1) = 0.000759
Iteration 211: norm(K_i - K_ip1) = 0.000039
Iteration 212: norm(K_i - K_ip1) = 0.000575
Iteration 213: norm(K_i - K_ip1) = 0.000011
Iteration 214: norm(K_i - K_ip1) = 0.000231
Iteration 215: norm(K_i - K_ip1) = 0.000360
Iteration 216: norm(K_i - K_ip1) = 0.000011
Iteration 217: norm(K_i - K_ip1) = 0.000015
Iteration 218: norm(K_i - K_ip1) = 0.000398
Iteration 219: norm(K_i - K_ip1) = 0.000688
Iteration 220: norm(K_i - K_ip1) = 0.000385
Iteration 221: norm(K_i - K_ip1) = 0.000054
Iteration 222: norm(K_i - K_ip1) = 0.000051
Iteration 223: norm(K_i - K_ip1) = 0.000713
Iteration 224: norm(K_i - K_ip1) = 0.001500
Iteration 225: norm(K_i - K_ip1) = 0.000888
Iteration 226: norm(K_i - K_ip1) = 0.000112
Iteration 227: norm(K_i - K_ip1) = 0.000014
Iteration 228: norm(K_i - K_ip1) = 0.000152
Iteration 229: norm(K_i - K_ip1) = 0.000603
Iteration 230: norm(K_i - K_ip1) = 0.000042
Iteration 231: norm(K_i - K_ip1) = 0.000343
Iteration 232: norm(K_i - K_ip1) = 0.000446
Iteration 233: norm(K_i - K_ip1) = 0.000255
```

```
Iteration 234: norm(K_i - K_ip1) = 0.000302
Iteration 235: norm(K_i - K_ip1) = 0.000012
Iteration 236: norm(K_i - K_ip1) = 0.000419
Iteration 237: norm(K_i - K_ip1) = 0.000002
Iteration 238: norm(K_i - K_ip1) = 0.000062
Iteration 239: norm(K_i - K_ip1) = 0.000672
Iteration 240: norm(K_i - K_ip1) = 0.000598
Iteration 241: norm(K_i - K_ip1) = 0.000038
Iteration 242: norm(K_i - K_ip1) = 0.000027
Iteration 243: norm(K_i - K_ip1) = 0.000282
Iteration 244: norm(K_i - K_ip1) = 0.000045
Iteration 245: norm(K_i - K_ip1) = 0.000533
Iteration 246: norm(K_i - K_ip1) = 0.000465
Iteration 247: norm(K_i - K_ip1) = 0.000021
Iteration 248: norm(K_i - K_ip1) = 0.000228
Iteration 249: norm(K_i - K_ip1) = 0.000391
Iteration 250: norm(K_i - K_ip1) = 0.000165
Iteration 251: norm(K_i - K_ip1) = 0.000430
Iteration 252: norm(K_i - K_ip1) = 0.000351
Iteration 253: norm(K_i - K_ip1) = 0.000694
Iteration 254: norm(K_i - K_ip1) = 0.000180
Iteration 255: norm(K_i - K_ip1) = 0.000347
Iteration 256: norm(K_i - K_ip1) = 0.000321
Iteration 257: norm(K_i - K_ip1) = 0.000822
Iteration 258: norm(K_i - K_ip1) = 0.000698
Iteration 259: norm(K_i - K_ip1) = 0.000330
Iteration 260: norm(K_i - K_ip1) = 0.000176
Iteration 261: norm(K_i - K_ip1) = 0.000377
Iteration 262: norm(K_i - K_ip1) = 0.001624
Iteration 263: norm(K_i - K_ip1) = 0.000593
Iteration 264: norm(K_i - K_ip1) = 0.000293
Iteration 265: norm(K_i - K_ip1) = 0.000510
Iteration 266: norm(K_i - K_ip1) = 0.000427
Iteration 267: norm(K_i - K_ip1) = 0.000630
Iteration 268: norm(K_i - K_ip1) = 0.000502
Iteration 269: norm(K_i - K_ip1) = 0.000111
Iteration 270: norm(K_i - K_ip1) = 0.000346
Iteration 271: norm(K_i - K_ip1) = 0.000768
Iteration 272: norm(K_i - K_ip1) = 0.000229
Iteration 273: norm(K_i - K_ip1) = 0.000155
Iteration 274: norm(K_i - K_ip1) = 0.000182
Iteration 275: norm(K_i - K_ip1) = 0.000802
Iteration 276: norm(K_i - K_ip1) = 0.000752
Iteration 277: norm(K_i - K_ip1) = 0.000454
Iteration 278: norm(K_i - K_ip1) = 0.000259
Iteration 279: norm(K_i - K_ip1) = 0.000039
Iteration 280: norm(K_i - K_ip1) = 0.000054
Iteration 281: norm(K_i - K_ip1) = 0.000124
Iteration 282: norm(K_i - K_ip1) = 0.000411
Iteration 283: norm(K_i - K_ip1) = 0.000009
Iteration 284: norm(K_i - K_ip1) = 0.000520
Iteration 285: norm(K_i - K_ip1) = 0.000140
Iteration 286: norm(K_i - K_ip1) = 0.000133
Iteration 287: norm(K_i - K_ip1) = 0.000297
```

```
Iteration 288: norm(K_i - K_ip1) = 0.000451
Iteration 289: norm(K_i - K_ip1) = 0.001806
Iteration 290: norm(K_i - K_ip1) = 0.000976
Iteration 291: norm(K_i - K_ip1) = 0.000389
Iteration 292: norm(K_i - K_ip1) = 0.000271
Iteration 293: norm(K_i - K_ip1) = 0.000425
Iteration 294: norm(K_i - K_ip1) = 0.000018
Iteration 295: norm(K_i - K_ip1) = 0.000269
Iteration 296: norm(K_i - K_ip1) = 0.000018
Iteration 297: norm(K_i - K_ip1) = 0.000090
Iteration 298: norm(K_i - K_ip1) = 0.000970
Iteration 299: norm(K_i - K_ip1) = 0.000591
Iteration 300: norm(K_i - K_ip1) = 0.000415
Iteration 301: norm(K_i - K_ip1) = 0.000080
Iteration 302: norm(K_i - K_ip1) = 0.000078
Iteration 303: norm(K_i - K_ip1) = 0.000308
Iteration 304: norm(K_i - K_ip1) = 0.000064
Iteration 305: norm(K_i - K_ip1) = 0.000079
Iteration 306: norm(K_i - K_ip1) = 0.000040
Iteration 307: norm(K_i - K_ip1) = 0.000510
Iteration 308: norm(K_i - K_ip1) = 0.000604
Iteration 309: norm(K_i - K_ip1) = 0.000319
Iteration 310: norm(K_i - K_ip1) = 0.000064
Iteration 311: norm(K_i - K_ip1) = 0.000336
Iteration 312: norm(K_i - K_ip1) = 0.000814
Iteration 313: norm(K_i - K_ip1) = 0.000274
Iteration 314: norm(K_i - K_ip1) = 0.000450
Iteration 315: norm(K_i - K_ip1) = 0.000623
Iteration 316: norm(K_i - K_ip1) = 0.000448
Iteration 317: norm(K_i - K_ip1) = 0.000329
Iteration 318: norm(K_i - K_ip1) = 0.000168
Iteration 319: norm(K_i - K_ip1) = 0.000384
Iteration 320: norm(K_i - K_ip1) = 0.000288
Iteration 321: norm(K_i - K_ip1) = 0.000428
Iteration 322: norm(K_i - K_ip1) = 0.000374
Iteration 323: norm(K_i - K_ip1) = 0.000020
Iteration 324: norm(K_i - K_ip1) = 0.000121
Iteration 325: norm(K_i - K_ip1) = 0.000156
Iteration 326: norm(K_i - K_ip1) = 0.000118
Iteration 327: norm(K_i - K_ip1) = 0.000013
Iteration 328: norm(K_i - K_ip1) = 0.000025
Iteration 329: norm(K_i - K_ip1) = 0.000312
Iteration 330: norm(K_i - K_ip1) = 0.000096
Iteration 331: norm(K_i - K_ip1) = 0.000151
Iteration 332: norm(K_i - K_ip1) = 0.000349
Iteration 333: norm(K_i - K_ip1) = 0.000365
Iteration 334: norm(K_i - K_ip1) = 0.000468
Iteration 335: norm(K_i - K_ip1) = 0.000898
Iteration 336: norm(K_i - K_ip1) = 0.000118
Iteration 337: norm(K_i - K_ip1) = 0.000235
Iteration 338: norm(K_i - K_ip1) = 0.000028
Iteration 339: norm(K_i - K_ip1) = 0.000226
Iteration 340: norm(K_i - K_ip1) = 0.000086
Iteration 341: norm(K_i - K_ip1) = 0.000317
```

```
Iteration 342: norm(K_i - K_ip1) = 0.000438
Iteration 343: norm(K_i - K_ip1) = 0.000376
Iteration 344: norm(K_i - K_ip1) = 0.000081
Iteration 345: norm(K_i - K_ip1) = 0.000309
Iteration 346: norm(K_i - K_ip1) = 0.000389
Iteration 347: norm(K_i - K_ip1) = 0.000088
Iteration 348: norm(K_i - K_ip1) = 0.000270
Iteration 349: norm(K_i - K_ip1) = 0.000126
Iteration 350: norm(K_i - K_ip1) = 0.000146
Iteration 351: norm(K_i - K_ip1) = 0.000239
Iteration 352: norm(K_i - K_ip1) = 0.000735
Iteration 353: norm(K_i - K_ip1) = 0.000753
Iteration 354: norm(K_i - K_ip1) = 0.000564
Iteration 355: norm(K_i - K_ip1) = 0.000187
Iteration 356: norm(K_i - K_ip1) = 0.000038
Iteration 357: norm(K_i - K_ip1) = 0.000109
Iteration 358: norm(K_i - K_ip1) = 0.000195
Iteration 359: norm(K_i - K_ip1) = 0.000383
Iteration 360: norm(K_i - K_ip1) = 0.000144
Iteration 361: norm(K_i - K_ip1) = 0.000120
Iteration 362: norm(K_i - K_ip1) = 0.000105
Iteration 363: norm(K_i - K_ip1) = 0.000051
Iteration 364: norm(K_i - K_ip1) = 0.000135
Iteration 365: norm(K_i - K_ip1) = 0.000225
Iteration 366: norm(K_i - K_ip1) = 0.000277
Iteration 367: norm(K_i - K_ip1) = 0.000262
Iteration 368: norm(K_i - K_ip1) = 0.000340
Iteration 369: norm(K_i - K_ip1) = 0.000448
Iteration 370: norm(K_i - K_ip1) = 0.000003
Iteration 371: norm(K_i - K_ip1) = 0.000011
Iteration 372: norm(K_i - K_ip1) = 0.000022
Iteration 373: norm(K_i - K_ip1) = 0.000187
Iteration 374: norm(K_i - K_ip1) = 0.000507
Iteration 375: norm(K_i - K_ip1) = 0.001056
Iteration 376: norm(K_i - K_ip1) = 0.000397
Iteration 377: norm(K_i - K_ip1) = 0.000364
Iteration 378: norm(K_i - K_ip1) = 0.000747
Iteration 379: norm(K_i - K_ip1) = 0.000433
Iteration 380: norm(K_i - K_ip1) = 0.000154
Iteration 381: norm(K_i - K_ip1) = 0.000407
Iteration 382: norm(K_i - K_ip1) = 0.000038
Iteration 383: norm(K_i - K_ip1) = 0.000435
Iteration 384: norm(K_i - K_ip1) = 0.000085
Iteration 385: norm(K_i - K_ip1) = 0.000217
Iteration 386: norm(K_i - K_ip1) = 0.000347
Iteration 387: norm(K_i - K_ip1) = 0.000844
Iteration 388: norm(K_i - K_ip1) = 0.000630
Iteration 389: norm(K_i - K_ip1) = 0.000307
Iteration 390: norm(K_i - K_ip1) = 0.000358
Iteration 391: norm(K_i - K_ip1) = 0.000039
Iteration 392: norm(K_i - K_ip1) = 0.000248
Iteration 393: norm(K_i - K_ip1) = 0.000326
Iteration 394: norm(K_i - K_ip1) = 0.000987
Iteration 395: norm(K_i - K_ip1) = 0.000519
```

```
Iteration 396: norm(K_i - K_ip1) = 0.000212
Iteration 397: norm(K_i - K_ip1) = 0.000320
Iteration 398: norm(K_i - K_ip1) = 0.000313
Iteration 399: norm(K_i - K_ip1) = 0.000167
Iteration 400: norm(K_i - K_ip1) = 0.000488
Iteration 401: norm(K_i - K_ip1) = 0.000764
Iteration 402: norm(K_i - K_ip1) = 0.000004
Iteration 403: norm(K_i - K_ip1) = 0.000301
Iteration 404: norm(K_i - K_ip1) = 0.000616
Iteration 405: norm(K_i - K_ip1) = 0.000321
Iteration 406: norm(K_i - K_ip1) = 0.000100
Iteration 407: norm(K_i - K_ip1) = 0.000148
Iteration 408: norm(K_i - K_ip1) = 0.000430
Iteration 409: norm(K_i - K_ip1) = 0.000226
Iteration 410: norm(K_i - K_ip1) = 0.000015
Iteration 411: norm(K_i - K_ip1) = 0.000170
Iteration 412: norm(K_i - K_ip1) = 0.000089
Iteration 413: norm(K_i - K_ip1) = 0.000009
Iteration 414: norm(K_i - K_ip1) = 0.000116
Iteration 415: norm(K_i - K_ip1) = 0.000198
Iteration 416: norm(K_i - K_ip1) = 0.000194
Iteration 417: norm(K_i - K_ip1) = 0.000150
Iteration 418: norm(K_i - K_ip1) = 0.000093
Iteration 419: norm(K_i - K_ip1) = 0.000361
Iteration 420: norm(K_i - K_ip1) = 0.000224
Iteration 421: norm(K_i - K_ip1) = 0.000494
Iteration 422: norm(K_i - K_ip1) = 0.000766
Iteration 423: norm(K_i - K_ip1) = 0.000319
Iteration 424: norm(K_i - K_ip1) = 0.000250
Iteration 425: norm(K_i - K_ip1) = 0.000785
Iteration 426: norm(K_i - K_ip1) = 0.000458
Iteration 427: norm(K_i - K_ip1) = 0.000093
Iteration 428: norm(K_i - K_ip1) = 0.000158
Iteration 429: norm(K_i - K_ip1) = 0.000101
Iteration 430: norm(K_i - K_ip1) = 0.000461
Iteration 431: norm(K_i - K_ip1) = 0.000163
Iteration 432: norm(K_i - K_ip1) = 0.000312
Iteration 433: norm(K_i - K_ip1) = 0.000466
Iteration 434: norm(K_i - K_ip1) = 0.000537
Iteration 435: norm(K_i - K_ip1) = 0.000142
Iteration 436: norm(K_i - K_ip1) = 0.000810
Iteration 437: norm(K_i - K_ip1) = 0.000251
Iteration 438: norm(K_i - K_ip1) = 0.000012
Iteration 439: norm(K_i - K_ip1) = 0.000131
Iteration 440: norm(K_i - K_ip1) = 0.000251
Iteration 441: norm(K_i - K_ip1) = 0.000163
Iteration 442: norm(K_i - K_ip1) = 0.000036
Iteration 443: norm(K_i - K_ip1) = 0.000531
Iteration 444: norm(K_i - K_ip1) = 0.000260
Iteration 445: norm(K_i - K_ip1) = 0.000406
Iteration 446: norm(K_i - K_ip1) = 0.000769
Iteration 447: norm(K_i - K_ip1) = 0.000340
Iteration 448: norm(K_i - K_ip1) = 0.000279
Iteration 449: norm(K_i - K_ip1) = 0.000220
```

```
Iteration 450: norm(K_i - K_ip1) = 0.000326
Iteration 451: norm(K_i - K_ip1) = 0.000054
Iteration 452: norm(K_i - K_ip1) = 0.000683
Iteration 453: norm(K_i - K_ip1) = 0.000624
Iteration 454: norm(K_i - K_ip1) = 0.000045
Iteration 455: norm(K_i - K_ip1) = 0.000011
Iteration 456: norm(K_i - K_ip1) = 0.000255
Iteration 457: norm(K_i - K_ip1) = 0.000737
Iteration 458: norm(K_i - K_ip1) = 0.000198
Iteration 459: norm(K_i - K_ip1) = 0.000360
Iteration 460: norm(K_i - K_ip1) = 0.000110
Iteration 461: norm(K_i - K_ip1) = 0.000275
Iteration 462: norm(K_i - K_ip1) = 0.000058
Iteration 463: norm(K_i - K_ip1) = 0.000387
Iteration 464: norm(K_i - K_ip1) = 0.000169
Iteration 465: norm(K_i - K_ip1) = 0.000026
Iteration 466: norm(K_i - K_ip1) = 0.000366
Iteration 467: norm(K_i - K_ip1) = 0.000827
Iteration 468: norm(K_i - K_ip1) = 0.000383
Iteration 469: norm(K_i - K_ip1) = 0.000195
Iteration 470: norm(K_i - K_ip1) = 0.000467
Iteration 471: norm(K_i - K_ip1) = 0.000387
Iteration 472: norm(K_i - K_ip1) = 0.000422
Iteration 473: norm(K_i - K_ip1) = 0.000242
Iteration 474: norm(K_i - K_ip1) = 0.000075
Iteration 475: norm(K_i - K_ip1) = 0.000010
Iteration 476: norm(K_i - K_ip1) = 0.000319
Iteration 477: norm(K_i - K_ip1) = 0.000170
Iteration 478: norm(K_i - K_ip1) = 0.000381
Iteration 479: norm(K_i - K_ip1) = 0.000076
Iteration 480: norm(K_i - K_ip1) = 0.000163
Iteration 481: norm(K_i - K_ip1) = 0.000181
Iteration 482: norm(K_i - K_ip1) = 0.000199
Iteration 483: norm(K_i - K_ip1) = 0.000120
Iteration 484: norm(K_i - K_ip1) = 0.000523
Iteration 485: norm(K_i - K_ip1) = 0.000636
Iteration 486: norm(K_i - K_ip1) = 0.000173
Iteration 487: norm(K_i - K_ip1) = 0.000185
Iteration 488: norm(K_i - K_ip1) = 0.000362
Iteration 489: norm(K_i - K_ip1) = 0.000113
Iteration 490: norm(K_i - K_ip1) = 0.000241
Iteration 491: norm(K_i - K_ip1) = 0.000177
Iteration 492: norm(K_i - K_ip1) = 0.000452
Iteration 493: norm(K_i - K_ip1) = 0.000344
Iteration 494: norm(K_i - K_ip1) = 0.000230
Iteration 495: norm(K_i - K_ip1) = 0.000779
Iteration 496: norm(K_i - K_ip1) = 0.000501
Iteration 497: norm(K_i - K_ip1) = 0.000053
Iteration 498: norm(K_i - K_ip1) = 0.000284
Iteration 499: norm(K_i - K_ip1) = 0.000351
Iteration 500: norm(K_i - K_ip1) = 0.000202
Iteration 501: norm(K_i - K_ip1) = 0.000260
Iteration 502: norm(K_i - K_ip1) = 0.000706
Iteration 503: norm(K_i - K_ip1) = 0.000650
```

```
Iteration 504: norm(K_i - K_ip1) = 0.000027
Iteration 505: norm(K_i - K_ip1) = 0.000309
Iteration 506: norm(K_i - K_ip1) = 0.000469
Iteration 507: norm(K_i - K_ip1) = 0.000245
Iteration 508: norm(K_i - K_ip1) = 0.000238
Iteration 509: norm(K_i - K_ip1) = 0.000322
Iteration 510: norm(K_i - K_ip1) = 0.000666
Iteration 511: norm(K_i - K_ip1) = 0.000080
Iteration 512: norm(K_i - K_ip1) = 0.000321
Iteration 513: norm(K_i - K_ip1) = 0.000279
Iteration 514: norm(K_i - K_ip1) = 0.000752
Iteration 515: norm(K_i - K_ip1) = 0.000036
Iteration 516: norm(K_i - K_ip1) = 0.000421
Iteration 517: norm(K_i - K_ip1) = 0.000122
Iteration 518: norm(K_i - K_ip1) = 0.000213
Iteration 519: norm(K_i - K_ip1) = 0.000299
Iteration 520: norm(K_i - K_ip1) = 0.000155
Iteration 521: norm(K_i - K_ip1) = 0.000080
Iteration 522: norm(K_i - K_ip1) = 0.000242
Iteration 523: norm(K_i - K_ip1) = 0.000516
Iteration 524: norm(K_i - K_ip1) = 0.000067
Iteration 525: norm(K_i - K_ip1) = 0.001110
Iteration 526: norm(K_i - K_ip1) = 0.000550
Iteration 527: norm(K_i - K_ip1) = 0.000094
Iteration 528: norm(K_i - K_ip1) = 0.000092
Iteration 529: norm(K_i - K_ip1) = 0.000172
Iteration 530: norm(K_i - K_ip1) = 0.000462
Iteration 531: norm(K_i - K_ip1) = 0.000181
Iteration 532: norm(K_i - K_ip1) = 0.000314
Iteration 533: norm(K_i - K_ip1) = 0.001142
Iteration 534: norm(K_i - K_ip1) = 0.000560
Iteration 535: norm(K_i - K_ip1) = 0.000456
Iteration 536: norm(K_i - K_ip1) = 0.000354
Iteration 537: norm(K_i - K_ip1) = 0.000141
Iteration 538: norm(K_i - K_ip1) = 0.000281
Iteration 539: norm(K_i - K_ip1) = 0.000171
Iteration 540: norm(K_i - K_ip1) = 0.000379
Iteration 541: norm(K_i - K_ip1) = 0.000154
Iteration 542: norm(K_i - K_ip1) = 0.000232
Iteration 543: norm(K_i - K_ip1) = 0.000172
Iteration 544: norm(K_i - K_ip1) = 0.000437
Iteration 545: norm(K_i - K_ip1) = 0.001336
Iteration 546: norm(K_i - K_ip1) = 0.000789
Iteration 547: norm(K_i - K_ip1) = 0.000216
Iteration 548: norm(K_i - K_ip1) = 0.000331
Iteration 549: norm(K_i - K_ip1) = 0.000270
Iteration 550: norm(K_i - K_ip1) = 0.000250
Iteration 551: norm(K_i - K_ip1) = 0.000237
Iteration 552: norm(K_i - K_ip1) = 0.000122
Iteration 553: norm(K_i - K_ip1) = 0.000221
Iteration 554: norm(K_i - K_ip1) = 0.000223
Iteration 555: norm(K_i - K_ip1) = 0.000514
Iteration 556: norm(K_i - K_ip1) = 0.000682
Iteration 557: norm(K_i - K_ip1) = 0.000771
```

```
Iteration 558: norm(K_i - K_ip1) = 0.000063
Iteration 559: norm(K_i - K_ip1) = 0.000355
Iteration 560: norm(K_i - K_ip1) = 0.000503
Iteration 561: norm(K_i - K_ip1) = 0.000366
Iteration 562: norm(K_i - K_ip1) = 0.000202
Iteration 563: norm(K_i - K_ip1) = 0.000890
Iteration 564: norm(K_i - K_ip1) = 0.000351
Iteration 565: norm(K_i - K_ip1) = 0.000649
Iteration 566: norm(K_i - K_ip1) = 0.001067
Iteration 567: norm(K_i - K_ip1) = 0.000497
Iteration 568: norm(K_i - K_ip1) = 0.000343
Iteration 569: norm(K_i - K_ip1) = 0.000165
Iteration 570: norm(K_i - K_ip1) = 0.000272
Iteration 571: norm(K_i - K_ip1) = 0.000056
Iteration 572: norm(K_i - K_ip1) = 0.000240
Iteration 573: norm(K_i - K_ip1) = 0.000516
Iteration 574: norm(K_i - K_ip1) = 0.000500
Iteration 575: norm(K_i - K_ip1) = 0.000378
Iteration 576: norm(K_i - K_ip1) = 0.000394
Iteration 577: norm(K_i - K_ip1) = 0.000172
Iteration 578: norm(K_i - K_ip1) = 0.000059
Iteration 579: norm(K_i - K_ip1) = 0.000156
Iteration 580: norm(K_i - K_ip1) = 0.000095
Iteration 581: norm(K_i - K_ip1) = 0.000093
Iteration 582: norm(K_i - K_ip1) = 0.000409
Iteration 583: norm(K_i - K_ip1) = 0.000688
Iteration 584: norm(K_i - K_ip1) = 0.001122
Iteration 585: norm(K_i - K_ip1) = 0.000785
Iteration 586: norm(K_i - K_ip1) = 0.000019
Iteration 587: norm(K_i - K_ip1) = 0.000289
Iteration 588: norm(K_i - K_ip1) = 0.000518
Iteration 589: norm(K_i - K_ip1) = 0.000123
Iteration 590: norm(K_i - K_ip1) = 0.000095
Iteration 591: norm(K_i - K_ip1) = 0.000919
Iteration 592: norm(K_i - K_ip1) = 0.000364
Iteration 593: norm(K_i - K_ip1) = 0.000195
Iteration 594: norm(K_i - K_ip1) = 0.000175
Iteration 595: norm(K_i - K_ip1) = 0.000275
Iteration 596: norm(K_i - K_ip1) = 0.000205
Iteration 597: norm(K_i - K_ip1) = 0.000217
Iteration 598: norm(K_i - K_ip1) = 0.000118
Iteration 599: norm(K_i - K_ip1) = 0.000699
Iteration 600: norm(K_i - K_ip1) = 0.000620
Iteration 601: norm(K_i - K_ip1) = 0.000144
Iteration 602: norm(K_i - K_ip1) = 0.000450
Iteration 603: norm(K_i - K_ip1) = 0.000313
Iteration 604: norm(K_i - K_ip1) = 0.000043
Iteration 605: norm(K_i - K_ip1) = 0.000049
Iteration 606: norm(K_i - K_ip1) = 0.000318
Iteration 607: norm(K_i - K_ip1) = 0.000035
Iteration 608: norm(K_i - K_ip1) = 0.000470
Iteration 609: norm(K_i - K_ip1) = 0.000549
Iteration 610: norm(K_i - K_ip1) = 0.000569
Iteration 611: norm(K_i - K_ip1) = 0.000493
```

```
Iteration 612: norm(K_i - K_ip1) = 0.000037
Iteration 613: norm(K_i - K_ip1) = 0.000166
Iteration 614: norm(K_i - K_ip1) = 0.000091
Iteration 615: norm(K_i - K_ip1) = 0.000119
Iteration 616: norm(K_i - K_ip1) = 0.000291
Iteration 617: norm(K_i - K_ip1) = 0.000018
Iteration 618: norm(K_i - K_ip1) = 0.000481
Iteration 619: norm(K_i - K_ip1) = 0.000160
Iteration 620: norm(K_i - K_ip1) = 0.000188
Iteration 621: norm(K_i - K_ip1) = 0.000457
Iteration 622: norm(K_i - K_ip1) = 0.000393
Iteration 623: norm(K_i - K_ip1) = 0.000239
Iteration 624: norm(K_i - K_ip1) = 0.000197
Iteration 625: norm(K_i - K_ip1) = 0.000186
Iteration 626: norm(K_i - K_ip1) = 0.000273
Iteration 627: norm(K_i - K_ip1) = 0.000022
Iteration 628: norm(K_i - K_ip1) = 0.000321
Iteration 629: norm(K_i - K_ip1) = 0.000106
Iteration 630: norm(K_i - K_ip1) = 0.000516
Iteration 631: norm(K_i - K_ip1) = 0.000268
Iteration 632: norm(K_i - K_ip1) = 0.000053
Iteration 633: norm(K_i - K_ip1) = 0.000080
Iteration 634: norm(K_i - K_ip1) = 0.000320
Iteration 635: norm(K_i - K_ip1) = 0.000013
Iteration 636: norm(K_i - K_ip1) = 0.000939
Iteration 637: norm(K_i - K_ip1) = 0.002112
Iteration 638: norm(K_i - K_ip1) = 0.000790
Iteration 639: norm(K_i - K_ip1) = 0.000634
Iteration 640: norm(K_i - K_ip1) = 0.000273
Iteration 641: norm(K_i - K_ip1) = 0.000257
Iteration 642: norm(K_i - K_ip1) = 0.000591
Iteration 643: norm(K_i - K_ip1) = 0.000235
Iteration 644: norm(K_i - K_ip1) = 0.000045
Iteration 645: norm(K_i - K_ip1) = 0.000034
Iteration 646: norm(K_i - K_ip1) = 0.000051
Iteration 647: norm(K_i - K_ip1) = 0.000105
Iteration 648: norm(K_i - K_ip1) = 0.000052
Iteration 649: norm(K_i - K_ip1) = 0.000018
Iteration 650: norm(K_i - K_ip1) = 0.000496
Iteration 651: norm(K_i - K_ip1) = 0.000753
Iteration 652: norm(K_i - K_ip1) = 0.000425
Iteration 653: norm(K_i - K_ip1) = 0.000395
Iteration 654: norm(K_i - K_ip1) = 0.000267
Iteration 655: norm(K_i - K_ip1) = 0.000099
Iteration 656: norm(K_i - K_ip1) = 0.000283
Iteration 657: norm(K_i - K_ip1) = 0.000223
Iteration 658: norm(K_i - K_ip1) = 0.000349
Iteration 659: norm(K_i - K_ip1) = 0.000886
Iteration 660: norm(K_i - K_ip1) = 0.000296
Iteration 661: norm(K_i - K_ip1) = 0.000052
Iteration 662: norm(K_i - K_ip1) = 0.000377
Iteration 663: norm(K_i - K_ip1) = 0.000255
Iteration 664: norm(K_i - K_ip1) = 0.000186
Iteration 665: norm(K_i - K_ip1) = 0.000078
```

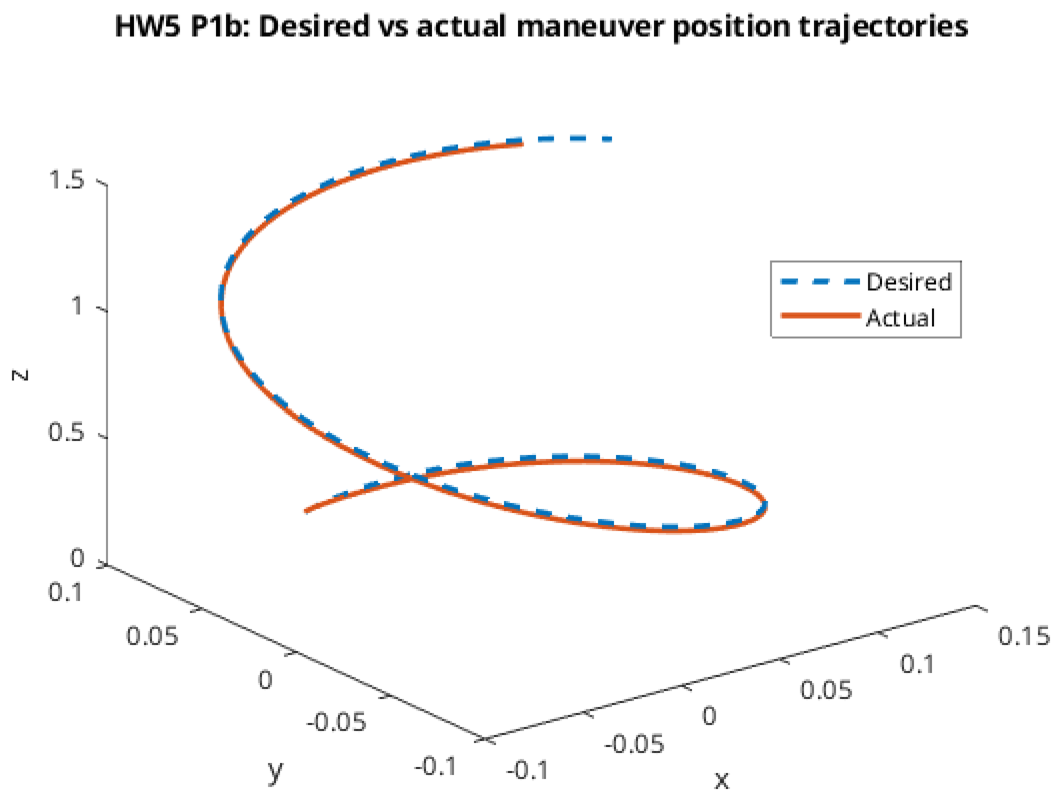
```
Iteration 666: norm(K_i - K_ip1) = 0.000134
Iteration 667: norm(K_i - K_ip1) = 0.000128
Iteration 668: norm(K_i - K_ip1) = 0.000186
Iteration 669: norm(K_i - K_ip1) = 0.000023
Iteration 670: norm(K_i - K_ip1) = 0.000044
Iteration 671: norm(K_i - K_ip1) = 0.000265
Iteration 672: norm(K_i - K_ip1) = 0.000583
Iteration 673: norm(K_i - K_ip1) = 0.000207
Iteration 674: norm(K_i - K_ip1) = 0.000528
Iteration 675: norm(K_i - K_ip1) = 0.000210
Iteration 676: norm(K_i - K_ip1) = 0.000028
Iteration 677: norm(K_i - K_ip1) = 0.000165
Iteration 678: norm(K_i - K_ip1) = 0.000460
Iteration 679: norm(K_i - K_ip1) = 0.000431
Iteration 680: norm(K_i - K_ip1) = 0.000132
Iteration 681: norm(K_i - K_ip1) = 0.000448
Iteration 682: norm(K_i - K_ip1) = 0.000081
Iteration 683: norm(K_i - K_ip1) = 0.000993
Iteration 684: norm(K_i - K_ip1) = 0.000809
Iteration 685: norm(K_i - K_ip1) = 0.000232
Iteration 686: norm(K_i - K_ip1) = 0.000151
Iteration 687: norm(K_i - K_ip1) = 0.000142
Iteration 688: norm(K_i - K_ip1) = 0.000322
Iteration 689: norm(K_i - K_ip1) = 0.000650
Iteration 690: norm(K_i - K_ip1) = 0.000440
Iteration 691: norm(K_i - K_ip1) = 0.000331
Iteration 692: norm(K_i - K_ip1) = 0.000649
Iteration 693: norm(K_i - K_ip1) = 0.000391
Iteration 694: norm(K_i - K_ip1) = 0.000206
Iteration 695: norm(K_i - K_ip1) = 0.000175
Iteration 696: norm(K_i - K_ip1) = 0.000063
Iteration 697: norm(K_i - K_ip1) = 0.000644
Iteration 698: norm(K_i - K_ip1) = 0.000175
Iteration 699: norm(K_i - K_ip1) = 0.000291
Iteration 700: norm(K_i - K_ip1) = 0.000166
Iteration 701: norm(K_i - K_ip1) = 0.000296
Iteration 702: norm(K_i - K_ip1) = 0.000344
Iteration 703: norm(K_i - K_ip1) = 0.000805
Iteration 704: norm(K_i - K_ip1) = 0.000420
Iteration 705: norm(K_i - K_ip1) = 0.000140
Iteration 706: norm(K_i - K_ip1) = 0.000004
Iteration 707: norm(K_i - K_ip1) = 0.000359
Iteration 708: norm(K_i - K_ip1) = 0.000848
Iteration 709: norm(K_i - K_ip1) = 0.000530
Iteration 710: norm(K_i - K_ip1) = 0.000183
Iteration 711: norm(K_i - K_ip1) = 0.000462
Iteration 712: norm(K_i - K_ip1) = 0.000252
Iteration 713: norm(K_i - K_ip1) = 0.000310
Iteration 714: norm(K_i - K_ip1) = 0.000442
Iteration 715: norm(K_i - K_ip1) = 0.000672
Iteration 716: norm(K_i - K_ip1) = 0.000237
Iteration 717: norm(K_i - K_ip1) = 0.000124
Iteration 718: norm(K_i - K_ip1) = 0.000360
Iteration 719: norm(K_i - K_ip1) = 0.000296
```

```
Iteration 720: norm(K_i - K_ip1) = 0.000540
Iteration 721: norm(K_i - K_ip1) = 0.001458
Iteration 722: norm(K_i - K_ip1) = 0.000617
Iteration 723: norm(K_i - K_ip1) = 0.000372
Iteration 724: norm(K_i - K_ip1) = 0.000062
Iteration 725: norm(K_i - K_ip1) = 0.000196
Iteration 726: norm(K_i - K_ip1) = 0.000635
Iteration 727: norm(K_i - K_ip1) = 0.000445
Iteration 728: norm(K_i - K_ip1) = 0.000077
Iteration 729: norm(K_i - K_ip1) = 0.000298
Iteration 730: norm(K_i - K_ip1) = 0.000359
Iteration 731: norm(K_i - K_ip1) = 0.000233
Iteration 732: norm(K_i - K_ip1) = 0.000166
Iteration 733: norm(K_i - K_ip1) = 0.000237
Iteration 734: norm(K_i - K_ip1) = 0.000125
Iteration 735: norm(K_i - K_ip1) = 0.000271
Iteration 736: norm(K_i - K_ip1) = 0.000285
Iteration 737: norm(K_i - K_ip1) = 0.000533
Iteration 738: norm(K_i - K_ip1) = 0.000246
Iteration 739: norm(K_i - K_ip1) = 0.000044
Iteration 740: norm(K_i - K_ip1) = 0.000058
Iteration 741: norm(K_i - K_ip1) = 0.000158
Iteration 742: norm(K_i - K_ip1) = 0.000558
Iteration 743: norm(K_i - K_ip1) = 0.000708
Iteration 744: norm(K_i - K_ip1) = 0.000180
Iteration 745: norm(K_i - K_ip1) = 0.000346
Iteration 746: norm(K_i - K_ip1) = 0.000380
Iteration 747: norm(K_i - K_ip1) = 0.000323
Iteration 748: norm(K_i - K_ip1) = 0.000248
Iteration 749: norm(K_i - K_ip1) = 0.000059
Iteration 750: norm(K_i - K_ip1) = 0.000047
Iteration 751: norm(K_i - K_ip1) = 0.000290
Iteration 752: norm(K_i - K_ip1) = 0.000072
Iteration 753: norm(K_i - K_ip1) = 0.000289
Iteration 754: norm(K_i - K_ip1) = 0.000724
Iteration 755: norm(K_i - K_ip1) = 0.000585
Iteration 756: norm(K_i - K_ip1) = 0.000663
Iteration 757: norm(K_i - K_ip1) = 0.000122
Iteration 758: norm(K_i - K_ip1) = 0.000336
Iteration 759: norm(K_i - K_ip1) = 0.000474
Iteration 760: norm(K_i - K_ip1) = 0.000292
Iteration 761: norm(K_i - K_ip1) = 0.000467
Iteration 762: norm(K_i - K_ip1) = 0.000250
Iteration 763: norm(K_i - K_ip1) = 0.000387
Iteration 764: norm(K_i - K_ip1) = 0.001259
Iteration 765: norm(K_i - K_ip1) = 0.000760
Iteration 766: norm(K_i - K_ip1) = 0.000472
Iteration 767: norm(K_i - K_ip1) = 0.000247
Iteration 768: norm(K_i - K_ip1) = 0.000054
Iteration 769: norm(K_i - K_ip1) = 0.000364
Iteration 770: norm(K_i - K_ip1) = 0.000740
Iteration 771: norm(K_i - K_ip1) = 0.000793
Iteration 772: norm(K_i - K_ip1) = 0.000713
Iteration 773: norm(K_i - K_ip1) = 0.003833
```

```
Iteration 774: norm(K_i - K_ip1) = 0.001880
Iteration 775: norm(K_i - K_ip1) = 0.000643
Iteration 776: norm(K_i - K_ip1) = 0.000400
Iteration 777: norm(K_i - K_ip1) = 0.000375
Iteration 778: norm(K_i - K_ip1) = 0.000056
Iteration 779: norm(K_i - K_ip1) = 0.000326
Iteration 780: norm(K_i - K_ip1) = 0.000195
Iteration 781: norm(K_i - K_ip1) = 0.000967
Iteration 782: norm(K_i - K_ip1) = 0.000270
Iteration 783: norm(K_i - K_ip1) = 0.000195
Iteration 784: norm(K_i - K_ip1) = 0.000654
Iteration 785: norm(K_i - K_ip1) = 0.000498
Iteration 786: norm(K_i - K_ip1) = 0.000161
Iteration 787: norm(K_i - K_ip1) = 0.000125
Iteration 788: norm(K_i - K_ip1) = 0.000117
Iteration 789: norm(K_i - K_ip1) = 0.000496
Iteration 790: norm(K_i - K_ip1) = 0.000104
Iteration 791: norm(K_i - K_ip1) = 0.000855
Iteration 792: norm(K_i - K_ip1) = 0.000683
Iteration 793: norm(K_i - K_ip1) = 0.000081
Iteration 794: norm(K_i - K_ip1) = 0.000431
Iteration 795: norm(K_i - K_ip1) = 0.000501
Iteration 796: norm(K_i - K_ip1) = 0.000233
Iteration 797: norm(K_i - K_ip1) = 0.000174
Iteration 798: norm(K_i - K_ip1) = 0.000192
Iteration 799: norm(K_i - K_ip1) = 0.000106
Iteration 800: norm(K_i - K_ip1) = 0.000547
Iteration 801: norm(K_i - K_ip1) = 0.000224
Iteration 802: norm(K_i - K_ip1) = 0.000075
Iteration 803: norm(K_i - K_ip1) = 0.000366
Iteration 804: norm(K_i - K_ip1) = 0.000640
Iteration 805: norm(K_i - K_ip1) = 0.000261
Iteration 806: norm(K_i - K_ip1) = 0.000297
Iteration 807: norm(K_i - K_ip1) = 0.000352
Iteration 808: norm(K_i - K_ip1) = 0.000199
Iteration 809: norm(K_i - K_ip1) = 0.000594
Iteration 810: norm(K_i - K_ip1) = 0.000744
Iteration 811: norm(K_i - K_ip1) = 0.000534
Iteration 812: norm(K_i - K_ip1) = 0.000242
Iteration 813: norm(K_i - K_ip1) = 0.000102
Iteration 814: norm(K_i - K_ip1) = 0.000462
Iteration 815: norm(K_i - K_ip1) = 0.000338
Iteration 816: norm(K_i - K_ip1) = 0.000437
Iteration 817: norm(K_i - K_ip1) = 0.000390
Iteration 818: norm(K_i - K_ip1) = 0.001205
Iteration 819: norm(K_i - K_ip1) = 0.000571
Iteration 820: norm(K_i - K_ip1) = 0.000158
Iteration 821: norm(K_i - K_ip1) = 0.000132
Iteration 822: norm(K_i - K_ip1) = 0.000242
Iteration 823: norm(K_i - K_ip1) = 0.000097
Iteration 824: norm(K_i - K_ip1) = 0.000006
Iteration 825: norm(K_i - K_ip1) = 0.000205
Iteration 826: norm(K_i - K_ip1) = 0.000534
Iteration 827: norm(K_i - K_ip1) = 0.000352
```

```
Iteration 828: norm(K_i - K_ip1) = 0.000077
Iteration 829: norm(K_i - K_ip1) = 0.000121
Iteration 830: norm(K_i - K_ip1) = 0.000502
Iteration 831: norm(K_i - K_ip1) = 0.000409
Iteration 832: norm(K_i - K_ip1) = 0.000205
Iteration 833: norm(K_i - K_ip1) = 0.000402
Iteration 834: norm(K_i - K_ip1) = 0.000237
Iteration 835: norm(K_i - K_ip1) = 0.000417
Iteration 836: norm(K_i - K_ip1) = 0.000037
Iteration 837: norm(K_i - K_ip1) = 0.000516
Iteration 838: norm(K_i - K_ip1) = 0.000594
Iteration 839: norm(K_i - K_ip1) = 0.000157
Iteration 840: norm(K_i - K_ip1) = 0.000284
Iteration 841: norm(K_i - K_ip1) = 0.000531
Iteration 842: norm(K_i - K_ip1) = 0.000440
Iteration 843: norm(K_i - K_ip1) = 0.000319
Iteration 844: norm(K_i - K_ip1) = 0.000376
Iteration 845: norm(K_i - K_ip1) = 0.000157
Iteration 846: norm(K_i - K_ip1) = 0.000259
Iteration 847: norm(K_i - K_ip1) = 0.000312
Iteration 848: norm(K_i - K_ip1) = 0.000474
Iteration 849: norm(K_i - K_ip1) = 0.000024
Iteration 850: norm(K_i - K_ip1) = 0.000037
Iteration 851: norm(K_i - K_ip1) = 0.000068
Iteration 852: norm(K_i - K_ip1) = 0.000551
Iteration 853: norm(K_i - K_ip1) = 0.000038
Iteration 854: norm(K_i - K_ip1) = 0.001136
Iteration 855: norm(K_i - K_ip1) = 0.000733
Iteration 856: norm(K_i - K_ip1) = 0.000069
Iteration 857: norm(K_i - K_ip1) = 0.000006
Iteration 858: norm(K_i - K_ip1) = 0.000328
Iteration 859: norm(K_i - K_ip1) = 0.001256
Iteration 860: norm(K_i - K_ip1) = 0.000786
Iteration 861: norm(K_i - K_ip1) = 0.000569
Iteration 862: norm(K_i - K_ip1) = 0.001174
Iteration 863: norm(K_i - K_ip1) = 0.000385
Iteration 864: norm(K_i - K_ip1) = 0.000551
Iteration 865: norm(K_i - K_ip1) = 0.000042
Iteration 866: norm(K_i - K_ip1) = 0.000203
Iteration 867: norm(K_i - K_ip1) = 0.000497
Iteration 868: norm(K_i - K_ip1) = 0.000244
Iteration 869: norm(K_i - K_ip1) = 0.000027
Iteration 870: norm(K_i - K_ip1) = 0.000324
Iteration 871: norm(K_i - K_ip1) = 0.000638
Iteration 872: norm(K_i - K_ip1) = 0.000632
Iteration 873: norm(K_i - K_ip1) = 0.000658
Iteration 874: norm(K_i - K_ip1) = 0.000308
Iteration 875: norm(K_i - K_ip1) = 0.000147
Iteration 876: norm(K_i - K_ip1) = 0.000067
Iteration 877: norm(K_i - K_ip1) = 0.000083
Iteration 878: norm(K_i - K_ip1) = 0.000575
Iteration 879: norm(K_i - K_ip1) = 0.000295
Iteration 880: norm(K_i - K_ip1) = 0.000190
Iteration 881: norm(K_i - K_ip1) = 0.000326
```

```
Iteration 882: norm(K_i - K_ip1) = 0.000738
Iteration 883: norm(K_i - K_ip1) = 0.000603
Iteration 884: norm(K_i - K_ip1) = 0.000052
Iteration 885: norm(K_i - K_ip1) = 0.000096
Iteration 886: norm(K_i - K_ip1) = 0.000146
Iteration 887: norm(K_i - K_ip1) = 0.000203
Iteration 888: norm(K_i - K_ip1) = 0.000529
Iteration 889: norm(K_i - K_ip1) = 0.000547
Iteration 890: norm(K_i - K_ip1) = 0.000087
Iteration 891: norm(K_i - K_ip1) = 0.000091
Iteration 892: norm(K_i - K_ip1) = 0.000322
Iteration 893: norm(K_i - K_ip1) = 0.000315
Iteration 894: norm(K_i - K_ip1) = 0.000014
Iteration 895: norm(K_i - K_ip1) = 0.000029
Iteration 896: norm(K_i - K_ip1) = 0.000001
Converged after 896 iterations
```



Published with MATLAB® R2023b

Problem 2

We are given the step response data of a stable system, which looks like this:

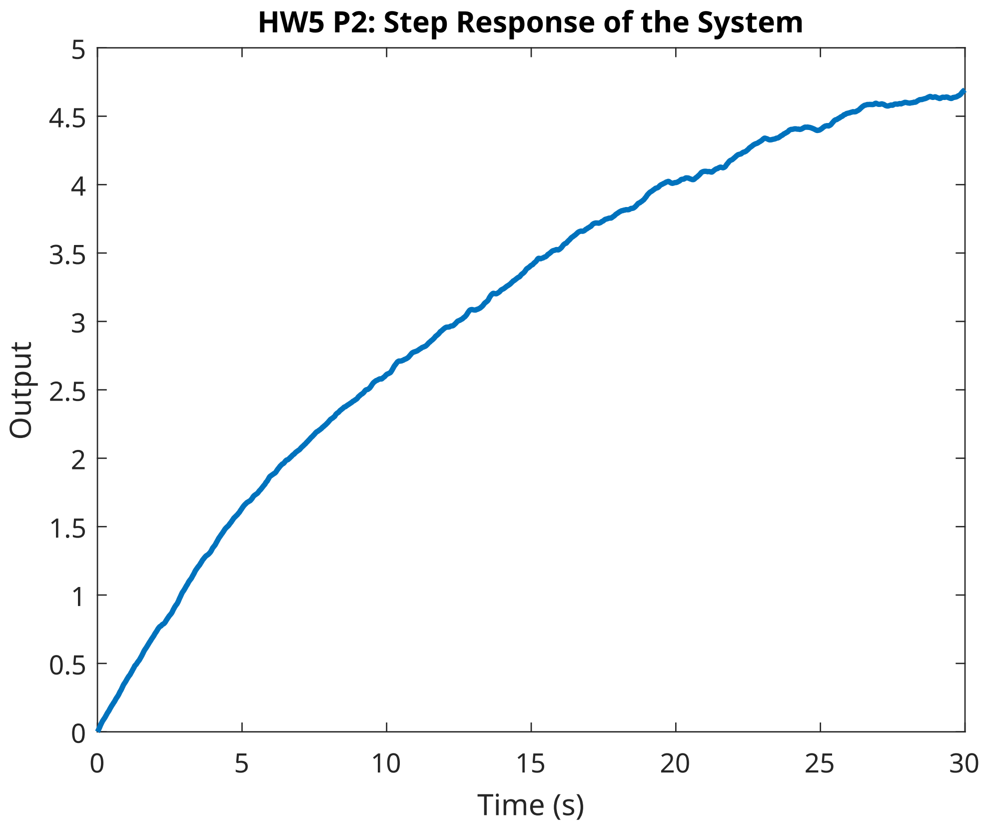


Figure 1: Stable system step response

We can observe that the system output response appears to exponentially approaches a steady state value. Hence we can assume that the system is a linear first order system. The ODE for a first order linear system is given by:

$$\tau \dot{x} + x(t) = Ku(t)$$

Where τ is the time constant, K is the gain, $u(t)$ is the system input, and $x(t)$ is the system output. Given the limited data, the system does not actually approach the steady state within the time span of the data so we cannot directly use the steady state value to find K and τ.

We can instead use the least squares method to fit the data to the first order system model. The first order system model can be rewritten as:

$$\begin{aligned} \tau \dot{x} + x(t) &= Ku(t) \\ \dot{x}\tau - u(t)K &= -x(t) \quad (1) \end{aligned}$$

Where we have the data for $x(t)$, $\dot{x}(t)$ can be computed by using the finite difference method $\dot{x}(t) = \frac{x(t+\Delta t)-x(t)}{\Delta t}$ and $u(t)$ is the step input

$$u(t) = \begin{cases} 0 & t < 0 \\ 1 & t \geq 0 \end{cases}$$

Since we only have data for $t \geq 0$, we can assume that $u(t) = 1$.

Using the finite difference method works when the data is not noisy, but we can see from the data that it is quite noisy and does not follow a smooth exponential curve. Hence, starting from the initial time t_0 we can integrate both sides of equation (1) till each timestep t of the data to get:

$$\int_{t_0}^{t} (\dot{x}\tau - u(t)K)dt = -\int_{t_0}^{t} x(t)dt$$
$$x(t)\tau - tK = -\int_{t_0}^{t} x(t)dt$$

Now we can stack all the data points from t_0 to t_N where t_N is the final time to form a matrix equation of the form:

$$\underbrace{\begin{bmatrix} x(t_0) & -t_0 \\ x(t_1) & -t_1 \\ \vdots & \vdots \\ x(t_N) & -t_N \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} \tau \\ K \end{bmatrix}}_{\theta} = \underbrace{\begin{bmatrix} -\int_{t_0}^{t_0} x(t)dt \\ -\int_{t_0}^{t_1} x(t)dt \\ \vdots \\ -\int_{t_0}^{t_N} x(t)dt \end{bmatrix}}_{b}$$

Where b can be constructed by sequentially adding the values of $x(t)$ to value of the previous timestep and multiplying by the time difference for a zero order hold integration.

We can find the least squares solution to the above equation, which gives us the following τ and K:

$$\tau = 14.790, \quad K = 5.412$$

We can now use the values of τ and K to construct the transfer function of our first order ODE

$$\tau\dot{x} + x(t) = Ku(t)$$
$$\tau sX(s) + X(s) = KU(s)$$
$$X(s)(\tau s + 1) = KU(s)$$
$$\implies \frac{X(s)}{U(s)} = \frac{K}{\tau s + 1}$$
$$\implies G(s) = \frac{K}{\tau s + 1}$$

Plotting the step response of the transfer function with the output data we get the following:

The model's step response matches the given data and we can be assured that the system has been identified correctly

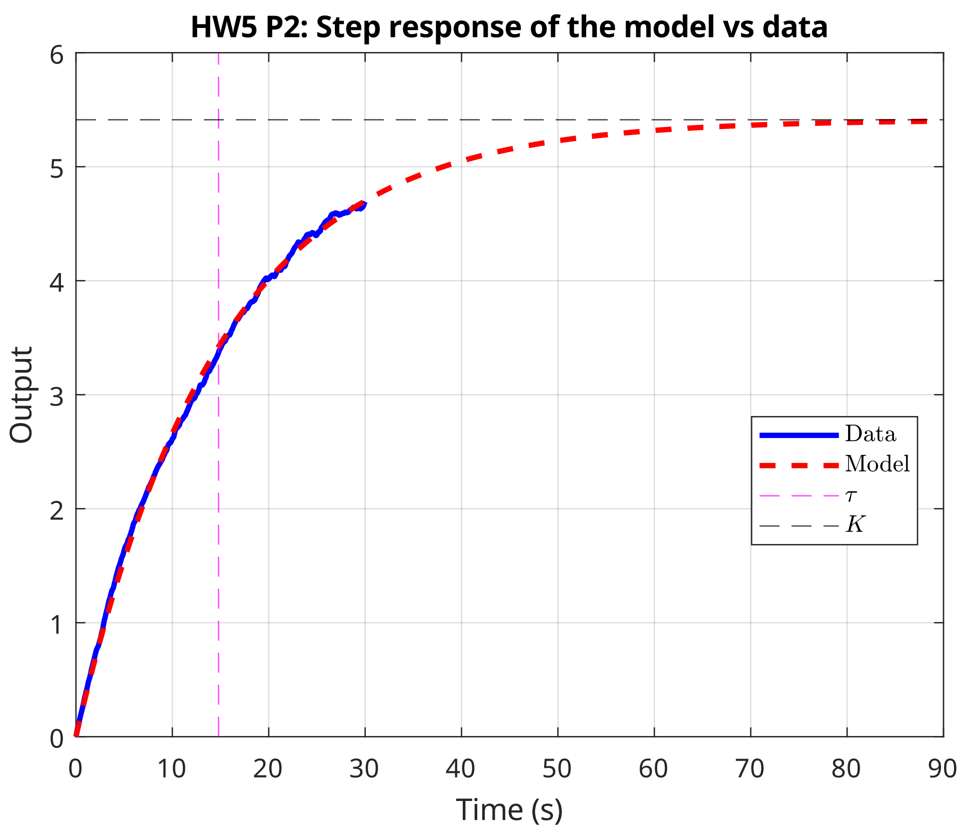


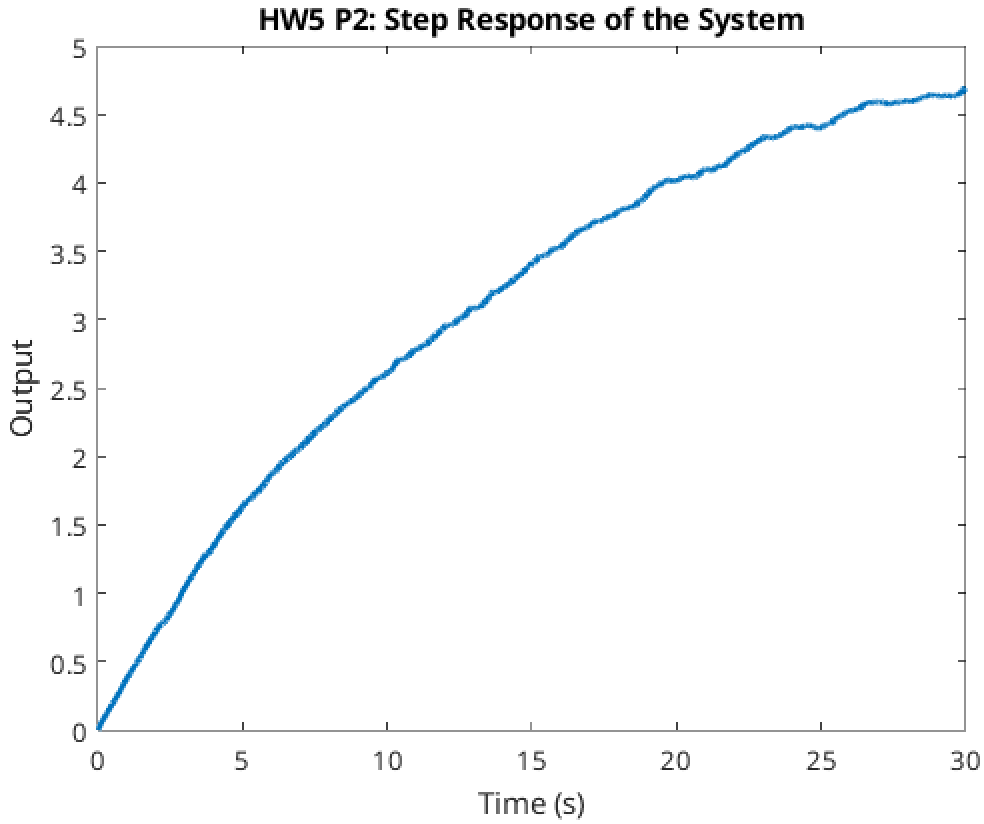
Figure 2: Step response of model vs data

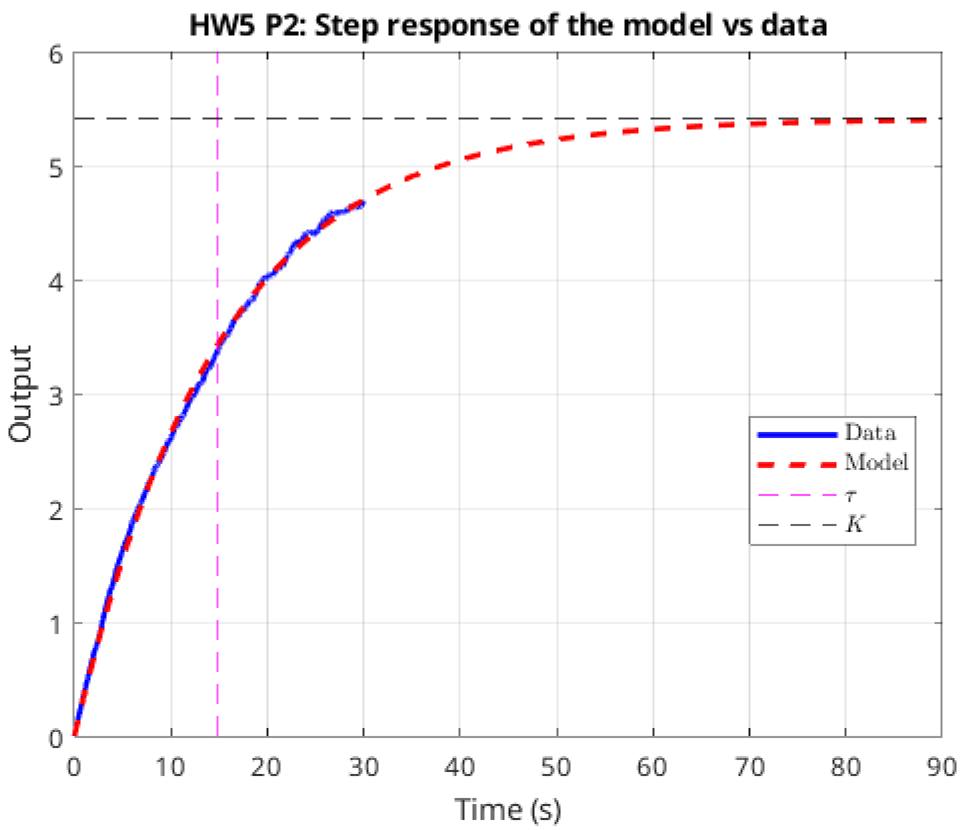
Table of Contents

ME599 HW5 Problem 2

```
clc; clear; close all;

load("System_step_response.mat");

% plot output data
fig = figure;
plot(time, output, 'LineWidth', 2);
xlabel('Time (s)')
ylabel('Output');
title('HW5 P2: Step Response of the System')
saveas(fig, 'figs/hw5p2_step.svg');

% construct A and b matrices
A = [output -time];
b = -cumtrapz(time, output);

% solve for theta
theta = A \ b;
tau = theta(1);
K = theta(2);
fprintf('tau = %.3f\nK = %.3f\n', tau, K);

% plot step response
t_settle = 6*tau; % should be 4x, but do 6x to show K
t_s = time(2) - time(1);
time_tf = 0:t_s:t_settle;
model  = tf(K, [tau 1]);
[model_out, ~] = step(model, time_tf);

fig = figure;
plot(time, output, 'b', 'LineWidth', 2, 'DisplayName', 'Data');
hold on;
plot(time_tf, model_out, 'r--', 'LineWidth', 2, 'DisplayName', 'Model');
xline(tau, 'm--', 'DisplayName', "$\tau$");
yline(K, 'k--', 'DisplayName', "$K$");
xlabel('Time (s)')
ylabel('Output');
title('HW5 P2: Step response of the model vs data')
legend('Location', 'best', 'Interpreter', 'latex');
grid on;
saveas(fig, 'figs/hw5p2_step_model.svg');
```

```
tau = 14.790
K = 5.412
```


HW5 P2: Step Response of the System
Output
Time (s)
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
5
0
5
10
15
20
25
30


HW5 P2: Step response of the model vs data
Output
Time (s)
0
1
2
3
4
5
6
0
10
20
30
40
50
60
70
80
90
Data
Model
τ
K

Published with MATLAB® R2023b