Adaptive Control - Assignment 3

Stochastic Self Tuning Regulators

Student: Murtaza Asaadi, mergenelos@gmail.com
Lecturer: Prof. Baqeri, peyman.bk@gmail.com

May, 2025

Contents

List of Figures

List of Program Codes

1 Questions 1 to 7

The Matlab implementation of the original system's transfer function is given in Code 1. Its step response, shown in Figure 1, indicates that the original system is unstable.

```
%% Setting up
clc, clear, close all

%% Required calculations
s = tf('s');
G = 3*(0.4*s+1)*(s+0.8)/((3*s+1)^2*(s+1));
% Settling time
step_info = stepinfo(G);
settling_time = step_info.SettlingTime;
% Sample time
Ts = floor(settling_time)*0.1;

%%  colored noise coefficients
C = [1,0.6,0.4];

%% Original transfer function
G_s = 3*(0.4*s+1)*(s+0.8)/((3*s+1)^2*(s-1));

%% Discrete transfer function
G_discrete = c2d(G_s,Ts,'zoh');
[B,A] = tfdata(G_discrete,'v');
B(1) = [];
```

Code 1: Basic system implementation

To enable the calculation of the system's settling time, the unstable pole was mirrored and the system was discretized, as implemented in the code shown in Code 1. The step response of the resulting, modified system is presented in Figure 2. The system rise time is 9.81 and settling time is 16.71 seconds.

The code for this section is available at assignment3/SSTR/SSTR_0.m.

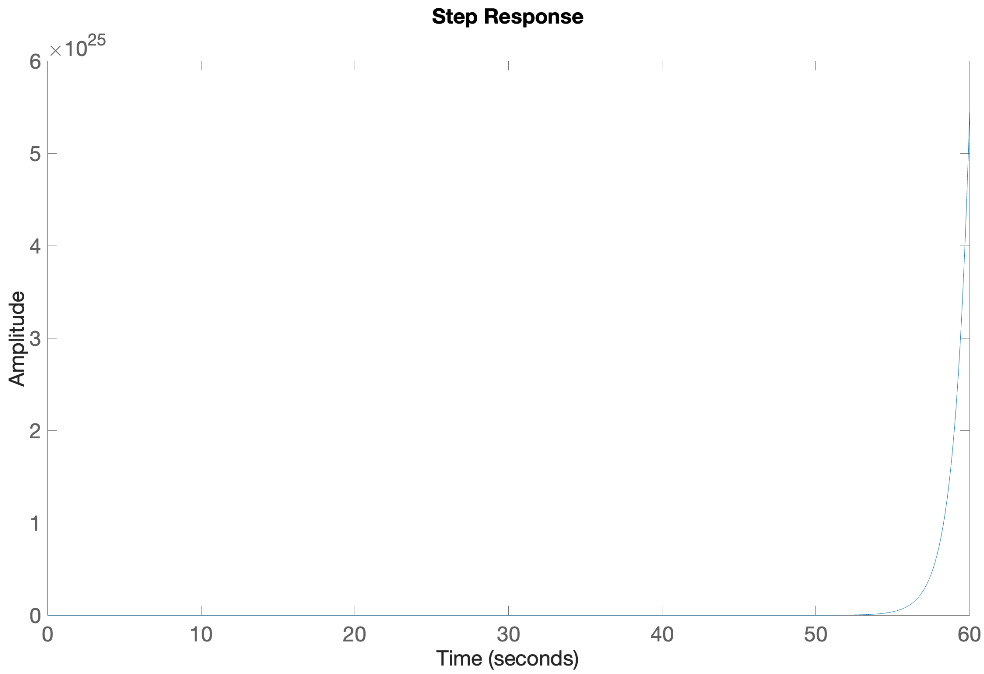

Figure 1: Original system step response

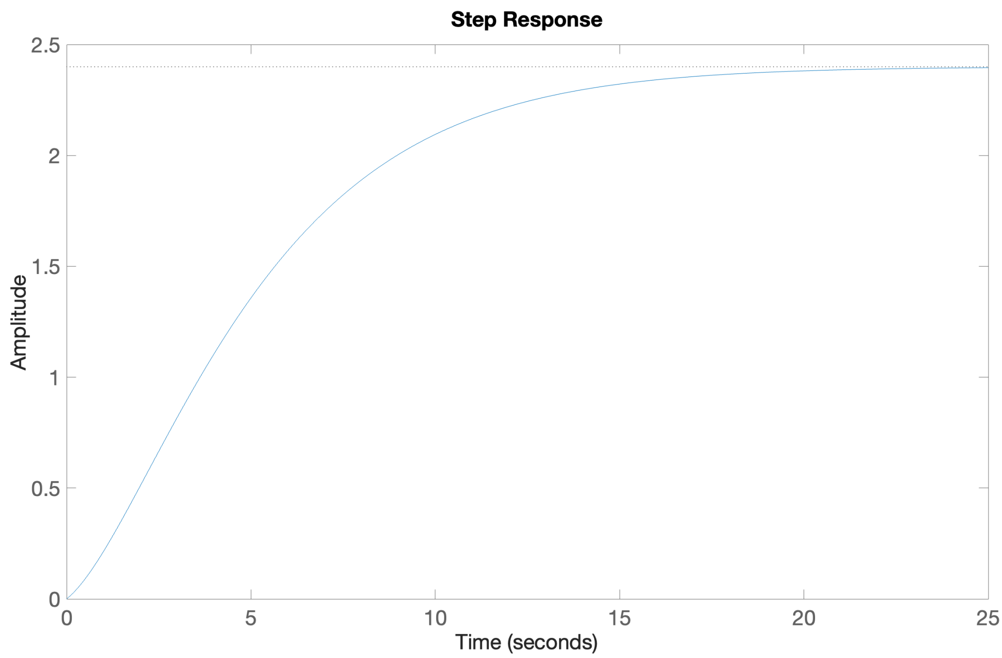

Figure 2: Discrete stable system step response

1.1 Question 1

First, we implement the open-loop system. To close the loop, we apply a minimum variance controller using Code 2. As shown in Figure 3, the system remains stable and, when actuated, returns to its equilibrium point at 0. As shown in Figure 4, cummulative loss of the system is bounded.

```
run('SSTR_0.m');

%% Diophantine equation
[F,G] = diophantine(A,C,1);

%% Open loop system
G_ol = minreal(G_discrete*tf(A,conv(B,F),Ts));

%% Closed loop system
G_cl = feedback(G_ol,1);
```

Code 2: Minimum variance controller

The code for this section is available at assignment3/SSTR/SSTR_1.m.

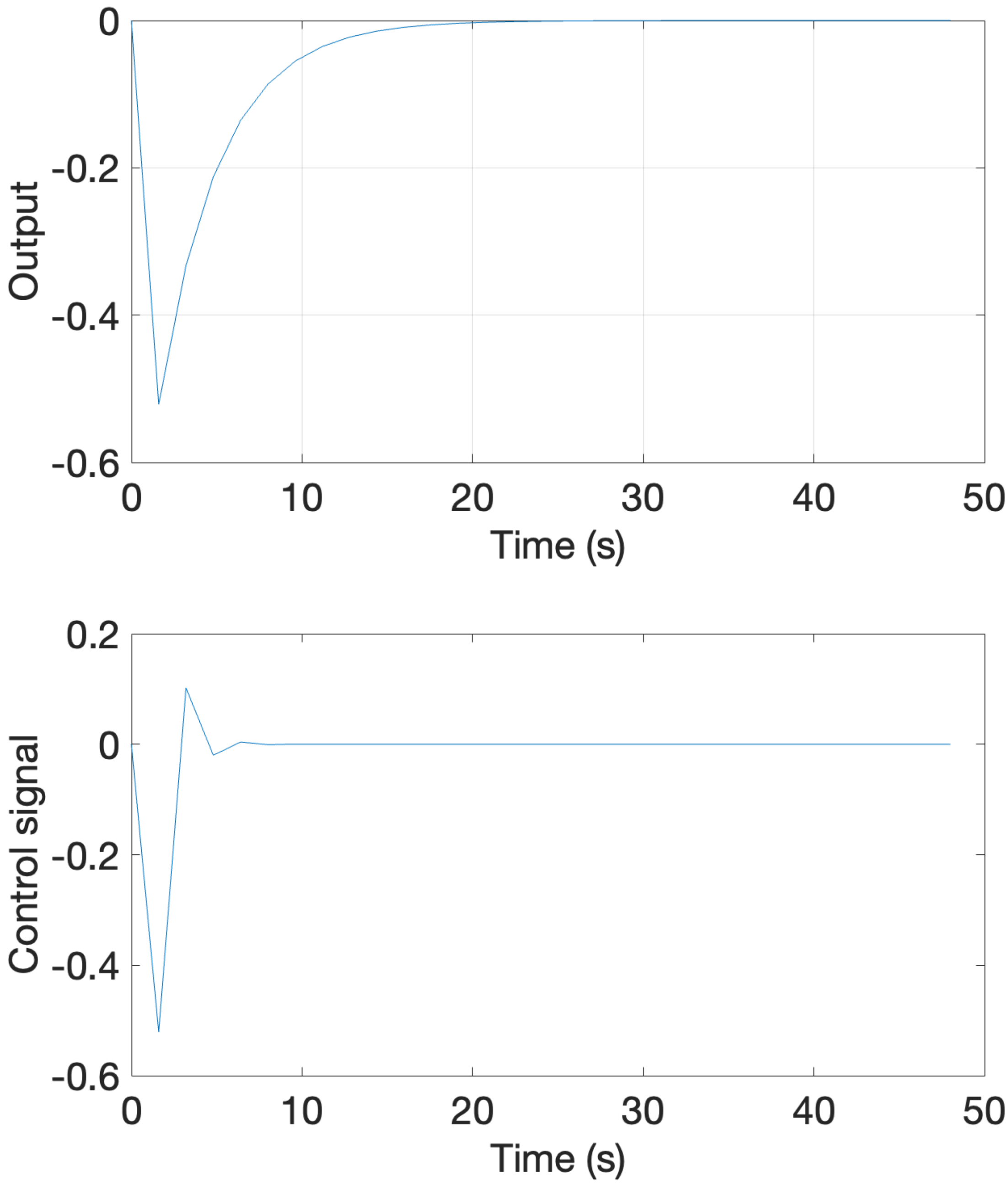

Figure 3: Output and control of minimum variance controller

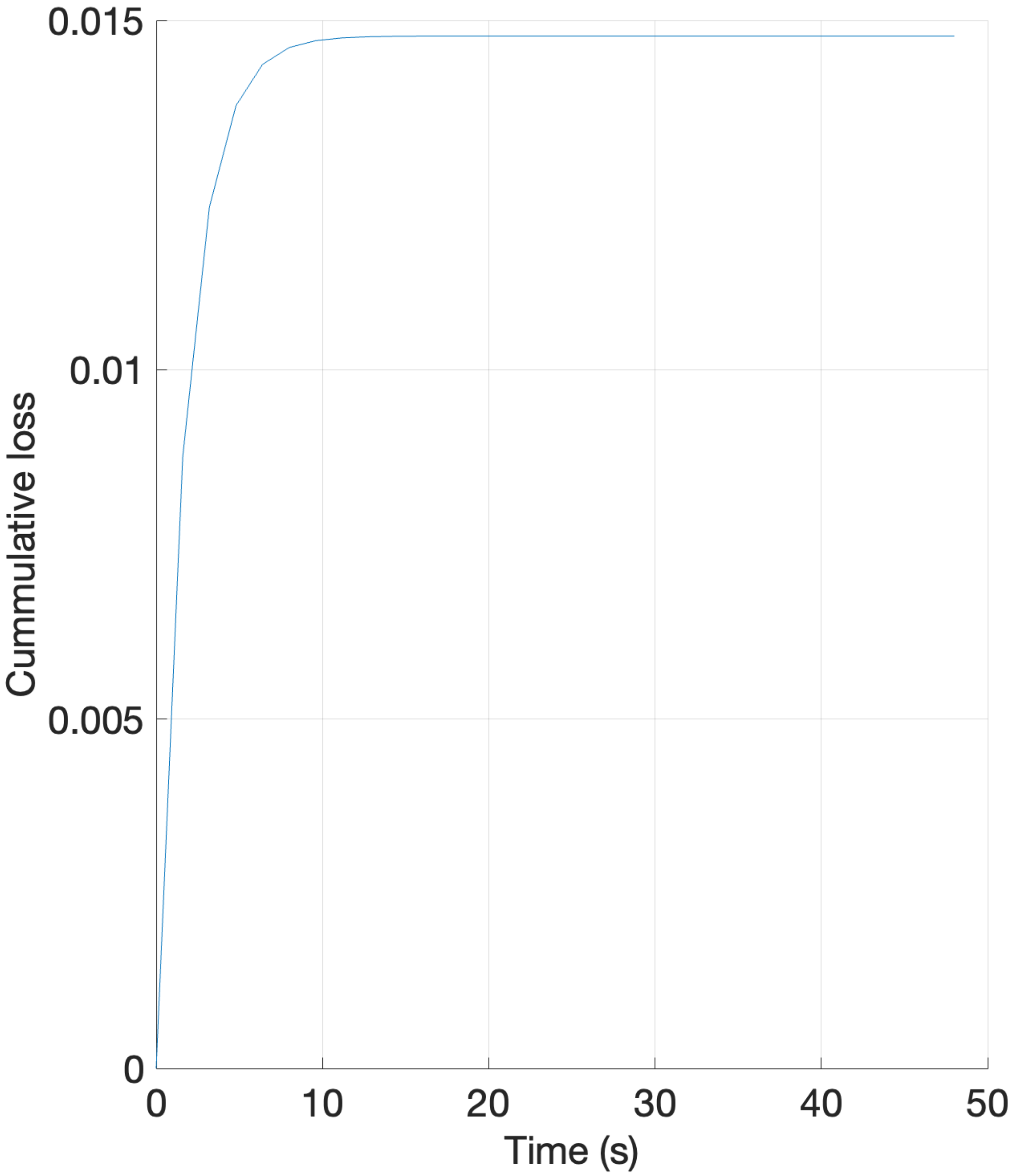

Figure 4: Cummulative loss of Minimum variance controller

1.2 Question 2

The required delay in the transfer function is implemented as shown in Code 3. Rest of the code remains as presented in Code 2. As shown in Figure 5, the implemented system is stable and returns to its equilibrium point at 0. And in Figure 6, cummulative loss of the system is bounded.

```
%% Transfer function
G_s = 3*(0.4*s+1)*(s+0.8)/((3*s+1)^2*(s-1))*tf(1,[1,0,0]);
```

Code 3: Minimum variance controller with delay

The code for this section is available at assignment3/SSTR/SSTR_2.m.

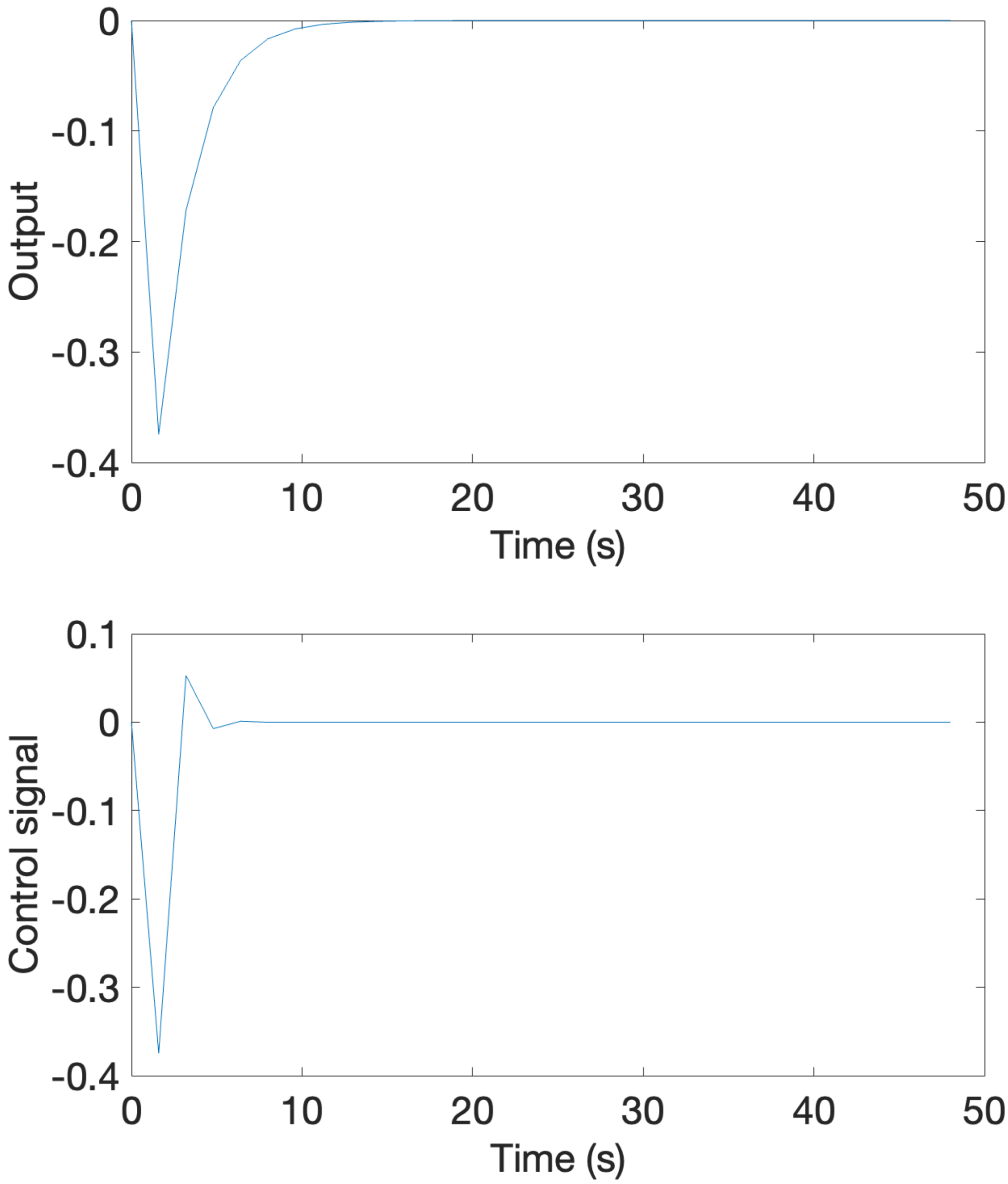

Figure 5: Minimum variance controller with delay, output and control

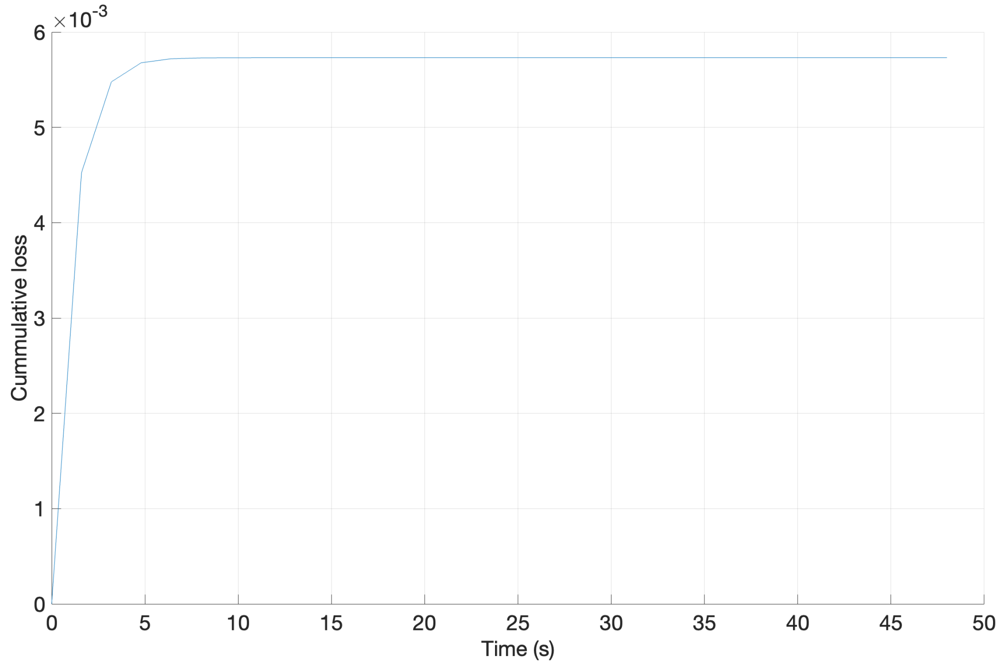

Figure 6: Minimum variance controller with delay, cummulative loss

1.3 Question 3

We implement the required indirect adaptive controller as shown in Code 4. As shown in Figure 7, the system remains stable and in Figure 6, cummulative loss of the system is bounded.

```
%% Identification parameters
na = 2; nb = 2;
theta = zeros(na+nb,1);
P = 1000*eye(na+nb);
lambda = 0.98;
%% Input and output data
N = 50;
u = randn(1,N);
y = zeros(1,N);
y_est = zeros(1,N);
for k = max(na,nb)+1:N
y(k) = -A(2)*y(k-1) - A(3)*y(k-2) + ...
B(2)*u(k-1) + B(3)*u(k-2);
end
%% RLS
for k = max(na,nb)+1:N
phi = [-y(k-1); -y(k-2); u(k-1); u(k-2)];
y_hat = theta'*phi;
e = y(k) - y_hat;
K = (P*phi)/(lambda + phi'*P*phi);
theta = theta + K*e;
P = (P - K*phi'*P)/lambda;

y_est(k) = y_hat;
THETA(:,k) = theta;
end

%% Recover A and B
A_id = [1; theta(1:na)];
B_id = theta(na+1:end);

%% Adaptive controller
[F,G] = diophantine(A_id, C, 1);

%% Open-loop and closed-loop systems
G_z_hat = tf(B_id', A_id', Ts);
G_ol = minreal(G_z_hat * tf(A_id', conv(B_id', F), Ts));
G_cl = feedback(G_ol, 1);
```

Code 4: Indirect adaptive implementation of minimum variance controller

The code for this section is available at assignment3/SSTR/SSTR_3.m.

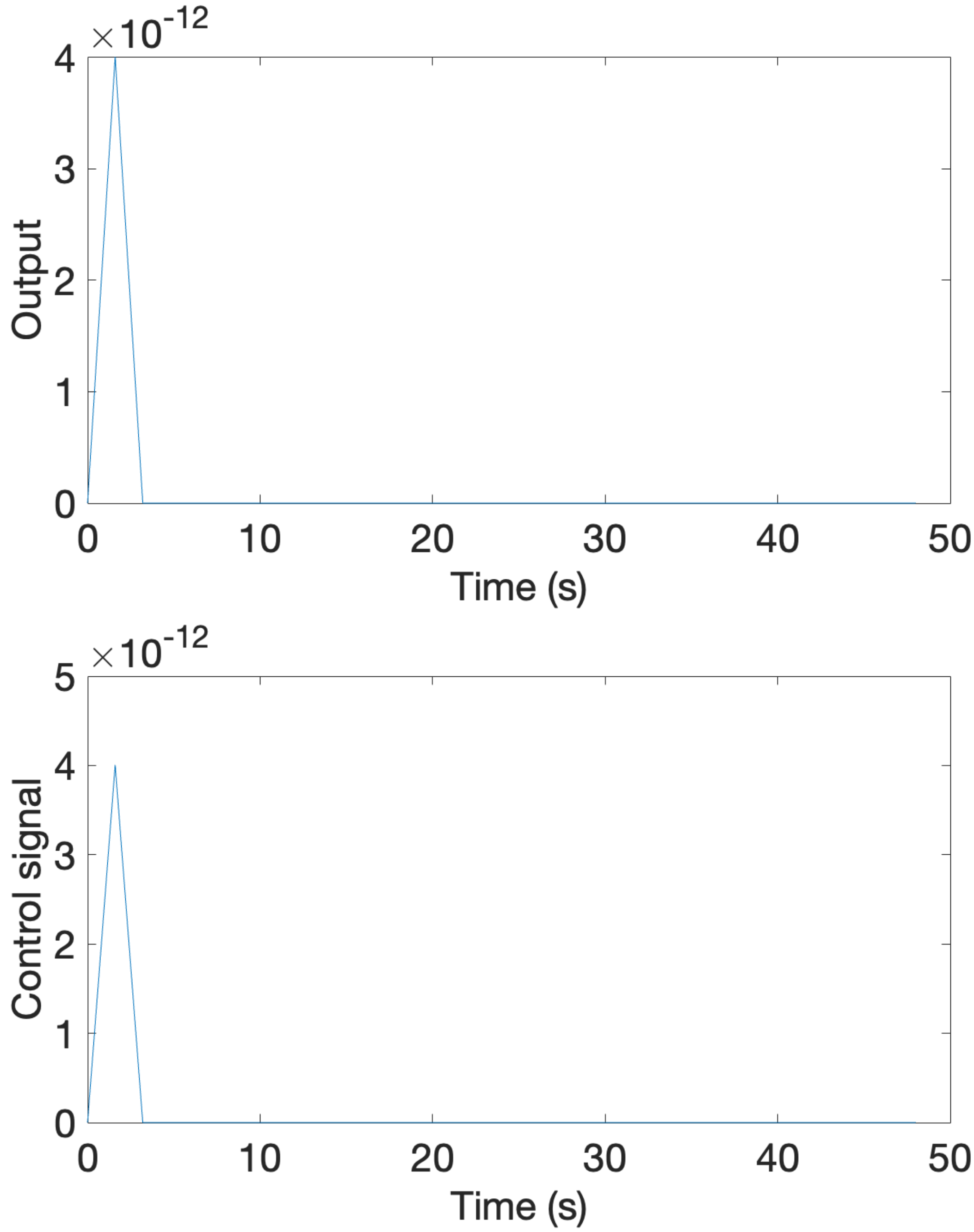

Figure 7: Output and control of indirect adaptive implementation of minimum variance controller

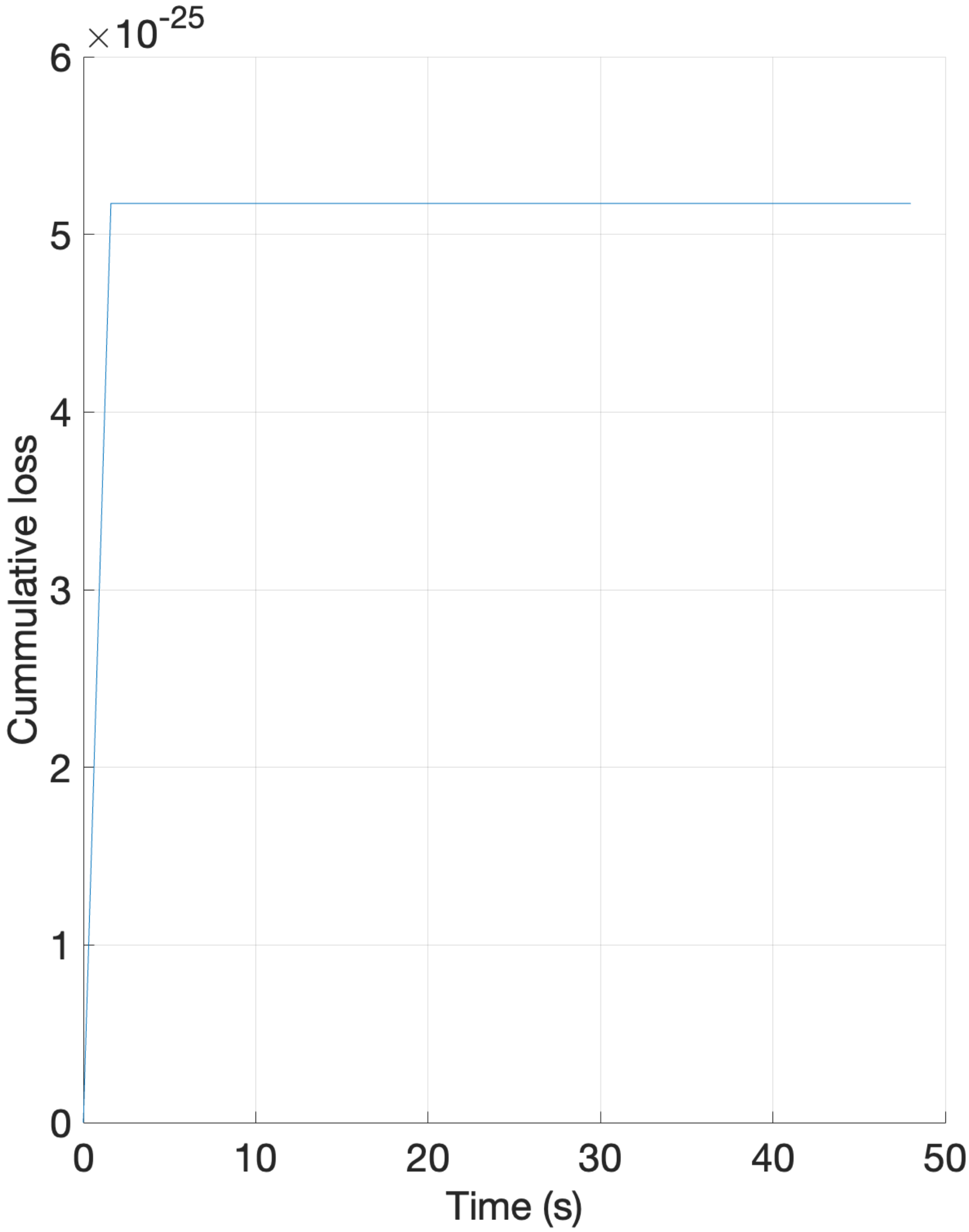

Figure 8: Cummulative loss of indirect adaptive implementation of minimum variance controller

1.4 Question 4

Direct adaptive controller codebase is presented in Code 5. As shown in Figure 9, the system is stable and in Figure 10, cummulative loss of the system is bounded. Figure 11 presents the parameters estimation plot of direct adaptive controller.

```
run('SSTR_0.m');

%%  Solve Diophantine equation
[F,G] = diophantine(A,C,1);

%%  Closed loop system
G_ol = minreal(G_discrete*tf(A,conv(B,F),Ts));

%%  closed loop system
G_cl = feedback(G_ol,1);
[B_true,A_true] = tfdata(G_cl,'v');

%% Simulation settings
na = length(A_true)-1; nb = length(B_true)-1;
N = 31;
theta = zeros(nb + na, 1);
gamma = 0.01;

u = ones(1,N);
y = zeros(1,N);
y_ref = ones(1,N);
phi = zeros(nb + na, 1);

%% Simulation loop
for k = 4:N
e = y_ref(k) - y(k);
phi = [-y(k-1); -y(k-2); -y(k-3); u(k-1); u(k-2); u(k-3)];
u(k) = theta' * phi;
y(k) = -A_true(2)*y(k-1) - A_true(3)*y(k-2) - A_true(4)*y(k-3) + ...
B_true(2)*u(k-1) + B_true(3)*u(k-2) + B_true(4)*u(k-3);

theta = theta - gamma * e * phi;
THETA(:,k) = theta;
end
```

Code 5: Direct adaptive implementation of minimum variance controller

The code for this section is available at assignment3/SSTR/SSTR_4.m.

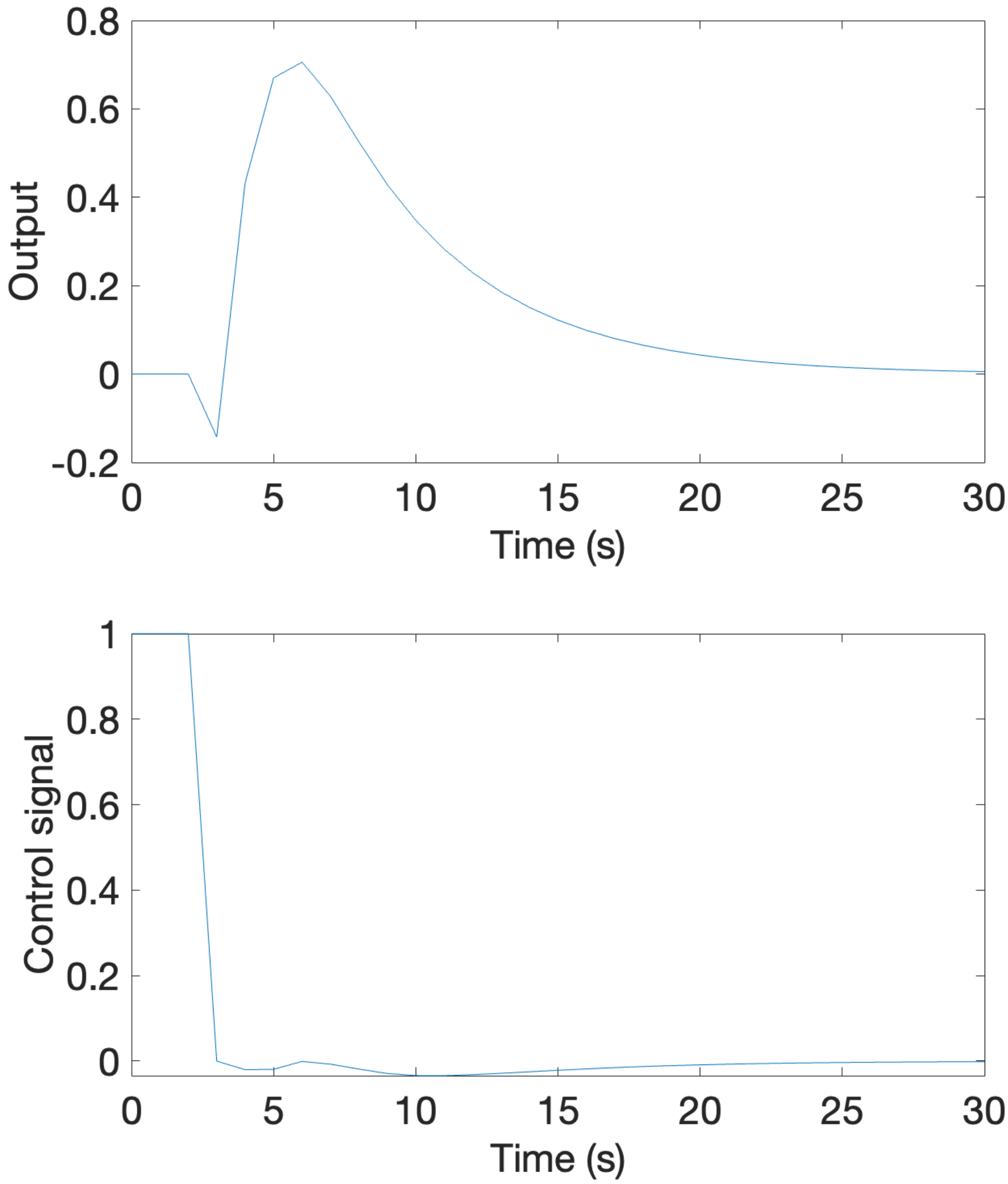

Figure 9: output and control of direct adaptive implementation of minimum variance controller

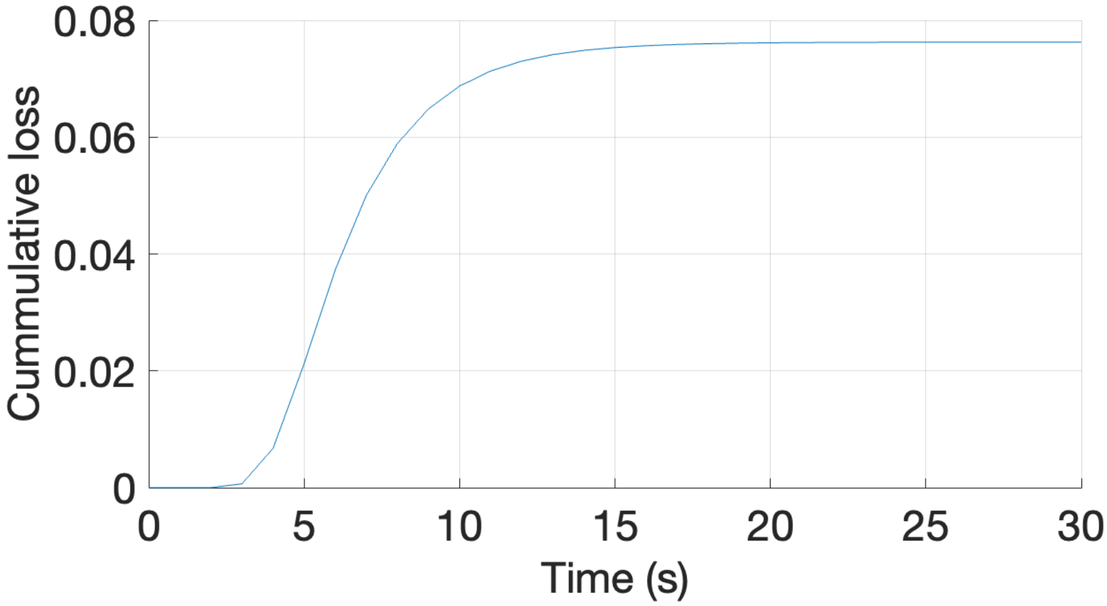

Figure 10: Cummulative loss of direct adaptive implementation of minimum variance controller

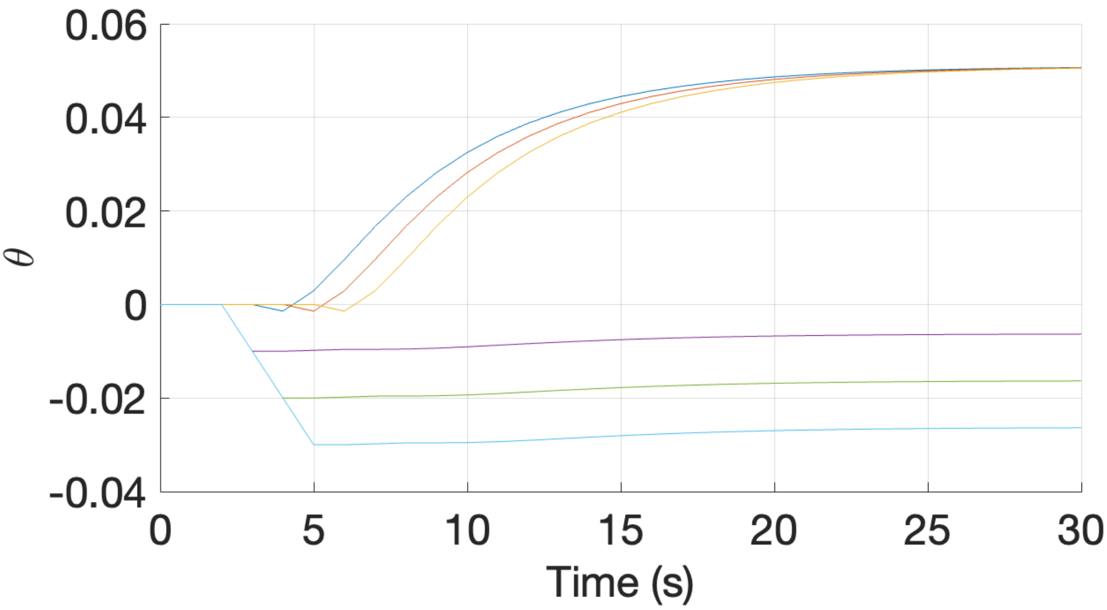

Figure 11: Estimated parameters of direct adaptive implementation of minimum variance controller

1.5 Question 5

We implement the open-loop system. To close the loop, we apply a moving average controller using Code 6. As shown in Figure 12, the system remains stable and, when actuated, returns to its equilibrium point at 0. As shown in Figure 13, cummulative loss of the system is bounded.

```
run('SSTR_0.m');

%%  Solve Diophantine equation for d = 2
d = 2;   % d0=1
[F, G] = diophantine(A, C, d);

%%  Define the MA controller
MA_controller = tf(conv(F,A), 1, Ts);

%%  Define the open loop system
G_ol = minreal(G_discrete * MA_controller);

%%  Closed-loop system
G_cl = feedback(G_ol, 1);
```

Code 6: Moving average controller

The code for this section is available at assignment3/SSTR/SSTR_5.m.

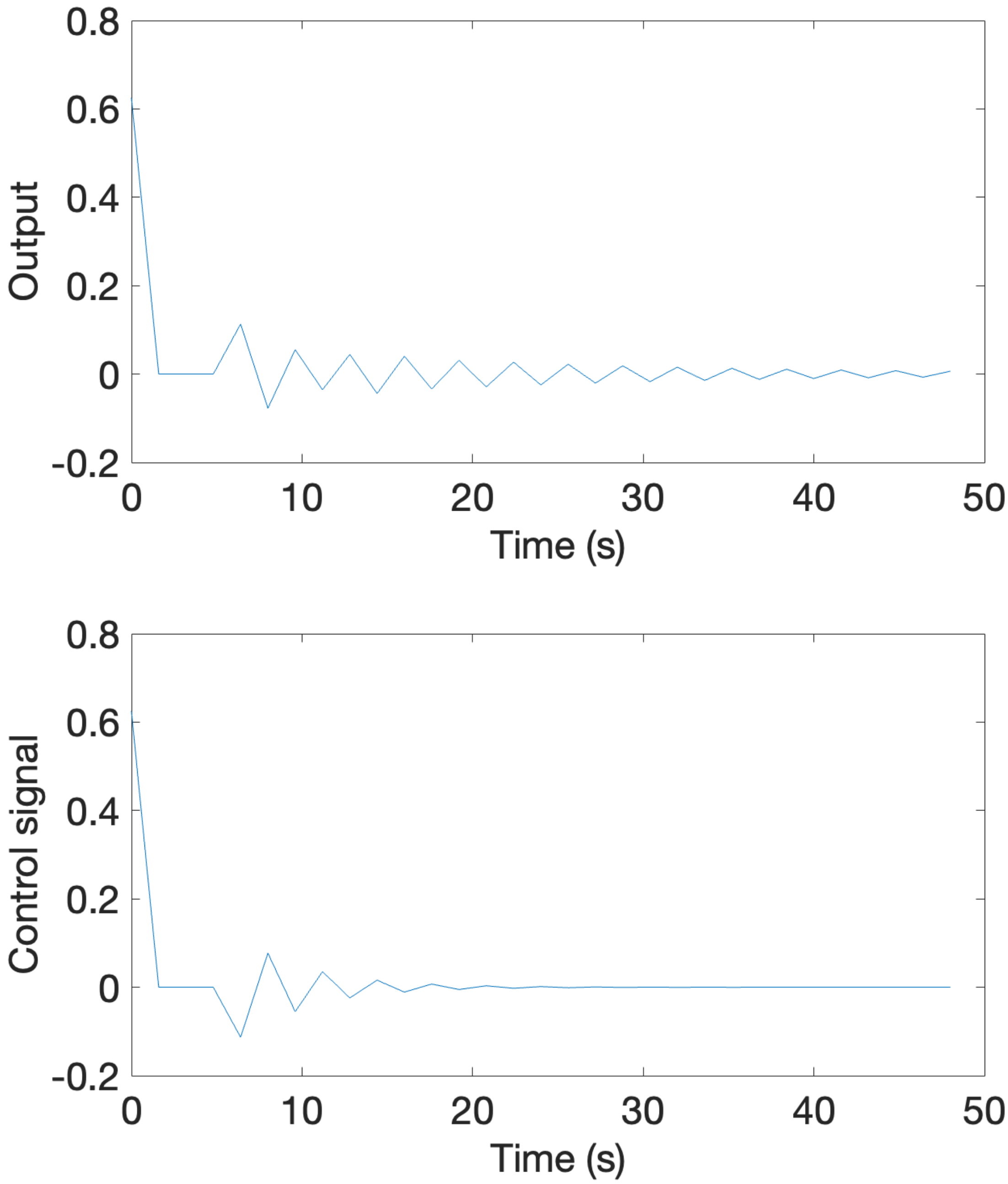

Figure 12: Output and control of moving average controller

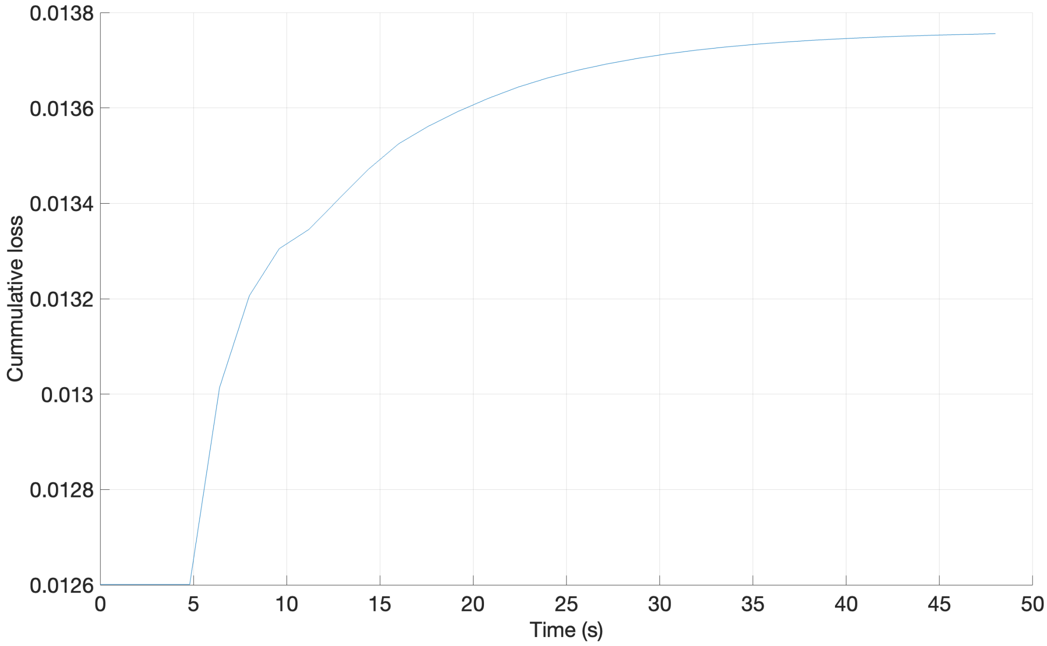

Figure 13: Cummulative loss of moving average controller

1.6 Question 6

We mirror zeros of the transfer function then we apply a moving average controller using Code 7. As shown in Figure 14, the system remains stable and, when actuated, returns to its equilibrium point at 0. As shown in Figure 15, cummulative loss of the system is bounded.

```
%%  Original unstable system
G_s = 3*(0.4*s-1)*(s-0.8)/((3*s+1)^2*(s-1));

%%  Discretize the system
G_z = c2d(G_s, Ts, 'zoh');
[B, A] = tfdata(G_z, 'v');
B(1) = [];

%%  Solve Diophantine equation for d = 2
d = 2;   % d0=1
[F, G] = diophantine(A, C, d);

%%  Define the MA controller
MA_controller = tf(conv(F,A), B, Ts);

%%  Define the open loop system
G_ol = minreal(G_z * MA_controller);

%%  Closed-loop system
G_cl = feedback(G_ol, 1);

```

Code 7: Moving average controller with mirrored zeros

The code for this section is available at assignment3/SSTR/SSTR_6.m.

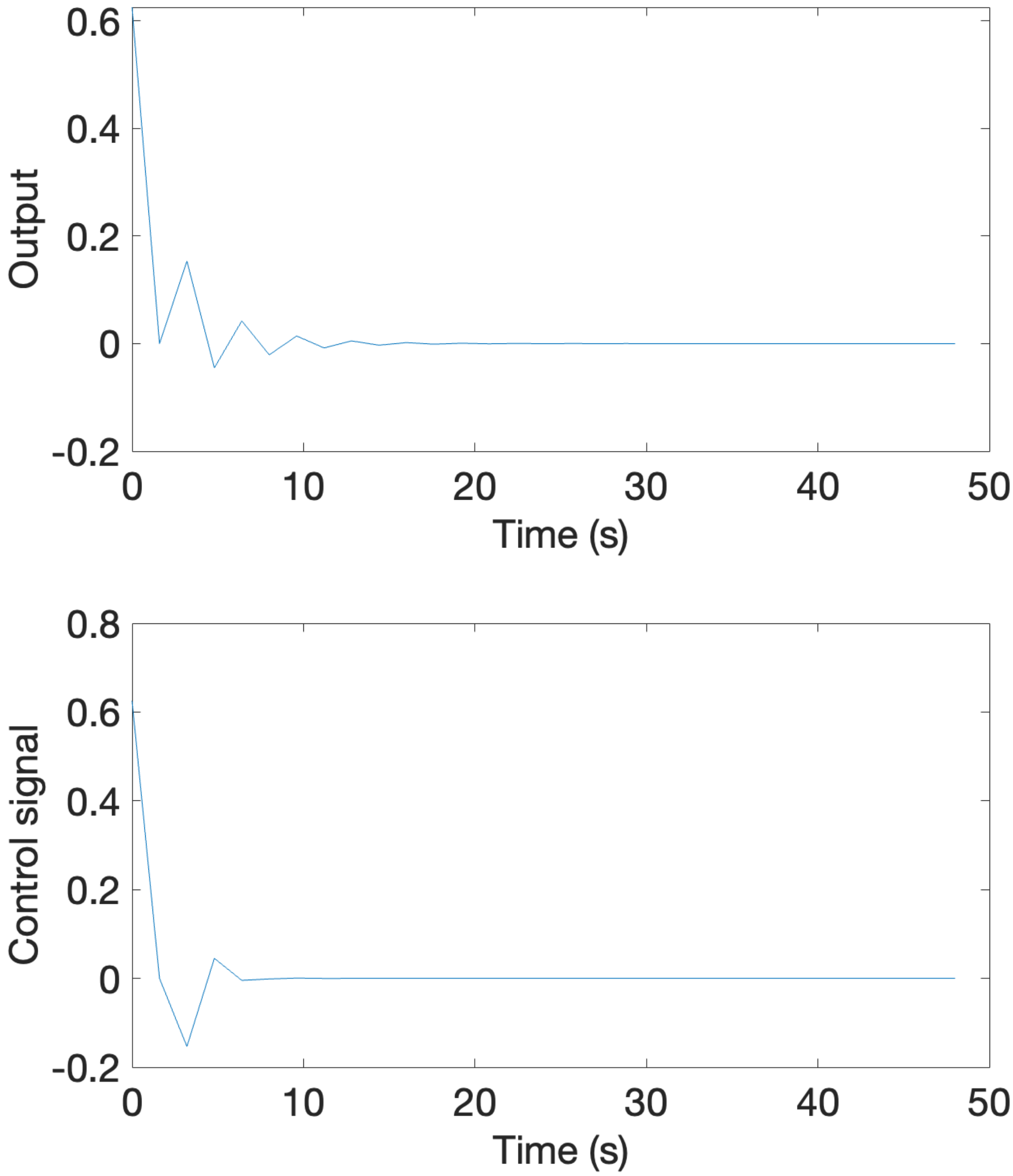

Figure 14: Output and control of moving average controller with mirrored zeros

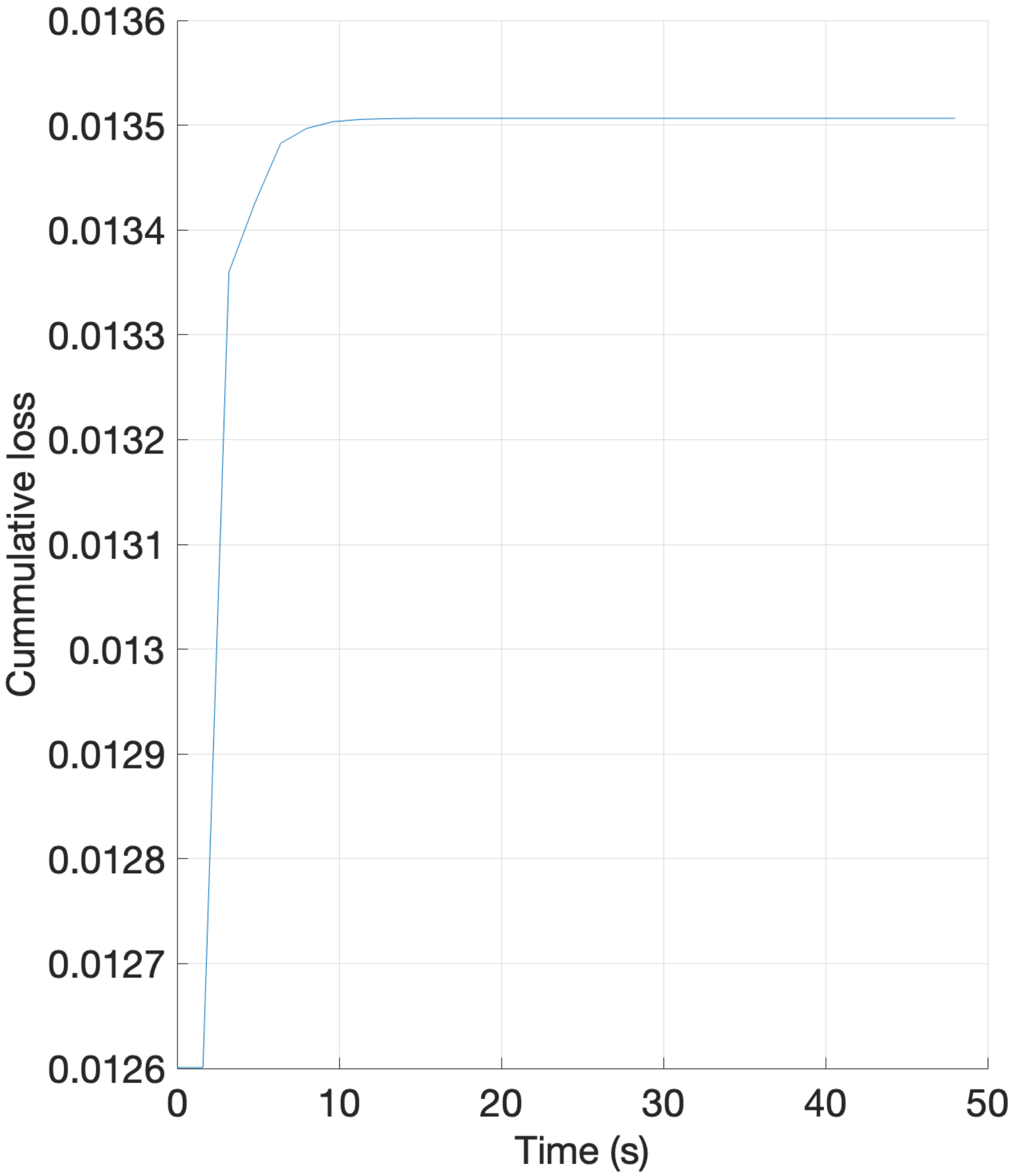

Figure 15: Cummulative loss of moving average controller with mirrored zeros

1.7 Question 7

?? implements the direct adaptive moving average controller. As shown in Figure 16, the system remains stable and, when actuated, returns to its equilibrium point at 0. As shown in Figure 17, cummulative loss of the system is bounded. Estimated parameters are presented in Figure 18. The code for this section is available at assignment3/SSTR/SSTR_7.m.

```
%% Solve Diophantine equation
d = 2;   % d0=1
[F, G] = diophantine(A, C, d);

%%  Define the MA controller
MA_controller = tf(conv(F,A), 1, Ts);

%% Open loop
G_ol = minreal(G_discrete * MA_controller);

%%  Closed loop
G_cl = feedback(G_ol, 1);
A_true = [-2.5373   -0.1906    0.0497    0.1744    1.0341];
B_true = [-2.5373   -0.1906    0.0497    0.1744    0.0341];

%% Simulation settings
na = length(A_true)-1; nb = length(B_true)-1;
N = 31;
theta = zeros(nb + na, 1);
gamma = 0.01;
u = ones(1,N);
y = zeros(1,N);
y_ref = ones(1,N);
phi = zeros(nb + na, 1);

for k = 5:N
e = y_ref(k) - y(k);
phi = [-y(k-1); -y(k-2); -y(k-3); -y(k-4); ...
u(k-1); u(k-2); u(k-3); u(k-4)];
u(k) = theta' * phi;
y(k) = -A_true(2)*y(k-1) - A_true(3)*y(k-2) - A_true(4)*y(k-3) + ...
B_true(2)*u(k-1) + B_true(3)*u(k-2) + B_true(4)*u(k-3);

theta = theta - gamma * e * phi;
THETA(:,k) = theta;
end
```

Code 8: Poles & zeros with a mirrored zero

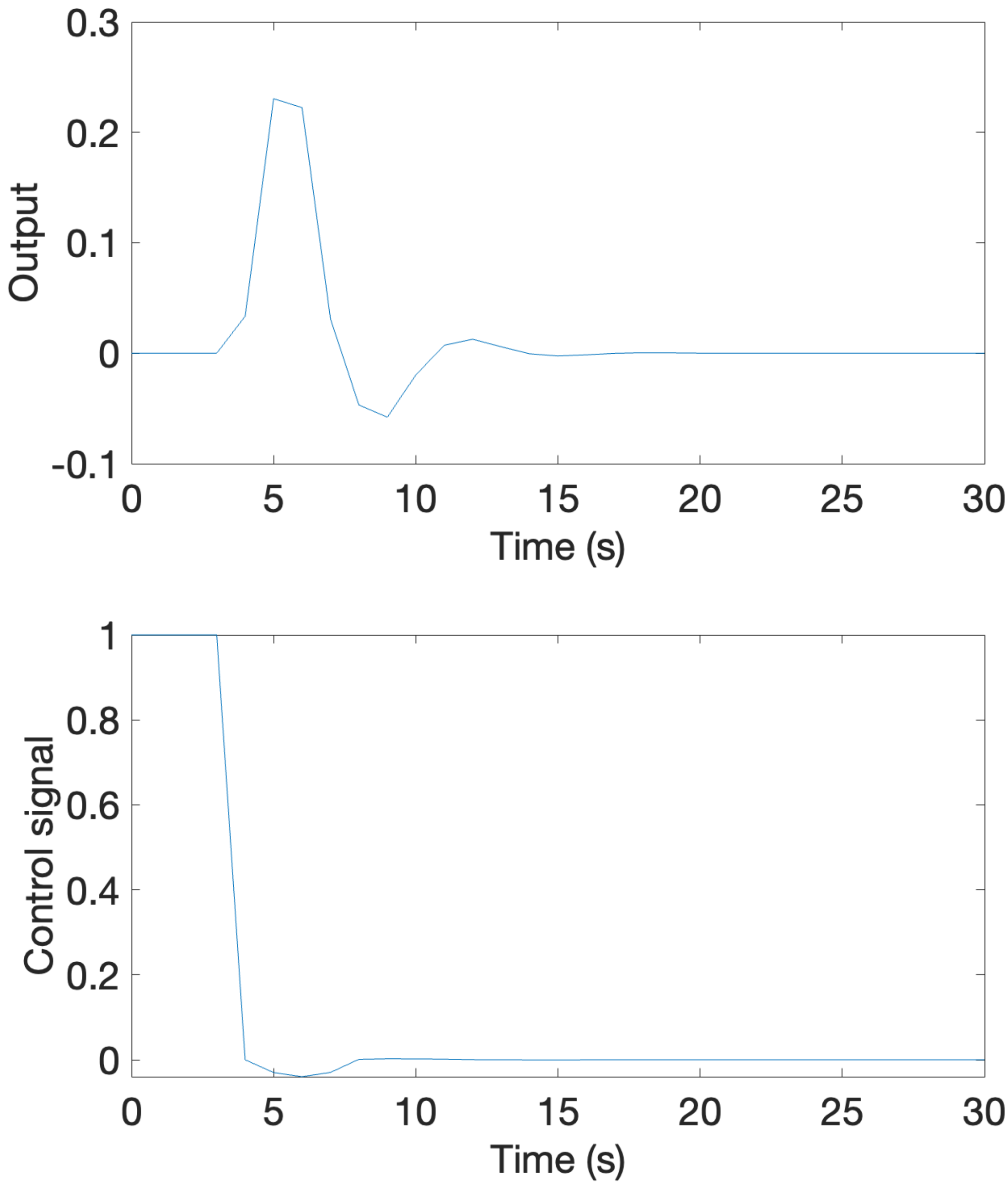

Figure 16: Output and control of direct adaptive implementation of moving average controller with mirrored zeros

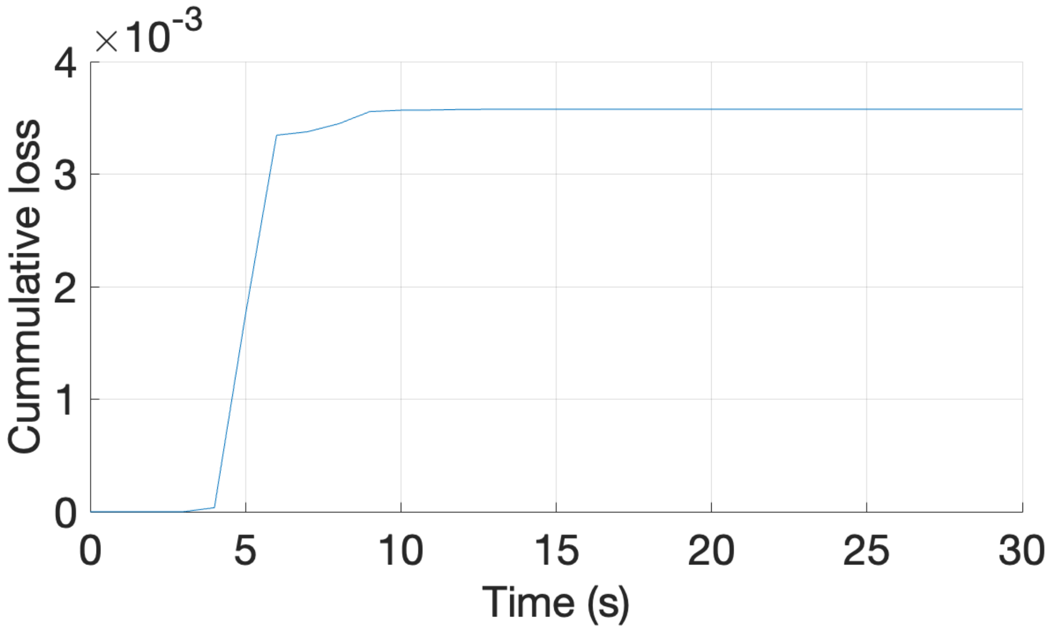

Figure 17: Cummulative loss of direct adaptive implementation of moving average controller

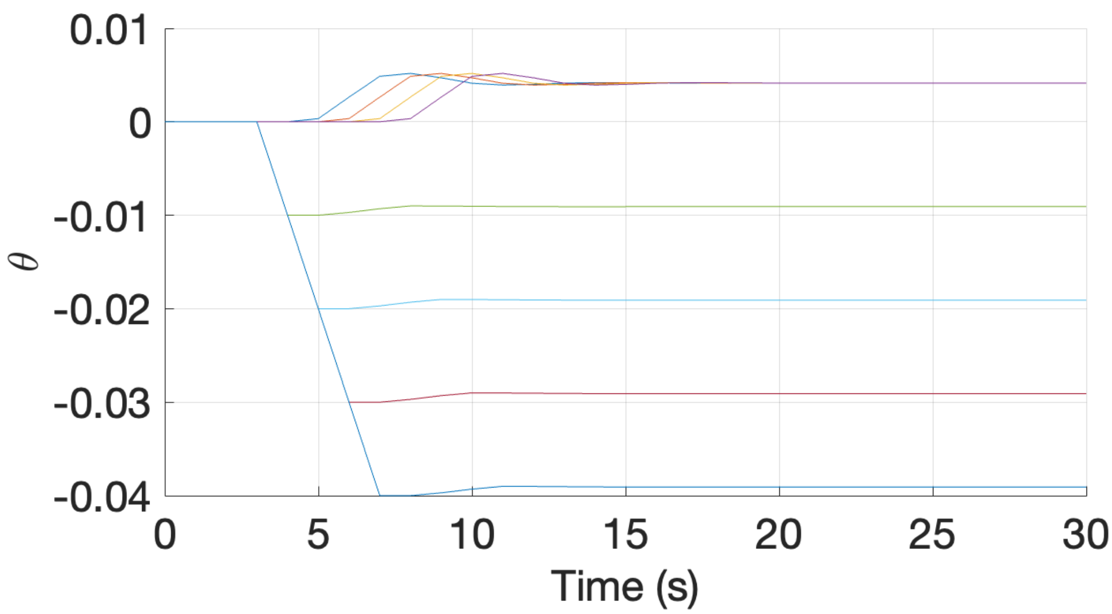

Figure 18: Estimated parameters of direct adaptive implementation of moving average controller