# CH5120: Course Project 2

# CH20B065 Mihir Singh

## Part - B : Model Predictive Control

## Problem Statement

a) Implement using actual calculations, the unconstrained MPC with prediction horizon and control horizon chosen with some heuristics for the following case:
    i) Controlled Variables are h1, h2 when all states are measured for a set-point for [h1 h2] of [13.4 13.7]
    ii) Give justification for the heuristics.
    iii) Analyze closed-loop stability of the unconstrained MPC system at two places:
        1) When the first move was implemented
        2) When the system stabilizes close to its set-point

**Heuristics for choosing the appropriate prediction and control horizon:**

1. **Prediction Horizon $N_p$**
    a. **System Dynamics**: Longer prediction horizons are typically used when the system dynamics are slow or exhibit long-term effects. This would allow the controller a longer time horizon to predict the future states.
    b. **Control objective:** If the objective is to track a set point then a longer prediction horizon would be beneficial for capturing the future evolution of the system.
    c. **Computational Complexity:** Longer prediction horizons increase the computational complexity of the optimization problem and therefore, its choice should be balanced with the available computational resources.
    d. **Noise and Disturbances:** If the system is affected by significant disturbances and measurement noise then a longer prediction horizon would help the system anticipate and mitigate the disturbance effects.

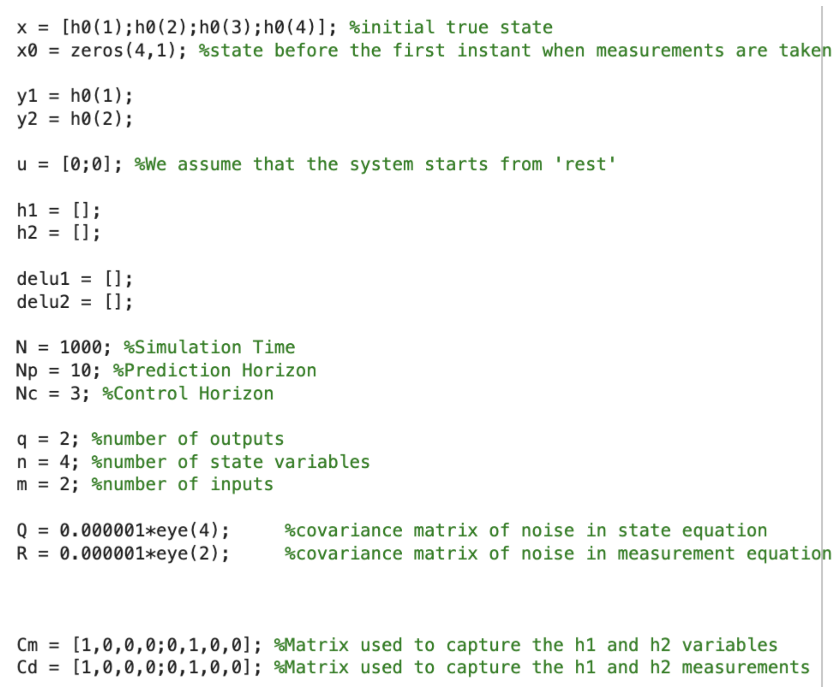
2. **Control Horizon $N_c$**
    a. **System Responsiveness:** Shorter control horizons make the system more responsive to changes in the system.
    b. **Constraints:** If the control horizon is too small, the controller might not be able to fulfill the constraints.
    c. **Computational Considerations:** Shorter control horizons reduce the computational burden of optimization.

With these heuristics in mind, the chosen values of the **prediction and control horizon** are **10 and 3** respectively.

Here in part a) we aim to implement an unconstrained **model predictive controller** to control the variables h1 and h2. The set point for these variables is [13.4,13.7].

## Approach:

The first step is initializing the initial state measurements, the simulation time, the prediction and control horizon, the covariance matrices of the state and measurement noise, and the C matrices needed to capture the values of h1 and h2 from our state vector.

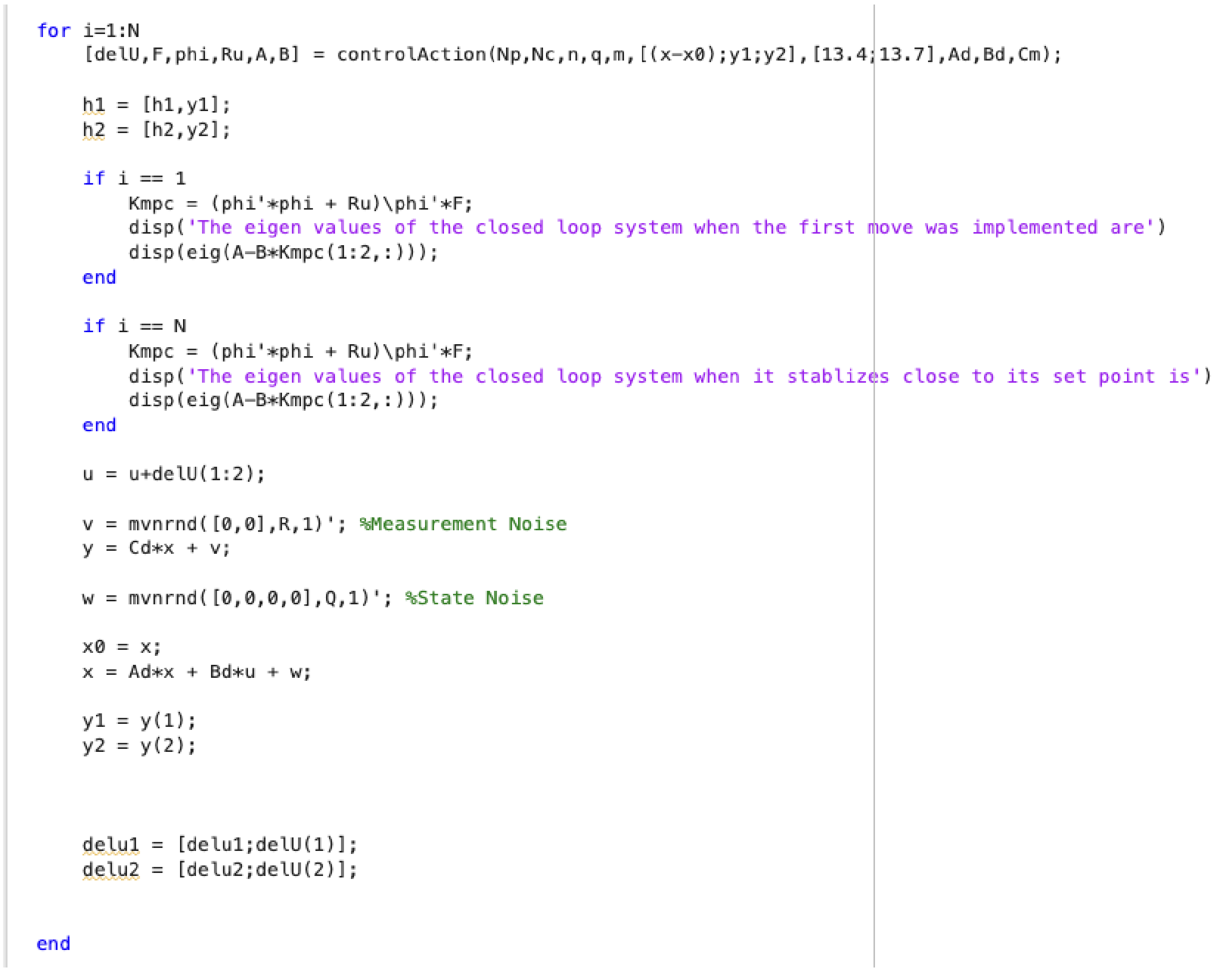


```
x = [h0(1);h0(2);h0(3);h0(4)]; %initial true state
x0 = zeros(4,1); %state before the first instant when measurements are taken

y1 = h0(1);
y2 = h0(2);

u = [0;0]; %We assume that the system starts from 'rest'

h1 = [];
h2 = [];

delu1 = [];
delu2 = [];

N = 1000; %Simulation Time
Np = 10; %Prediction Horizon
Nc = 3; %Control Horizon

q = 2; %number of outputs
n = 4; %number of state variables
m = 2; %number of inputs

Q = 0.000001*eye(4);     %covariance matrix of noise in state equation
R = 0.000001*eye(2);     %covariance matrix of noise in measurement equation



Cm = [1,0,0,0;0,1,0,0]; %Matrix used to capture the h1 and h2 variables
Cd = [1,0,0,0;0,1,0,0]; %Matrix used to capture the h1 and h2 measurements
```

The initial state is [12.4, 12.7, 1.8, 1.4].

In the subsequent steps, we iterate over a specified number of time steps (N), continuously solving an unconstrained Model Predictive Control (MPC) optimization problem based on the current state of the system. This optimization yields a sequence of Nc control inputs, from which only the first input is applied to the system at each iteration. Consequently, the system transitions to a new state, guided by the calculated control input and the underlying system dynamics. Through this iterative process, we dynamically generate a series of new measurements for variables h1 and h2.

Throughout the duration of the loop, a record is maintained, capturing the evolving measurements of h1 and h2, as well as the corresponding control inputs executed at each instance.



```
for i=1:N
    [delU,F,phi,Ru,A,B] = controlAction(Np,Nc,n,q,m,[(x-x0);y1;y2],[13.4;13.7],Ad,Bd,Cm);

    h1 = [h1,y1];
    h2 = [h2,y2];

    if i == 1
        Kmpc = (phi'*phi + Ru)\phi'*F;
        disp('The eigen values of the closed loop system when the first move was implemented are')
        disp(eig(A-B*Kmpc(1:2,:)));
    end

    if i == N
        Kmpc = (phi'*phi + Ru)\phi'*F;
        disp('The eigen values of the closed loop system when it stablizes close to its set point is')
        disp(eig(A-B*Kmpc(1:2,:)));
    end

    u = u+delU(1:2);

    v = mvnrnd([0,0],R,1)'; %Measurement Noise
    y = Cd*x + v;

    w = mvnrnd([0,0,0,0],Q,1)'; %State Noise

    x0 = x;
    x = Ad*x + Bd*u + w;

    y1 = y(1);
    y2 = y(2);



    delu1 = [delu1;delU(1)];
    delu2 = [delu2;delU(2)];


end
```

controlAction is the function used to solve the unconstrained MPC optimization problem. Within the function, the first thing done is defining a new incremental state space model of the system, and the corresponding A, B, and C matricesl.

```
A = [Am, zeros(n,q); Cm*Am, eye(q)];
B = [Bm; Cm*Bm];
C = [zeros(q,n), eye(q,q)];
```

Here, q represents the number of output variables, and n represents the number of state variables.

The matrices A, B, and C are then used to define larger and more complex matrices F and phi that are needed to make future predictions of the control variables given the system's initial state.

```
%defining F
F = [];
for i=1:Np
    F = [F;C*A^i];
end

%defining phi
phi = [];
for i=1:Nc
    col = zeros(q*(i-1),m);
    for j=i:Np
        col = [col;C*A^(j-i)*B];
    end
    phi = [phi,col];
end
```

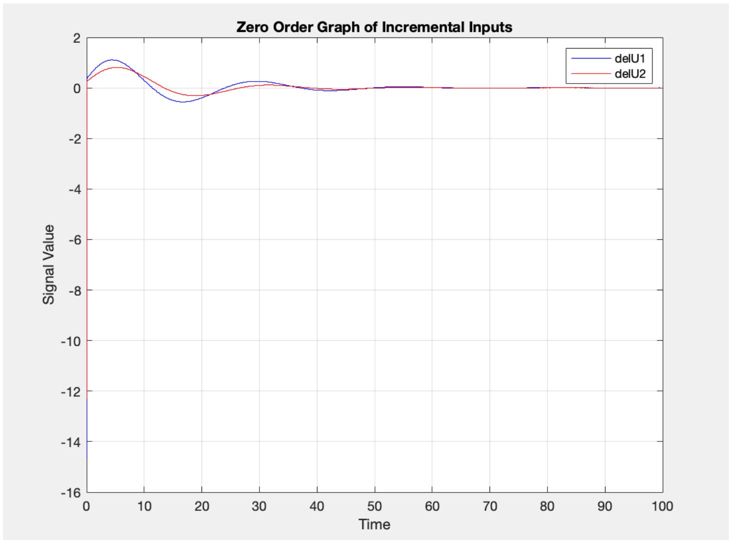
Next, we outline the set-point trajectory that we want our system to follow.

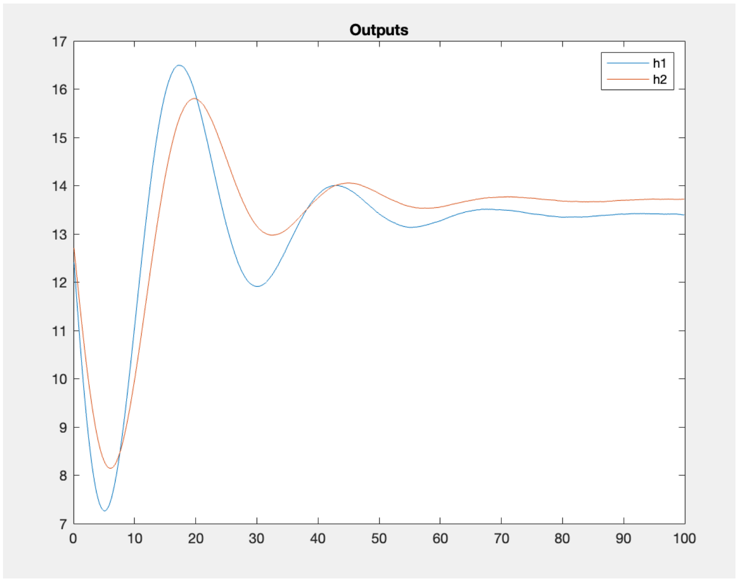
```
%defining the set point
Ru = eye(m*Nc);
Rs = [];
for i=1:Np
    Rs = [Rs;Rs0];
end
```

Rs0 here represents the final values we desire for the controlled variables of our choice.

The solution for the optimization problem in unconstrained model predictive control is a closed form expression that we use to calculate the most optimum series of Nc control inputs.

```
delU = inv(phi'*phi + Ru)*phi'*(Rs - F*x0);
```





Looking at the above figures, we see the system successfully reaches and maintains itself at the defined set point [13.7, 13.4].

To determine the closed stability of the MPC system when the first control move was implemented and when the system stabilizes close to its set point, the following snippet of code is used.

```
if i == 1
    Kmpc = (phi'*phi + Ru)\phi'*F;
    disp('The eigen values of the closed loop system when the first move was implemented are')
    disp(eig(A-B*Kmpc(1:2,:)));
end

if i == N
    Kmpc = (phi'*phi + Ru)\phi'*F;
    disp('The eigen values of the closed loop system when it stablizes close to its set point is')
    disp(eig(A-B*Kmpc(1:2,:)));
end
```

A and B are the matrices of the incremental state space model defined to design the unconstrained MPC.

The eigenvalues of the closed loop system when the first move was implemented are

```
0.9928 + 0.0246i
0.9928 - 0.0246i
0.9882 + 0.0104i
0.9882 - 0.0104i
0.9733 + 0.0000i
0.9572 + 0.0000i
```

The eigenvalues of the closed loop system when the system stabilizes close to its set point are

```
0.9928 + 0.0246i
0.9928 - 0.0246i
0.9882 + 0.0104i
0.9882 - 0.0104i
0.9733 + 0.0000i
0.9572 + 0.0000i
```

The real components of all the eigenvalues are less than 1 which suggests that closed loop stability is obtained for the unconstrained MPC.

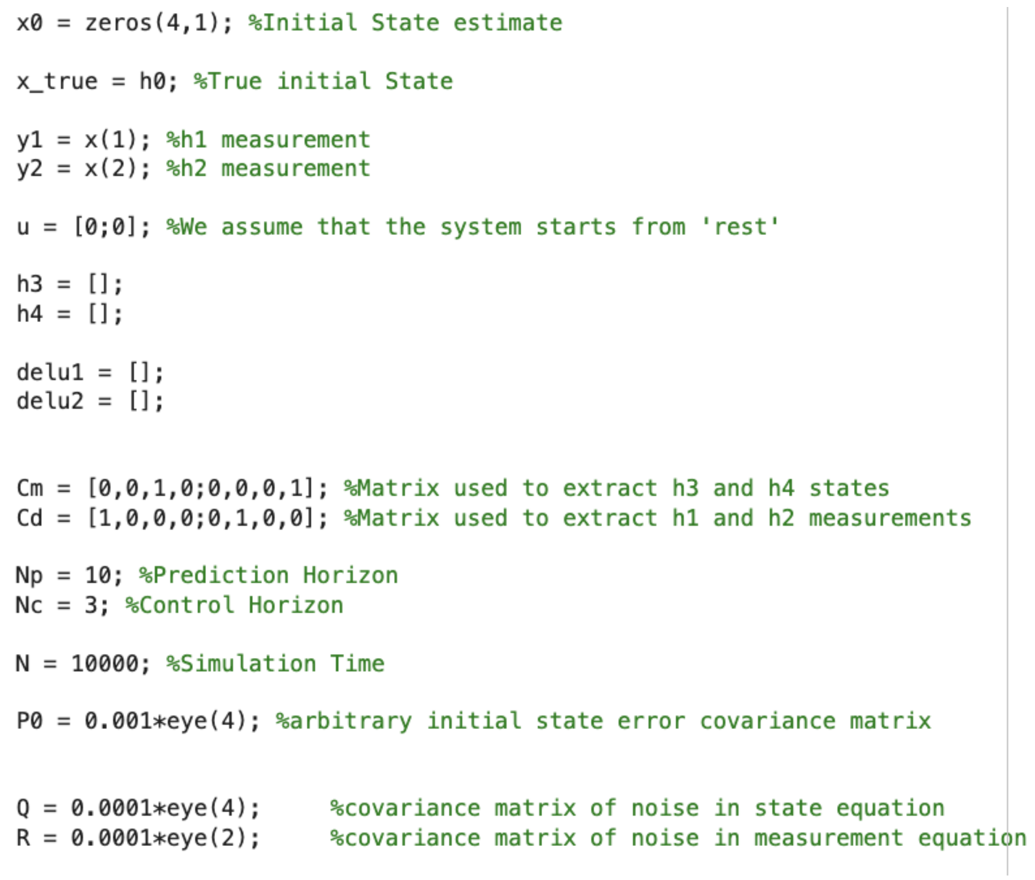
b) Implement Constraint MPC to control

a. Control h3, h4 when h1, h2 are measured; set-point for [h3 h4] is [2.8 2.4]

b. Analyze how Kalman filter performance affects MPC performance by experimenting with Kalman gain parameters.

**Approach:**

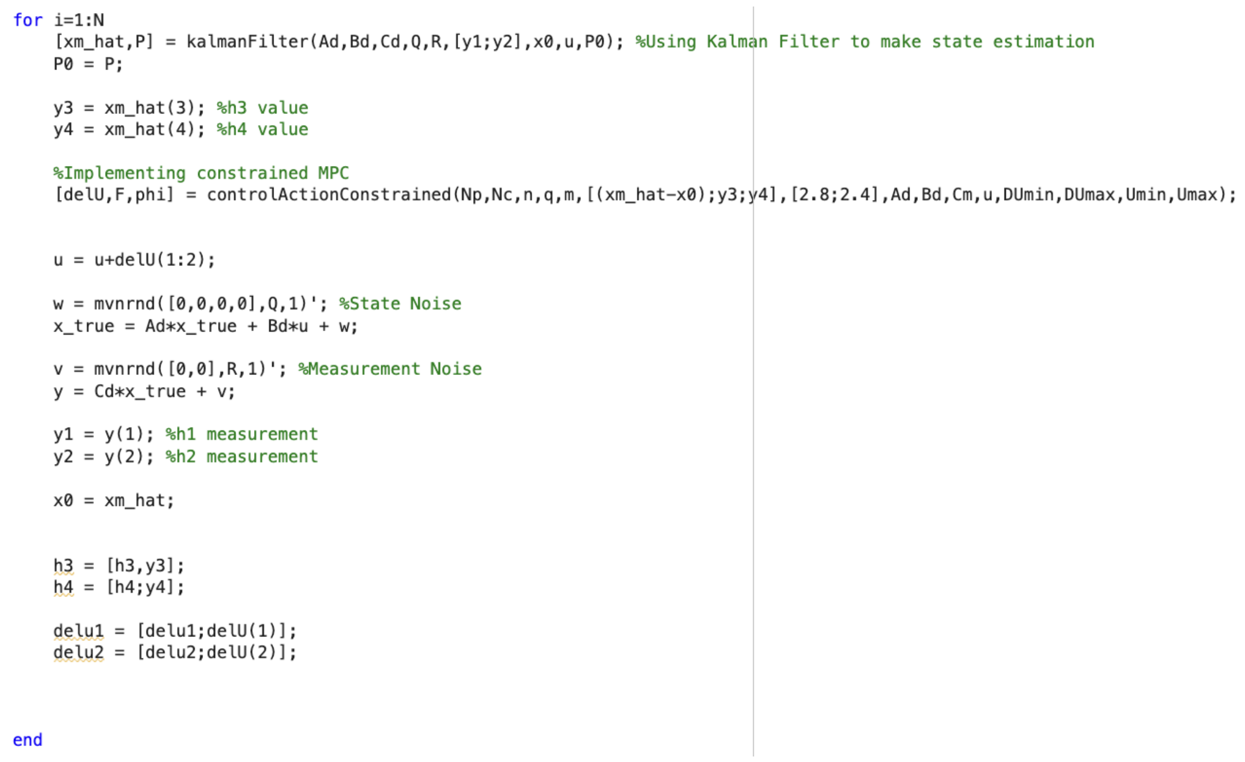
Like in part a) the first step involves initializing the true state vector, an estimate of the state vector (only h1 and h2 measurements are available), the prediction and control horizon, the covariance matrices of the state and measurement noise, the initial covariance matrix of the state estimation error, and the C matrix used to measure h1 and h2 which we need for our Kalman Filter, and the C matrix we need to estimate h3 and h4 which we need to design our model predictive controller.



```
x0 = zeros(4,1); %Initial State estimate

x_true = h0; %True initial State

y1 = x(1); %h1 measurement
y2 = x(2); %h2 measurement

u = [0;0]; %We assume that the system starts from 'rest'

h3 = [];
h4 = [];

delu1 = [];
delu2 = [];


Cm = [0,0,1,0;0,0,0,1]; %Matrix used to extract h3 and h4 states
Cd = [1,0,0,0;0,1,0,0]; %Matrix used to extract h1 and h2 measurements

Np = 10; %Prediction Horizon
Nc = 3; %Control Horizon

N = 10000; %Simulation Time

P0 = 0.001*eye(4); %arbitrary initial state error covariance matrix


Q = 0.0001*eye(4);     %covariance matrix of noise in state equation
R = 0.0001*eye(2);     %covariance matrix of noise in measurement equation
```

In the subsequent steps, we iterate over a specified number of time steps (N), first obtaining a more accurate estimate of our current state using the measurements of h1 and h2 in conjunction with a Kalman filter. This estimate of the state is used to solve the constrained MPC optimization problem and the resulting optimization yields a sequence of Nc control inputs from which only the first input is applied on the system. Consequently, the system transitions to a new true state and new measurements of h1 and h2 are obtained thanks to the stochastic state space model that defines the dynamics of our system. The values of h3 and h4 are extracted from our estimates of the system's state.

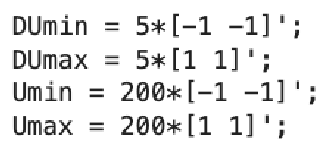
Throughout the duration of the loop, a record is maintained, capturing the evolving estimates of h3 and h4, as well as the corresponding control inputs executed at each instance.



```
for i=1:N
    [xm_hat,P] = kalmanFilter(Ad,Bd,Cd,Q,R,[y1;y2],x0,u,P0); %Using Kalman Filter to make state estimation
    P0 = P;

    y3 = xm_hat(3); %h3 value
    y4 = xm_hat(4); %h4 value

    %Implementing constrained MPC
    [delU,F,phi] = controlActionConstrained(Np,Nc,n,q,m,[(xm_hat-x0);y3;y4],[2.8;2.4],Ad,Bd,Cm,u,DUmin,DUmax,Umin,Umax);


    u = u+delU(1:2);

    w = mvnrnd([0,0,0,0],Q,1)'; %State Noise
    x_true = Ad*x_true + Bd*u + w;

    v = mvnrnd([0,0],R,1)'; %Measurement Noise
    y = Cd*x_true + v;

    y1 = y(1); %h1 measurement
    y2 = y(2); %h2 measurement

    x0 = xm_hat;


    h3 = [h3,y3];
    h4 = [h4;y4];

    delu1 = [delu1;delU(1)];
    delu2 = [delu2;delU(2)];


end
```

Constraints are set on the magnitude of the input control signal u and the incremental change in the input control signal del u.

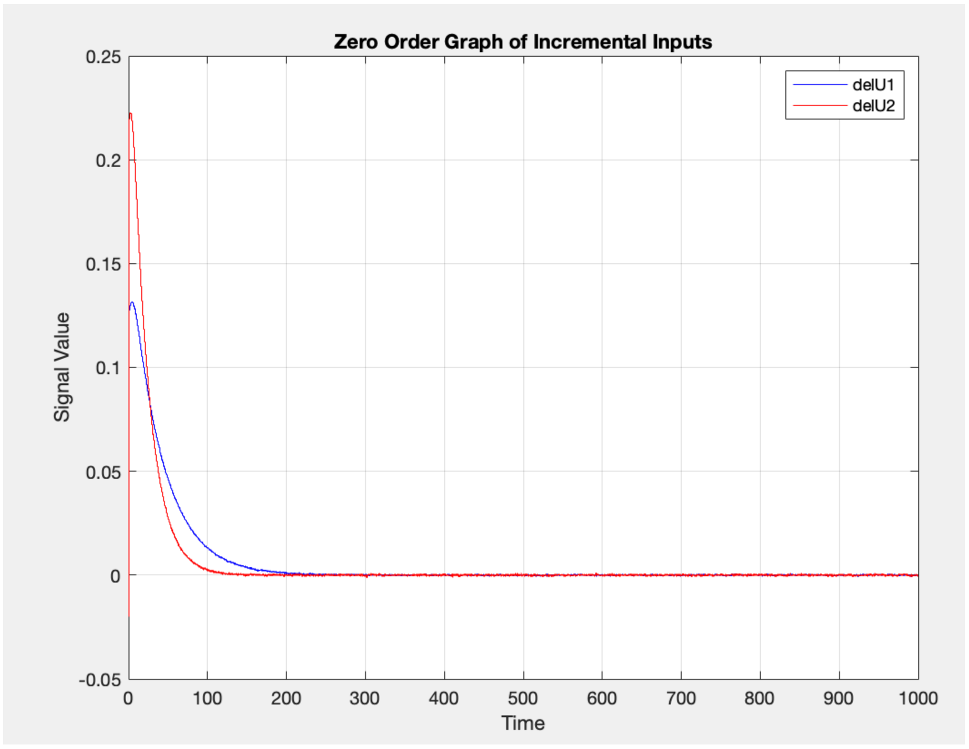
These constraints are given by the following snippet of code.

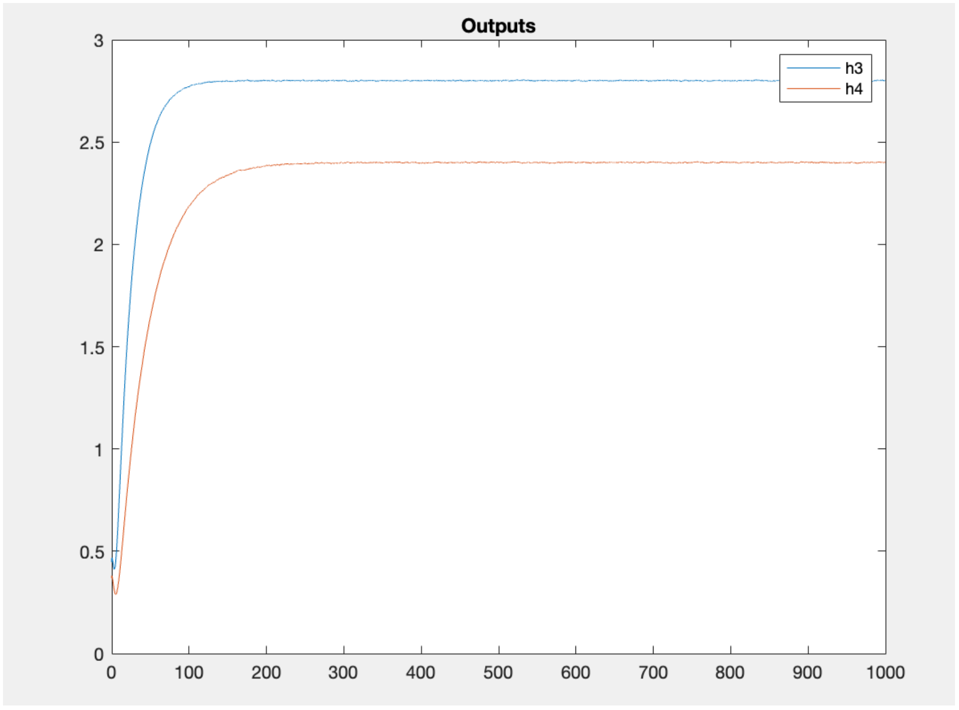


```
DUmin = 5*[-1 -1]';
DUmax = 5*[1 1]';
Umin = 200*[-1 -1]';
Umax = 200*[1 1]';
```

controlActionConstrained is the function used to solve the constrained MPC optimization problem. Similar to the solving unconstrained MPC optimization, the first steps here also include defining a new incremental state space model with new A, B, and C matrices, and using them to define even more complex matrices F and phi that we need to make predictions regarding the future of the system.

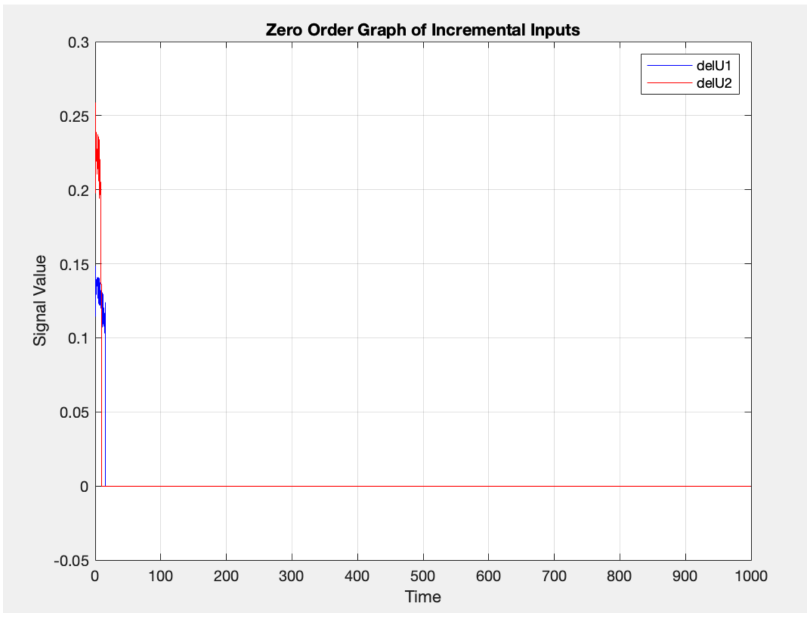
Next, a linear equality form of the constraints is outlined which are passed as input into MATLAB's quadprog function in addition to the overall cost function we aim to minimize. We use quadprog function here because the constrained MPC optimization problem does not have a closed form solution like its unconstrained counterpart. Finally, the quadratic programming solver yields a series of Nc optimum control inputs from which only the first one is applied on the system.

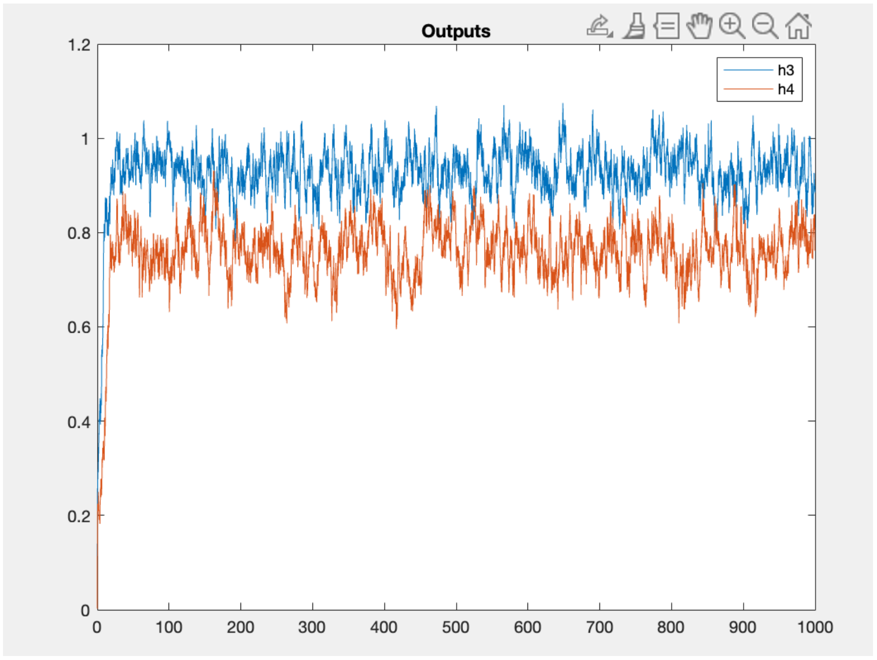




Here as well we see the system does not reach the set point. The set point defined for h3 and h4 is [2.8, 2.4].

The performance of the unconstrained model predictive controller is very clearly affected by the performance of the Kalman filter. The plots obtained above are for an MPC that uses the following covariance matrices for the state and measurement noise: 0.000001*eye(4) and 0.000001*eye(2). The Kalman gain depends on these covariance matrices, and consequently so do the Kalman gain parameters which happen to the elements of the Kalman gain matrix. Therefore, we can analyze the effect of the Kalman gain parameters by investigating the effects of the Q and R matrices on the MPC's performance, and to do so, we experiment by using larger matrices for Q and R and observe the impacts brought about. The new values of Q and R taken are 0.001*eye(4) and 0.001*eye(2), and the new performance plots we get are shown below.





The most significant things that we observe are that the incremental control signals have become larger in magnitude and aggressiveness, and similarly the controlled variables are not able to smoothly converge to their designated set points.

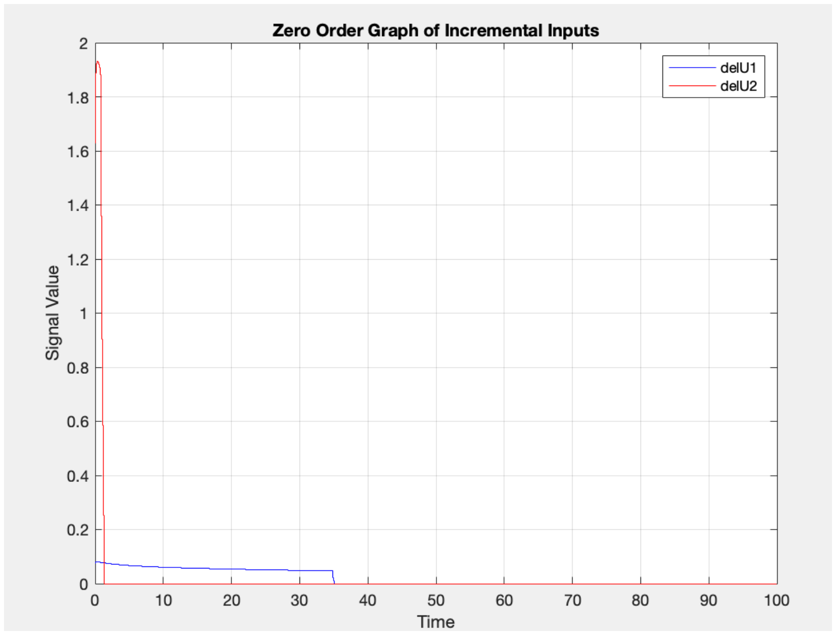
This observation implies that increased uncertainty in system dynamics compels the system to pursue a less predictable trajectory towards a designated set point, presenting challenges to its tracking capabilities.

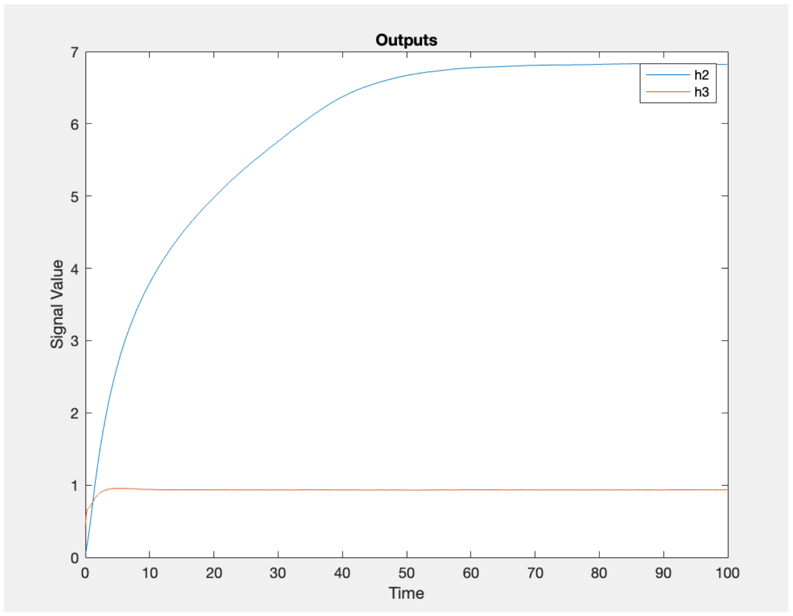
c) Implement Constraint MPC such that it can be used to control

a. h2, h3 when h1, h4 are measured with set-point for [h2 h3] as [13.7 2.8]
b. h1, h3 when h2, h4 are measured with set-point for [h1 h3] as [13.7 2.4]

Exactly what we did to solve the first question in part b) is repeated here except that the C matrices are changed appropriately to take into account the new variables that need to be controlled and the new ones that need to be measured.

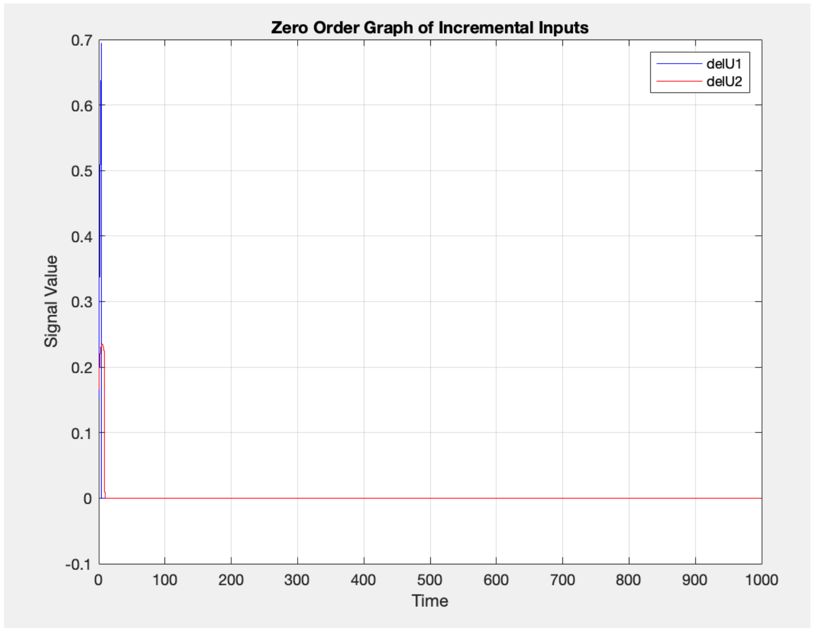
a. h2 and h3 are the control variables, and h1 and h4 are the measured variables

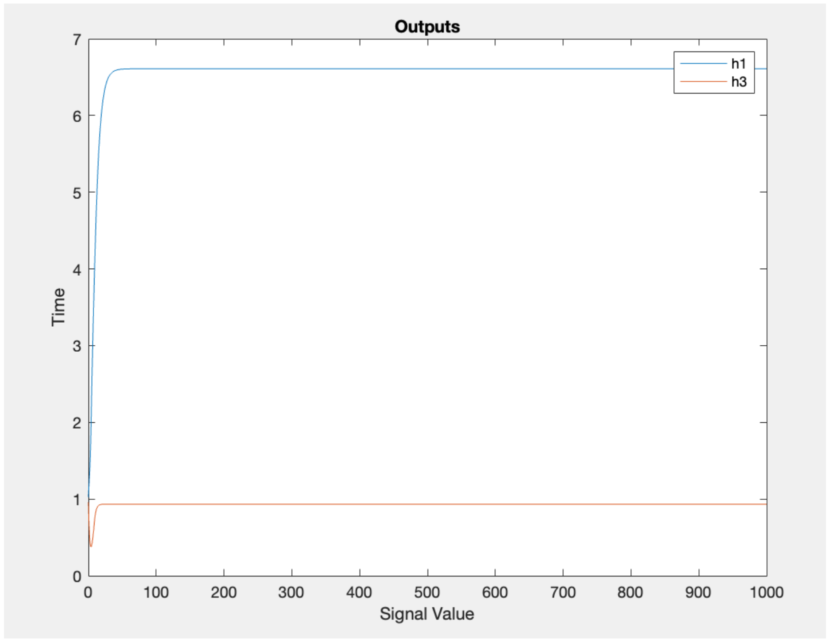




The control variables do not converge to their designated set points.

b. h1 and h3 are the control variables, and h2 and h4 are the measured variables.





Here also the control variables do not converge to their set points and one reason for this is most likely that the constraints placed on the input control signals may be preventing them from doing so.

Only in the unconstrained case, the system is able to achieve set point tracking. A possible reason for this is that constraints placed on the input control signals are too restrictive and inhibit the system from reaching its designated set points.

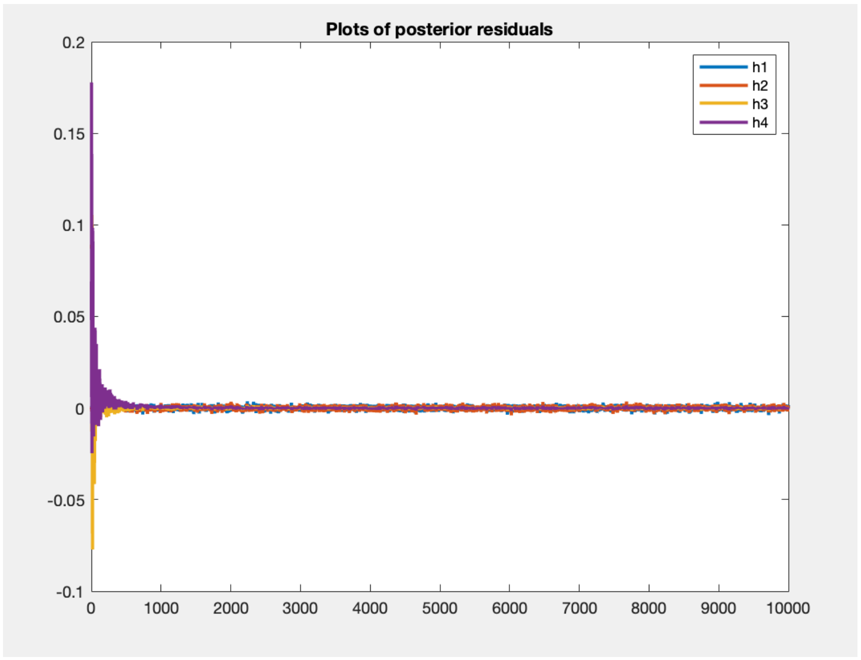
## Part - B : Extended Kalman Filter

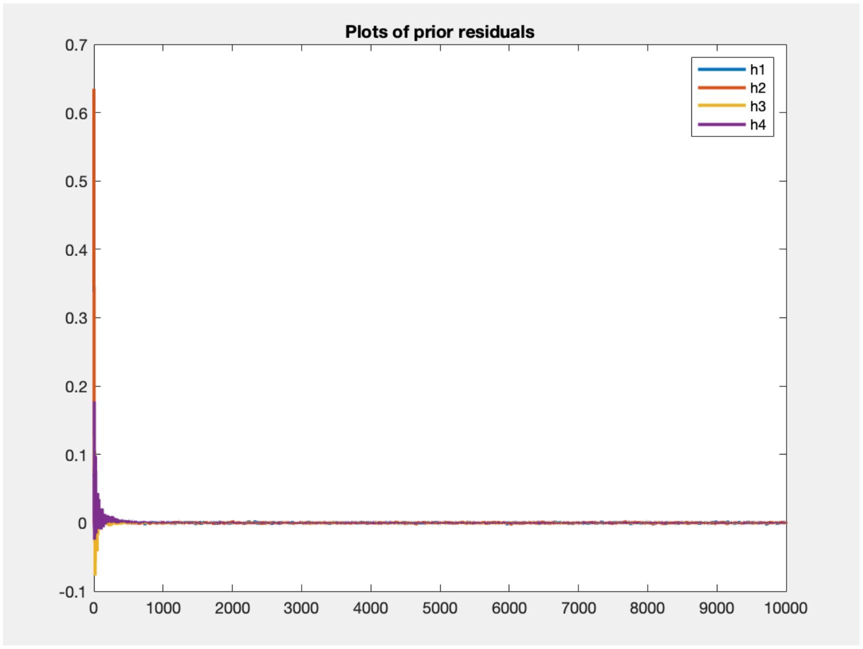
For the mini project 1 part A on Kalman filter, compare the plots by using an EKF, considering typically 10 discrete points for linearization(obtaining Jacobians). Use the same dataset.

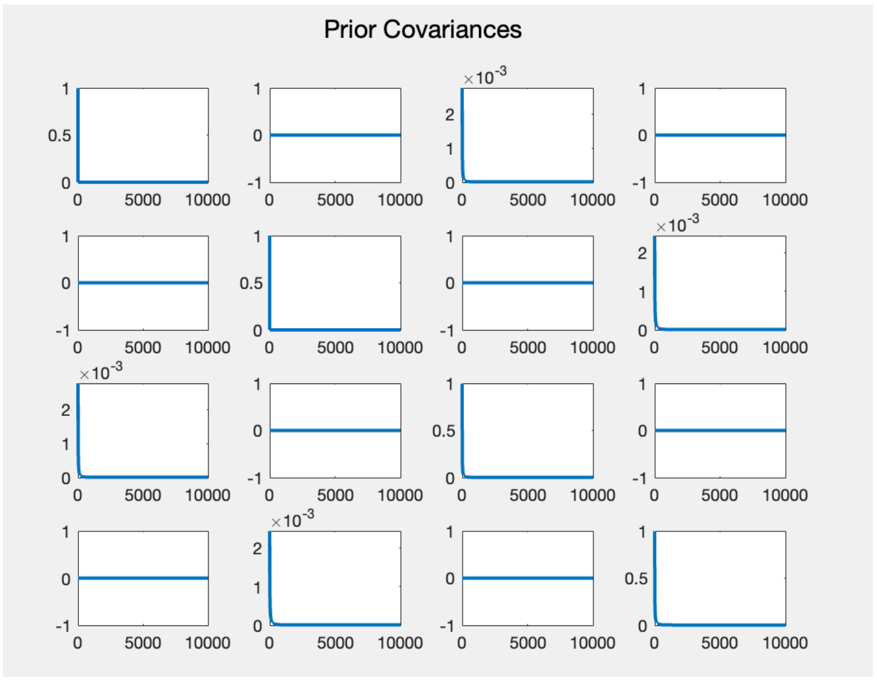
The dataset used in mini project 1 consisted of 10,000 points representing the results of simulating the system of ordinary differential equations that formulated the four tank system. The problem asks to arbitrarily choose 10 points for linearization and so we select every thousandth data point in the dataset.

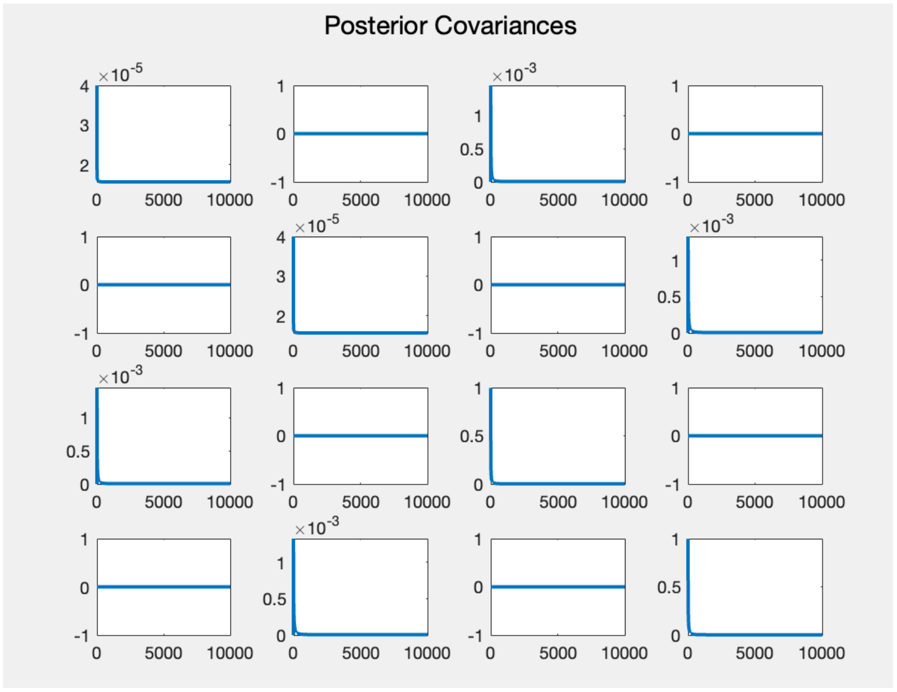
The same graphs that were plotted for the Kalman Filter in project 1 are plotted here for the extended Kalman Filter.

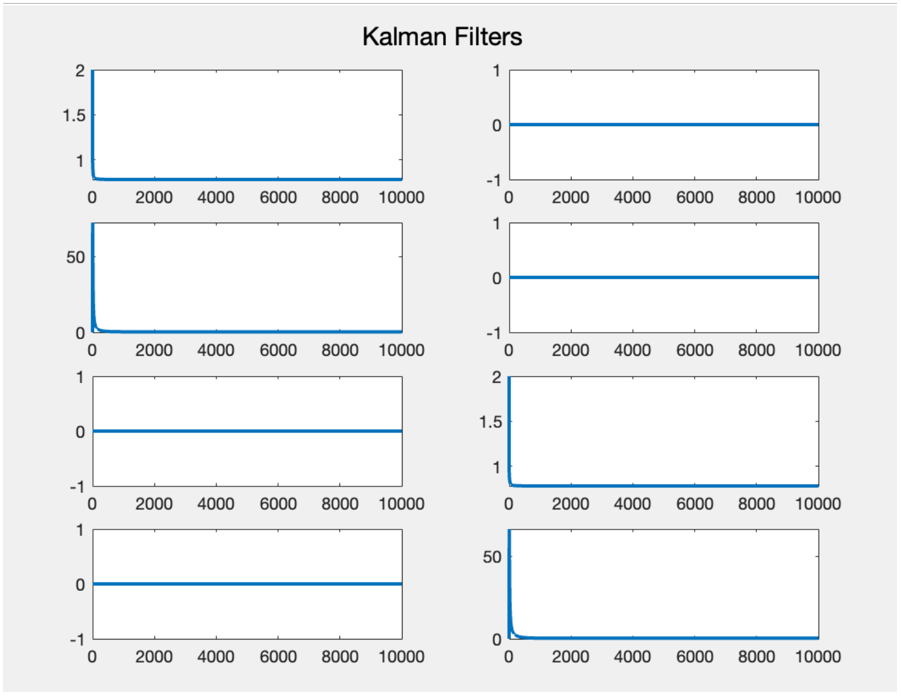
## Plots

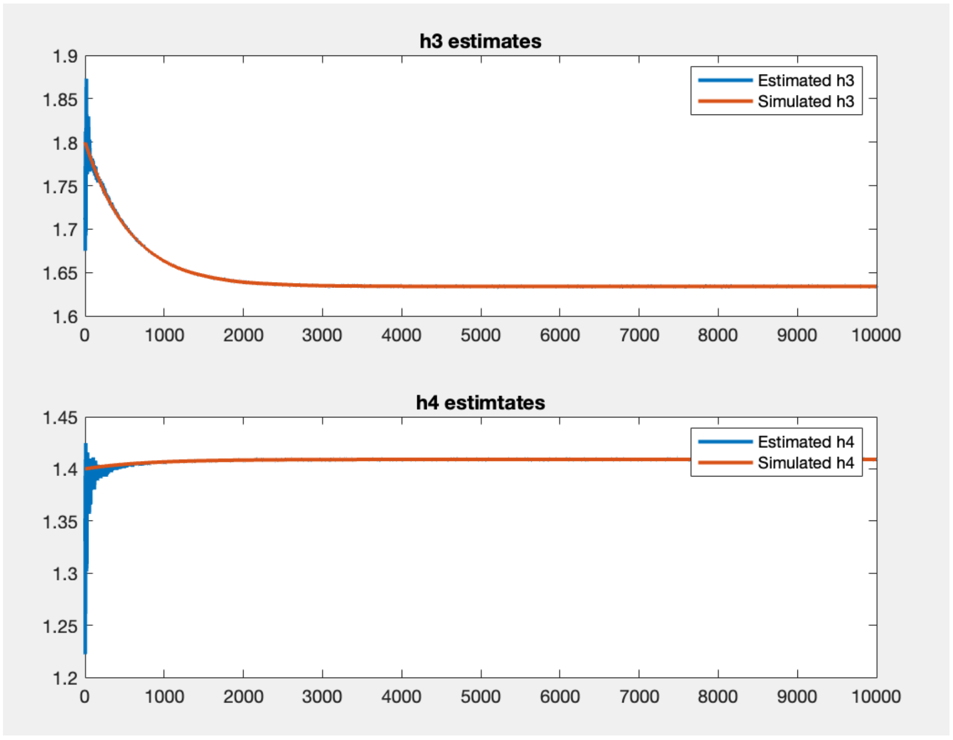



Plots of prior residuals
h1
h2
h3
h4
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
-0.1
0
1000
2000
3000
4000
5000
6000
7000
8000
9000
10000


Prior Covariances
$\times 10^{-3}$
0
5000
10000


Posterior Covariances
×10^-5
×10^-3
×10^-5
×10^-3
×10^-3
×10^-3
0
5000
10000


Kalman Filters
0
2000
4000
6000
8000
10000


h3 estimates
Estimated h3
Simulated h3
1.9
1.85
1.8
1.75
1.7
1.65
1.6
0
1000
2000
3000
4000
5000
6000
7000
8000
9000
10000
h4 estimtates
Estimated h4
Simulated h4
1.45
1.4
1.35
1.3
1.25
1.2
0
1000
2000
3000
4000
5000
6000
7000
8000
9000
10000