Name: Soumya Ganguly

PID : A53274333

Email : s1gangul@eng.ucsd.edu

The entire homework solution is arranged sequentially in terms of the tasks given. Execution of each task has been explained with reasons and the code to justify everything has been given in the end of this documentation. Explanations are supported with plots and figures, whenever necessary. Discreet time system under consideration here has a characteristics frequency of 200 Hz. $\emptyset, \theta$ are taken to be in usual sense.

TASK 0:

We run the 'mIpParameters.m' file once so that the necessary matrices are loaded in our workspace. We define the system with A,B,C,D matrices matrices that have been loaded in the workspace.

Then reference and wind generators are set to zero and system response has been recorded from the simulink file given. One sample response with initial condition on θ to be 0.1 and rest state coordinates as zero is given in the following :

Here the five states' evolution is recorded as time passes and we see that the final state is becoming zero as time progresses. The sequence of the states are in conventional sense of analysis of MIP.

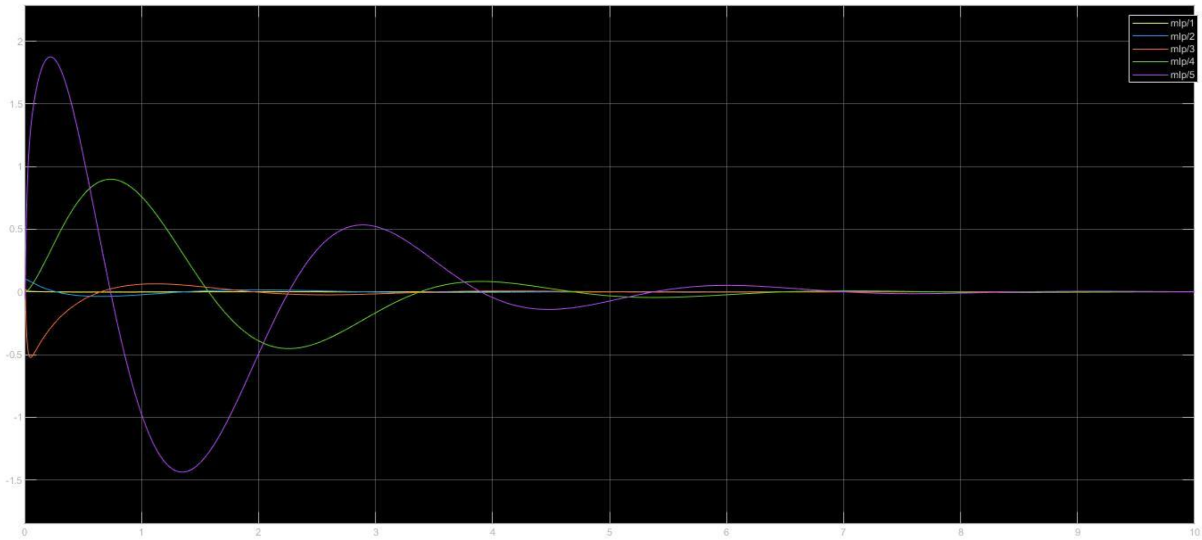


Figure 1: Before starting the analysis, this is the our output for initial condition zero

TASK 1:

We define the system and compute the rank of the observability and controllability matrices with the help of MATLAB commands and see that both the matrices are full rank, so our system is both reachable and observable. Now we have taken the poles of K and L to be arbitrarily placed within the unit circle in complex plane. For a reference we have simply multiplied the eigenvalues of ‘A’ matrix of system, with real factors that are lesser than 1 and greater than zero (0.95 in both cases here).

TASK 2:

We have defined a controller that we can feed back to our system in order to make the closed-loop-system stable and using MATLAB’s ‘*feedback’* command, we can create a ‘closed loop system’ that will be stabilised. To see whether the system is actually stabilised or not, we plot the ‘Ø (wheel angle)’ and ‘θ (MIP orientation with respect to ground)’ with the help of the MATLAB’s ‘*lsim’* command. We have plotted the plots upto time= 20 units. The sample plots are given in Figure 2.

TASK 3:

The *‘d2c’* command in MATLAB is used on our closed loop system to get a continuous equivalent system. ‘Z O H’ is taken into account while doing this and we see from Figure 2. below that ‘Ø’ and ‘θ’ outputs for the continuous system kind of agree with the discreet system’s outputs.

It is noted that the nonlinear simulink model also used ‘Zero Order hold’.

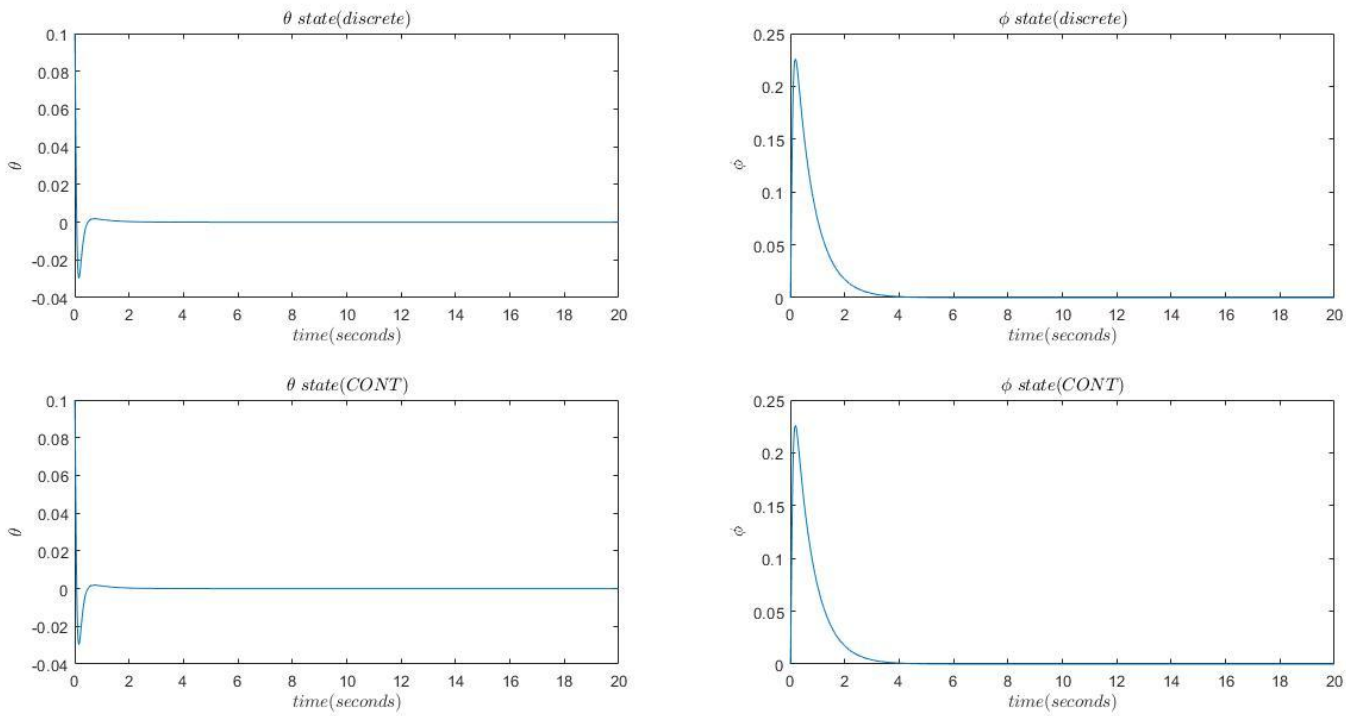


Figure 2: The plots of θ and Ø for both continuous and discreet systems

Task 4:

Subpart (i):

Our computed K,L matrices are used to compute ALDmIp, BLDmIp, ALCBKLDmIp. Then the simulink file is run after setting the amplitudes and biases of winds and references to zero and setting the noises to zero power as well. We get the a zero equilibrium for all the states as the initial conditions are made zero. (Figure 3). The state estimates are also found to be zeroes.

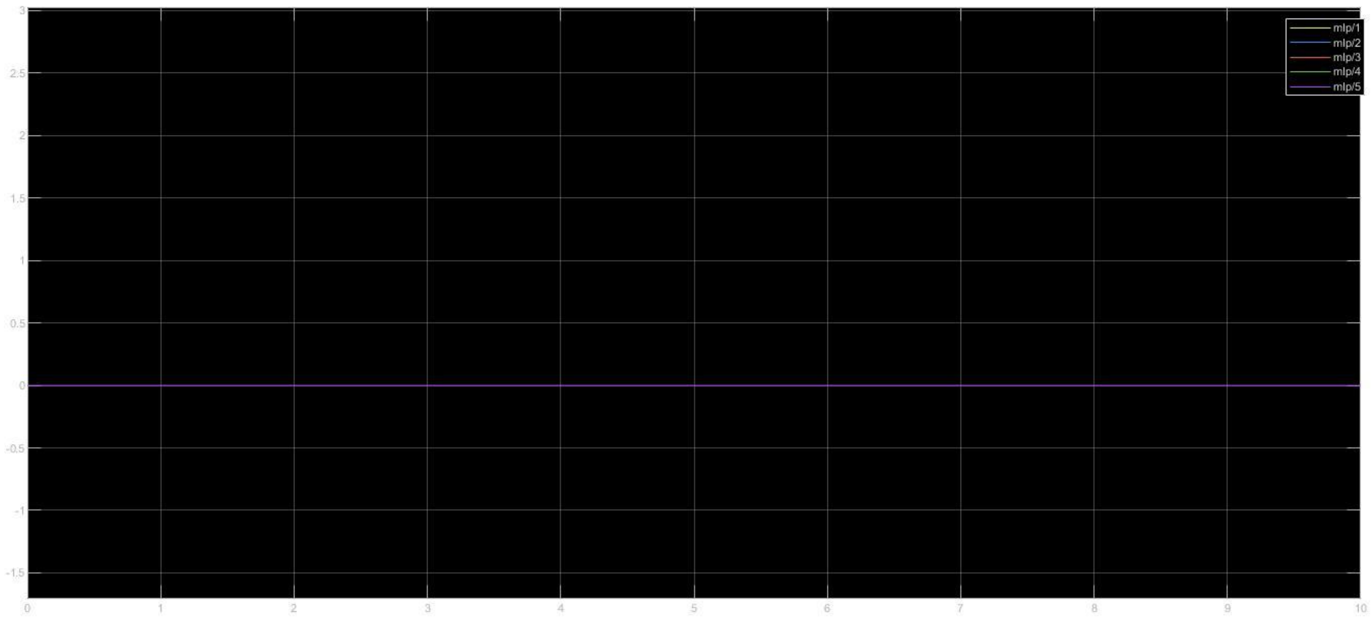


Figure 3 : Getting zero as our equilibrium for all states

In subpart (ii):

We try to find RANGE of initial conditions on θ forund by inspection:

We get that within [0,0.9810] radians: the system is stable and we see at θ_initial of 0.9811, the system is unstable.

So we can safely assume, the range of initial condition of θ for stability : [0, 0.9810]

In subpart (iii):

We are including some reference noise signal by setting nonzero amplitude. We vary the reference first in a very small range then beyond the linearization range and after that we take it beyond the saturation range itself. We see that beyond the saturation range, the system's states and estimates differ widely. The comparison of states obtained and the states estimated for some specific values of the amplitude are given below:

<u>NOTE :</u> We can attain a better range of agreement between the states and their estimates if we let the input of the system enter directly into the controller of the simulink file. Here in this case we have done that only so that our observer gives us a better approximation of the system. This is needed because of the 'saturation' term in the input which is nonlinear term and can't be estimated properly with our linear observer-controller.

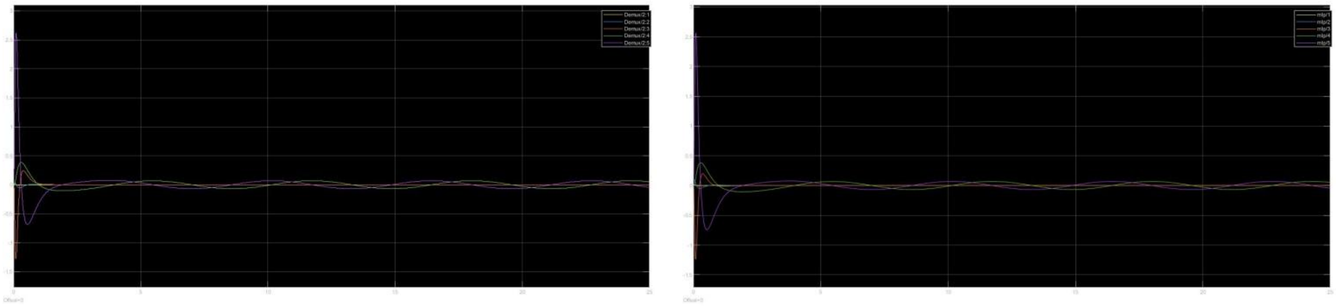


Figure 4 : States and their estimates for reference amplitude 0.1

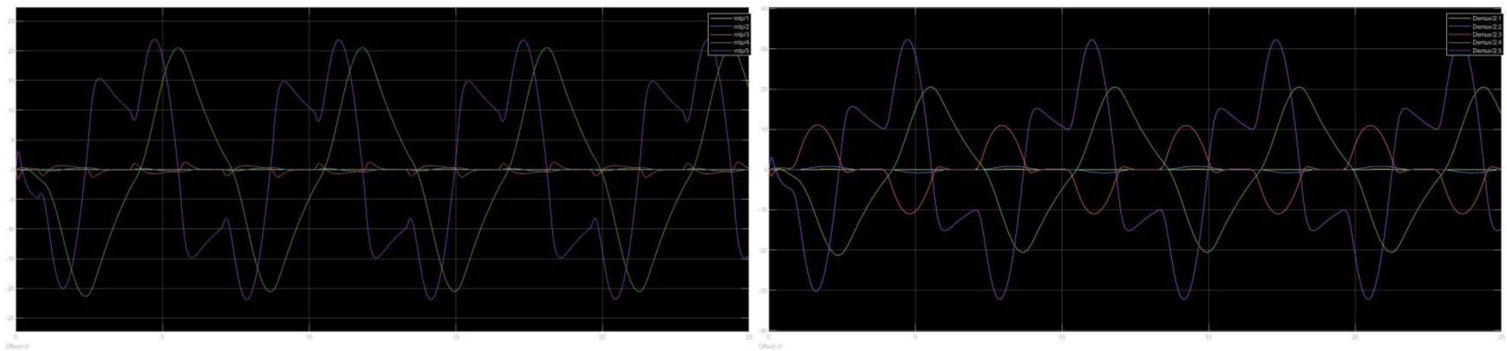


Figure 5 : States and their estimates for reference amplitude 9

<u>In subpart (iv):</u>

We reset initial condition of θ to zero and we start raising the θ noise power and observe:

In θ noise power range [0,0.001041] : the system is stable and at θ noise of 0.001042, the system is unstable.

So we can safely assume, the range of noise power for stability: [0, 0.001041]

NOTE : From our code given below one may find in the simulink model that the system stable for more than this noise power range given above but if the run time of the simulation is increased from '10' to say '80', the whole picture becomes clear.

<u>In subpart (v):</u>

To make our observer better for withstanding more noise, we can simply make the poles of the observer smaller than the previous one, or by making the observer 'faster'. As we start doing it, we get that the system becomes more and more immune to noise. Here two comparative figures are given below, the left one being the condition of the system's states at noise power 0.001042 with the previous observer of TASK 1 and the right one is the condition of the system the system's states at noise power 0.001042 when the eigenvalues of the K,L matrices are changed (as given in the code in the end). The θ state is stable in the right plot but not in the left plot.

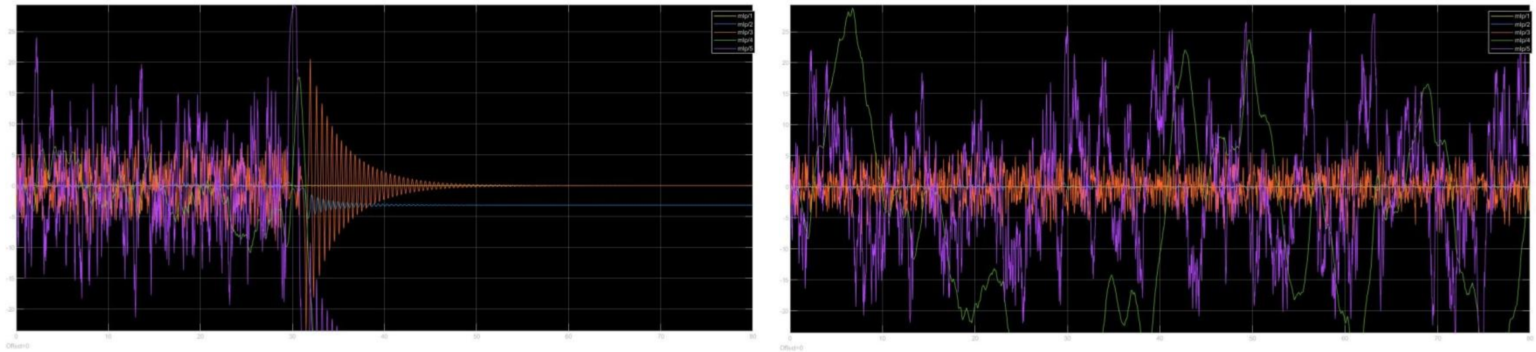


Figure 6 : Comparative behaviour at same noise level for two different controllers

TASK 5 :

The controller we developed in TASK 1 is already better than the one of Neeraj's because there the K matrix has very similar elements resulting in comparable control efforts for both but my controller is stabilising the system much faster than Neeraj's controller in current case. Also the overshoot in 'Ø' for Neeraj's is much more than that of my own system. More overshoot can result in high jerk or acceleration which can be deleterious for the machine's hardware. In the following figure , two comparative graphs for both the controllers are given for 'θ' and 'Ø' respectively. We can see the verification of the statements made above (Neeraj's controller's plots are the ones with red dots. Mine is the one with blue dots).

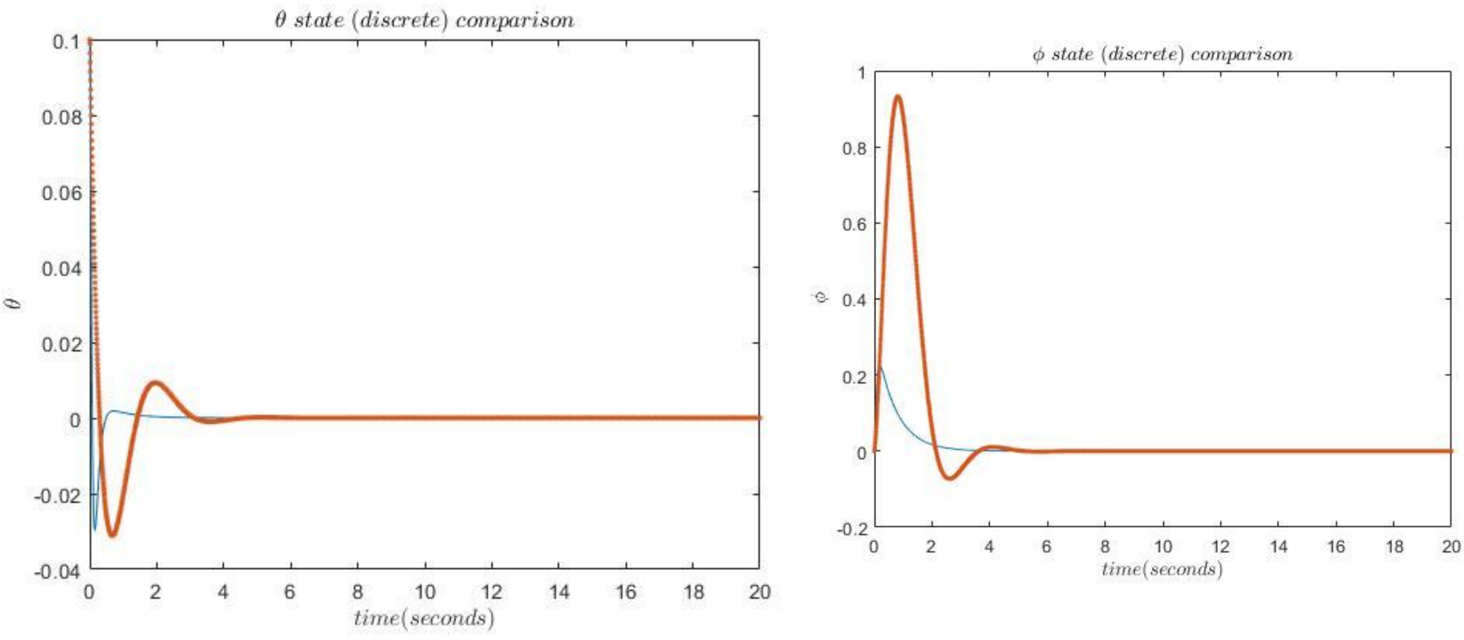


Figure 7 : Comparison between Neeraj's and my controller (Neeraj's controller's plots are the ones with red dots. Mine is the one with blue dots)

TASK 6:

The Documentation is being done here with plots and explanation of design conditions.

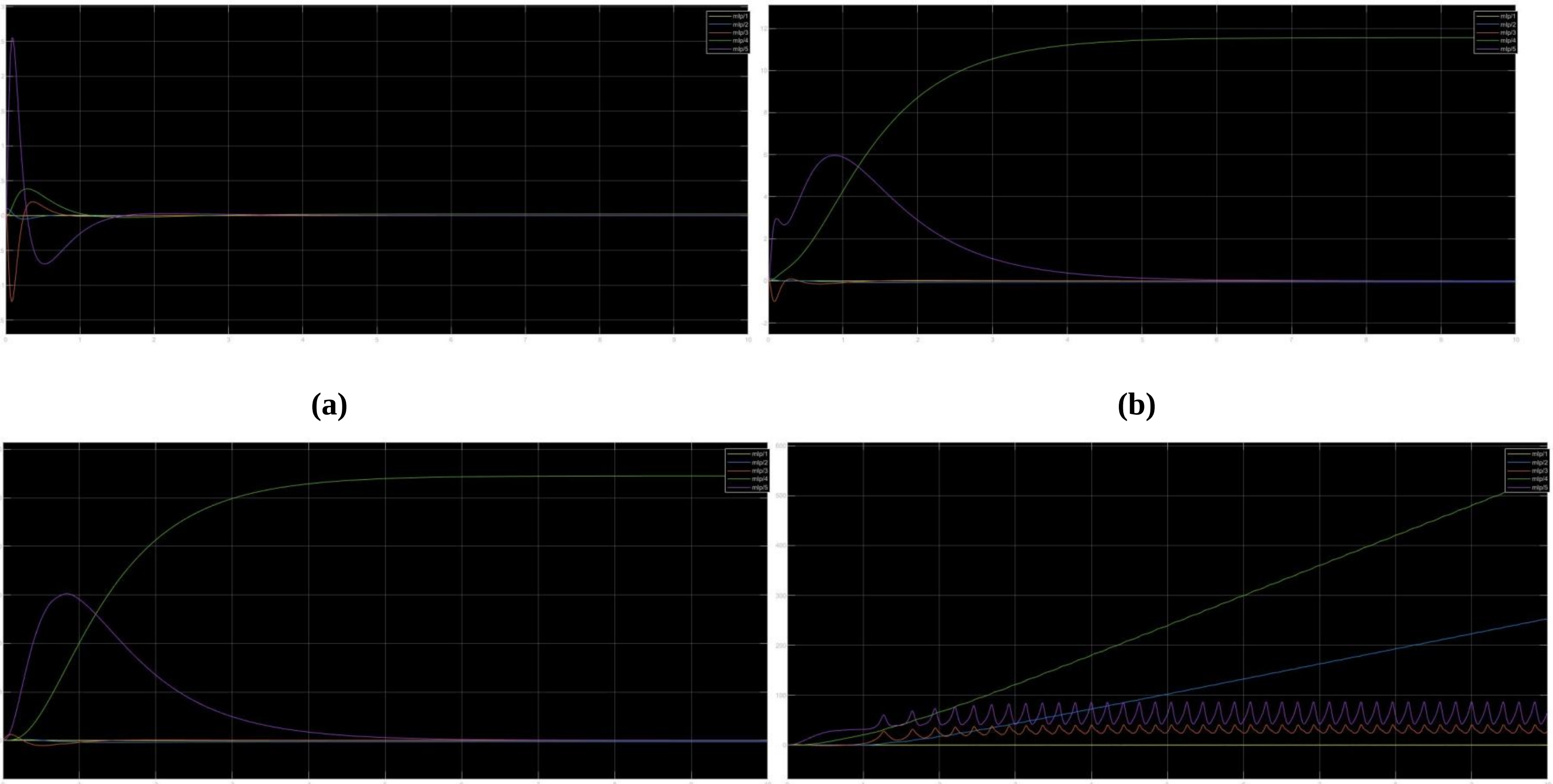
TASK 7:

After making the reference, noise at zero again, we now increase the bias of the wind to give a constant wind disturbance to the system. We get the anticipated result from the plots of the system's states. We see that '∅' and 'θ' equilibrium positions have been changed from zero. This behaviour is due to facts that :

i) MIP will be leaning towards the wind source so that gravity's torque can balance the wind torque and the system can remain at equilibrium.

ii) As the initial conditions were zero, the MIP has to travel some distance as 'θ' is stabilising at a nonzero value and '∅' has to change so that the base movement can make the MIP, not fall over completely.

In the following plots we can find the effect of wind-bias on the system as it is increased. We see as we increase the bias, ∅'s equilibrium position will shift further and further away from zero. After a certain value of the bias (in our system it is around 23.5), the system will be destabilised completely and fall over.



(a) **(b)**

(c) **(d)**

Figure 8: (a) ,(b), (c) : System's stable yet nonzero equilibrium state vector's plots. And (d) is the unstable behaviour as we cross a certain value of the bias.

TASK 8: Making a HOMING MIP

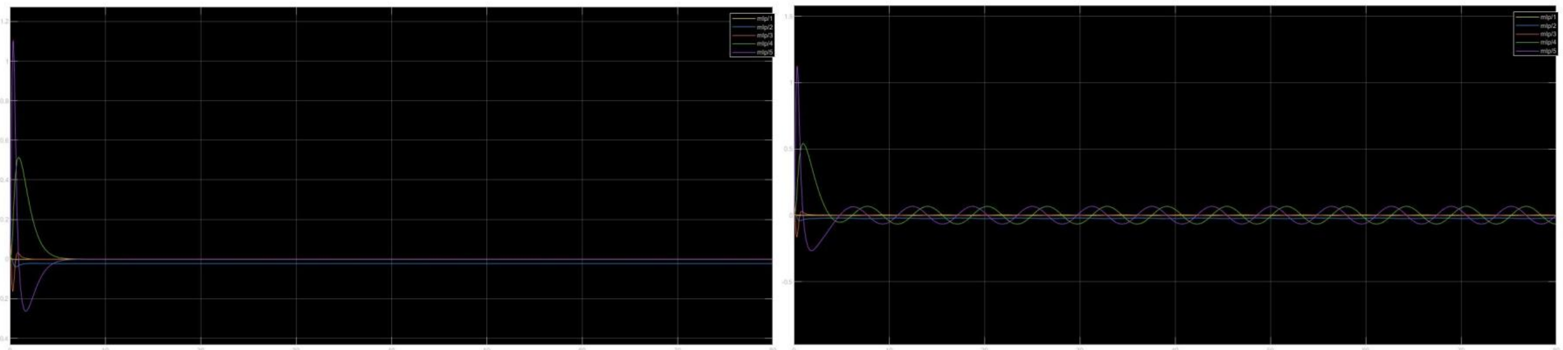
Part 1: WITH Constant wind:

The wind is an unknown disturbance affecting the angular acceleration associated with θ or the orientation of the MIP and then eventually affecting ∅. So we can include this disturbance as an extra entry in our state equations and create a higher dimensional system where we can estimate this disturbance (since we can calculate the observability of the system and see the new state is observable, although it is not controllable) and then subtract the estimate while giving the feedback through the controller. (As we can define a new system in this way In this way we can gradually make the ∅ to be equal to zero.

Part 2: WITH Constant + Sinusoidal wind:

Here there are 3 unknowns in the system that are the constant bias of the wind, the wind amplitude and the wind phase. So three new states have to be inserted in the system so that we can estimate it (luckily, in our case this new, bigger system is observable and not controllable) and only control the states of the MIP we need to control by subtracting the effect of the wind from the equation of .angular acceleration corresponding to θ.

But in our case we can deal with both the systems with a simple consideration. When there is wind , ∅ is getting stabilised around a nonzero value which gives a nonzero area of ∅-t plot .We can insert only one new state in the system for both cases which is the area under ∅-t plot (basically the system acts on an integrator on ∅) and now we can stabilise this new state (the state of area under the ∅-t plot) to zero (by inserting a new stable pole in the new K matrix). This new system is not observable yet controllable. Now as this new state is stabilised, the ∅ value also gets stabilised around 0. (Although for any sinusoidal type disturbance of the wind ∅ will keep changing as θ changes and ∅ is measured from the body axis of the MIP). In the last figure in the following, we can see for both cases of the wind, that the system is getting stabilised around ∅ =0 (it may not stay at zero always but the MIP will be 'homing' always).



(a) **(b)**

Figure 9 : (a) denotes the states with constant bias wind and (b) the states with constant + sinusoidal wind

Now the MATLAB code will be given below for verification of all the actions I have taken in MATLAB, to perform all the tasks. This matlab code, if run without any changes may not give the desired result for all of the tasks in the assignment as the code was getting updated as newer tasks were being attempted.

```
% Student Name: Soumya Ganguly
% PID : A53274333
% Final Project
% Linear Systems Theory (MAE 280A), Fall 2018
% Final Assignment

clc
% TASK 0: reference and wind generators are set to zero and system response
% has been recorded by defining the following:

sys=ss(ALDmIp,BLDmIp,CLDmIp,DLDmIp,0.005)


% TASK 1:
% part 1:
o1=obsv(sys)    % observability matrix
r_o=rank(o1)
if(r_o==5)
    disp('The system is observable')
else
    disp('The system is not observable')
end

%part2:
r1=ctrb(sys) %reachability matrix
r_r=rank(r1)
if(r_r==5)
    disp('The system is reachable')
else
    disp('The system is not reachable')
end

% part3 :
e_given=eig(ALDmIp);
eigIlike_K=0.95*e_given;
eigIlike_L=0.95*e_given;

K_new=place(ALDmIp,BLDmIp,eigIlike_K) % K Matrix for the feedback
L_new=(place(ALDmIp',CLDmIp',eigIlike_L))' % L matrix for our controller-
observer


%TASK2
sys2=ss(ALDmIp-BLDmIp*K_new-L_new*CLDmIp, L_new, K_new, 0,0.005);

%in the last command we are defining the sys2 which acts as the
%feedback-gain
sys_feedback=feedback(sys,sys2);%command to get a feedback system

% simulation of feedback system
ti=0:0.005:20;
x0=[0;0.1;0;0;0;0;0.1;0;0;0];
u=zeros(1,length(ti));
plot1=lsim(sys_feedback,u,ti,x0,'zoh'); %simulating with zero order hold

figure(1)
subplot(2,2,1);
```

```
plot(ti,plot1(:,1))
title('$\theta$ $state (discrete)$','Interpreter','latex');
ylabel('$\theta$','Interpreter','latex');
xlabel('$time(seconds)$','Interpreter','latex');

subplot(2,2,2);
plot(ti,plot1(:,2))
title('$\phi$ $state (discrete)$','Interpreter','latex');
ylabel('$\phi$','Interpreter','latex');
xlabel('$time(seconds)$','Interpreter','latex');

%TASK 3
syscont=d2c(sys_feedback,'zoh');
plot2=lsim(syscont,u,ti,x0,'zoh');

subplot(2,2,3);
plot(ti,plot2(:,1))
title('$\theta$ $state (CONT)$','Interpreter','latex');
ylabel('$\theta$','Interpreter','latex');
xlabel('$time(seconds)$','Interpreter','latex');

subplot(2,2,4);
plot(ti,plot2(:,2))
title('$\phi$ $state (CONT)$','Interpreter','latex');
ylabel('$\phi$','Interpreter','latex');
xlabel('$time(seconds)$','Interpreter','latex');

%TASK 4
% eigIlike_K=0.955*e_given
% eigIlike_L=0.9545*e_given

K_new=place(ALDmIp,BLDmIp,eigIlike_K); % K Matrix for the feedback
L_new=(place(ALDmIp',CLDmIp',eigIlike_L))'; % new L matrix for our
controller-observer
KLDmIp=K_new;
LLDmIp=L_new;
ALCBKLDmIp =ALDmIp-BLDmIp*KLDmIp-LLDmIp*CLDmIp;



%TASK 5
sys3=ss(ALDmIp-BLDmIp*K-L*CLDmIp, L, K, 0,0.005);
sys_feedback2=feedback(sys,sys3);
plot3=lsim(sys_feedback2,u,ti,x0,'zoh');

figure(2)
plot4=plot(ti,plot1(:,1))
title('$\theta$ $state$ $(discrete)$ $comparison$','Interpreter','latex');
ylabel('$\theta$','Interpreter','latex');
xlabel('$time(seconds)$','Interpreter','latex');
hold on;
plot(ti,plot3(:,1),'.')

figure(3)
plot5=plot(ti,plot1(:,2))
title('$\phi$ $state$ $(discrete)$ $comparison$','Interpreter','latex');
ylabel('$\phi$','Interpreter','latex');
xlabel('$time(seconds)$','Interpreter','latex');
hold on;
plot(ti,plot3(:,2),'.')
```

```
% Task 7 : done in simulink
% TASK 8 :
ALDmIp1=[ALDmIp [0;0;0;0;0];[0 0 0 0.005 0 1]];
BLDmIp1=[BLDmIp;0];
CLDmIp1=[CLDmIp [0;0]];
syswind=ss(ALDmIp1,BLDmIp1,CLDmIp1,DLDmIp,0.005);
% owind=obsv(syswind);   % observability matrix
% r_owind=rank(owind)
%
% rwind=ctrb(syswind); %reachability matrix
% r_cwind=rank(rwind)
eigIlike_kwind=[eigIlike_K; 0.951];
eigIlike_lwind=[eigIlike_L;1];
K_wind=place(ALDmIp1, BLDmIp1, eigIlike_kwind); % K Matrix for the feedback
L_wind=(place(ALDmIp1',CLDmIp1',eigIlike_lwind))'; % new L matrix for our
controller-observer
KLDmIp1=K_wind;
LLDmIp1=L_wind;
ALCBKLDmIp1=ALDmIp1-BLDmIp1*KLDmIp1-LLDmIp1*CLDmIp1;
```