

ELEC-E8101 Group project:

Lab B report

Group 05

Swaminathan Aravind, Kamath Abhijit Krishnananda, Karal Puthanpura JithinLal Dev

November 20, 2019

Reporting of Task 5.1

Initially we faced issues on communicating with the robot as we have Ubuntu systems. After the updated model files from Lukas and team the connection issues were solved.

The PID controller was tested with our parameters but it did not work as expected. The robot was not able to self balance properly. We started tuning the PID parameters and spent some time on analysing the performance of the robot. Finally the PID was tuned with the following values: $kP = 274$; $kI = 530$; $kD = 0.1312$

With these values the robot was performing better than our initial case. It was able to balance for about 5-8 seconds. But it was observed that the robot was not smooth enough and the time for which the robot could balance itself was varying in each experiment.

Below given are the screenshots of x_w, θ_b , and u plots for different trials. We had to run it using the Monitor&Tune functionality and the screenshots were taken from the scope. Note: the different screenshots are for different trials. Figure 1,2,3 represent trial 1,2,3 with different kP, kI, kD values.

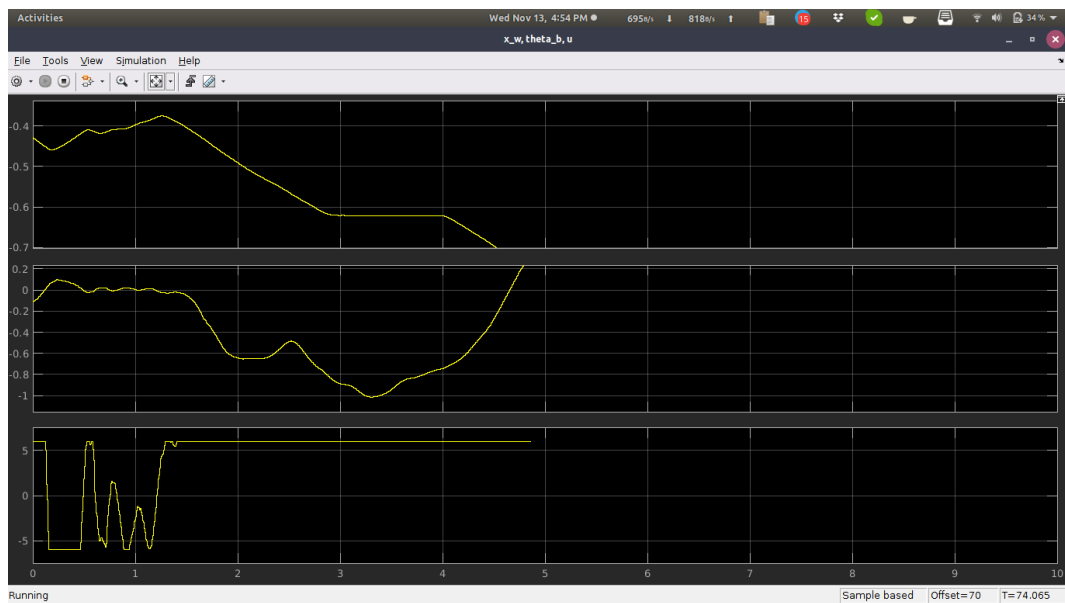


Figure 1: x_w, θ_b and u for trial 1



Figure 2: x_w, θ_b and u for trial 2

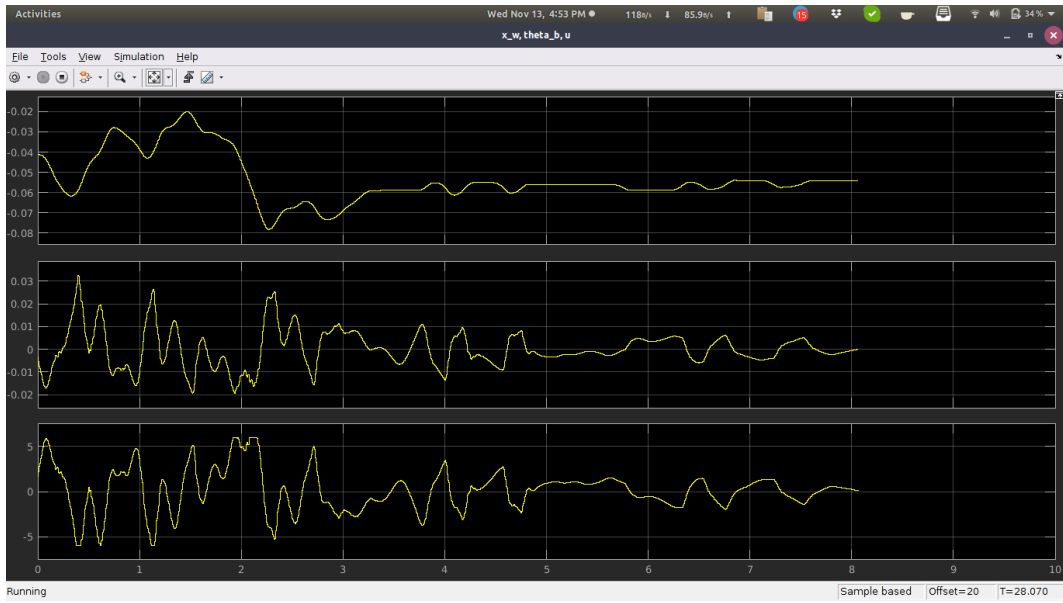


Figure 3: x_w, θ_b and u for trial 3

Reporting of Task 5.2

5.2.1 : To calculate the controllability matrix , the following formula is used

$$Co = [B \quad AB \quad AB^2 \quad AB^3]$$

The values for A and B are obtained from Report 1 in task 4.1 (which is after linearising the equations).

The numerical value of Controllability matrix is

$$Co = \begin{bmatrix} 0 & 36.5979568516 & 890- & -30865.3766078564 & 26031758.4724007 \\ 36.5979568516 & -30865.3766078564 & 26031758.4724007 & -21955189521.4189 & \\ 0 & -156.7072230094 & 132161.133337975 & -111469744.004147 & \\ -156.7072230094 & 132161.133337975 & -111469744.004147 & 94013600843.3055 & \end{bmatrix}$$

To calculate the Observability matrix , the following formula is used

$$Ob = \begin{bmatrix} C \\ CA \\ CA^2 \\ CA^3 \end{bmatrix}$$

The values for A and C are obtained from Report 1 in task 4.1 (which is after linearising the equations)

The numerical value of Observability matrix is

$$Ob = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 3313.2384293412 & 63.0719380049227 & -69.5780070161652 \\ 0 & -2794263.9620029 & -26168.0482800051 & 58742.6151400659 \end{bmatrix}$$

5.2.2:

a. In the A matrix which was designed initially, it is seen that system is not controllable for \dot{x}_w . i.e there is no input signal u to control the wheel velocity The system is also not observable for wheel velocity state \dot{x}_w because the wheel position term x_w is zero in C matrix.

b. The system is controllable for wheel acceleration \ddot{x}_w because of the non zero term is second column in B matrix, but it is not observable, as the corresponding term in C matrix is zero.

c. The body angle velocity $\dot{\theta}_b$ is not controllable because of zero term in the corresponding element in B matrix but it is observable because of non zero θ_b term in C matrix.

d. The body angle acceleration $\ddot{\theta}_b$ is controllable because of non-zero term in the corresponding element in B matrix but it is not observable because of zero θ_b term in C matrix.

Reporting of Task 5.3

5.3.1 :

We selected the poles as [-5.6 -843.1 -4 -4]. There is no specific algorithm for the pole location process. Since we don't have much depth knowledge on pole placement techniques, we had followed the strategies mentioned in the hint as to move the poles close to slowest pole. We had poles initially at [-5.6 -843.1 +5.6 0], as the hint suggested that we had to move the poles to the slowest one. We assumed that in the list of poles obtained initially, the slowest pole(dominant pole) is approximately -5.6 and we tried to move the poles close to it. Also the fastest pole is kept as such and it is not moved. The speed of the slower pole is a good starting point to proceed with. Also it is a fact that poles should not be moved too much to avoid saturation. So only poles at $s = +5.6$, $s = 0$ was moved to $s = -4$, $s = -4$ respectively.

5.3.2:

There are two functions available in matlab to compute the gain matrix, either acker() or place(). We used the acker() function to compute the gain matrix K as the poles we selected are in same position. place() doesn't accept poles in same location.

The value of gain matrix K is [-59.1007 -61.3170 -87.7624 -14.4053]

5.3.3: The plots for the values of θ_b and v_m are given in Figure 4 and Figure 5 respectively.

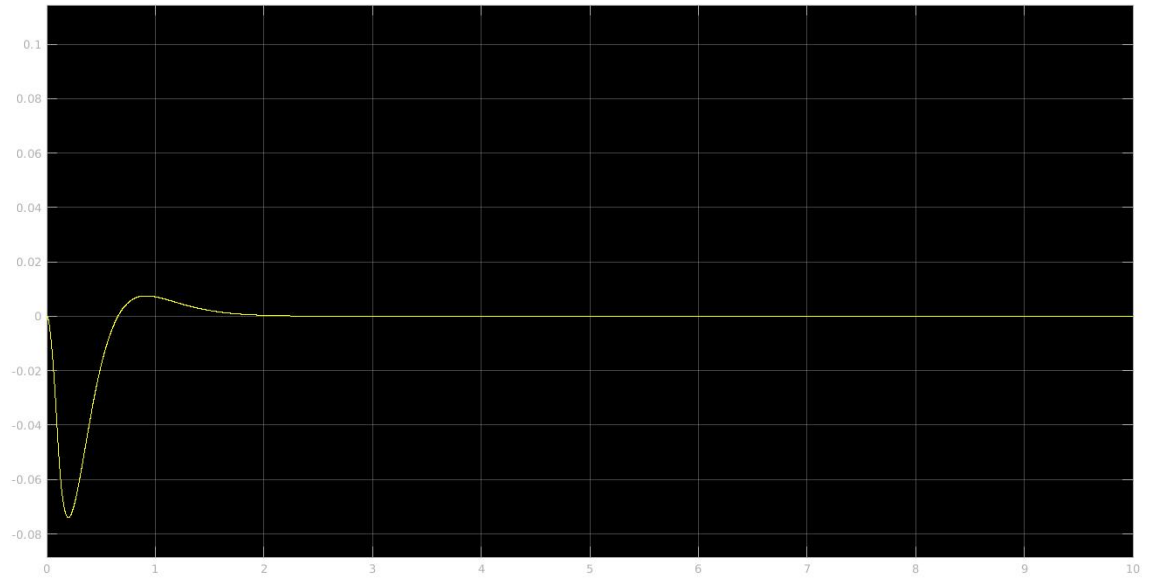


Figure 4: θ_b for new poles

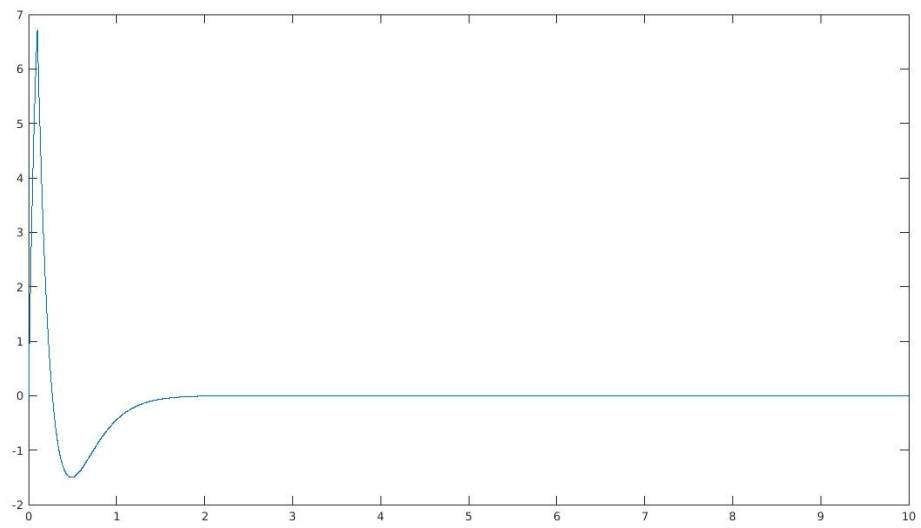


Figure 5: v_m for new poles

Reporting of Task 5.4

5.4.1: Full Order Luenberger Estimator:

For the Luenberger Observer, we had considered the pole location ten times than the original poles of the system. We got this information from online sources about the Luenberger observer. We are not sure of the exact reason to do this.

Then gain matrix is

$$L = \begin{bmatrix} 384.941250082315 & -1567.75501669812 \\ -278857.59262065 & 1268903.31030057 \\ -1590.8873439352 & 7337.99536946813 \\ 1189444.05488631 & -5409778.94606582 \end{bmatrix}$$

Reduced Order Estimator :

The C matrix was given in the lab book. And the derivations were not easy for the calculations of $M_{1...7}$ Matrices.

The following are considered

$$C_{acc} = \begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix}$$

$$T^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

and the further derivations are proceeded with as in the lab book.

The numerical forms of $M_{1...7}$ matrices are:

$$M1 = \begin{bmatrix} -8388.3898002557 & -487.343355617259 & 16.2494928420906 \\ -497.325737760301 & -69.132192728133 & 1 \\ 6961.80252242957 & 406.450785980973 & -69.5780070161652 \end{bmatrix}$$

$$M2 = \begin{bmatrix} 36.5979568515555 \\ 0 \\ -156.707223009381 \end{bmatrix} \quad M3 = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \quad M4 = \begin{bmatrix} 480.769838799966 \\ 69.132192728133 \\ -343.37884797605 \end{bmatrix}$$

$$M5 = \begin{bmatrix} 7614.60442682281 \\ 497.325737760301 \\ -3648.56409308837 \end{bmatrix} \quad M6 = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad M7 = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

5.4.2 : The plots of θ_b and x_w for the system and their estimates from the full and reduced order estimators are shown in the figure 6 and Figure 7 respectively.

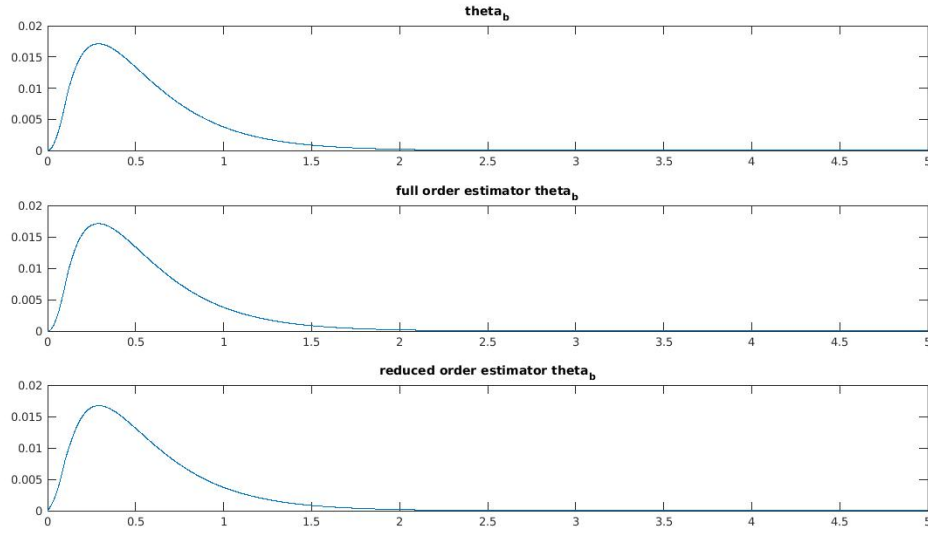


Figure 6: θ_b for system and estimators

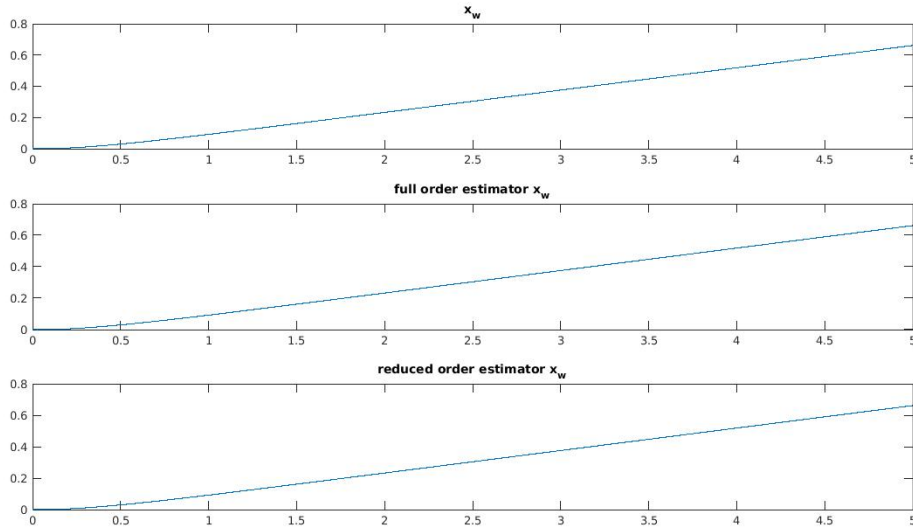


Figure 7: v_m for system and estimators

From the graphs, we can conclude that the estimator's performance is good and the error from both the estimators are small.

5.4.3: The maximum error values in the estimation of the θ_b and x_w calculations are given in table below

max error	θ_b (in radians)	x_w (in meters)
full estimator	0.000125016539042703	0.000186744795577537
reduced estimator	0.000416522270894195	0

Reporting of Task 5.5

5.5.1: To convert the system into a discrete one, we used the Matlab function `c2d(system,sampling period)`. The sampling period chosen was 0.01s. The exact matlab code used for the discretization of the system is:

```
fSamplingPeriod = 0.01;
sys = ss(A,B,C,D);
sysd = c2d(sys,fSamplingPeriod);
```

The exact code is found in *LabB_ControllerOverSimulator_Discrete_Parameters.m* in the appendix column.

The sysd computed will have the A,B,C,D matrices,

```
A_d= sysd.A;
B_d = sysd.B;
C_d= sysd.C;
D_d= sysd.D
```

And the values of the matrices are,

$$A_d = \begin{bmatrix} 1 & 0.00191196177196008 & -4.21218748235277e-05 & 0.000169621066998641 \\ 0 & 0.082546983900223 & -0.00186998343993353 & 0.0192243914632718 \\ 0 & 0.0346482610531582 & 1.00192723405754 & 0.00927918378200914 \\ 0 & 3.93383068165404 & 0.357495177512275 & 0.919316789742805 \end{bmatrix}$$

$$B_d = \begin{bmatrix} 0.00038254234862351 \\ 0.0433930480587732 \\ -0.00163876910386559 \\ -0.186059559267421 \end{bmatrix}$$

$$C_d = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

$$D_d = [0]$$

5.5.2: The values of K_d , L_d and $M_{d1...7}$ are computed as follows.

To compute the values of K_d , the poles in z domain are computed first using the formula $z = e^{p\Delta}$ where Δ is sampling period and p represent poles in continuous domain.

The computed values of poles for discrete system are

$$polesd = [0.0002 \quad 0.9455 \quad 0.9607 \quad 0.9607]$$

With these poles, the acker command in Matlab was used to get the K_d value. The matlab code to compute is

```
polesd = exp(oldpoles*fSamplingPeriod);
Kd = acker(sysd.A,sysd.B,polesd);
```

The value of K_d is

$$K_d = [-55.2247 \quad -58.9646 \quad -84.4460 \quad -13.8549]$$

The values of L_d and the other values of $M_{d1...7}$ are computed the same way how we did for the task 5.4.1

The values of L_d and $M_{d1...7}$ are as follows:

$$L_d = \begin{bmatrix} 0.0392281407642617 & 0.00112085704411749 \\ 0.00666472203676182 & 0.0120308255026752 \\ 0.00126018689514763 & 0.0972268493632884 \\ -0.0230793367436434 & 0.547097936486273 \end{bmatrix}$$

$$M_{d1} = \begin{bmatrix} 0.0827153794708072 & -0.32947909087155 & 0.0192393307977184 \\ 0.033921468457569 & 0.816624058422631 & 0.00921470585292158 \\ 3.92831121355546 & 0.439014349621558 & 0.918827126151264 \end{bmatrix}$$

$$M_{d2} = \begin{bmatrix} 0.0434267403835957 \\ -0.00178418464017825 \\ -0.187163885891807 \end{bmatrix}$$

$$M_{d3} = \begin{bmatrix} 0.0880747581117465 \\ -0.380129250619974 \\ -2.88680881570271 \end{bmatrix}$$

$$M_{d4} = \begin{bmatrix} 0.32760539755768 \\ 0.18531918739162 \\ -0.0813975743097094 \end{bmatrix}$$

$$M_{d5} = \begin{bmatrix} -0.0880747581117465 \\ 0.380129250619974 \\ 2.88680881570271 \end{bmatrix}$$

$$M_{d6} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

$$M_{d7} = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

5.5.3:

The Graphs of x_w , θ_b and u for the complete system which uses observer and the controller are displayed below in Figure 8,9 and 10 respectively.

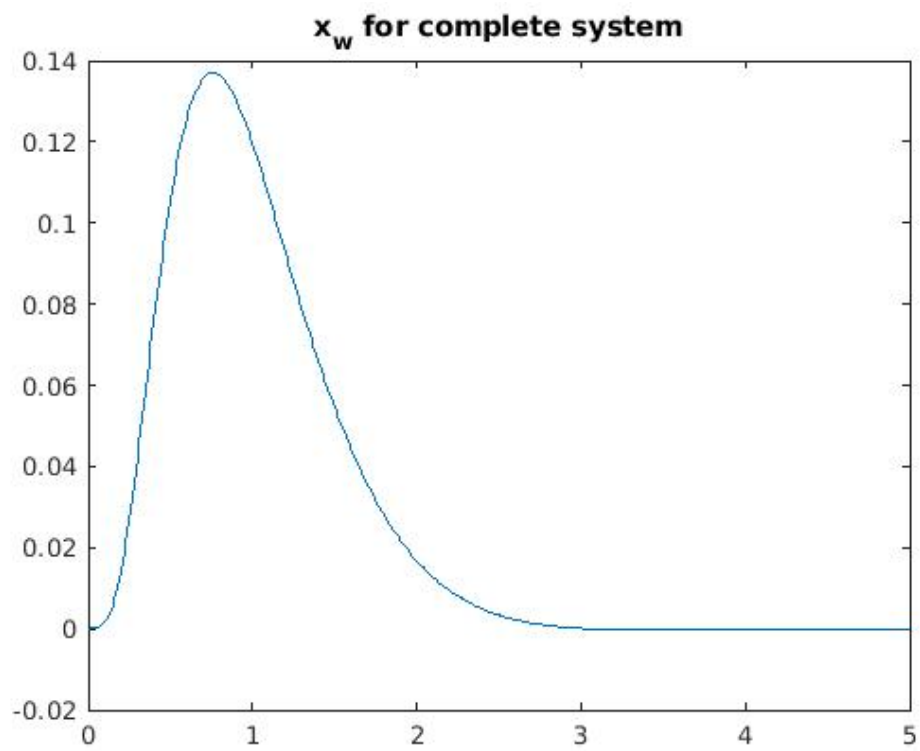


Figure 8: x_w for complete system

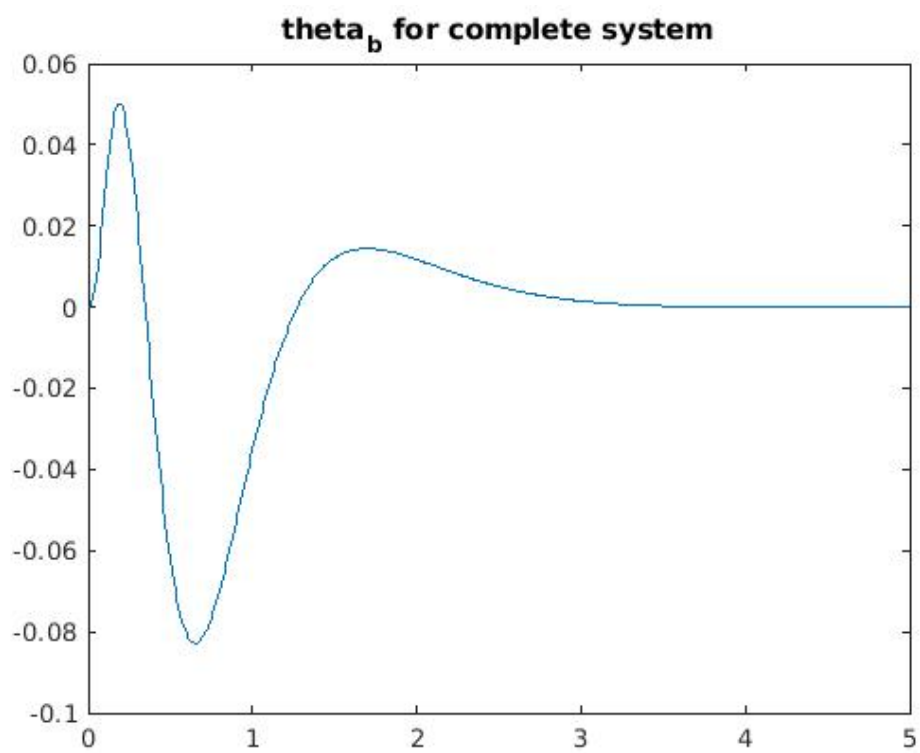


Figure 9: θ_b for complete system

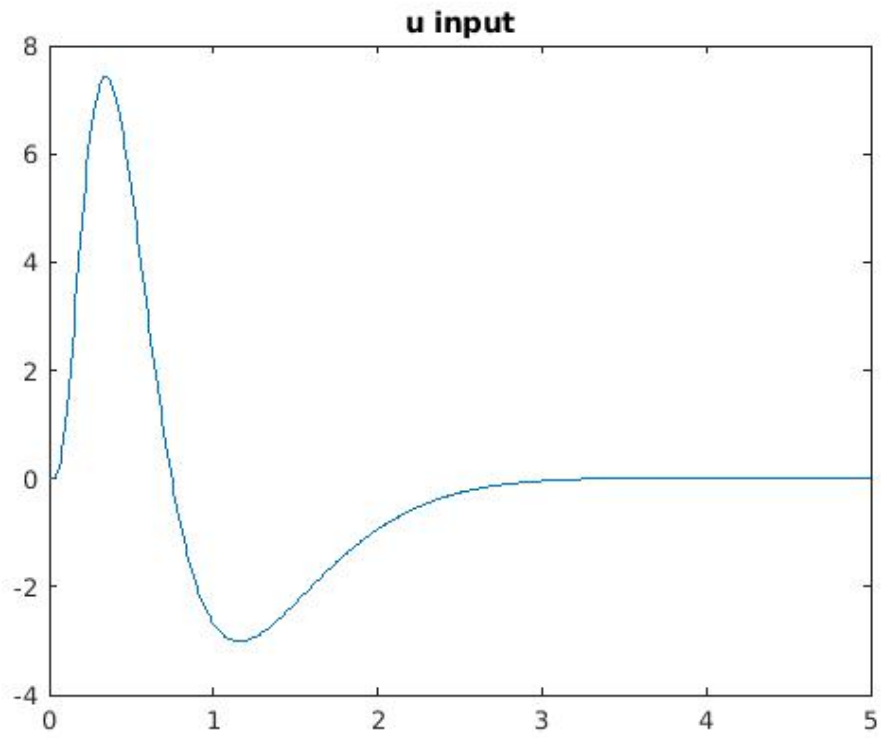


Figure 10: u for complete system

Reporting of Task 5.6

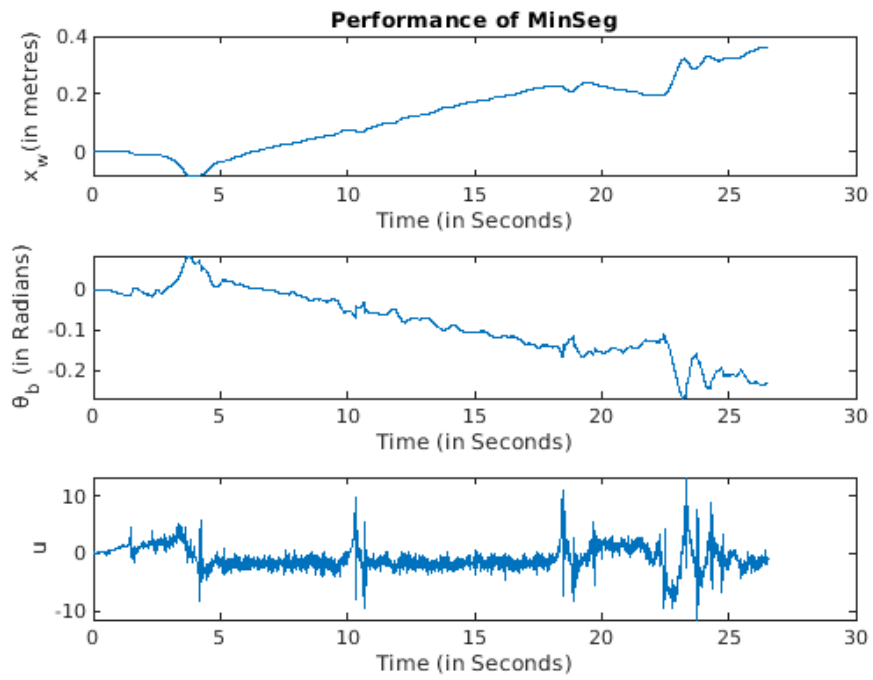


Figure 11: x_w, θ_b, u

A full order Luenberger observer was used in the robot as it showed better performance

in the robot. The reduced order Luenberger observer was also tested in the robot, but it was not working as it was working in the full order case. The drift in the gyro was causing the robot to move away but was corrected by calibration of the gyro as mentioned in the Troubleshoot 5.5. Still a small amount of drift was observed which could be neglected on the basis of poor sensor accuracy. The values of x_w , θ_b and u values from the robot are shown in Figure 11.