Control Systems 414 Practical 1

Xari Mouchtouris
23795824

Chapter 1

Introduction:

This practical considered designing and testing a discrete feedback control system for a simulated plant. Two different controllers are designed, namely a discrete state space controller with reference input, and a discrete state space controller with integral control. The controllers are evaluated to identify which performs the best at tracking a reference single. Furthermore an external disturbance gets introduced, to test the disturbance rejection of each controller. The results for disturbance rejections are discussed, as well as the transient response and steady-state tracking error. Figure 1.1 below illustrates the Laser following the input reference signal (Alien).

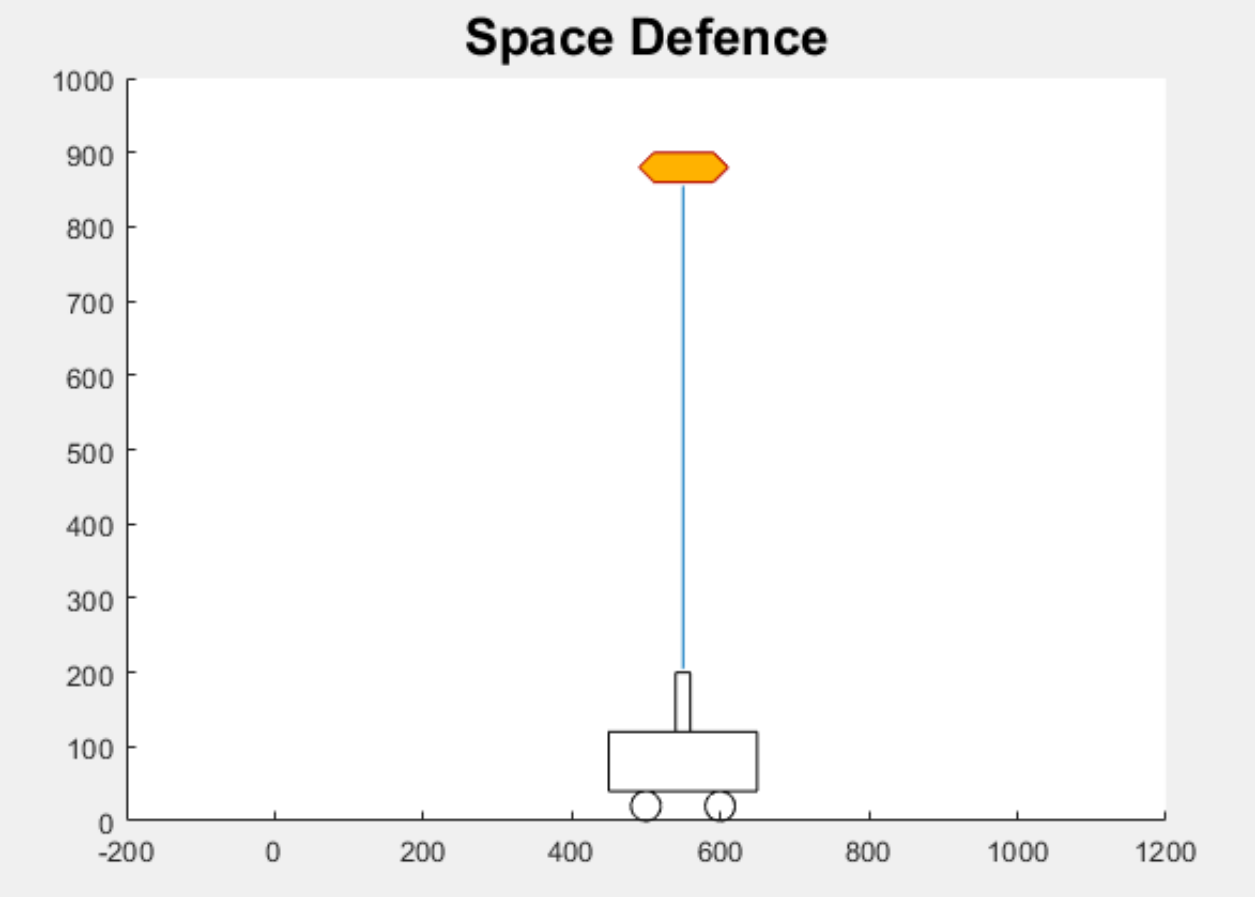

Figure 1.1: Laser and Alien:

Q1: Discrete State Space Model

State Vectors: $u(t) = f(t)$
See below $\mathbf{X}$ and $\dot{\mathbf{X}}$ for the laser cannon motion:

$$\mathbf{X} = \begin{bmatrix} x \\ \dot{x} \end{bmatrix} \quad \dot{\mathbf{X}} = \begin{bmatrix} \dot{x} \\ \ddot{x} \end{bmatrix}$$

Using the horizontal motion differential equation of the laser, we can calculate the matrices **A**, **b**, **c** and d.

$$m\ddot{x} = f(t)$$

$$\ddot{x} = \frac{1}{m} f(t)$$

Now, the continuous state space model is the following:

$$\begin{bmatrix} \dot{x} \\ \ddot{x} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \times \begin{bmatrix} x \\ \dot{x} \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{m} \end{bmatrix} \times u(t)$$

$$y(t) = \begin{bmatrix} 1 & 0 \end{bmatrix} \times \begin{bmatrix} x \\ \dot{x} \end{bmatrix} + 0 \cdot u(t)$$

Given: $m = 0.02\,\text{kg}$

The final matrices are with the values plugged in:

$$\mathbf{A} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}; \mathbf{b} = \begin{bmatrix} 0 \\ 50 \end{bmatrix}; \mathbf{c} = \begin{bmatrix} 1 & 0 \end{bmatrix}; \mathbf{d} = 0$$

Q2: Discrete State Space Model

In this question, the continuous state space model will be converted to the discrete state space model, using Matlab's **ss** & **c2d** functions. The sampling period used is specified as **T** = 0.1 s. The matrices for **F**, **g**, **c** and d will be listed.

The following matrices for **F**, **g**, **c** and d, is achieved after being converted using Matlab:

$$\mathbf{F} = \begin{bmatrix} 1 & 0.1 \\ 0 & 1 \end{bmatrix}; \mathbf{g} = \begin{bmatrix} 0.25 \\ 5 \end{bmatrix}; \mathbf{c} = \begin{bmatrix} 1 & 0 \end{bmatrix}; \mathbf{d} = 0$$

The Matlab code to convert it to a discrete state space model, can be seen below in figure 1.2.

```
m = 0.02;
T = 0.1;
% Continuous State Space Model
A = [0 1; 0 0];
b = [0; 1/m];
c = [1 0];
d = 0;
% Discrete State Space Model
sysC = ss(A, b,c, d);
sysD = c2d(sysC, T, 'zoh');
% Extract discrete matrices
F = sysD.A;
g = sysD.B;
c = sysD.C;
d = sysD.D;
```

Figure 1.2: Discrete State Space Conversion Matlab Code

Q3: State Feedback with Reference Input:

The goal is to design a discrete state space controller with a reference input that satisfies the following closed loop specifications:

- $\zeta = 0.707$ (Optimally Damped)
- 2 % settling time: $t_S = 1s$

To achieve this, we first write Matlab code to calculate the closed loop poles for the discrete state space controller, and then we calculate the feedback gain vector **k**. Lastly an expression is derived to calculate the feedforward gain $\bar{N}$. The **k** feedback gain vector and the feedforward gain $\bar{N}$, can be seen below.

$$k = \begin{bmatrix} 0.4292 & 0.1316 \end{bmatrix} \qquad \bar{N} = 0.4292$$

Thus, the feedforward gain $\bar{N} = 0.4292$, $k_1 = 0.4292$ and $k_2 = 0.1316$. The Matlab code to calculate the feedforward gain and the feedback gain vector can be seen below, in Figure 1.3.

```
% Closed Loop Poles Calculations
zeta = 0.707;
ts = 1; % 2% settling time
wn = 4/(zeta*ts);
s_cl = [-(zeta*wn)+(i*wn*sqrt(1-zeta^2));-(zeta*wn)-(i*wn*sqrt(1-zeta^2))];
z_cl = exp(s_cl * T);
I = [1 0 ; 0 1];
%Determine k
k = place(F,g,z_cl);
% Feedforward gain
N = 1/(c*inv(I - F + g*k)*g);
```

Figure 1.3: **k** and $\bar{N}$ Matlab Calculation Code

Q4: State Feedback with Reference Input:

The goal is to design a current state observer so that the observer error dynamics will converge within 2 sample periods. For the implementation of this, both the observer poles should be placed at z = 0. The prediction observer gain vector m_p will be calculated using Matlab. Then an expression will be used to calculate the current observer gain m_c. The prediction observer gain vector m_p and the current observer gain m_c can be seen below.

$$m_p = \begin{bmatrix} 2 \\ 10 \end{bmatrix} \qquad m_c = \begin{bmatrix} 1 \\ 10 \end{bmatrix}$$

The Matlab code to calculate m_p and m_c can be seen below in Figure 1.4.

```
% Current state observer
V = obsv(F,c);
if det(V) ~= 0
    disp("Observable");
end
Observer_Poles = [0;0];
mp = acker(F', c', Observer_Poles)';
mc = inv(F)*mp;
```

Figure 1.4: m_p and m_c Matlab Calculation Code

Q5: State Feedback with Reference Input:

The discrete state space controller with reference input was implemented in the **SpaceDefence.slx** code and the simulation was run, to evaluate if the controller works. The Simulink model can be seen below in figure 1.5.

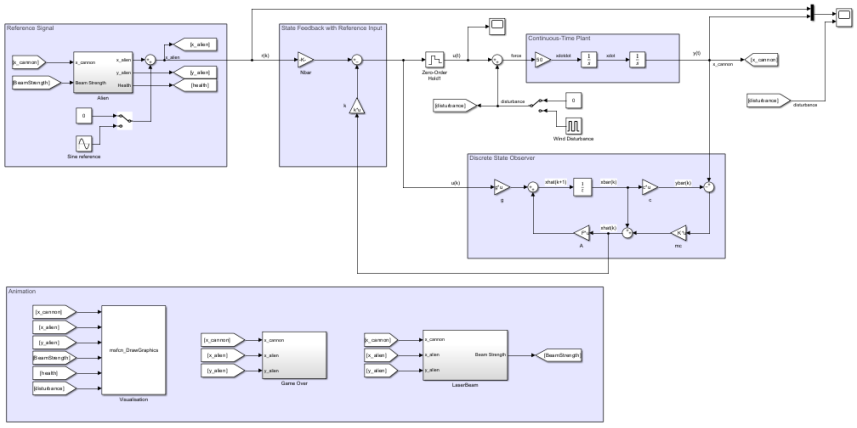

Figure 1.5: Simulink Matlab Model (Without Integral Controller)

Q6: State Feedback with Reference Input:

The system is simulated with a step reference signals and no external disturbances (No Wind). The simulated response can be seen in figure 1.6 below. As can be seen in the figure without any disturbance, the steady-state tracking error is low, and the laser controller tracks the refrence input good.

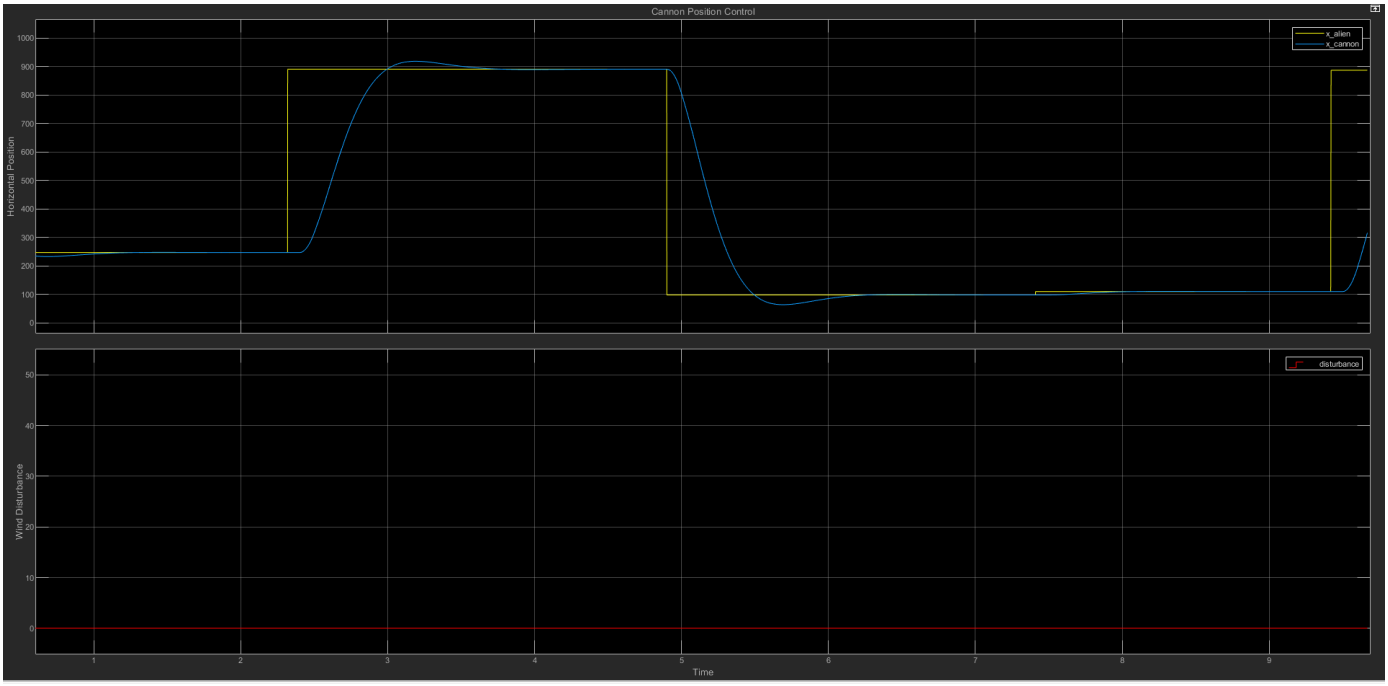

Figure 1.6: State Feedback Controller Simualtion (Without Disturbance)

Figure 1.7 below illustrates the maximum overshoot and Figure 1.8 illustrates the settling time of the state feedback controller.

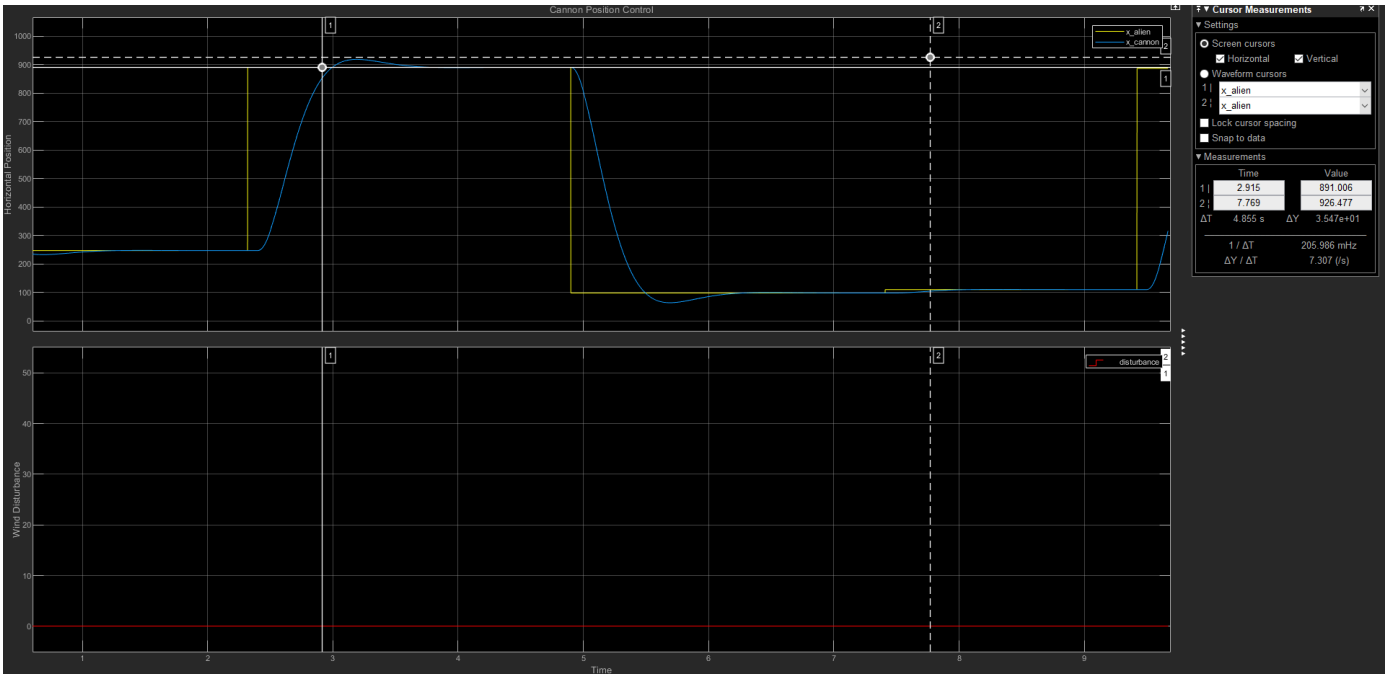

Figure 1.7: State Feedback Controller Maximum Overshoot

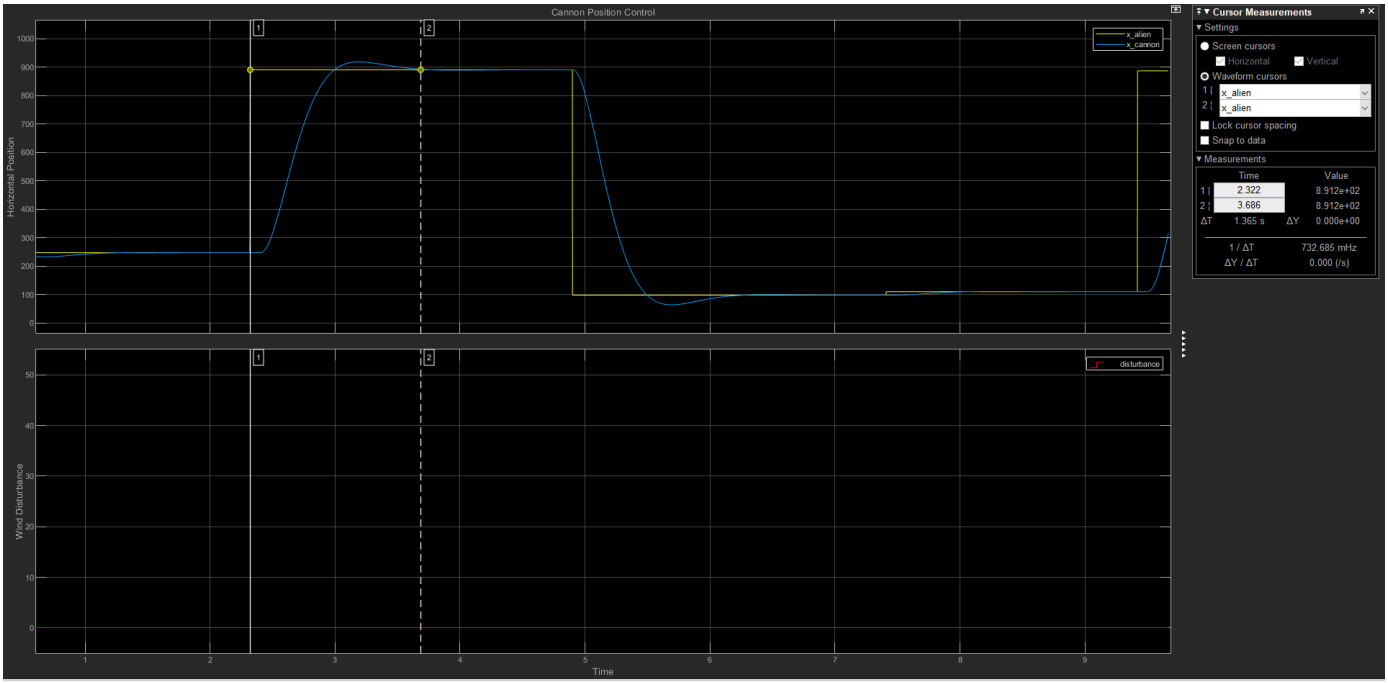

Figure 1.8: State Feedback Controller Simulation Settling time

The results for the simulink Maximum Overshoot and Settling Time is the following. The settling time is $t_s = \mathbf{1.365\ s}$ and a maximum overshoot of $M_p = 35.47$. The simulated $t_s = \mathbf{1.365\ s}$ is close to the design 2% settling time of $t_s = \mathbf{1\ s}$. The theoretical M_p can be calculated as $M_p = e^{-\frac{\zeta\pi}{\sqrt{1-\zeta^2}}}$. The theoretical value is calculated as $M_p = 43.25$, which is very close to the simulated value of $M_p = 35.47$. Thus, this means that the closed loop response satisfies the specification, for the transient response, because the simulated values agree with the theoretical calculated overshoot and settling time. The steady tracking error is low, and

the state feedback follows the reference input well, when there is no disturbance present.

Q7: State Feedback with Reference Input:

The system is simulated with a step reference signals and external disturbances (Wind). The simulated response can be seen in figure 1.9.

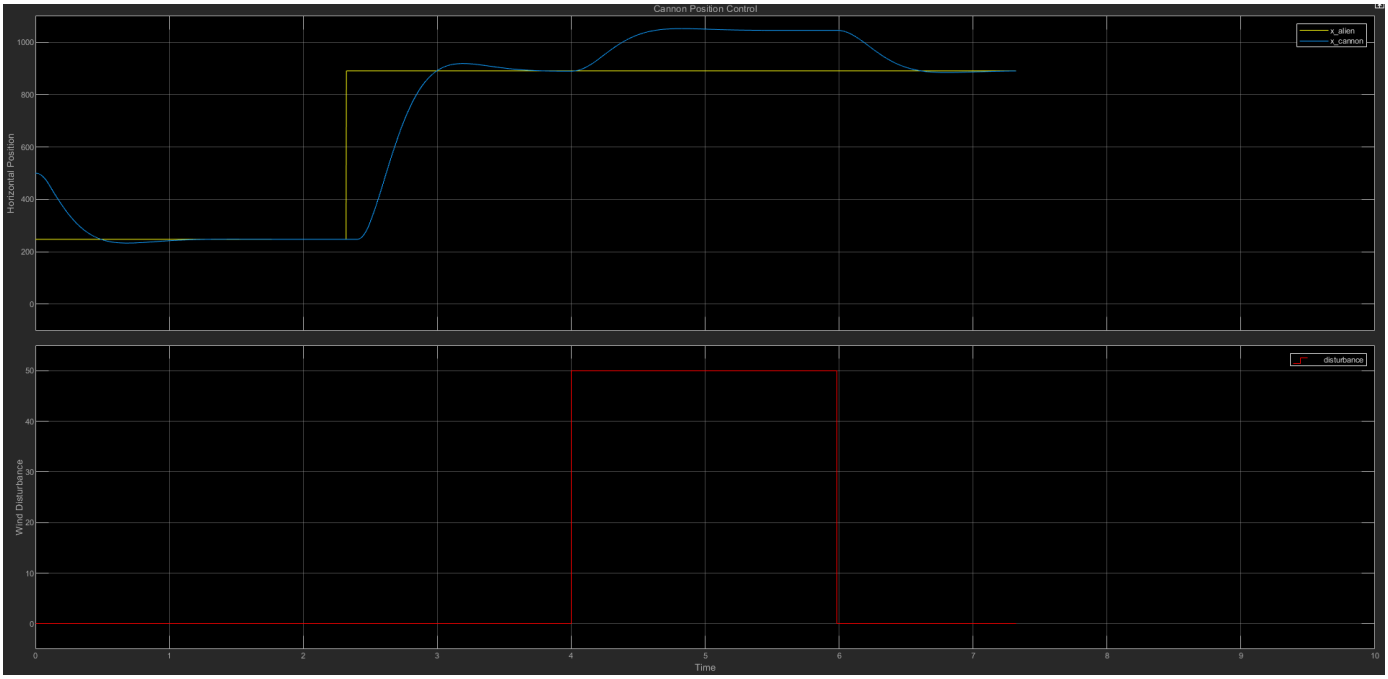

Figure 1.9: State Feedback Controller Simualtion (With Disturbance)

As can be seen above in Figure 1.9 with disturbance, the laser controller struggles to track the reference input when the external disturbance (wind) is applied. The steady tracking error is high when the disturbance step (wind) is applied to the state feedback controller, and the feedback controller can't follow the reference input anymore. once the disturbance (wind) isn't applied anymore, the state feedback controller follows the input reference well again. If a constant disturbances (wind) is applied to the controller, the feedback controller won't be able to track the the reference input at all.

Q8: State Feedback with Reference Input:

The reference input of the system is now changed to a **sinusoidal reference signals** with changing frequency. The different frequencies are f = (0.2 Hz; 0.4 Hz ; 1 Hz; 2 Hz). The figures below illustrates the different feedback controller simulations at the different frequencies.

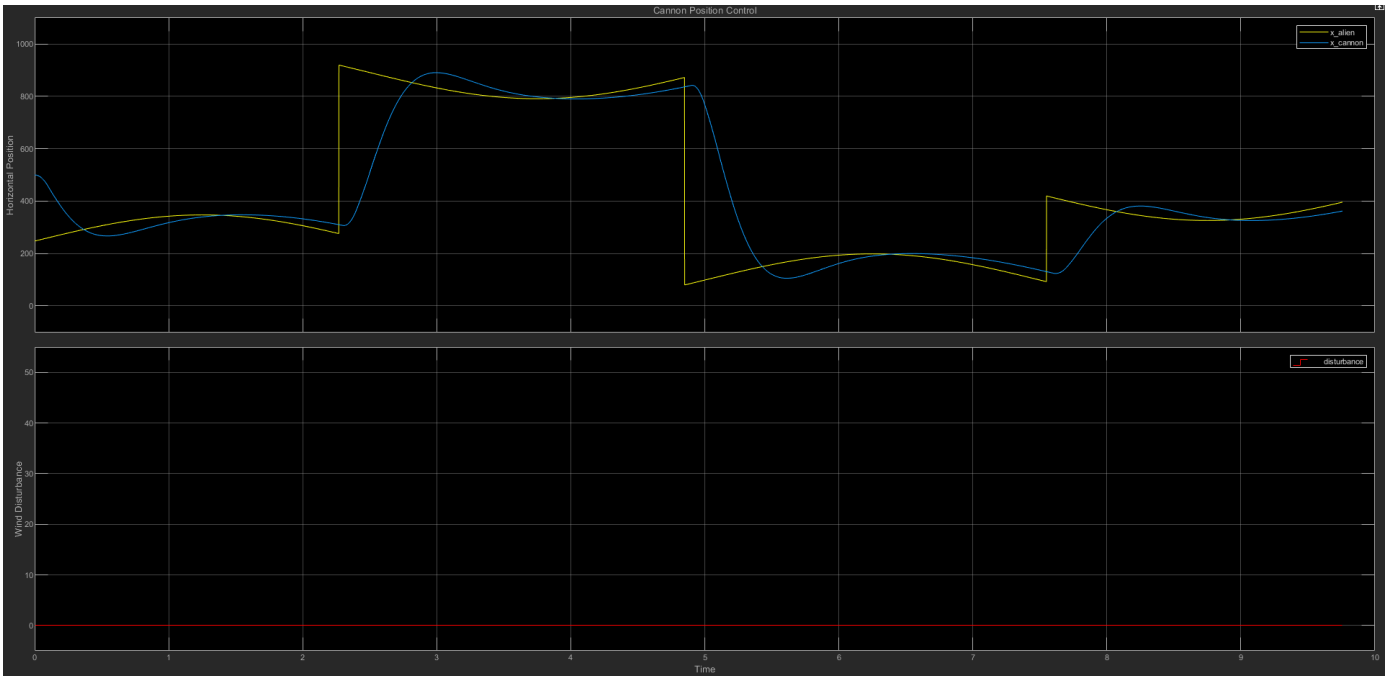

Figure 1.10: Sinusoidal Reference at f = 0.2 Hz

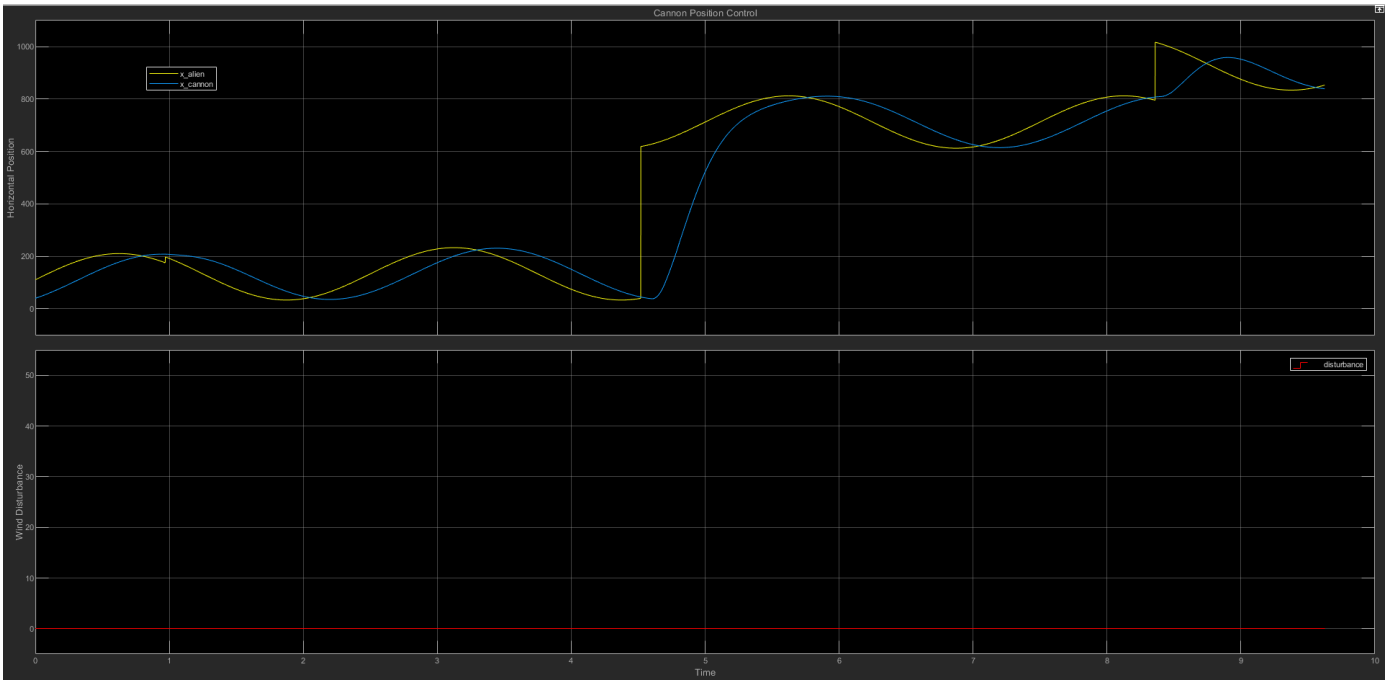

Figure 1.11: Sinusoidal Reference at f = 0.4 Hz

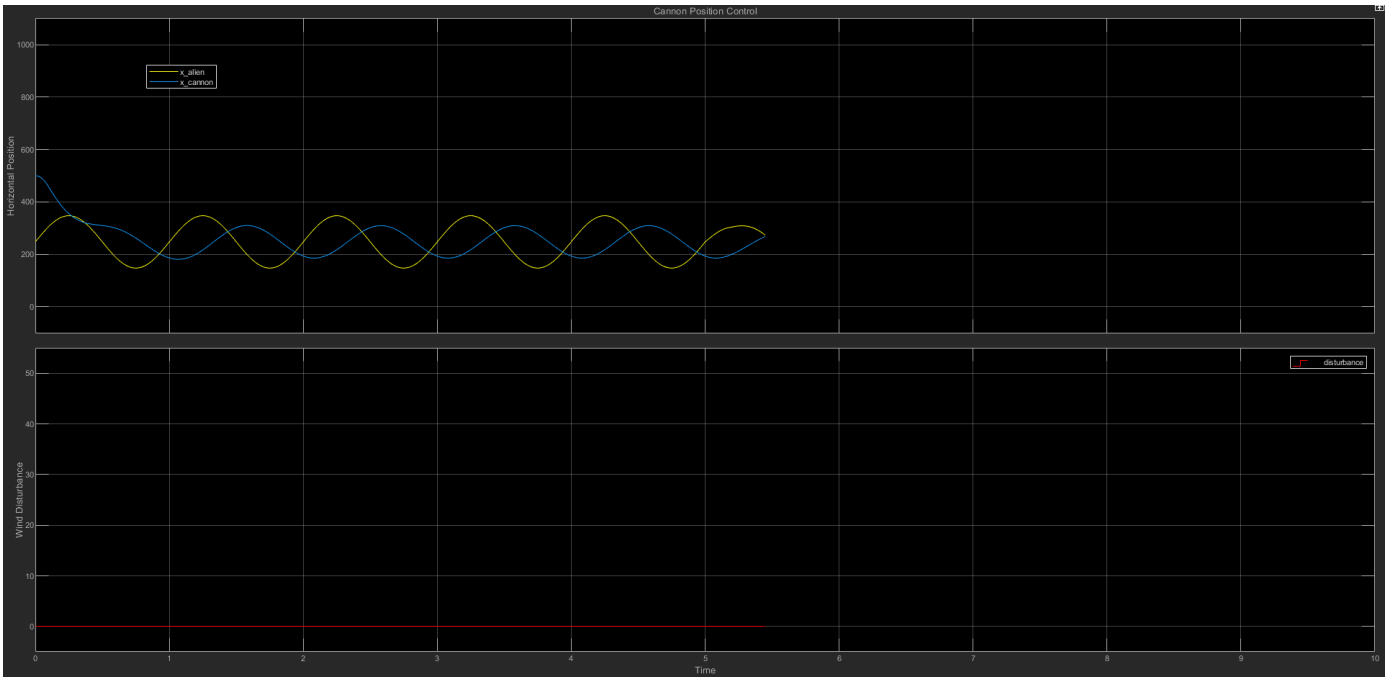

Figure 1.12: Sinusoidal Reference at f = 1 Hz

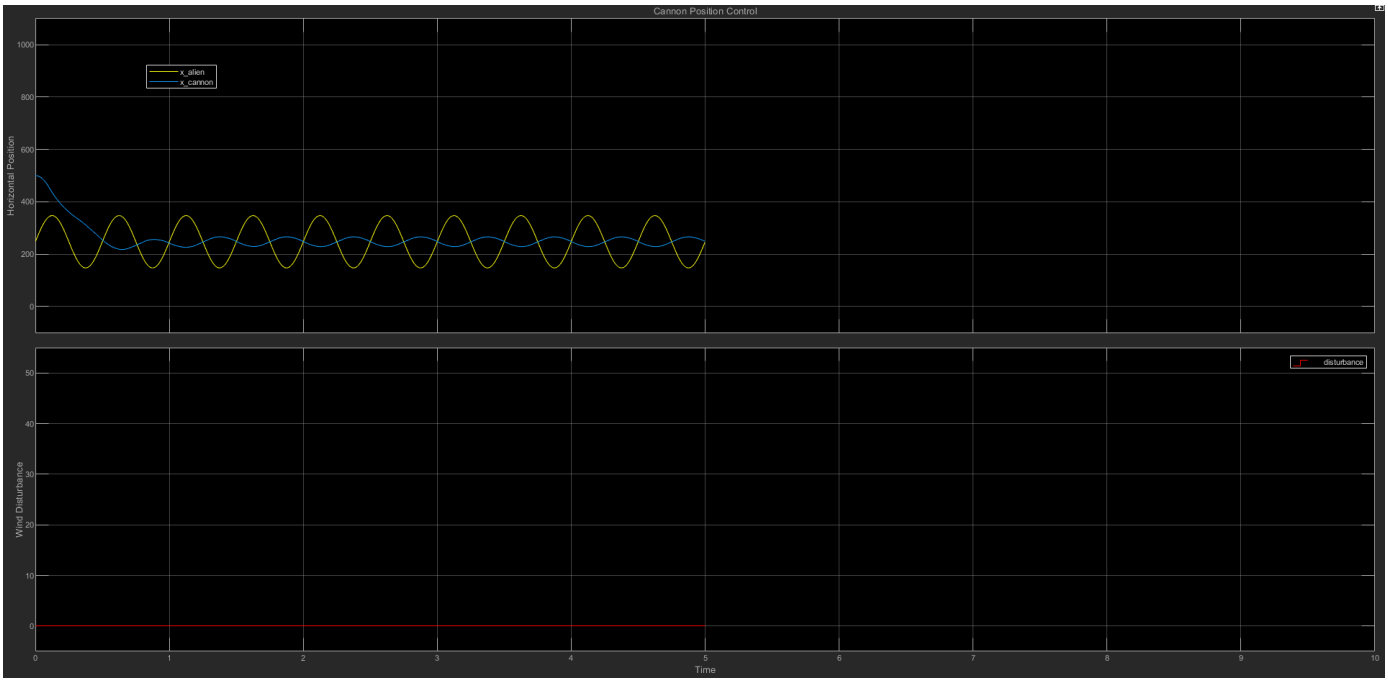

Figure 1.13: Sinusoidal Reference at f = 2 Hz

As can be seen in figure 1.10 above the feedback controller follows the input Sinusoidal Reference at f = 0.2 Hz well. The amplitude of both the input and the feedback controller is almost the same. The phase of both is almost the same of both input reference and the controller. The controller reacts quite fast to a change in amplitude of the reference input.

Figure 1.11 at f = 0.4 Hz, also still show the feedback controller follows the reference quite well. The amplitude is also almost the same, but there is a bit of a noticeable phase shift,

with a slight delay between the controller and the input reference.

For Figure 1.12 (f = 1 Hz) and 1.13 (f = 2 Hz), the feedback controller doesn't follow the input reference well at all. There is a big difference in amplitude between the feedback controller and the input reference. There is also a big phase shift between both. The controller takes a long time to respond to any amplitude change of the reference.

Q9: State Feedback with Integral Control:

Now, an Integral Controller is designed with the integrator closed closed loop pole at $Z_{Integrator} = 0.9$.

Q10: State Feedback with Integral Control:

Now, I wrote Matlab code to calculate the augmented matrix, F ($\bar{F}$) and the augmented matrix for g ($\bar{g}$). The augmented matrices for F and g can be seen below.

$$\bar{\mathbf{F}} = \begin{bmatrix} 1 & 0.1 & 0 \\ 0 & 1 & 0 \\ 1 & 0.1 & 1 \end{bmatrix}; \bar{\mathbf{g}} = \begin{bmatrix} 0.25 \\ 5 \\ 0.25 \end{bmatrix}$$

The Matlab code to calculate ($\bar{F}$) and ($\bar{g}$) can be seen below in figure 1.14.

```
% Integral Control
z_integral = 0.9;
zero = [0;0];
Fbar = [F zero ; c*F 1];
gbar = [g ; c*g];
```

Figure 1.14: $\bar{F}$ and $\bar{g}$ Matrices Matlab Code

Q11: State Feedback with Integral Control:

Now, I wrote Matlab code to calculate the augmented feedback gain vector, ($\bar{K}$) and the state feedback gain **k** and the integral gain **ki**. $\bar{K}$, **k** and **ki** can be seen below.

$$\bar{\mathbf{K}} = \begin{bmatrix} 0.5178 & 0.1450 & 0.0429 \end{bmatrix}; \bar{\mathbf{k}} = \begin{bmatrix} 0.5178 & 0.1450 \end{bmatrix}; \bar{\mathbf{ki}} = \begin{bmatrix} 0.0429 \end{bmatrix}$$

The Matlab code to calculate $\bar{K}$,$\bar{k}$ and $\bar{ki}$ can be seen below in figure 1.15.

```
z_i = [z_cl; z_integral];
kbar = place(Fbar, gbar, z_i);
kp = kbar(1:2);
ki = kbar(3);
```

Figure 1.15: $\bar{K}$,$\bar{k}$ and $\bar{ki}$ Matlab Code

Q12: State Feedback with Integral Control:

Next, I wrote Matlab code to calculate the the feedforward gain, $\bar{N}$ for the integral controller. The equation below illustrates how $\bar{N}$ is calculated.

$$\bar{N} = \frac{k_i \cdot z_{\text{Integral}}}{1 - z_{\text{Integral}}} = 0.3862$$

The Matlab code to calculate $\bar{N}$, can be seen below in Figure 1.16.

```
Nbar = (ki*z_integral)/(1- z_integral);
NbarInt = Nbar;
```

Figure 1.16: $\bar{N}$ Matlab Code

Q13: State Feedback with Integral Control:

The discrete state space controller with Integral Control was implemented in the **SpaceDefenceIntegral.slx** code and the simulation was run, to evaluate if the controller works. The Simulink model can be seen below in figure 1.17.

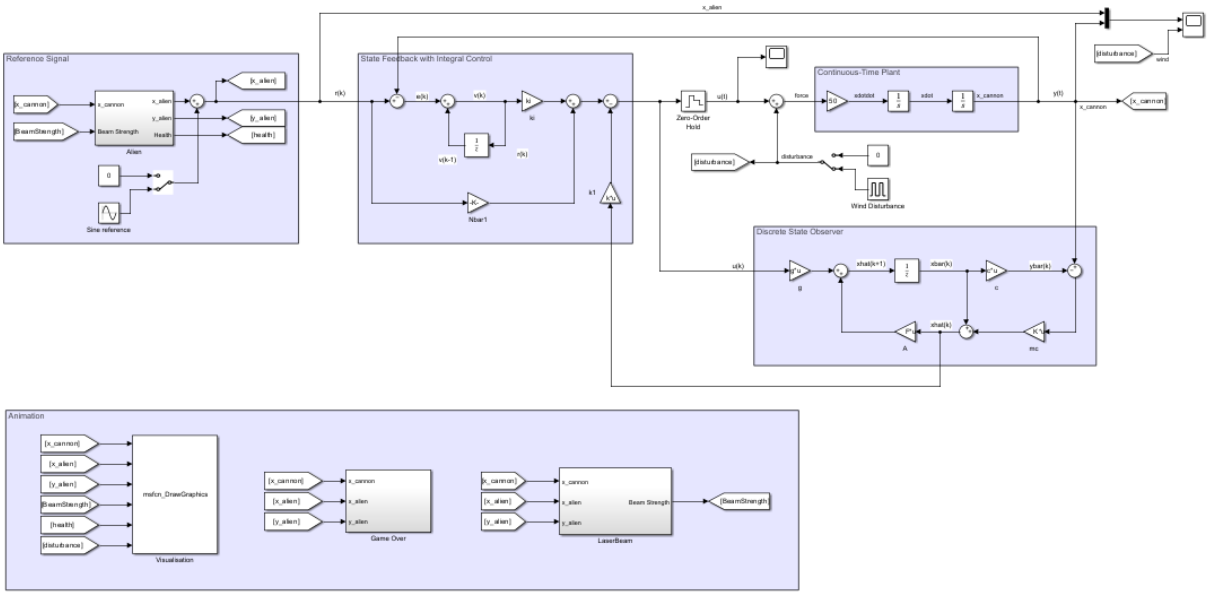

Figure 1.17: Simulink Matlab Model (Integral Controller)

Q14: State Feedback with Integral Control:

(a). Step reference signals and no external disturbances.

The state feedback with Integral Control is simulated with no external disturbances (No Wind). The simulated response can be seen in figure 1.18 below. As can be seen in the figure

without any disturbance, the steady-state tracking error is low, and the laser controller tracks the reference input well.

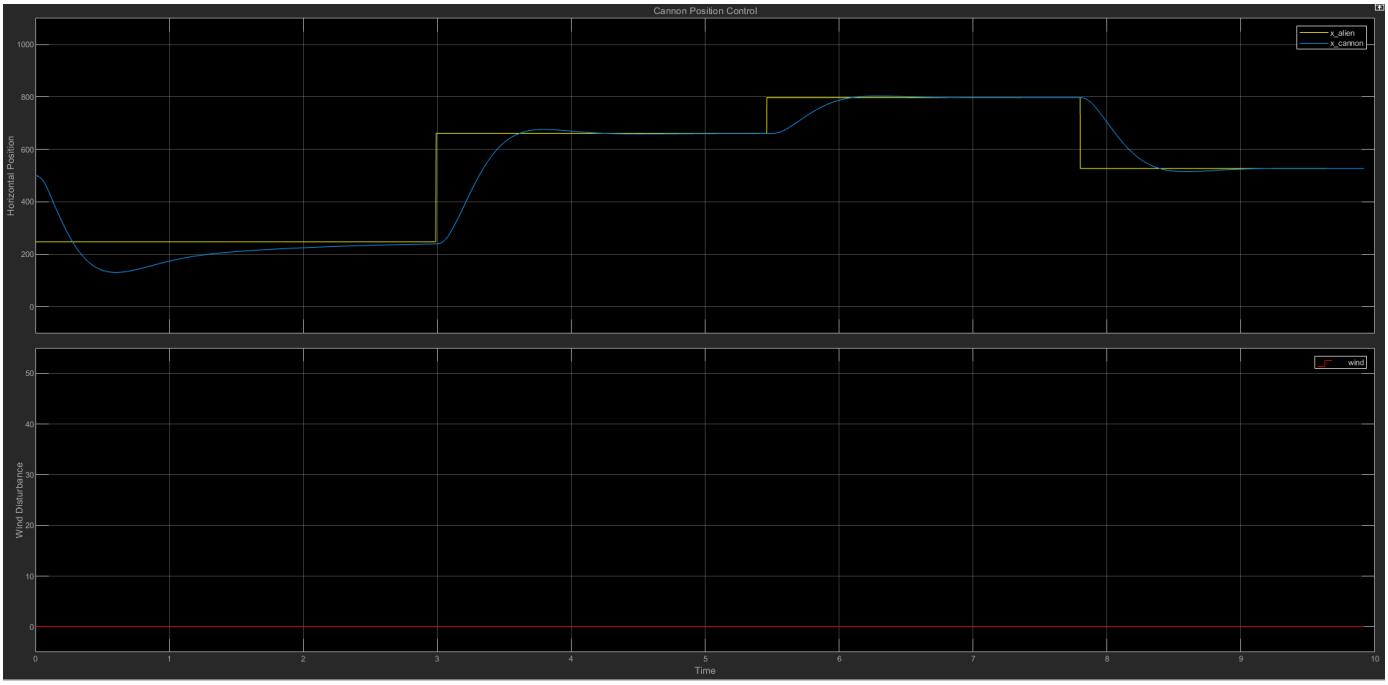

Figure 1.18: State Feedback Integral Controller Simulation (Without Disturbance)

Figure 1.19 below illustrates the maximum overshoot and Figure 1.20 illustrates the settling time of the state feedback controller.

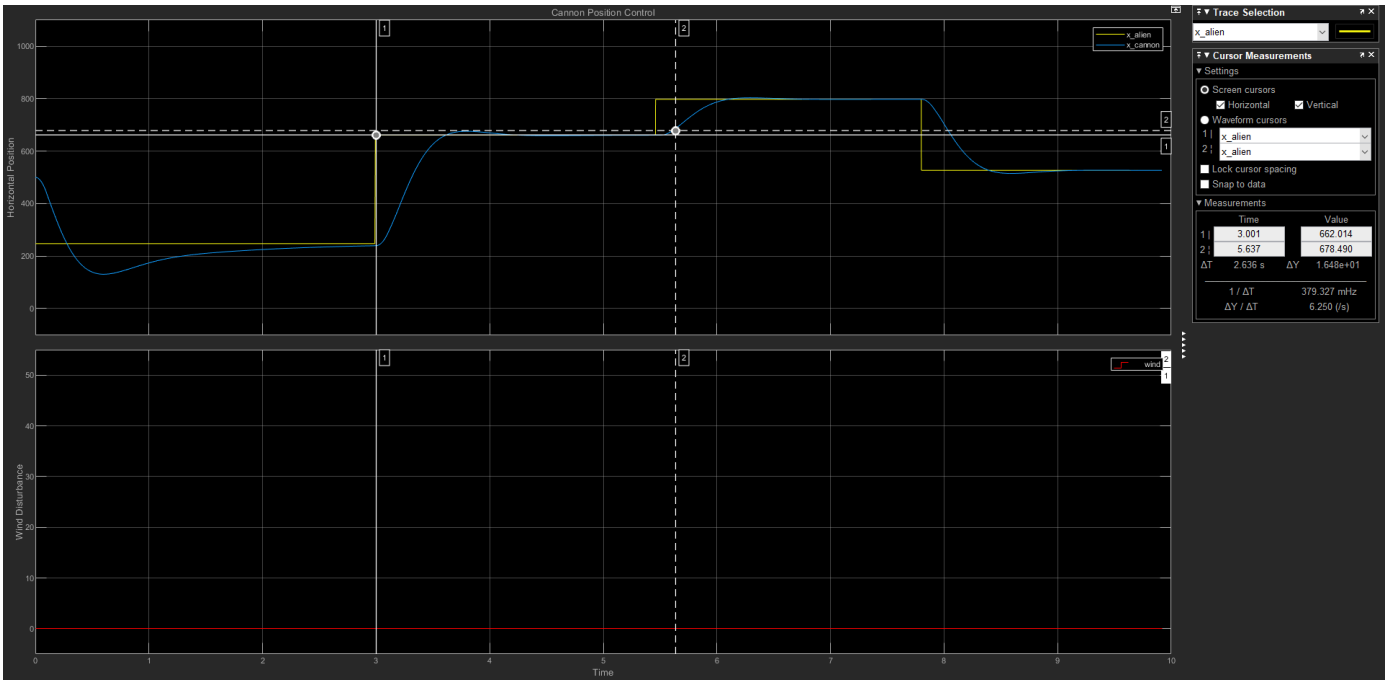

Figure 1.19: State Feedback Integral Controller Maximum Overshoot

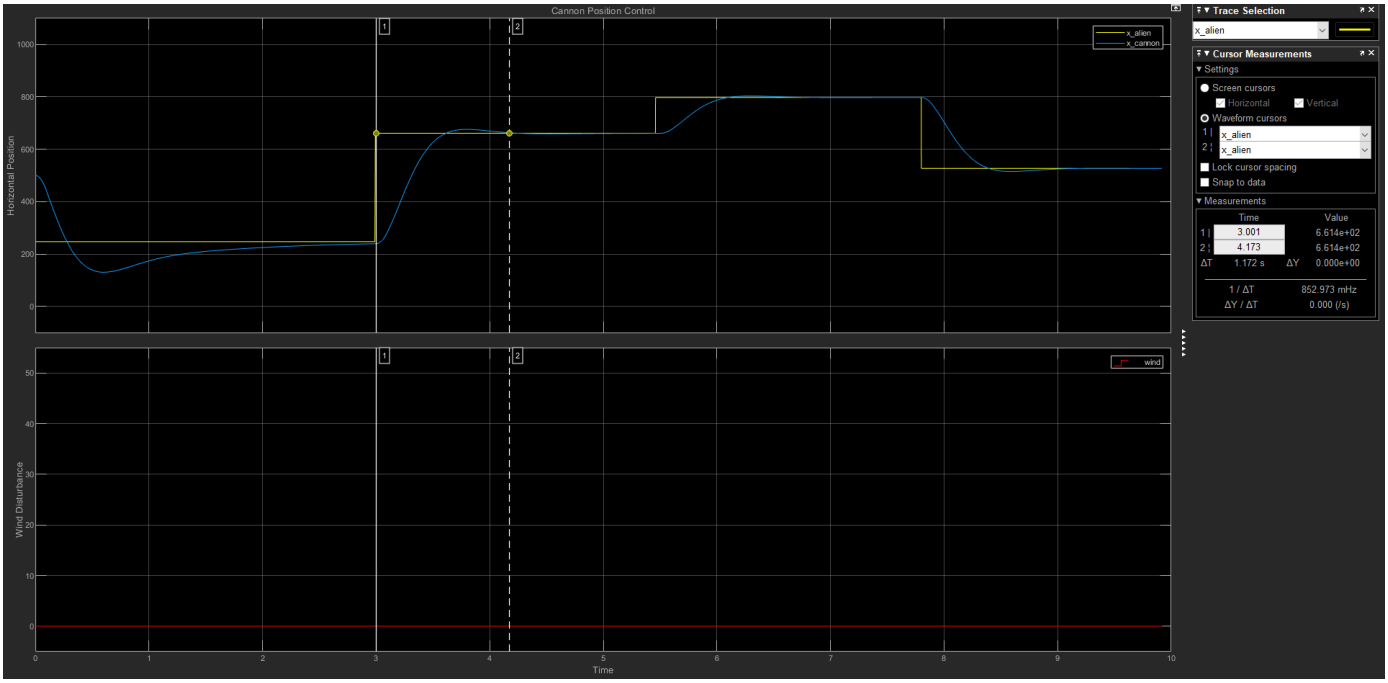

Figure 1.20: State Feedback Integral Controller Simulation Settling time

The results for the simulink Maximum Overshoot and Settling Time is the following. The settling time is t_s = **1.172 s** and a maximum overshoot of $M_p = 16.48$. The simulated t_s = **1.172 s** is close to the design 2% settling time of t_s = **1 s**. The simulated value of the integral controller is $M_p = 16.48$. Thus, the maximum overshoot for the integral controller is better than the controller without the integral. The maximum overshoot decreases a lot.

The steady tracking error is also better than the controller without the integral. The state feedback with integral follows the reference input well, when there is no disturbance present.

(b). Step reference signals with external disturbances.

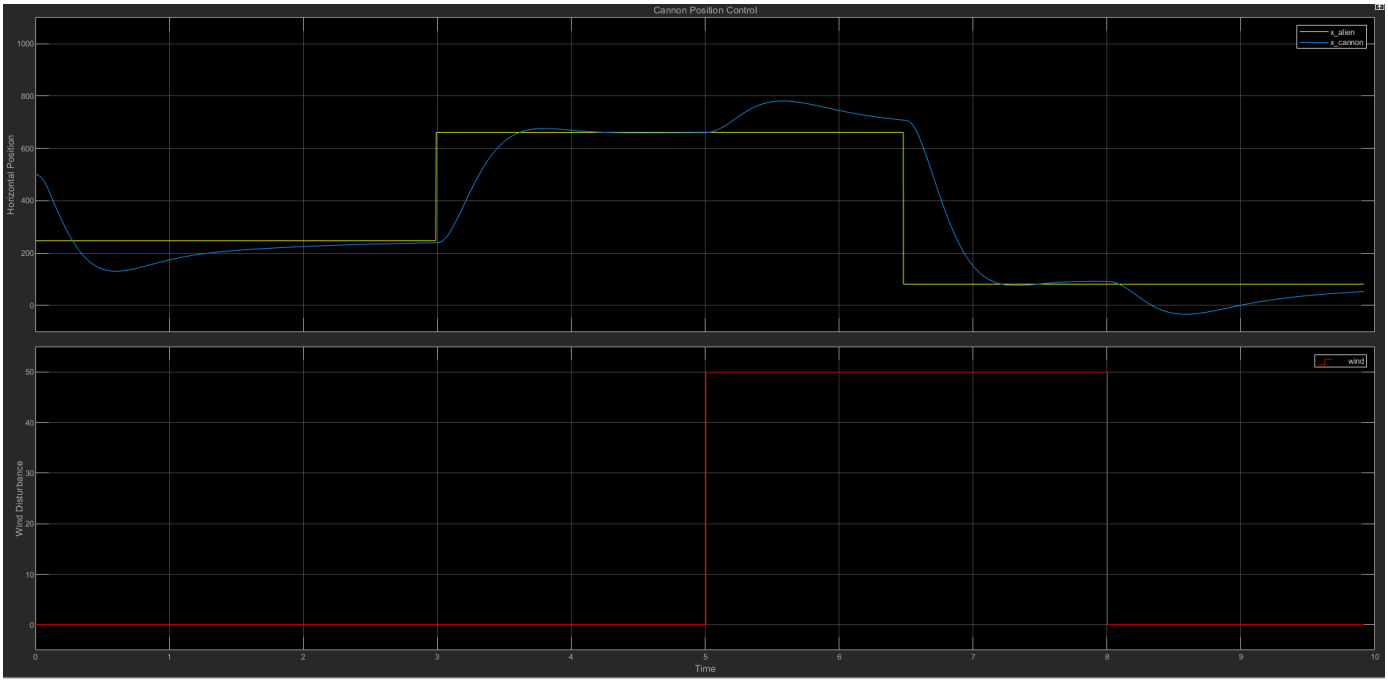

Figure 1.21: State Feedback Integral Controller Simulation Settling time

As can be seen above in Figure 1.21 with the disturbance (wind), the laser controller tracks the reference input well, when the external disturbance (wind) is applied. The steady tracking error is still low when the disturbance step (wind) is applied to the integral controller. This is different to the previous controller, who struggled to follow the reference signals when there was a step disturbance. The new added integrator pole of z = 0.9, helps with the disturbance rejection. Thus, choosing to add the integrator pole was a good choice. If we evaluate a constant disturbance (Wind), we notice that the controller stills works, even during constant disturbance. Thus, the constant disturbance rejection of the controller works well. I evaluate this by setting the phase delay to 0 seconds and period to 11 seconds, to evaluate a constant disturbance. This test illustrates how the controller would perform during a constant disturbance. Figure 1.22 below illustrates the performance of the controller during constant disturbance.

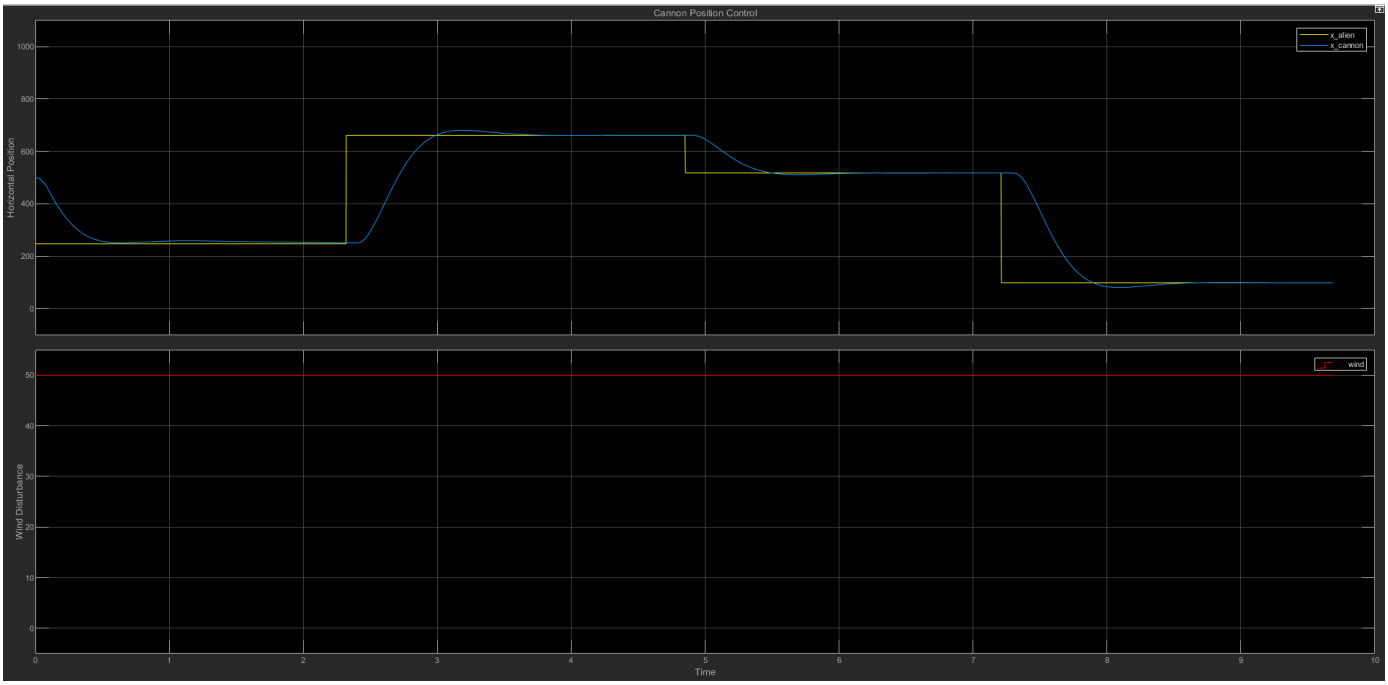

Figure 1.22: Constant Disturbance Rejection Simulation

Figure 1.22 above illustrates that during the 10 seconds period, the controller was able to reject the constant disturbance (Wind).

Conclusion:

In conclusion, the state feedback controller without integral demonstrated a satisfactory performance with a settling time close to the desired 2% settling time of 1.365 seconds and a maximum overshoot of 35.47, aligning well with the theoretical value calculated using the overshoot formula. It exhibited low steady-state tracking error and effectively followed the reference input in the absence of disturbances. However, the controller struggled to maintain tracking accuracy when external disturbances (Wind), were introduced. The steady-state tracking error increased during disturbance periods, highlighting a limitation in disturbance rejection capabilities.

On the other hand, the state feedback controller with integral showcased improvements. It achieved a faster settling time of 1.172 seconds and significantly reduced the maximum overshoot to 16.48, indicating enhanced transient response and smoother system behavior. Importantly, the integral controller exhibited superior disturbance rejection capabilities, maintaining low tracking error even during external disturbances. Moreover, it demonstrated effectiveness in rejecting a constant disturbance (wind), highlighting its robustness in handling sustained disturbances. Overall, the addition of an integrator pole ($z = 0.9$) proved to be a good choice, contributing to a more resilient and stable control system, particularly advantageous in scenarios with disturbances.

Matlab Code:

```
m = 0.02;
T = 0.1;
% Continuous State Space Model
A = [0 1; 0 0];
b = [0; 1/m];
c = [1 0];
d = 0;
% Discrete State Space Model
sysC = ss(A, b,c, d);
sysD = c2d(sysC, T, 'zoh');
% Extract discrete matrices
F = sysD.A;
g = sysD.B;
c = sysD.C;
d = sysD.D;
% Closed Loop Poles Calculations
zeta = 0.707;
ts = 1; % 2% settling time
wn = 4/(zeta*ts);
s_cl = [-(zeta*wn)+(i*wn*sqrt(1-zeta^2));-(zeta*wn)-(i*wn*sqrt(1-zeta^2))];
z_cl = exp(s_cl * T);
I = [1 0 ; 0 1];
%Determine k
k = place(F,g,z_cl);
% Feedforward gain
N = 1/(c*inv(I - F + g*k)*g);
Nbar = N;
% Current state observer
V = obsv(F,c);
if det(V) ~= 0
disp("Observable");
end
Observer_Poles = [0;0];
mp = acker(F', c', Observer_Poles)';
mc = inv(F)*mp;

% Integral Control
z_integral = 0.9;
zero = [0;0];
Fbar = [F zero ; c*F 1];
gbar = [g ; c*g];
z_i = [z_cl; z_integral];
kbar = place(Fbar, gbar, z_i);
kp = kbar(1:2);
ki = kbar(3);
Nbar = (ki*z_integral)/(1- z_integral);
NbarInt = Nbar;
k = kp;
```

Figure 23: Matlab Code for Practical 1: