Hard Disk Drive Read Header Controller Design Project

*EE 141 – Principles of Feedback Control*

*Spring 2015*

*University of California, Los Angeles*

Baixiao Huang 504313981

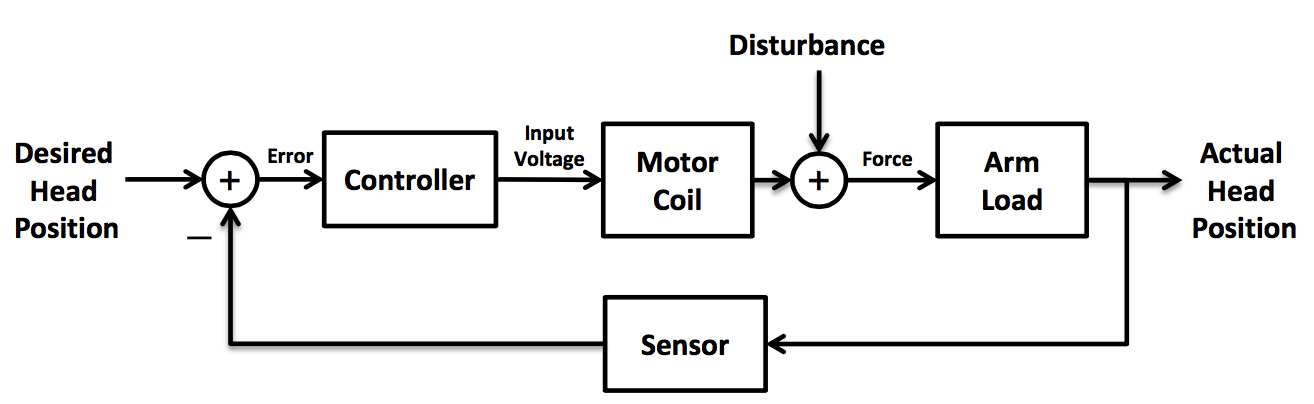
Weiqian Xu 404297854

Xingjian Yan 104304403

05/21/2015

**1.0 OBJECTIVES**

The objective of this project is to design a control system for hard disk drive read header. Knowledge of feedback control theories and skills of using Matlab are crucial in designing. The system contains a controller, a motor coil, a reader arm and a sensor forming a closed loop system. In this project, we are going to analyze the relationship between system performance and different compensators or sensors.



**Figure 1**

**2.0 TASKS**

2.1 Task 1

For this part, we are provided with the physical model relating the input torque to the header position with the inertia of arm and head and friction .

(1)

Simply apply Laplace transform to get the transfer function:  
 (2)

2.2 Task 2

The transfer function of the motor coil that relates the input voltage to the output torque is related by:

(3)

In this task, we are going to obtain the open-loop transfer function of the cascaded HDD head reader assembly and then use Matlab to generate the plot of the system’s step response.

(4)

The Matlab code with the constants plugged in and the plot obtained are presented below:

s = tf('s');

J = 1;

b = 20;

R = 1;

L = 0.001;

Km = 5;

G1 = Km/(L\*s+R)

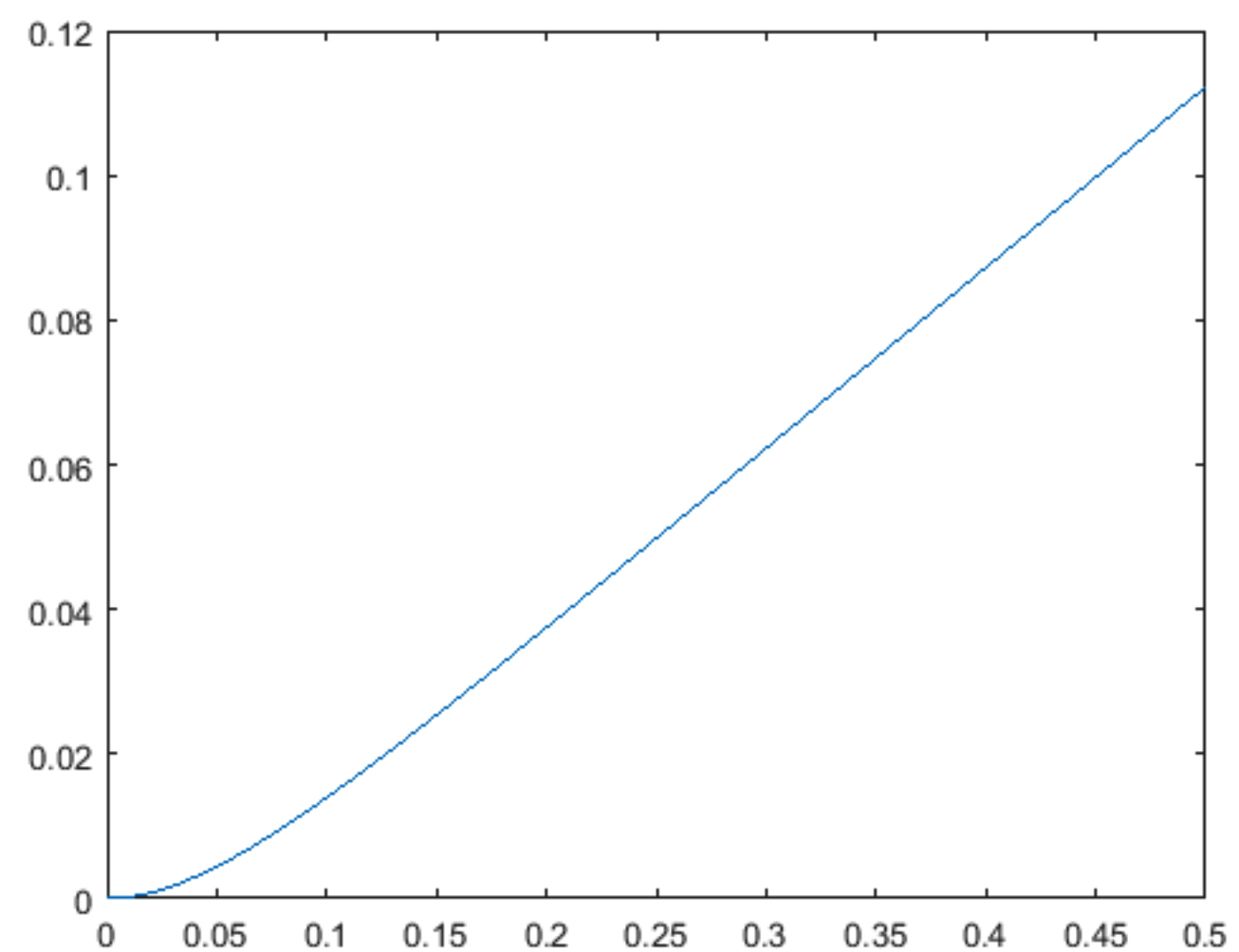
G2 = 1/(J\*s^2+b\*s)

openTF = G1\*G2

t=[0:0.005:0.5];

y = step(openTF,t);

plot(t,y);

****

**Figure 2**

As we can see, if a constant voltage is applied, then the read head will moves with a constant speed. However, at the beginning when the voltage is applied, there is a curvature indicating the read head is accelerating.

2.3 Task 3 – Proportional Compensator

In this part, we are going to make the closed-loop system satisfies several transient response performance specifications: Percent overshoot less than 5%; Settling time (2% deviation) less than 250ms; Maximum value of response to a unit step disturbance less than . Assume the feedback sensor’s transfer function is H(s) = 1 and the candidate controller is .

**Part A**

We are going to determine and evaluate the responses of the closed-loop system to the unit step reference input without disturbance. If the proportional compensator applied, the transfer function is given by:

(5)

Then the closed-loop transfer function is given by:

(6)

After plugging in (2) and (3), the equation becomes:

(7)

Try , the Matlab code (continuous after the previous code) and plot are shown below:

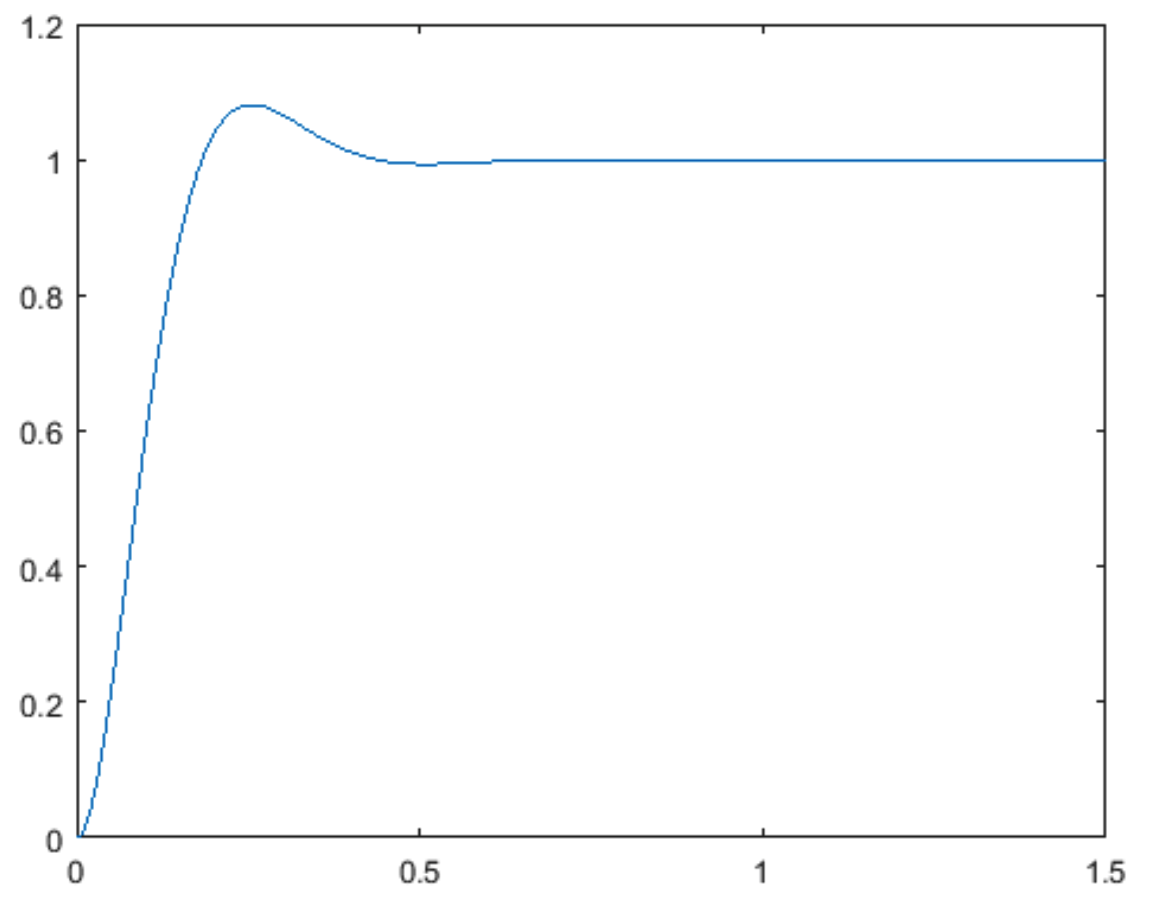
Ka = 50;

t=[0:0.005:1.5];

ProportionalTF =(Ka\*openTF)/(1+Ka\*openTF)

y = step(ProportionalTF, t);

plot(t,y);



**Figure 3**

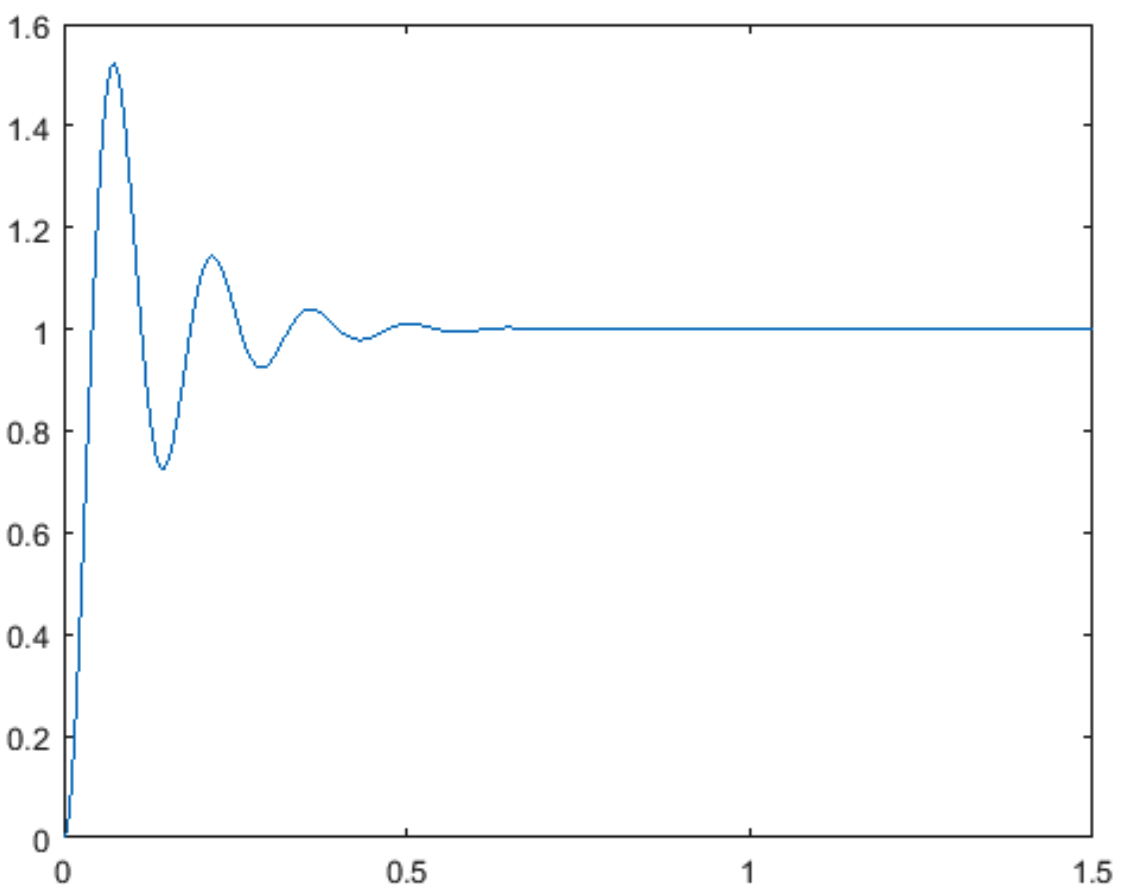
If , the Matlab code and plot are shown below:

Ka = 400;

ProportionalTF = (Ka\*openTF)/(1+Ka\*openTF)

y = step(ProportionalTF, t);

plot(t,y);

****

**Figure 4**

It is clear that the system overshoot when is much lesser than that when . If the feedback amplification is too big, the system will be too sensitive.

**Part B**

In this part, we are going to determine and evaluate the responses of the closed-loop system to the unit step disturbance, assuming zero reference input. The transfer function for disturbance is:

(8)

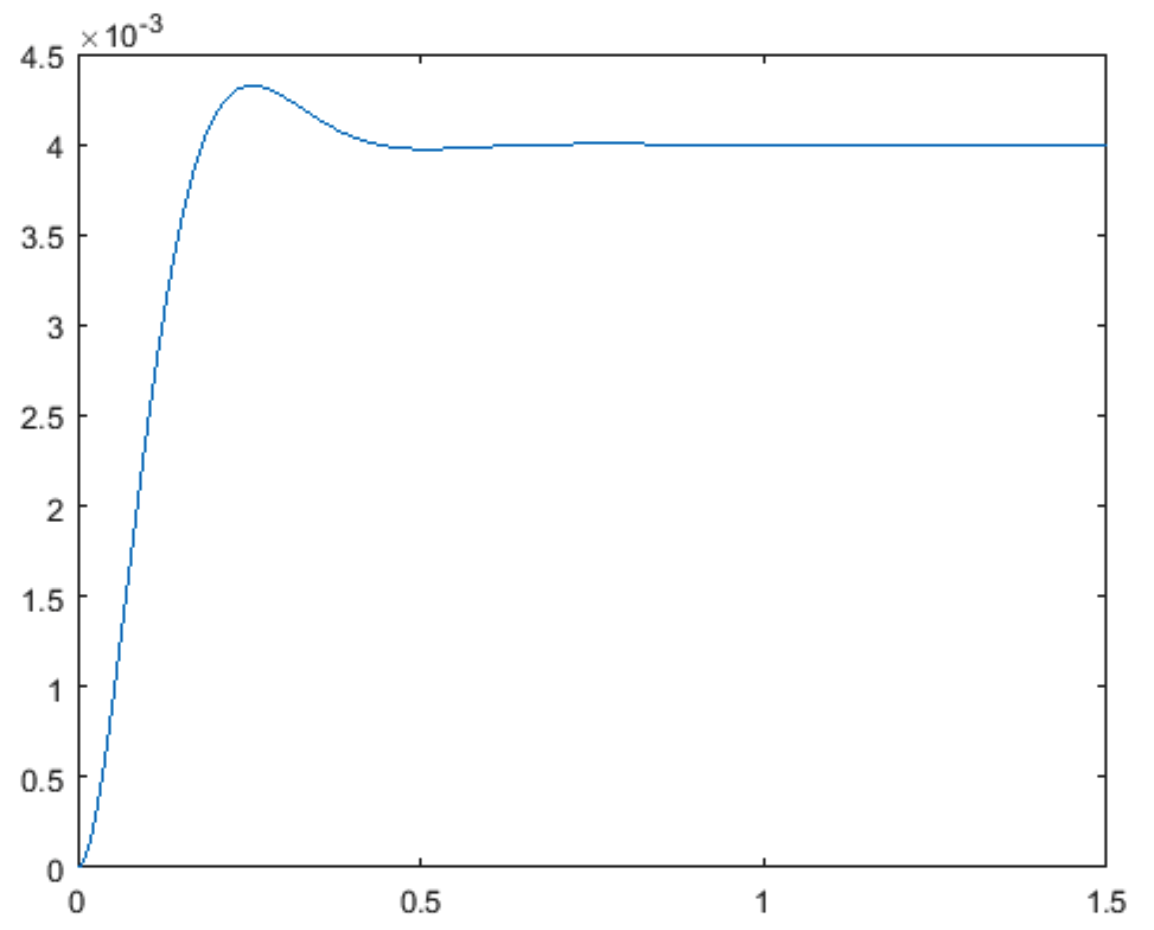
As above, we first plot the response with .

Ka = 50;

Tw = G2/(1+Ka\*G1\*G2)

y = step(Tw, t);

plot(t,y);



**Figure 5**

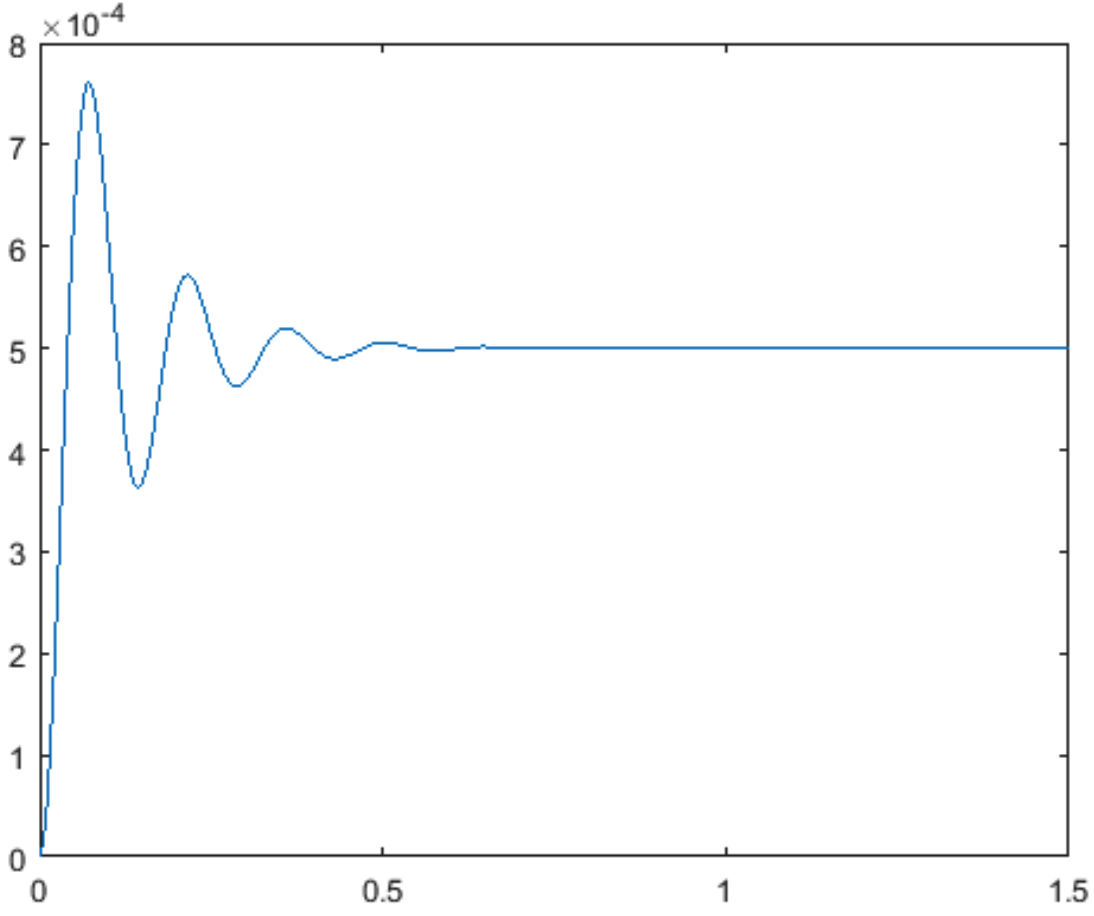
Similarly, we then try .

Ka = 400;

Tw = G2/(1+Ka\*G1\*G2)

y = step(Tw, t);

plot(t,y);



**Figure 6**

Same as Part A, 0 has less overshoot than

**Part C**

In this part, the objective is to find a value for that satisfies the performance specifications above. We can plot overshoot percentage and settling time as a function of , and analyze the optimal for the system.

overshoot = [];

settlingTime = [];

maxError = [];

n=[1:100];

for Ka=n

ProportionalTF = (Ka\*openTF)/(1+Ka\*openTF);

y = step(ProportionalTF, t);

info = stepinfo(y, t, 'SettlingTimeThreshold', 0.02);

overshoot = [overshoot, info.Overshoot];

settlingTime = [settlingTime info.SettlingTime];

Tw = G2/(1+Ka\*G1\*G2);

y = step(Tw, t);

maxError = [maxError max(y)];

end

subplot(3,1,1);

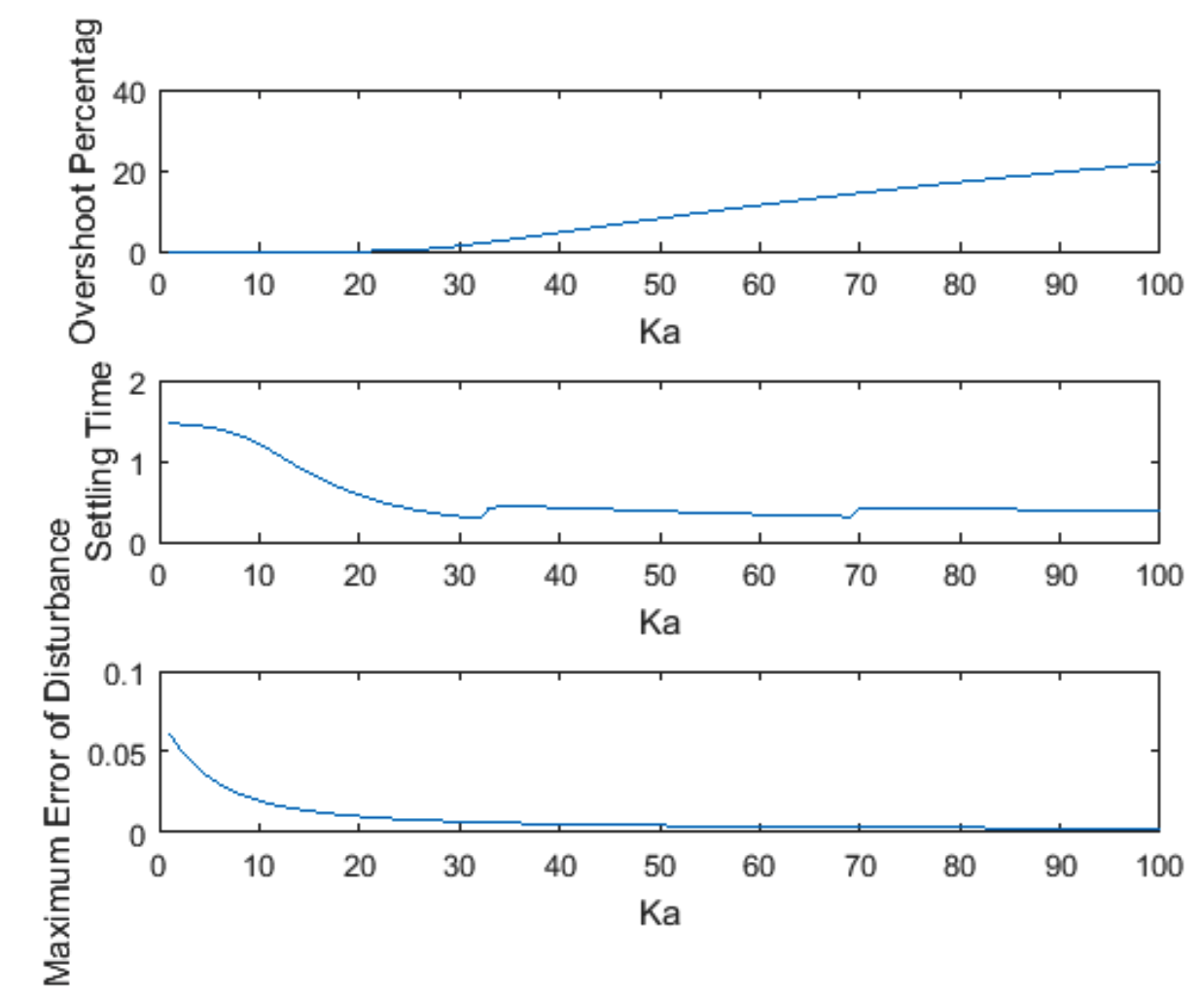
plot(n,overshoot);xlabel('Ka');ylabel('Overshoot Percentage');

subplot(3,1,2);

plot(n,settlingTime);xlabel('Ka');ylabel('Settling Time');

subplot(3,1,3);

plot(n,maxError);xlabel('Ka');ylabel('Maximum Error of Disturbance');

****

**Figure 7**

For overshoot less than 5%, is required to be equal or less than 41. The value of that satisfies settling time requirement is more than 100 and is not in the graph. For disturbance less than 0.005, is required to be equal or bigger than 41. As a result, the three requirements cannot be satisfied at the same time.

2.4 Task 4 – Positional and Velocity Sensor

In this part, we substitute the sensor with that measures the position as well as the velocity of the head reader. Then we need to find and such that the system is stable and satisfies the performance specifications mentioned above. The closed loop transfer function is:

(9)

The fundamental idea for finding the solution is that since both and are varying, we can plot a 3-D graph in which x-axis is , y-axis is , and z-axis is the property under examination including settling time, overshoot and disturbance. Then we can find out all of the combinations of and that satisfy all performance requirements.

[KaRange, KhRange] = meshgrid(55:65, 0:0.01:0.1);

overshootMatrix = [];

settlingTimeMatrix = [];

candidatePairs = [];

t=[0:0.05:0.8];

G12= G1\*G2;

for Kh = KhRange(:,1)'

%overshootArr=[];

%settlingTimeArr=[];

for Ka = KaRange(1,:)

CLTF = (Ka\*G12)/(1+Ka\*(1+Kh\*s)\*G12);

y = step(CLTF, t);

info = stepinfo(y, t, 'SettlingTimeThreshold', 0.02);

%overshootArr = [overshootArr, info.Overshoot];

%settlingTimeArr = [settlingTimeArr, info.SettlingTime];

Tw = G2/(1+Ka\*(1+Kh\*s)\*G12);

y = step(Tw,t);

maxDisturbance = max(y);

if(info.Overshoot <= 5 & info.SettlingTime <0.25 & y<0.005)

candidatePairs = [candidatePairs; Ka, Kh];

end

end

%overshootMatrix = [overshootMatrix; overshootArr];

%settlingTimeMatrix = [settlingTimeMatrix; settlingTimeArr];

end

%mesh(KaRange, KhRange, settlingTimeMatrix);

%mesh(KaRange, KhRange, overshootMatrix);

candidatePairs

*candidatePairs =*

*56.0000 0.0300*

*57.0000 0.0300*

*58.0000 0.0300*

*59.0000 0.0300*

*60.0000 0.0300*

*61.0000 0.0300*

*62.0000 0.0300*

*63.0000 0.0300*

*64.0000 0.0300*

*65.0000 0.0300*

There are a bunch of valid pairs of and that satisfies the design. We can select and to check the result.

Ka = 60; Kh = 0.03;

CLTF = (Ka\*G12)/(1+Ka\*(1+Kh\*s)\*G12);

y = step(CLTF, t);

figure(6);

plot(t,y);title('System Step Response');

info = stepinfo(y, t, 'SettlingTimeThreshold', 0.02)

Tw = G2/(1+Ka\*(1+Kh\*s)\*G12);

y = step(Tw,t);

figure(7);

plot(t,y);title('System Step Disturbance Response');

maxDisturbance = max(y)

*info =*

*RiseTime: 0.1621*

*SettlingTime: 0.2380*

*SettlingMin: 0.9401*

*SettlingMax: 1.0084*

*Overshoot: 0.8390*

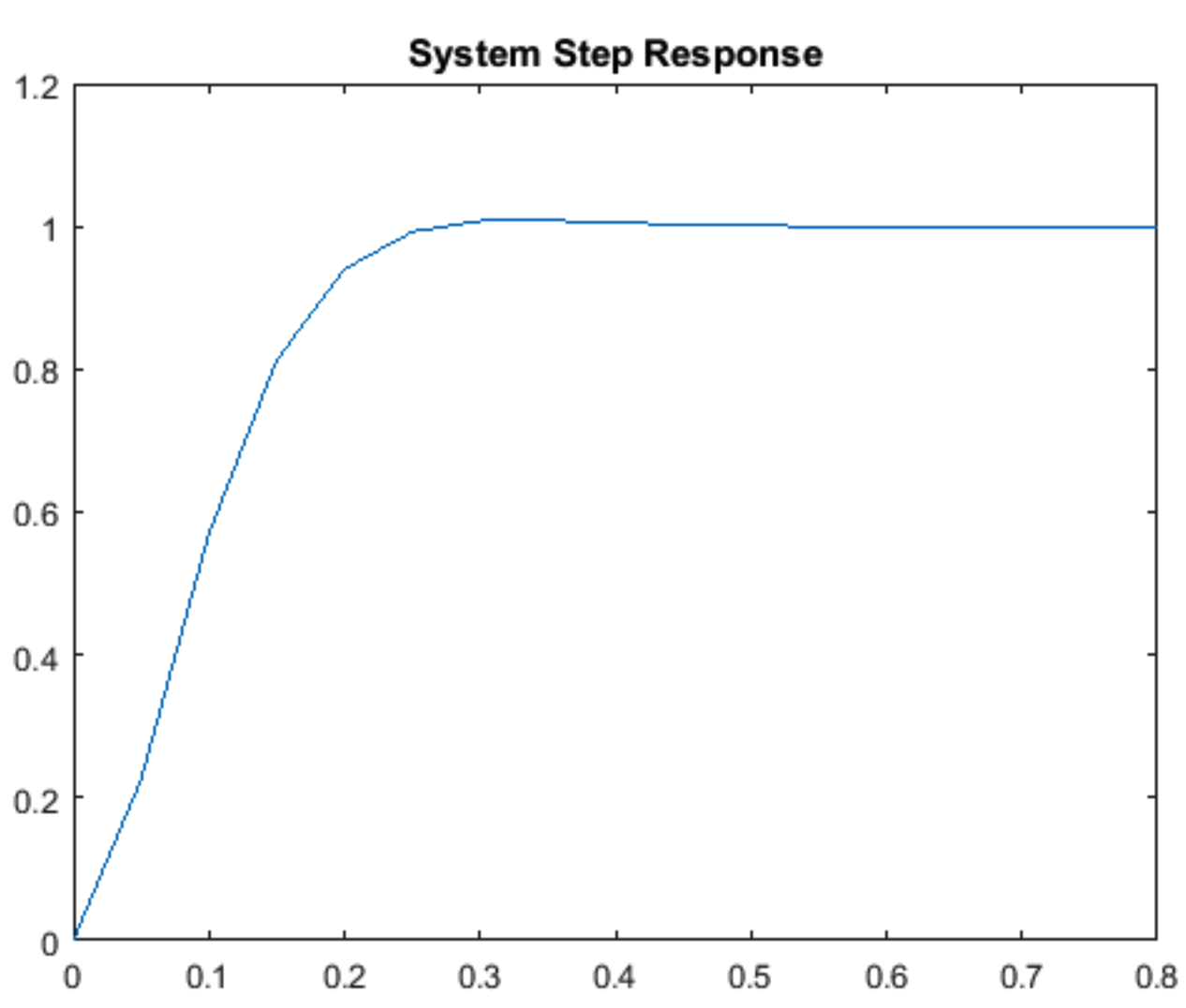
*Undershoot: 0*

*Peak: 1.0084*

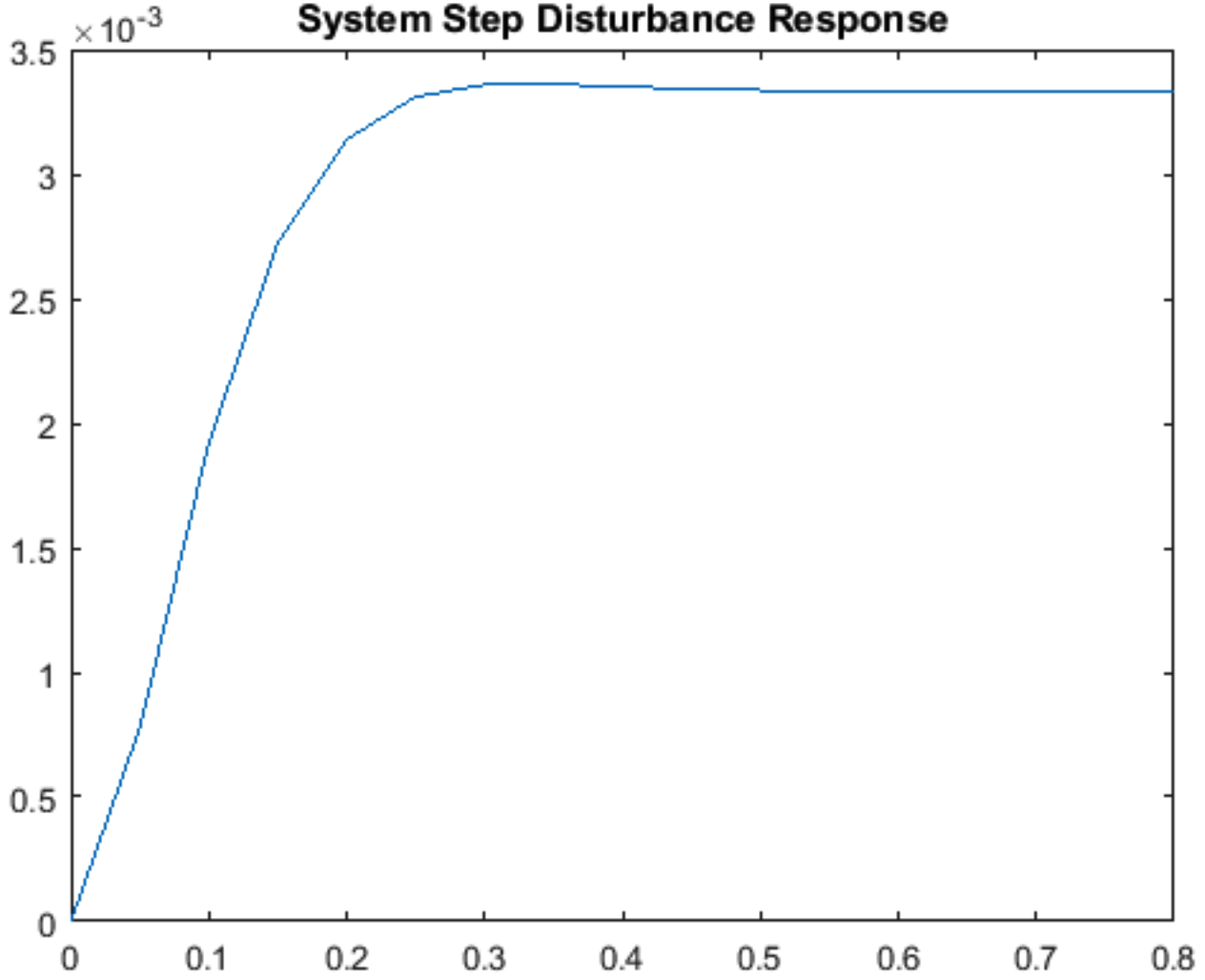
*PeakTime: 0.3500*

*maxDisturbance = 0.0034*

The graphs of responses are presented below.



**Figure 8**

****

**Figure 9**

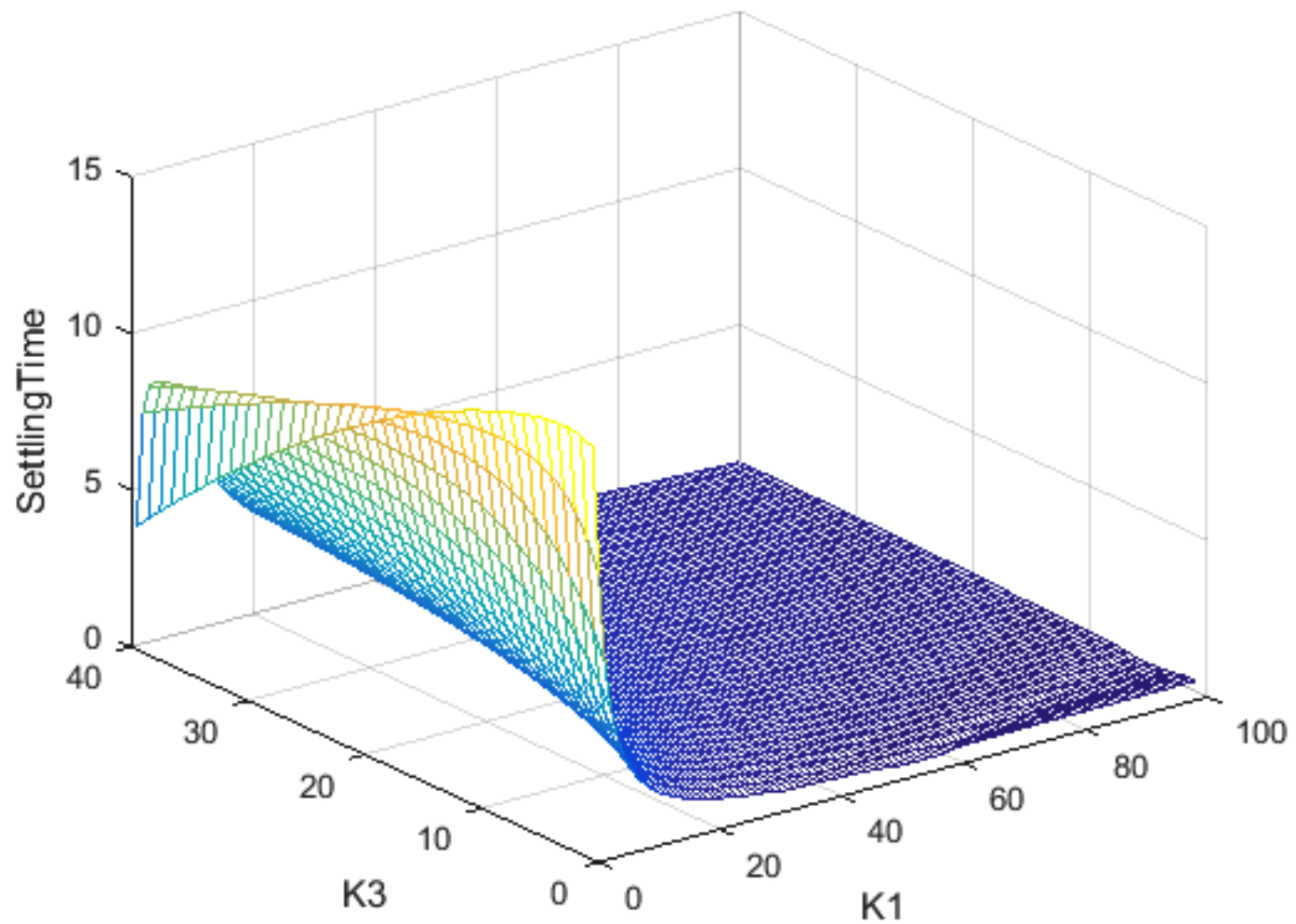
We can then conclude that the pairs of results listed above are valid for the system to meet all requirements.

2.5 Task 5 – PID Compensator

In this part, we revert the sensor back to H(s) = 1 but substitute the controller with . Since the arm inherently has an integrating term, we assume . The closed-loop transfer function then becomes:

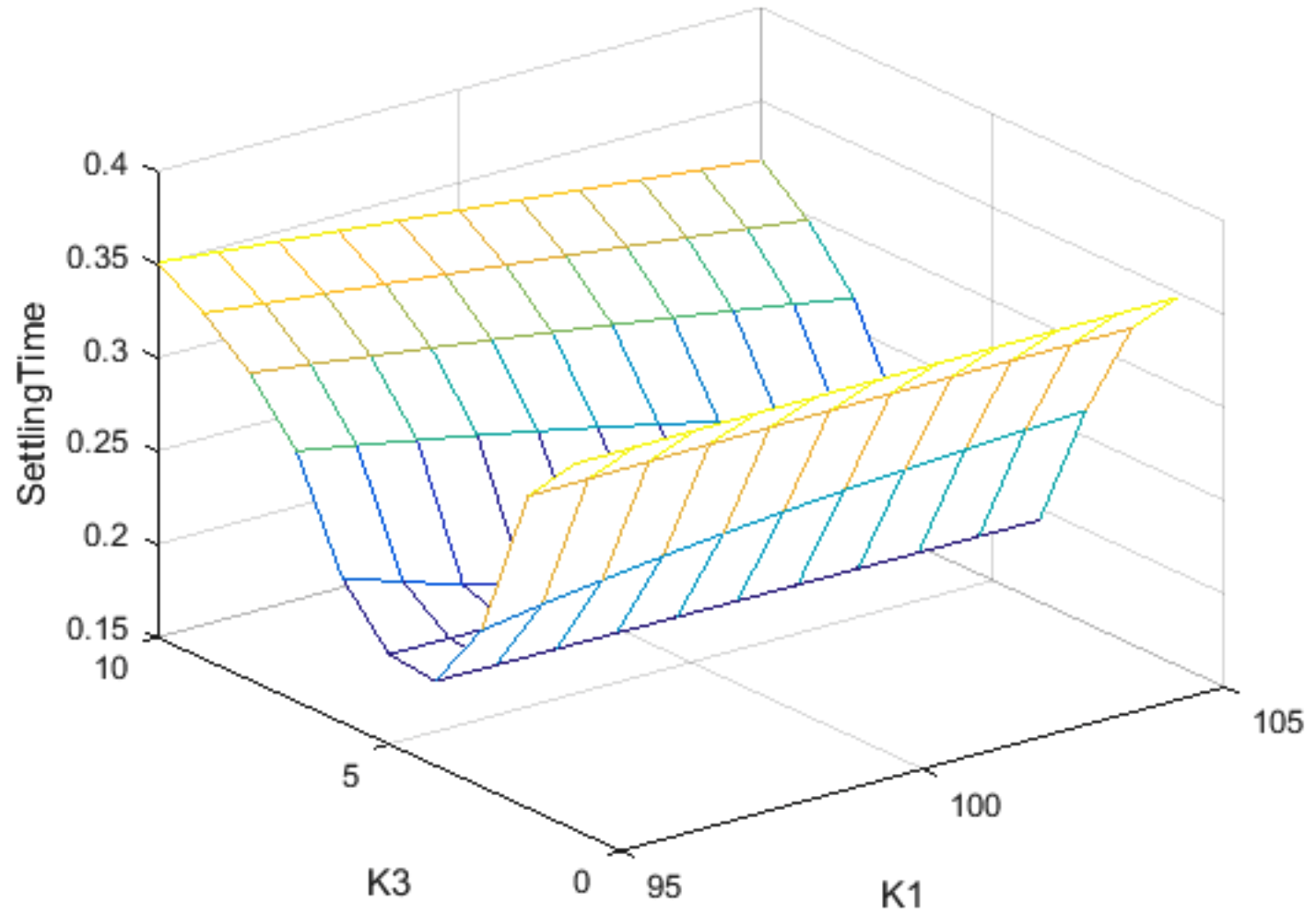
(10)

We can apply the same procedure to generate 3-D graph of constraints.



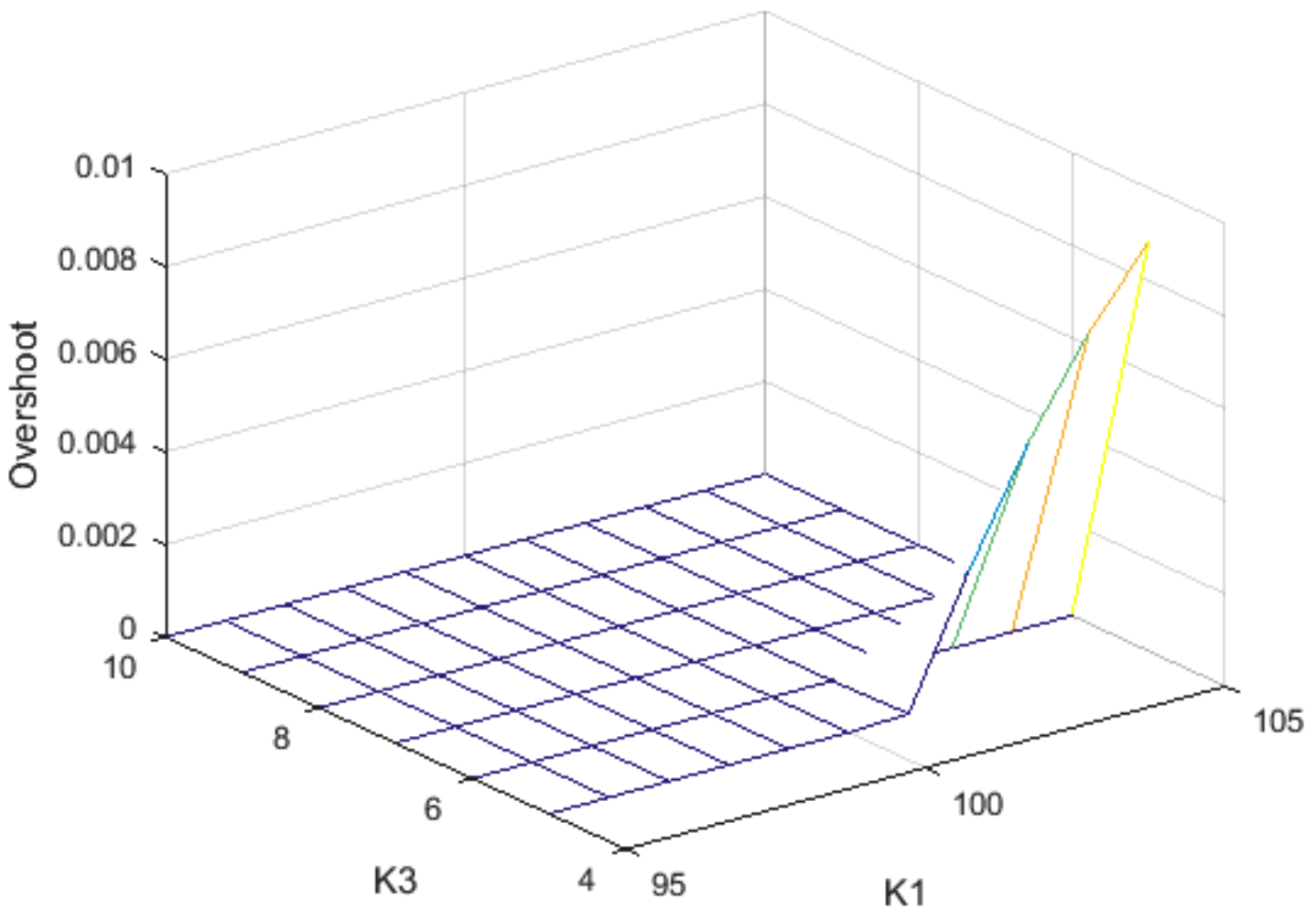
**Figure 10** – Settling time

When is around 100 and is around 5 to 10, the settling time requirement can be satisfied. With the range of values, we are able to further derive detailed solution. If we take a closer look at the range of values that satisfy the constraints, the graph becomes the one shown by Figure 11.



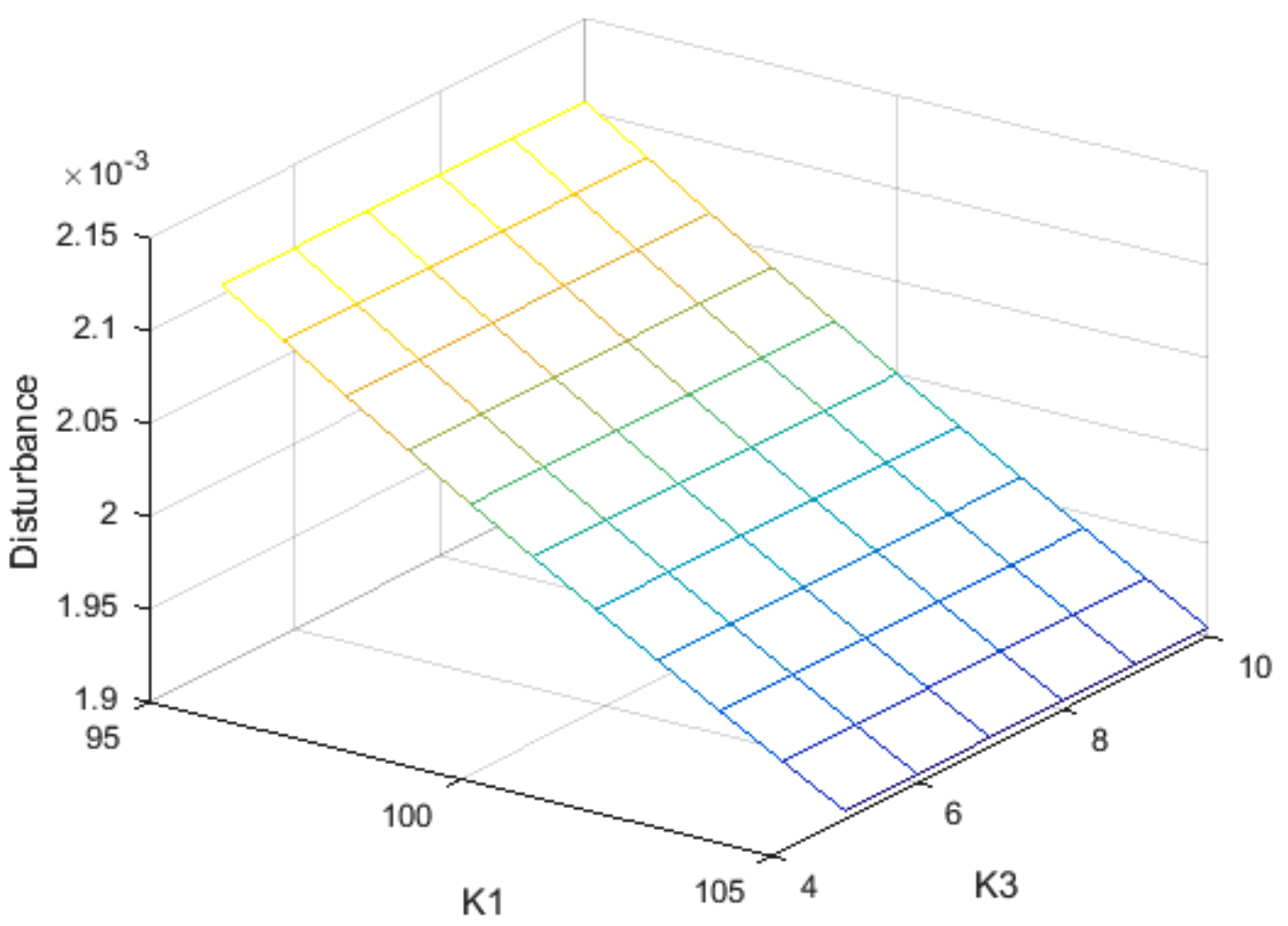
**Figure 11** – Settling time

As shown in Figure 11, there are couples of and pairs that meet the settling time requirement. If we exam and for example, the settling time will under 250ms. The next step is to evaluate the overshoot of the system.



**Figure 12** – Overshoot

As we can see, for and , there is no apparent overshoot. We can then examine the disturbance of this range.



**Figure 13** – Disturbance

The disturbance of and meets the requirement. The code for generates graphs above are presented below:

[K1Range K3Range] = meshgrid(95:105, 5:10);

overshootMatrix = [];

settlingTimeMatrix = [];

disturbanceMatrix=[];

candidatePairs = [];

t=[0:0.2:15];

G12= G1\*G2;

for K3 = K3Range(:,1)'

overshootArr=[];

settlingTimeArr=[];

disturbanceArr=[];

* for K1 = K1Range(1,:)
* CLTF = ((K1+K3\*s)\*G12)/(1+(K1+K3\*s)\*G12);
* y = step(CLTF, t);
* info = stepinfo(y, t, 'SettlingTimeThreshold', 0.02);
* overshootArr = [overshootArr, info.Overshoot];
* settlingTimeArr = [settlingTimeArr, info.SettlingTime];
* Tw = G2/(1+(K1+K3\*s)\*G12);
* y = step(Tw,t);
* maxDisturbance = max(y);
* disturbanceArr = [disturbanceArr, maxDisturbance];
  + - 1. if(info.Overshoot <= 5 & info.SettlingTime <0.25 & y<0.005)
* candidatePairs = [candidatePairs; Ka, Kh];

end

end

* overshootMatrix = [overshootMatrix; overshootArr];
* settlingTimeMatrix = [settlingTimeMatrix; settlingTimeArr];
* disturbanceMatrix = [disturbanceMatrix; disturbanceArr];
* end
* mesh(K1Range, K3Range,

overshootMatrix);xlabel('K1');ylabel('K3');zlabel('Overshoot')

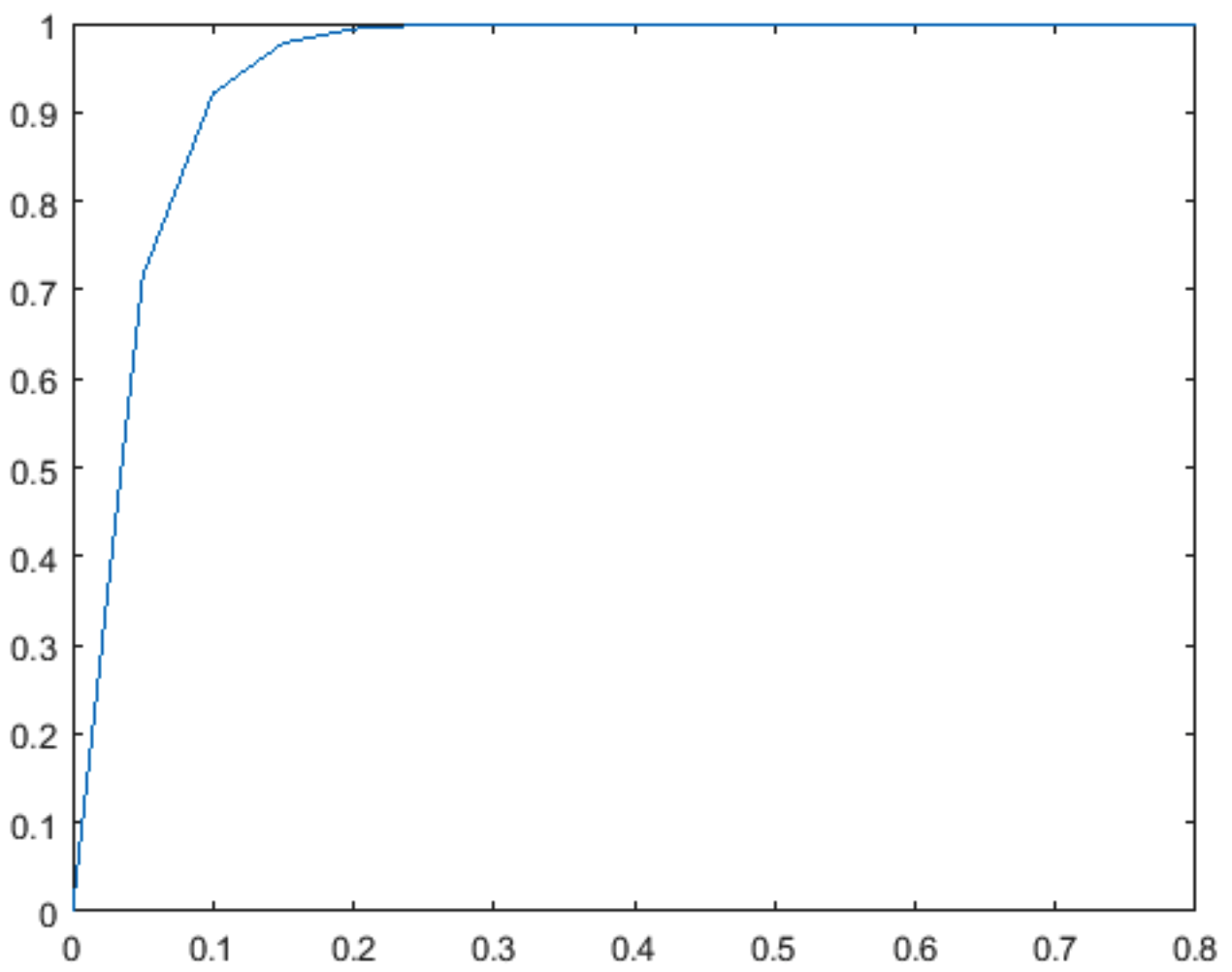
* mesh(K1Range, K3Range,

settlingTimeMatrix);xlabel('K1');ylabel('K3');zlabel('SettlingTime')

* mesh(K1Range,
* K3Range,disturbanceMatrix);xlabel('K1');ylabel('K3');zlabel('Disturbance');view(3
* candidatePairs

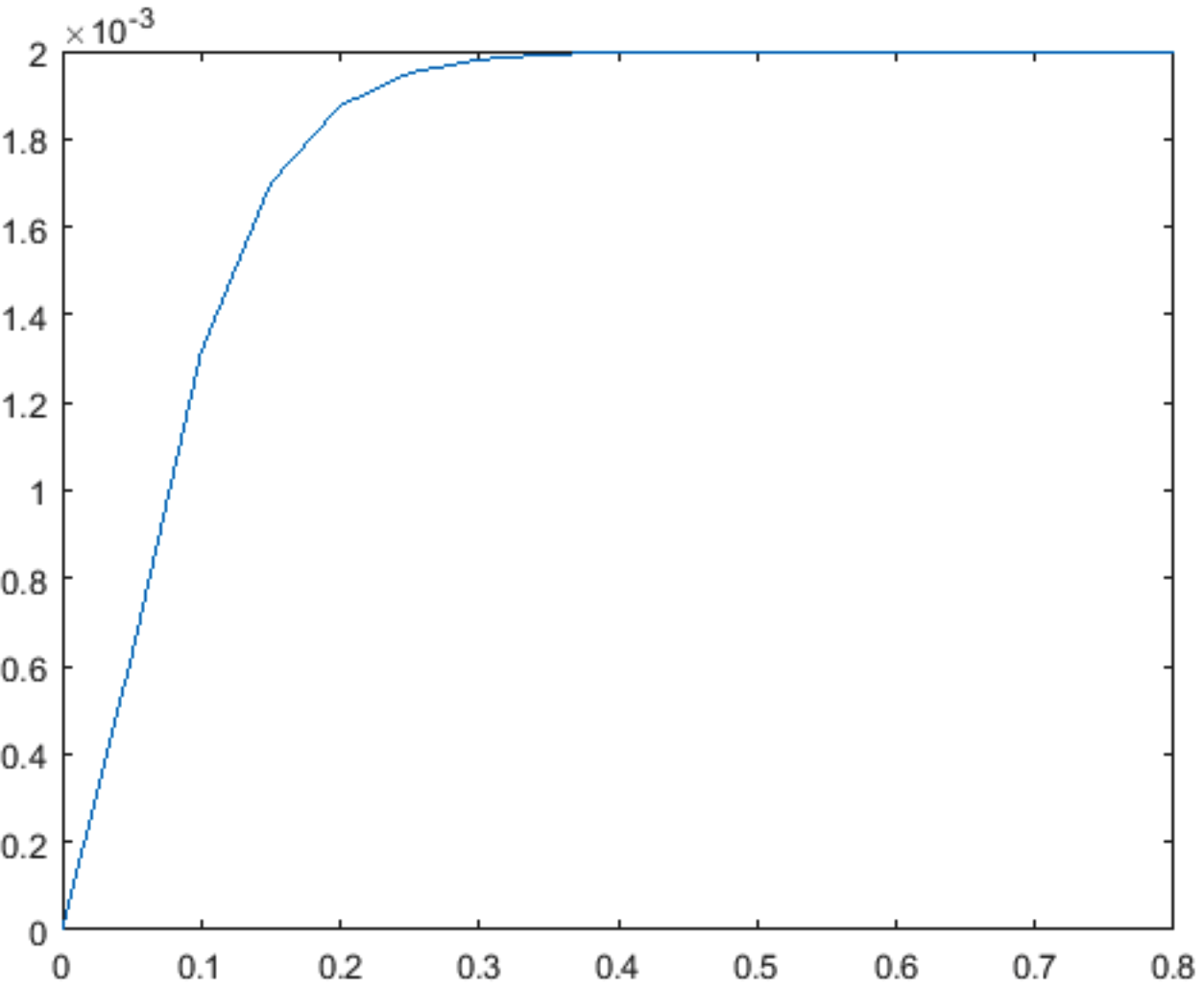
To examine the correctness of the answer, we select and to evaluate more specifically

* K1=100; K3=5;
* CLTF = ((K1+K3\*s)\*G12)/(1+(K1+K3\*s)\*G12);
* y = step(CLTF, t);
* info = stepinfo(y, t, 'SettlingTimeThreshold', 0.02)
* plot(t,y);
* *info =*   *RiseTime: 0.0879*
* *SettlingTime: 0.1560*
* *SettlingMin: 0.9211*
* *SettlingMax: 1.0000*
* *Overshoot: 0*
* *Undershoot: 0*
* *Peak: 1.0000*
* *PeakTime: 0.8000*



**Figure 14** – System overshoot

With the information generated and the curve shown in Figure 14, we can discover that at the same time satisfying other system requirements, the system has no overshoot. The reason is that acts as a damping factor that prevents the response from overshooting. We can then analyze the disturbance.



**Figure 15** – System disturbance

The maximum disturbance only reaches , within the range of requirement. We can conclude for this task that, compared to the proportional compensator, the PID compensator can be much more effective in controlling the system especially reducing the overshoot.

2.6 Task 6 – PID Compensator

**Part A**

**Part B**

**Part C**

**3.0 CONCLUTION**

The final architecture of the system that satisfies all of the performance requirements is shown below:

Actual Head

Position

Spring Load

Arm

Motor Coil

Controller F(s)

Desired

Head

Position

Sensor

H(s)

Transfer function of each block are presented below:

(11)

(12)

(13)

(14)

(15)

The closed-loop transfer function for the entire system is:

(16)

Parameters are shown in the table below:

**Table 1**

|  |  |
| --- | --- |
| Parameters | Value |
| Inertia of arm and head  Friction b  Field resistance R  Field inductance L  Motor constant | 1 /rad  20 Kg/m/s  1  0.001 H  5  0.3  18850 rad  0 |

We compared the system performance with the given specifications in **Table 2**

**Table 2**

|  |  |  |
| --- | --- | --- |
| Specification Variable Name | Specification Variable Value | Designed System Value |
| Percent Overshoot  Settling time(2% deviation)  Maximum value of response to a unit step disturbance | < 5%  < 250ms  < |  |