4/14/2012

|  |
| --- |
| Final Report | Hersha Bhagwan, 5537924 & Nicholas Kilingi, 6133169 |



Image from: <http://photographicdictionary.com/images/e/elevator.jpg>

|  |  |
| --- | --- |
| Concordia University | SOEN 385 – Elevator Project |

Contents

[Table of Figures 3](#_Toc322190528)

[List of Tables 5](#_Toc322190529)

[Introduction 6](#_Toc322190530)

[Modeling 8](#_Toc322190531)

[La Place and State Space Models 9](#_Toc322190532)

[Open loop response and root locus 10](#_Toc322190533)

[MAX MASS 10](#_Toc322190534)

[Parameters: 10](#_Toc322190535)

[Transient response 10](#_Toc322190536)

[Root locus 11](#_Toc322190537)

[MIN MASS 12](#_Toc322190538)

[Parameters: 12](#_Toc322190539)

[Transient response 12](#_Toc322190540)

[Root locus 12](#_Toc322190541)

[Unit step 14](#_Toc322190542)

[Input 14](#_Toc322190543)

[Max mass Output 15](#_Toc322190544)

[Min mass Output 16](#_Toc322190545)

[Ramp 16](#_Toc322190546)

[Input 17](#_Toc322190547)

[Max mass Output 17](#_Toc322190548)

[Min mass Output 18](#_Toc322190549)

[Elevator system(representing parabolic function) 18](#_Toc322190550)

[Input 19](#_Toc322190551)

[Max mass output velocity 21](#_Toc322190552)

[Max mass output position 21](#_Toc322190553)

[Min mass output velocity 22](#_Toc322190554)

[Min mass output position 22](#_Toc322190555)

[Feed-back loop control 22](#_Toc322190556)

[Block diagram 22](#_Toc322190557)

[Strategy of Improvement: PID Controller 23](#_Toc322190558)

[Matlab tools 23](#_Toc322190559)

[Elevator World 23](#_Toc322190560)

[Simulink design 24](#_Toc322190561)

[GUI 26](#_Toc322190562)

[Setup of the project files 27](#_Toc322190563)

[File directory structure: 27](#_Toc322190564)

[Data table and graphs 27](#_Toc322190565)

[Input velocity and position responses 28](#_Toc322190566)

[Output velocity and position responses for max mass 30](#_Toc322190567)

[Data columns 32](#_Toc322190568)

[Data table (for 1 position up) 32](#_Toc322190569)

[Output velocity and position responses for min mass 45](#_Toc322190570)

[Data columns 47](#_Toc322190571)

[Data table (for 1 position up) 47](#_Toc322190572)

[Tuning samples 48](#_Toc322190573)

[Discussion 60](#_Toc322190574)

[Analysis 60](#_Toc322190575)

[Conclusions and General Discussion 60](#_Toc322190576)

[References (outside of class material) 61](#_Toc322190577)

# Table of Figures

[Figure 0 - 1: Elevator Model 6](#_Toc322171183)

[Figure 0 - 2: Pulley System 6](#_Toc322171184)

[Figure 1 - 1: Free Body Diagrams 7](#_Toc322171260)

[Figure 2 - 1: Maximum Mass Root Locus 11](#_Toc322171627)

[Figure 2 - 2: Root Locus at Minimum Mass 13](#_Toc322171628)

[Figure 2 ‑3 13](#_Toc322171629)

[Figure 2 - 4 13](#_Toc322171630)

[Figure 2 - 5 13](#_Toc322171631)

[Figure 2 - 6 13](#_Toc322171632)

[Figure 2 - 7 13](#_Toc322171633)

[Figure 2 - 8 13](#_Toc322171634)

[Figure 2 - 9 13](#_Toc322171635)

[Figure 2 - 10 13](#_Toc322171636)

[Figure 2 - 11 13](#_Toc322171637)

[Figure 2 - 12 13](#_Toc322171638)

[Figure 2 - 13 13](#_Toc322171639)

[Figure 2 - 14 13](#_Toc322171640)

[Figure 2 - 15 13](#_Toc322171641)

[Figure 2 - 16 13](#_Toc322171642)

[Figure 2 - 17 13](#_Toc322171643)

[Figure 3 - 1 13](#_Toc322171650)

[Figure 3 - 2 13](#_Toc322171651)

[Figure 3 - 3 13](#_Toc322171652)

[Figure 3 - 4 13](#_Toc322171653)

[Figure 3 - 5 13](#_Toc322171654)

[Figure 3 - 6 13](#_Toc322171655)

[Figure 3 - 7 13](#_Toc322171656)

[Figure 3 - 8: *Input velocity 1 floor up* 13](#_Toc322171657)

[Figure 3 - 9: *Input position 1 floor up*  13](#_Toc322171658)

[Figure 3 - 10: *Input velocity 2 floors up* 13](#_Toc322171659)

[Figure 3 - 11: *Input position 2 floors up*  13](#_Toc322171660)

[Figure 3 - 12: *Uncontrolled output velocity 1 floor up* 13](#_Toc322171661)

[Figure 3 - 13 *Uncontrolled output position 1 floor up* 13](#_Toc322171662)

[Figure 3 - 14 *Uncontrolled output velocity 2 floors up* 13](#_Toc322171663)

[Figure 3 - 15: *Uncontrolled output position 2 floors up* 13](#_Toc322171664)

[Figure 3 - 16: Controlled Output velocity 1 floor up 13](#_Toc322171665)

[Figure 3 - 17: Controlled Output position 1 floor up 13](#_Toc322171666)

[Figure 3 - 18: Controlled Output velocity 2 floors up 13](#_Toc322171667)

[Figure 3 - 19: Controlled Output position 2 floors up 13](#_Toc322171668)

[Figure 3 - 20: Controlled Output velocity 1 floor up 13](#_Toc322171669)

[Figure 3 - 21: Controlled Output position 1 floor up 13](#_Toc322171670)

[Figure 3 - 22: *Controlled Output velocity 2 floors up* 13](#_Toc322171671)

[Figure 3 - 23: *Controlled Output position 2 floors up* 13](#_Toc322171672)

[Figure 3 - 24: *Controlled Output velocity 1 floor up* 13](#_Toc322171673)

[Figure 3 - 25 *Controlled Output position 1 floor up* 13](#_Toc322171674)

[Figure 3 - 26 *Controlled Output velocity 2 floors up* 13](#_Toc322171675)

[Figure 3 - 27 *Controlled Output position 2 floors up* 13](#_Toc322171676)

[Figure 3 - 28 *&* Figure 3 - 29 13](#_Toc322171677)

[Figure 3 - 30 *Controlled Output velocity 2 floors up* 13](#_Toc322171678)

[Figure 3 - 31 *Controlled Output position 2 floors up* 13](#_Toc322171679)

[Figure 3 - 32 *&* Figure 3 - 33 13](#_Toc322171680)

[Figure 3 - 34 *Controlled Output velocity 2 floors up* 13](#_Toc322171681)

[Figure 3 - 35 *Controlled Output position 2 floors up* 13](#_Toc322171682)

[Figure 3 - 36 *&* Figure 3 - 37 13](#_Toc322171683)

[Figure 3 - 38 *Controlled Output velocity 2 floors up* 13](#_Toc322171684)

[Figure 3 - 39 *Controlled Output position 2 floors up* 13](#_Toc322171685)

[Figure 3 - 40 & Figure 3 - 41: Uncontrolled 13](#_Toc322171686)

[Figure 3 - 42 & Figure 3 - 43 13](#_Toc322171687)

[Figure 3 - 44 *Controlled Output velocity 1 floor up* 13](#_Toc322171688)

[Figure 3 - 45 *Controlled Output position 1 floor up* 13](#_Toc322171689)

[Figure 3 - 46 Controlled Output velocity 2 floors up 13](#_Toc322171690)

[Figure 3 - 47 Controlled Output position 2 floors up 13](#_Toc322171691)

[Figure 3 - 48 *&* Figure 3 - 49 *&* Figure 3 - 50 13](#_Toc322171692)

[Figure 3 - 51 Controlled Output position 2 floors up 13](#_Toc322171693)

[Figure 3 - 52 *Controlled Output velocity 1 floor up* 13](#_Toc322171694)

[Figure 3 - 53 *Controlled Output position 1 floor up* 13](#_Toc322171695)

[Figure 3 - 54 Controlled Output velocity 2 floors up 13](#_Toc322171696)

[Figure 3 - 55 Controlled Output position 2 floors up 13](#_Toc322171697)

[Figure 3 - 56 *Controlled Output velocity 1 floor up* 13](#_Toc322171698)

[Figure 3 - 57 *Controlled Output position 1 floor up* 13](#_Toc322171699)

[Figure 3 - 58 Controlled Output velocity 2 floors up 13](#_Toc322171700)

[Figure 3 - 59 Controlled Output position 2 floors up 13](#_Toc322171701)

[Figure 3 - 60 Position *&* Figure 3 - 61 velociy 13](#_Toc322171702)

[Figure 3 - 62 Controlled Output velocity 2 floors up 13](#_Toc322171703)

[Figure 3 - 63 Controlled Output position 2 floors up 13](#_Toc322171704)

[Figure 3 - 64 *&* Figure 3 - 65 13](#_Toc322171705)

[Figure 3 - 66 Controlled Output velocity 2 floors up 13](#_Toc322171706)

[Figure 3 - 67 Controlled Output position 2 floors up 13](#_Toc322171707)

# List of Tables

[Table 1 13](#_Toc322172373)

[Table 2 13](#_Toc322172374)

[Table 3 13](#_Toc322172375)

# Introduction

For our project we modeled the elevator as follows:

Figure 0 - 1: Elevator Model

K

D

M1

M2

Where M1 represents the mass of the elevator shaft and M­2 represents the mass of the elevator. K represents the spring constant and D represents a hydraulic actuator with a passive damper. In our case we modeled the elevator this way to represent an elevator system that has a pulley and either a weight (counter mass) or hydraulic system applying a force through the actuator with a damping ratio to counter balance the tension force created by M­2. This hydraulic actuator basically consists of a cellunoid valve that applies force upward with pistons. Here K actually represents two tensions (springs) of equal value. (1/2K +1/2K = K), as in reality an elevator would have two tensions of equal value.

Figure 0 - 2: Pulley System

Figure 0.2 above shows a simple pulley system with a mass hanging down - this is how most elevator systems are with a pulley above the elevator. On the other end there is usually a counter weight, or there is a hydraulic system on the bottom. In our case the tensions of the strings have been replaced by our spring. The system we designed is balanced out by a damper, to represent the counter balance created by the counter weight. The damper, as would a counter mass, reduces the oscillations of the spring, meaning that there will be no shaking, so to speak, when a desired floor is reached. “The counterweight weighs about the same as the car filled to 40-percent capacity. In other words, when the car is 40 percent full (an average amount), the counterweight and the car are perfectly balanced.” The reason for having this balance is so that that the a constant potential energy can be maintained at all times, in this way the only additional force that needs to be overcome for the motion is friction. For the modeling of our system we assumed zero friction because it does not have any contact with the shaft. In general, a normal force can be felt by passengers within the elevator. The normal force balances out the force of gravity (mg) pulling the passenger down. If the elevator is at rest or moving at a constant velocity then N is equal to mg, and the passenger will not feel “heavier” or “lighter”. However, when there is acceleration upward, the passenger will feel a bit “heavier” due to N = mg + ma. In the same way when there is a downward acceleration the passenger will feel “lighter, as N = mg – ma. In our system, one of our aims is to minimize the discomfort during the start and stop accelerations of the elevator, so that it is safe and comfortable for the people using it. For this reason the accelerations are gradual and only to reach the desired velocity and to stop at the desired floor.

Our input in general is a magnitude of force applied depending on the floor selected, and our expected output is position. This system input is a force driven by the hydraulic actuator movement up or down that works to counter or release the elevator mass respectively. This input in our design however, is modelled by a complex signal(fig 2-12) instead of a constant one. We wished to ensure that the rate of change of actuator force from one position to another is not constant but one that rises, becomes constant and then drops because the simple version of this would be a constant force signal which is easier to deal with but hazardous to passengers due to inertia when it stops or starts acting. Hence we realized velocity of force applied needs to be considered and opted for this solution. This complex signal contains information on direction of movement(if graph is above or below x-axis ie +y or –y for up or down respectively) and change in distance(difference between the current position height and selected position height which can be seen by length of signal on graph) which is set depending on the floor selected. The input has been configured to allow the force to gradually increase from 0 to a desired constant velocity, hold in that constant velocity for a period of time(depending on how far up or down the elevator is going) and drops equally gradually to 0 when about to reach the destination floor as in fig 2-12. It is then pushed through the feedback loop and finally given as an unstable output velocity signal that needs to be then integrated to obtain position(as in fig 2-13) once again, so as to move the elevator to the desired vertical position. Our controller works to ensure that this position is the right position without errors, instability or overshoot and to model the required rise time and settling time. This complex input feature helps in real-life situation to ensure comfortable rise and drop of the elevator. Hence after integrating the velocity output signal from the loop, our final position output signal as in fig 2.13 contains information on the gradual increase and not instantaneous rise to the desired position. It also contains the direction of motion(+y or -y) and the length of motion(amount of change in position from old one to new one is represented by the y-value the initially rising or dropping signal becomes constant). This signal is then summed after to the height of the current position in order to run from that current position as the starting point to the final desired position. For example if one is on floor 2 and wishes to go to floor 1, the difference in height is 5m and the direction desired is down, then the output position signal will be as in fig 2-13 but inverted along the x-axis, which indicates the downward movement and with the constant y-value at -5. This -5 would then finally be summed to the current position which is 10m(floor 2) to get to the final position at 5m, but to get there, the elevator would gradually move downward as per the signal does from floor 2 to floor 1. Refer to fig 2-12 for the ideal complex input and/or output graphs, and fig 2-13 for the integrated output signal(position) graph.

# Modeling

In our system we chose the mass of our elevator shaft to be 500kg, and the mass of the empty elevator to be 100kg and the maximum mass of passengers to be 260kg. Consecutive floors are 5m apart. Currently our building has a total of three floors, hence 15m.

To calculate how much the spring constant, K, should be we used the potential energy of our elevator to be equal to mgh = 360kg(9.8N/m)(15m) = 52920J. We then made that energy equal to the potential energy of the spring. The Energy of a spring is equal to ½(Kx­­­2), where x is the length the spring would stretch. Therefore 52920J = ½ K (152) to get approximately 470N/m for K. For the project we opted to use negative K because during the demo the instructor asked us to use lower value.

As mentioned, in industry, the counter weight would balance out the potential energy; therefore in our system we made the damping constant to be approximately 85% of our spring constant. The lack of equality gives us more of a challenge to stabilize the system.

f

Figure 1 - 1: Free Body Diagrams

From the free body diagrams of our system in **Error! Reference source not found.**, and letting direction down to be positive, we get the following three differential equations:

# La Place and State Space Models

Based on the last sections differential equations we can find the transfer function. The following is our La Place transform equations:

From there we can calculate our delta equations using Cramer’s Rule:

We can then get our transfer function to be G(s) = Y3(s)/F(s), where Y3(s)= ∆Y3/∆.

Therefore, our transfer function is:

To get our state space we first divided the numerator and the denominator of our transfer function by D and got the following:

Changing the denominator into a differential equation and rearranging:

Then:

Open loop response and root locus

To find the root locus and control our system, we create a closed loop system by adding a control block K. Our new formula would account for the unity-gain outer loop. G(s) will now be controlled by our K block. The closed loop transfer function will be KG(s)/ 1+KG(s). From that we can get our characteristic equation to be 1+KG(s) = 0. From there we can find our roots and poles and get our calculations of departure angle, breakpoint, and imaginary axis and sketch the graph. From there we can analyze the system. In our case we used MatLab for all of that.

## MAX MASS

### Parameters:

Rise time < 10 sec higher because of greater mass

Overshoot < 10%

Steady state error < 2%

### Transient response

>> T1=tf(1,[450 860 423 470])

Transfer function:

1

-------------------------------

450 s^3 + 860 s^2 + 423 s + 470

>> P=pole(T1)

P =

-1.7178

-0.0966 + 0.7737i

-0.0966 - 0.7737i

>> Pre=abs(real(P(1)));

>> Pim=abs(imag(P(1)));

>> wn=sqrt(Pre\*Pre+Pim\*Pim)

wn =1.7178

>> damping\_ratio=(Pre/wn)

damping\_ratio =1

>> OS=(exp(-1\*Pre\*pi/Pim))\*100

OS =0

>> Tp=pi/Pim

Tp =Inf

>> Ts=4/Pre

Ts = 2.3285

### 

### Root locus

>> rlocus(num,den)

>> zeta=1; Wn=1.7178; sgrid(zeta, Wn)

>> [kd,poles]= rlocfind(num,den)

Select a point in the graphics window

selected\_point = -0.7476 - 0.0078i

kd = 446.4102

poles =

-1.9613

0.0251 + 1.0187i

0.0251 - 1.0187i

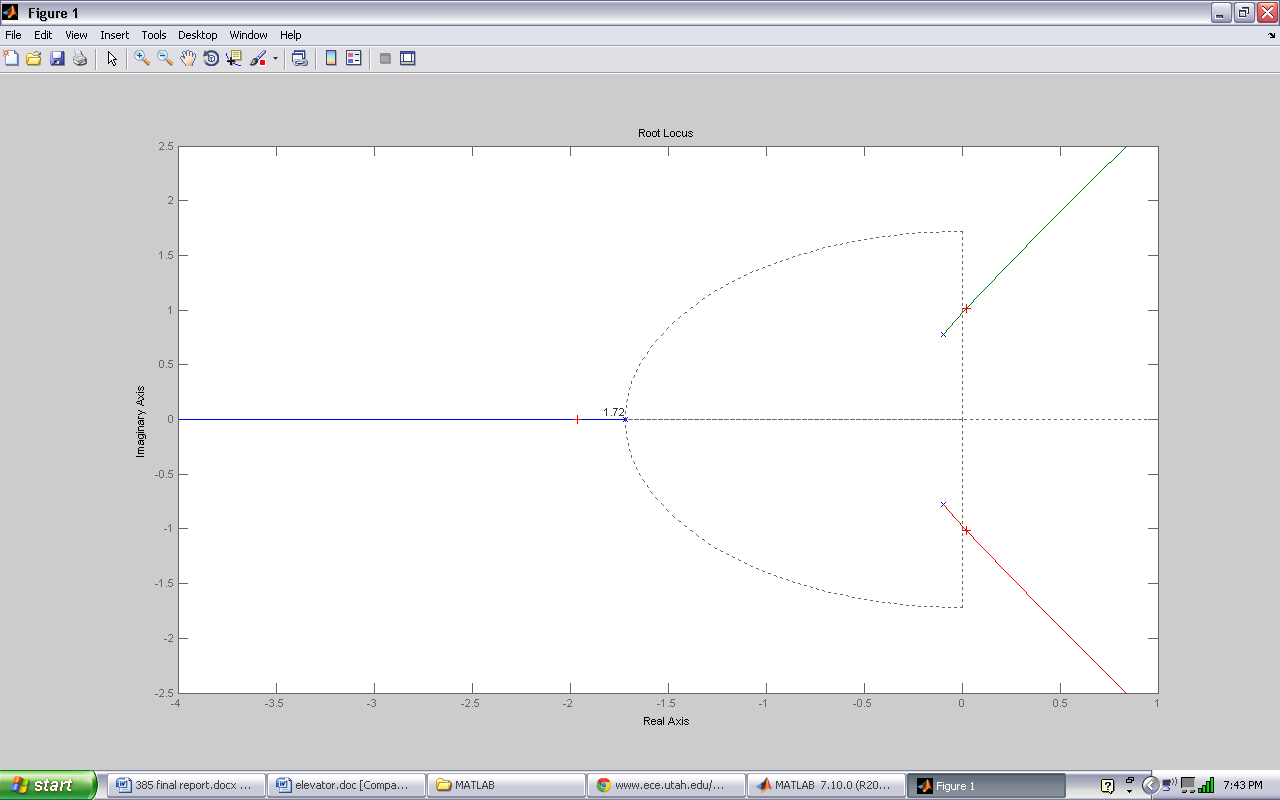


Figure 2 - 1: Maximum Mass Root Locus

After plotting root locus for the max mass transfer function, refer to Figure 2 - 1: Maximum Mass Root Locus ,we realized that our entire locus region exists in the left half of plane hence shifting is not really necessary to make the system stable.

## MIN MASS

### Parameters:

Rise time < 5 sec lower because of lower mass

Overshoot < 10%

Steady state error < 2%

### Transient response

>> T1=tf(1,[125 600 117 470])

Transfer function:

1

-------------------------------

125 s^3 + 600 s^2 + 117 s + 470

>> P=pole(T1)

P =

-4.7691

-0.0155 + 0.8878i

-0.0155 - 0.8878i

>> Pre=abs(real(P(1)));

>> Pim=abs(imag(P(1)));

>> wn=sqrt(Pre\*Pre+Pim\*Pim)

wn =4.7691

>> damping\_ratio=(Pre/wn)

damping\_ratio =1

>> OS=(exp(-1\*Pre\*pi/Pim))\*100

OS =0

>> Tp=pi/Pim

Tp =Inf

>> Ts=4/Pre

Ts = 0.8387

### Root locus

>> rlocus(num,den)

>> zeta=1; Wn=4.7691; sgrid(zeta, Wn)

>> [kd,poles]= rlocfind(num,den)

Select a point in the graphics window

selected\_point = -1.2500 - 0.0548i

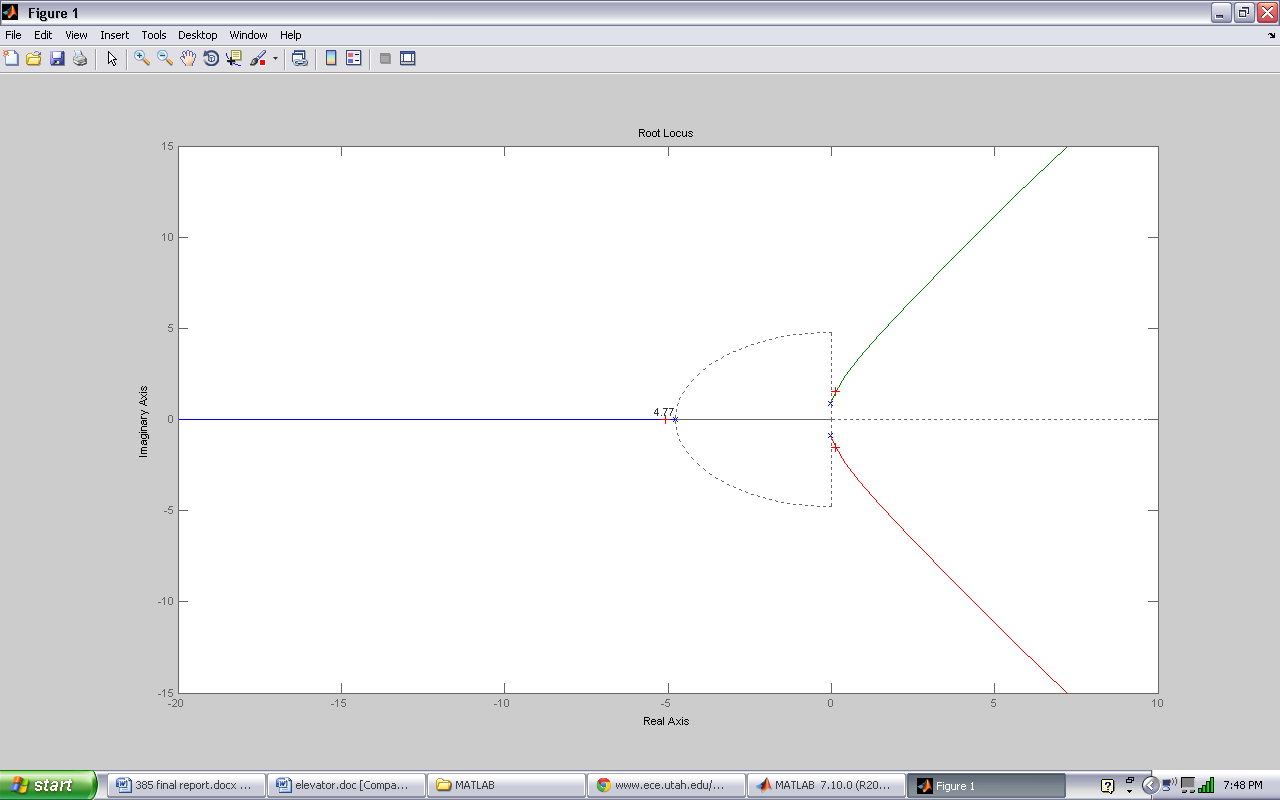
kd = 1.0177e+003

poles =

-5.0773

0.1387 + 1.5247i

0.1387 - 1.5247i

Figure 2 - 2: Root Locus at Minimum Mass

After plotting root locus for the min mass transfer, shown in Figure 2 - 2: Root Locus at Minimum Mass, we realize that our entire locus region exists in the left half of plane hence shifting is not really necessary to make the system stable.

## Unit step

C:\Documents and Settings\nali katana\My Documents\MATLAB\elevator_doc_files\image-005-sl.emf

Figure 2 ‑3

### Input

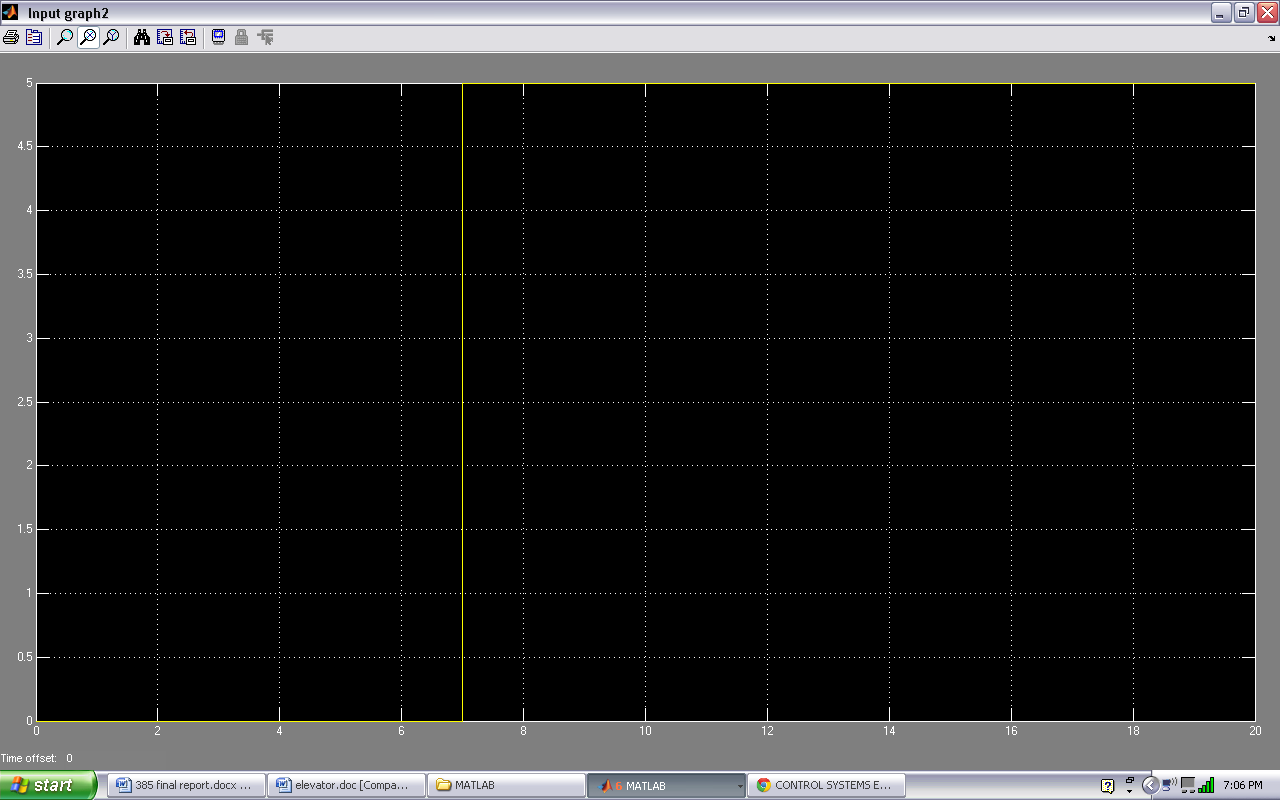


Figure 2 - 4

### Max mass Output

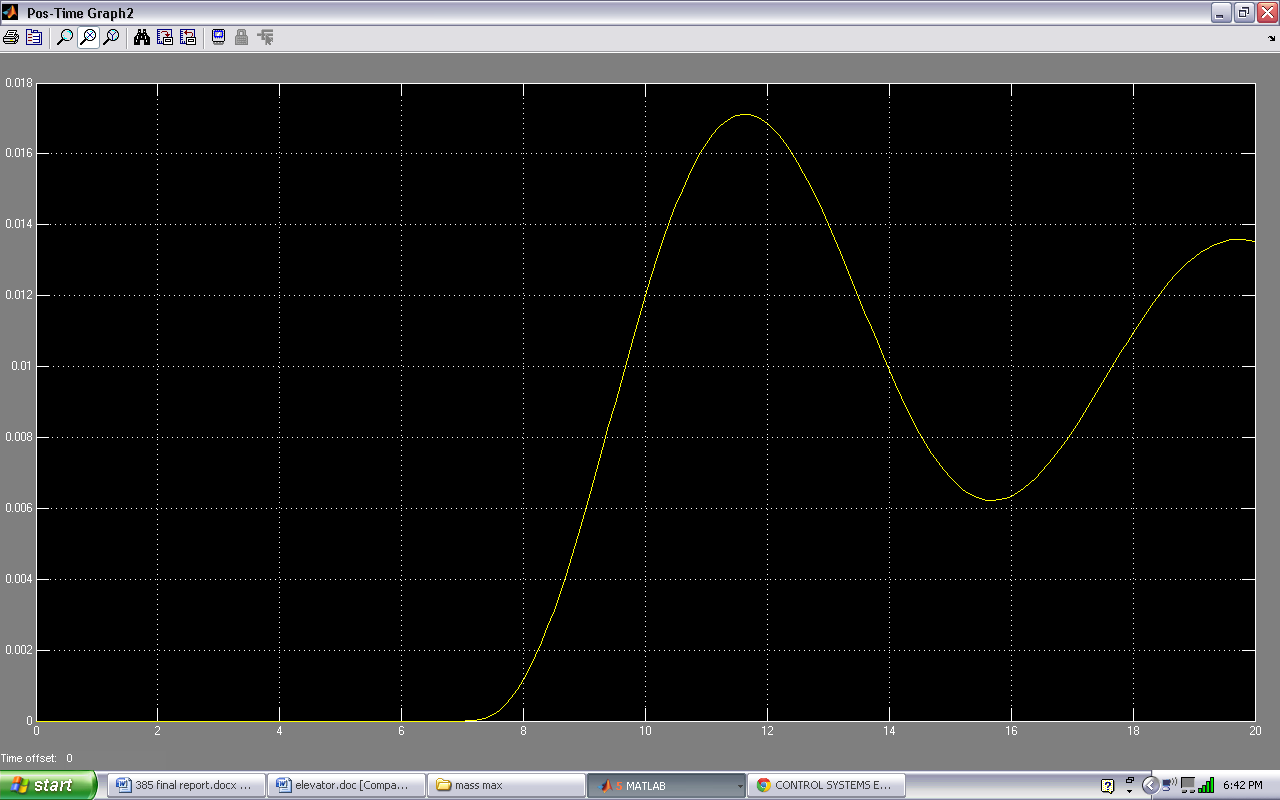


Figure 2 - 5

Rise time > 10 sec

Overshoot > 10%

Steady state error > 2%

### Min mass Output

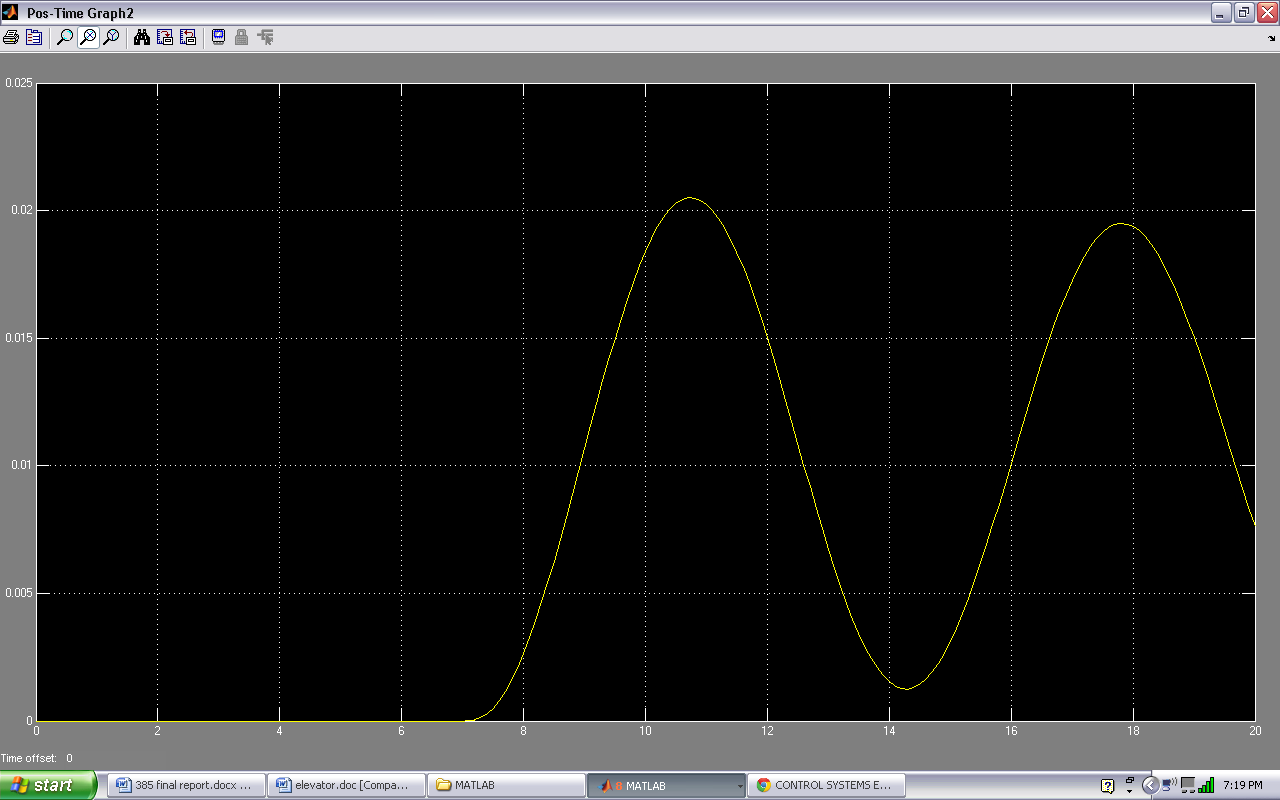


Figure 2 - 6

Rise time > 5 sec

Overshoot > 10%

Steady state error > 2%

## 

## Ramp

Figure 2 - 7C:\Documents and Settings\nali katana\My Documents\MATLAB\elevator_doc_files\image-005-sl.emf

### Input

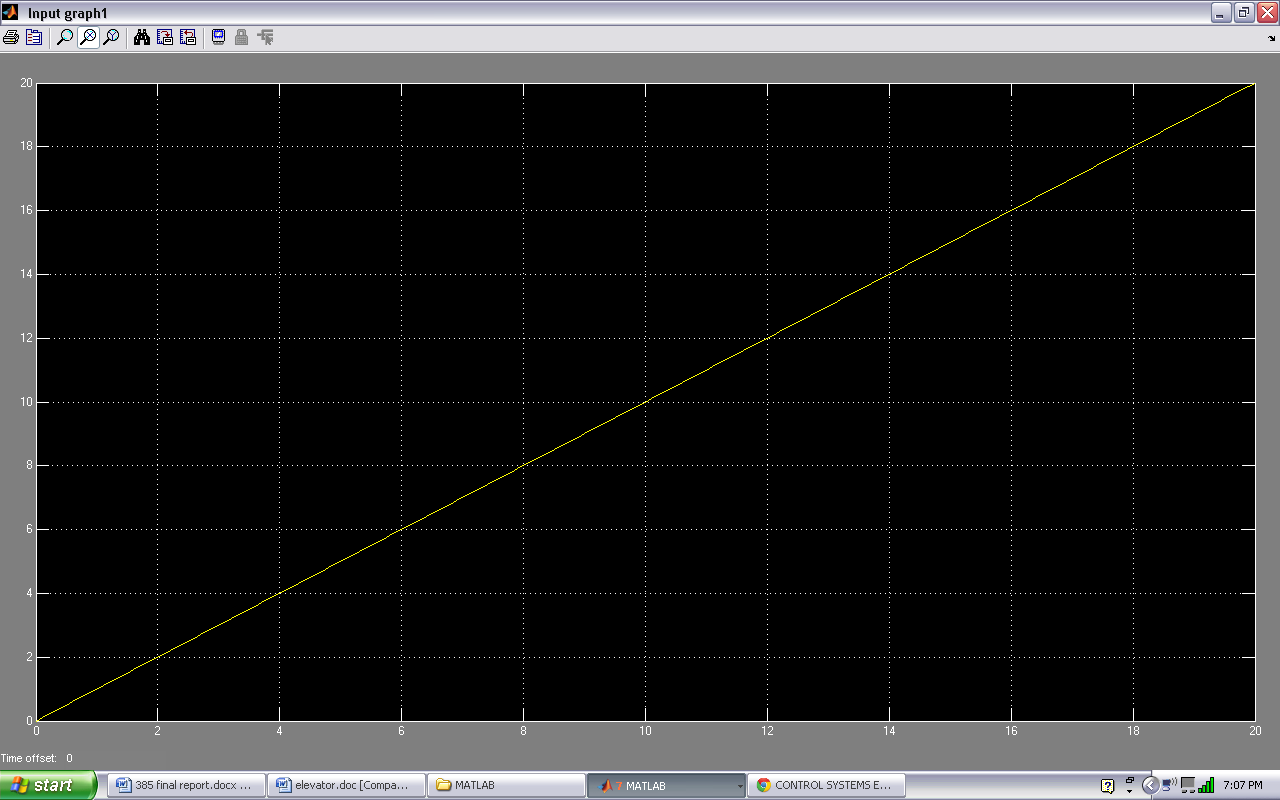


Figure 2 - 8

### Max mass Output

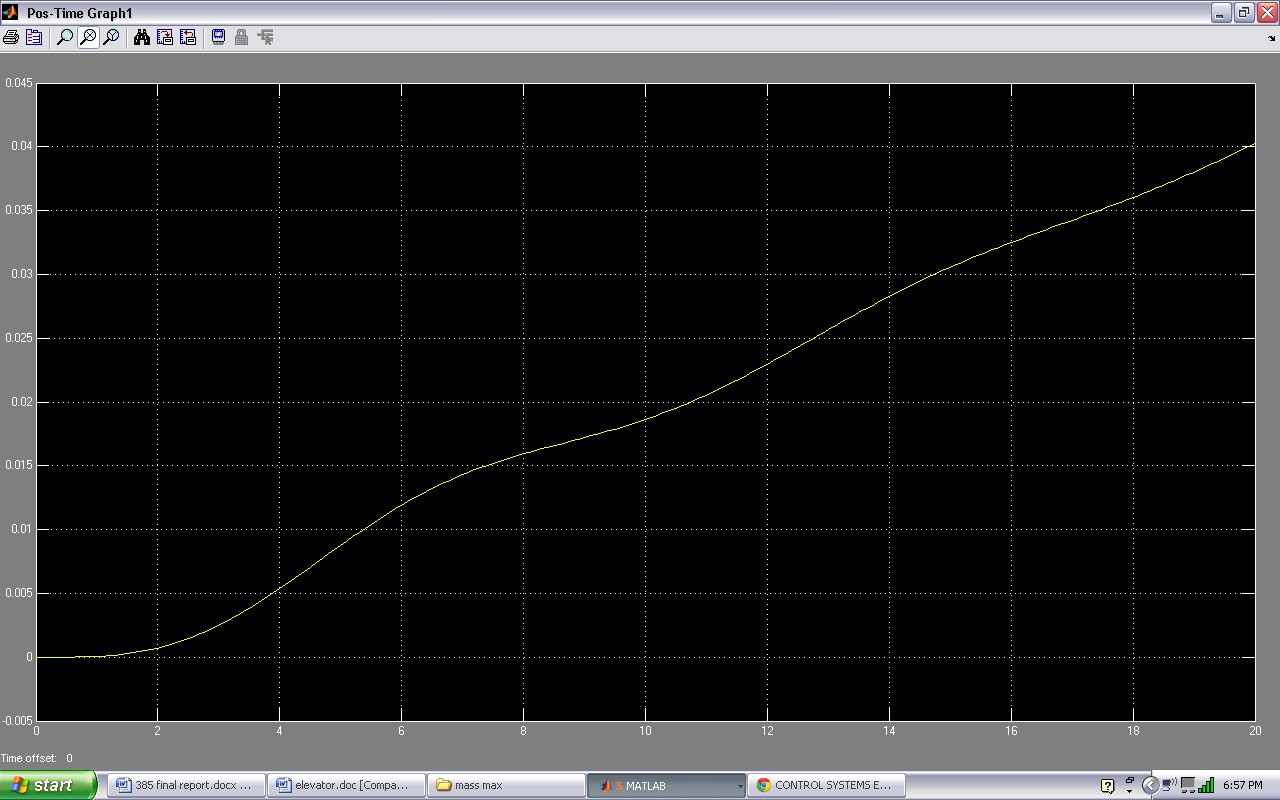


Figure 2 - 9

Rise time > 10 sec

Overshoot > 10%

Steady state error > 2%

### Min mass Output

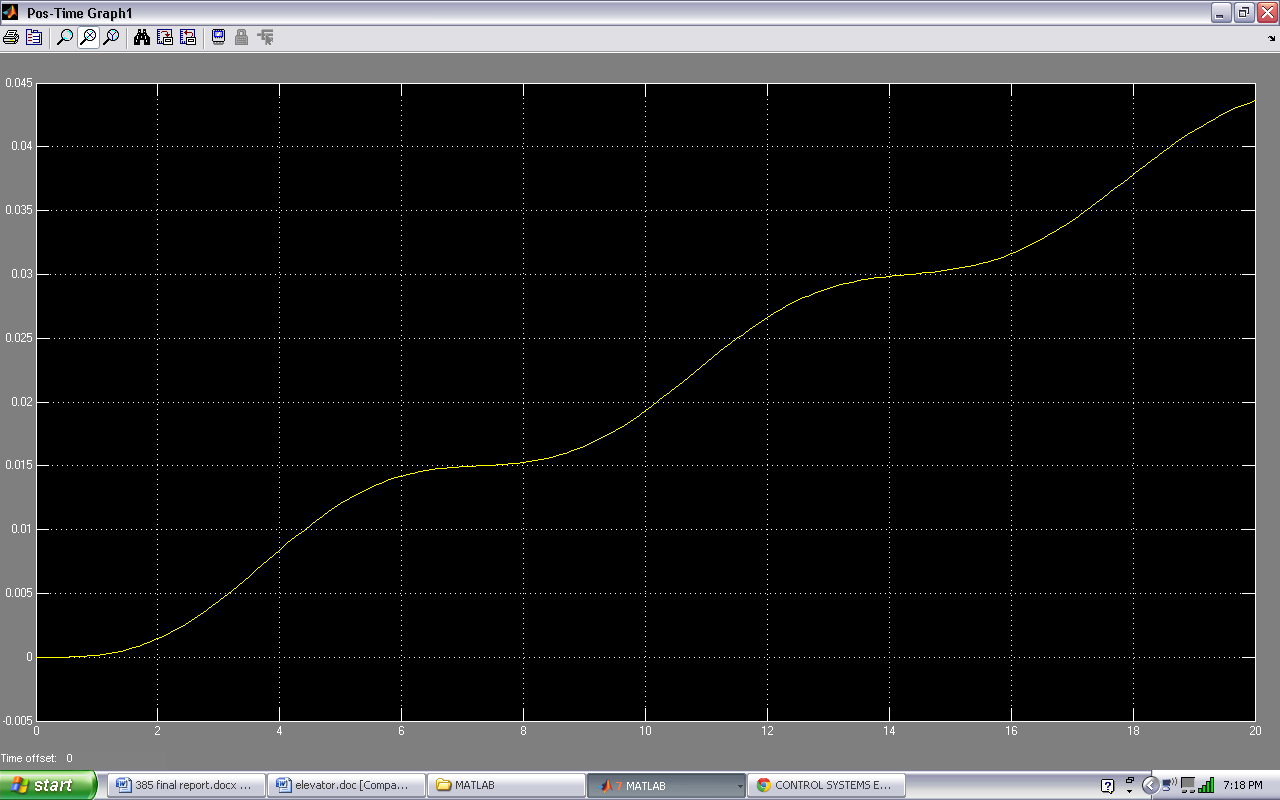


Figure 2 - 10

Rise time > 5 sec

Overshoot > 10%

Steady state error > 2%

## 

## Elevator system(representing parabolic function)

**C:\Documents and Settings\nali katana\My Documents\MATLAB\elevator_doc_files\image-005-sl.emf**

Figure 2 - 11

### Input

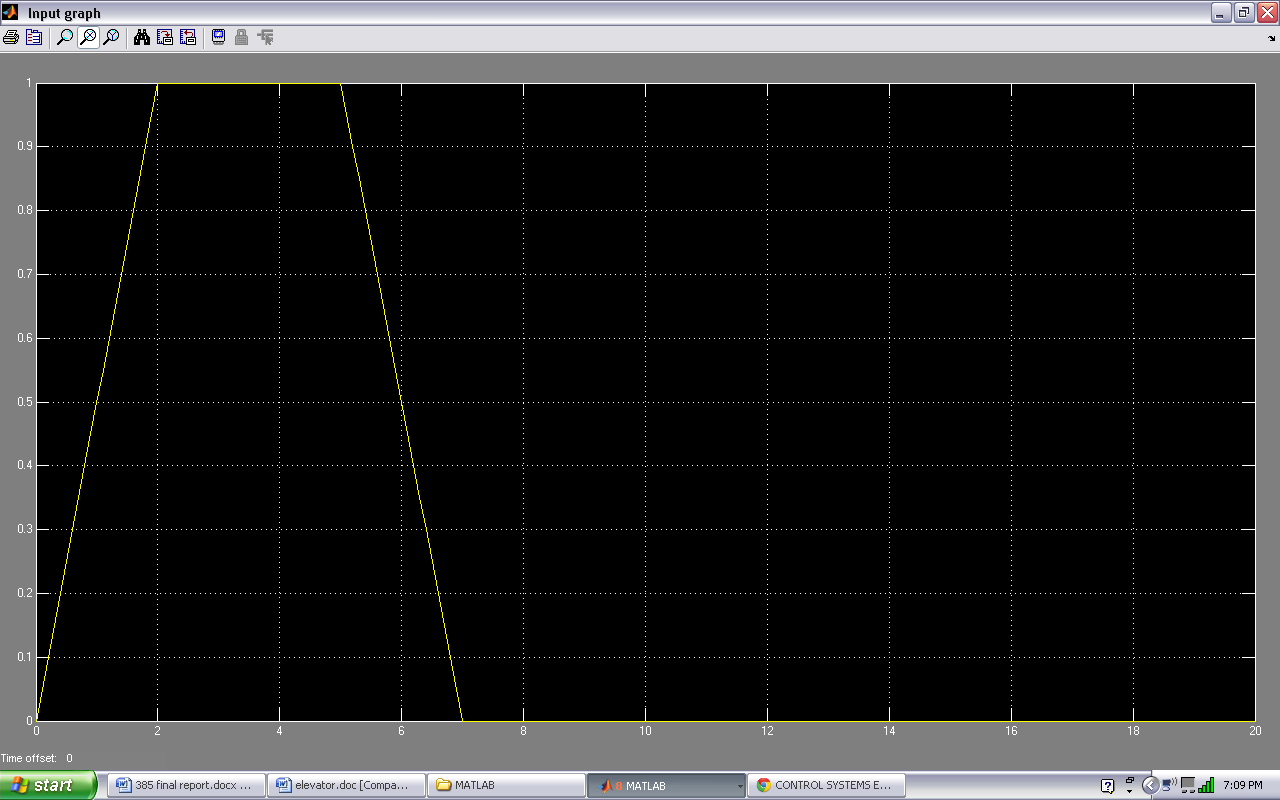


Figure 2 - 12

Rise time <10 sec

Overshoot < 10%

Steady state error < 2%

The above represents the velocity curve we desire and when we pass this signal through an integrator, it is translated to the position signal which we require to be passed to the world. We only required this velocity input so as to be able to allow a smooth rise and drop of speed of elevator as it is in real life situation because inertia can be a hazard to passengers if velocity is instant.

The graph of the ideal expected output signal after integration is below where it settles at 5 which represents a gradual movement of elevator 5m vertically and stops. This 5m in this case, is obtained after a gui floor button is clicked which is then processed in the m-file depending on which floor one is currently in hence if the movement is one floor up or down, it is registered as -5 or 5 respectively. If two it will be -10 or 10 and so on so forth. This value is then set in the necessary input simulink blocks as:

-Velocity rise:

We perform a product of a ramp block of slope 0.5 and a step block of time 2 and initial value of -1 or 1 depending on direction of elevator expected. Hence the rise in the graph above to 1 from time 0 to 2.

-Velocity constant:

We perform a product of two step blocks, one with time 2 and initial value of 0 and final 1 while the other of time 7 and initial value 1 and final 0. This difference in time is where our position input of 5 from the gui comes to play. So this is where any change occurs in these two. Hence if this was two floors up or down, the 1st step block would have time 2 while the other has time 12 i.e. 5x2.

-Velocity drop:

We perform a product of a step block of initial value 1 and final 0 with step time of 7 with a ramp of start time of 5 and slope of -0.5. This is the opposite of the velocity rise set of blocks with difference that it is set to drop to 0 by time 7.

Conclusively, we assume that this set up is our hydraulic actuator force that acts at a specific velocity and translated by integration to a specific position we wish to move the elevator to. We could have used a plain input signal without the rate of change but we figured it a better challenge to do it this way as per the requirements we set initially of a comfortable ride. Below is the graph of the integrated velocity.

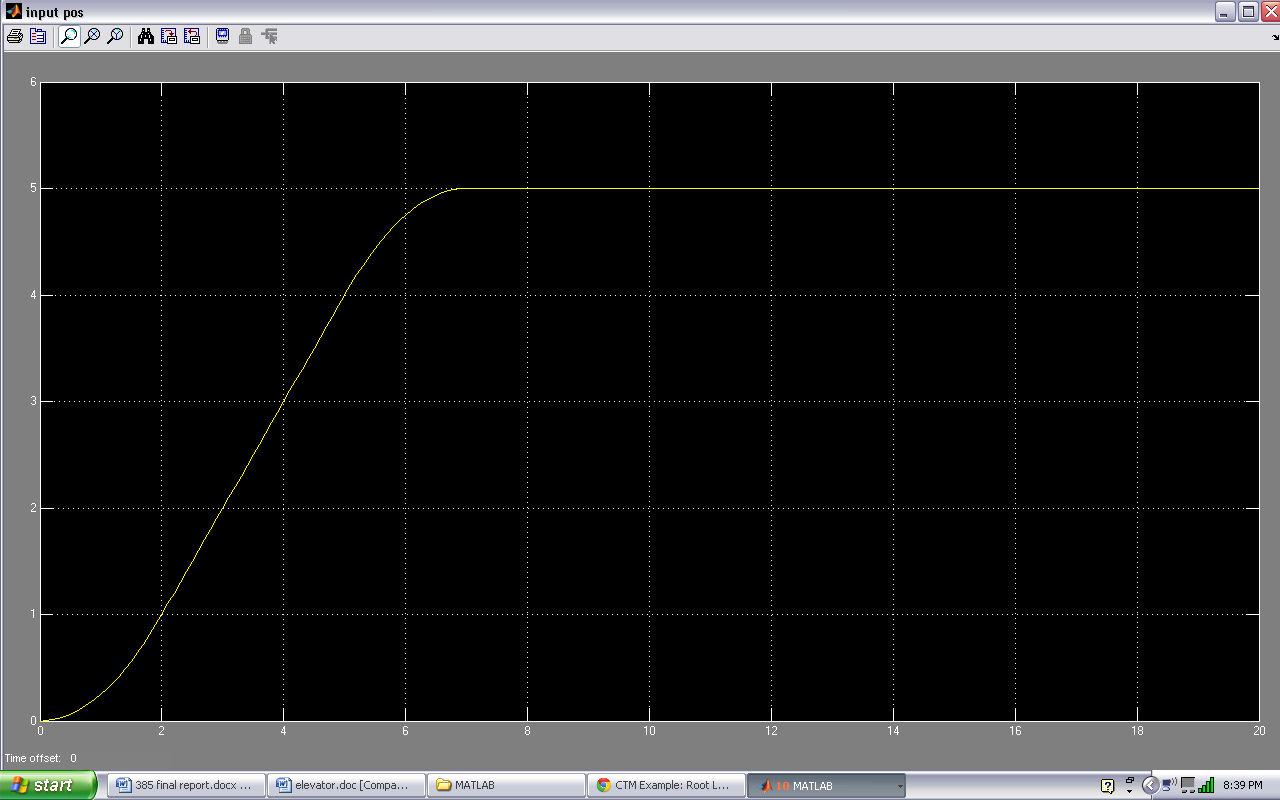


Figure 2 - 13

Rise time <10 sec

Overshoot < 10%

Steady state error < 2%

### Max mass output velocity

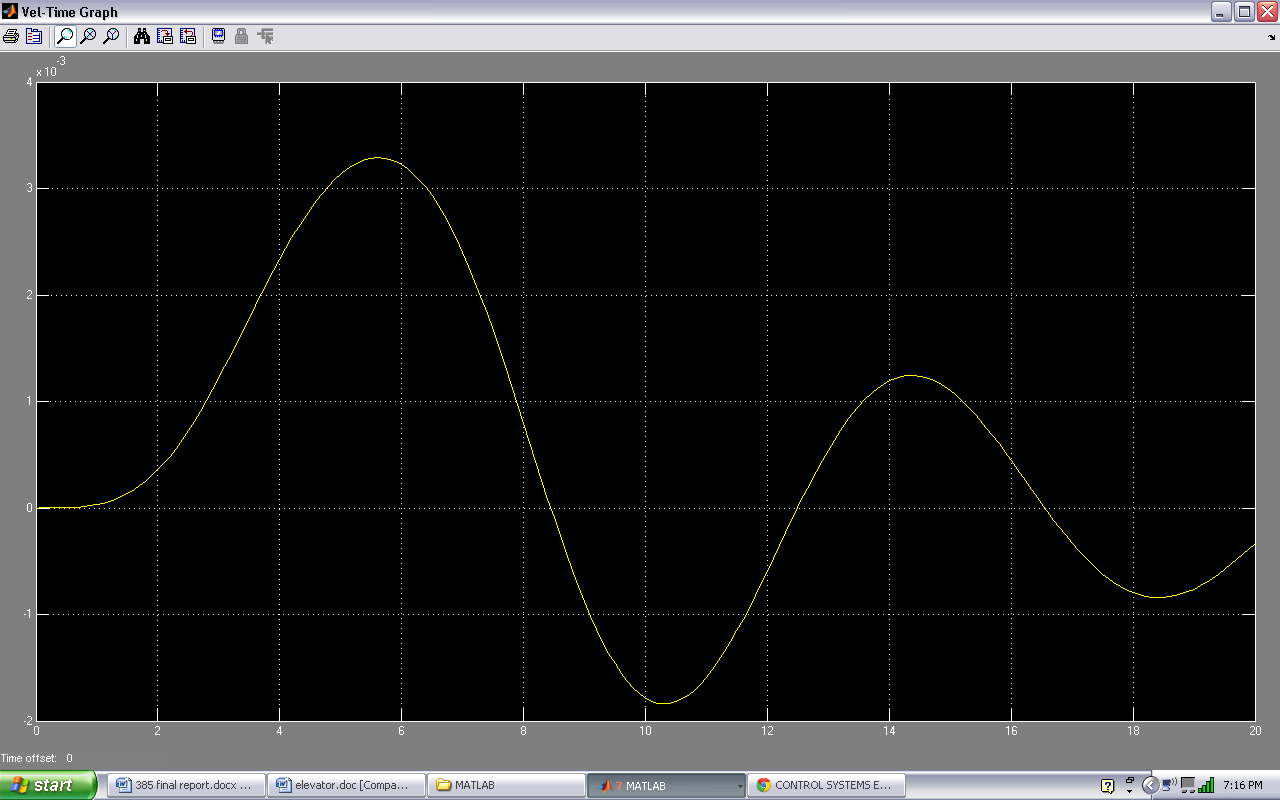


Figure 2 - 14

Rise time <10 sec

Overshoot > 10%

Steady state error > 2%

### Max mass output position

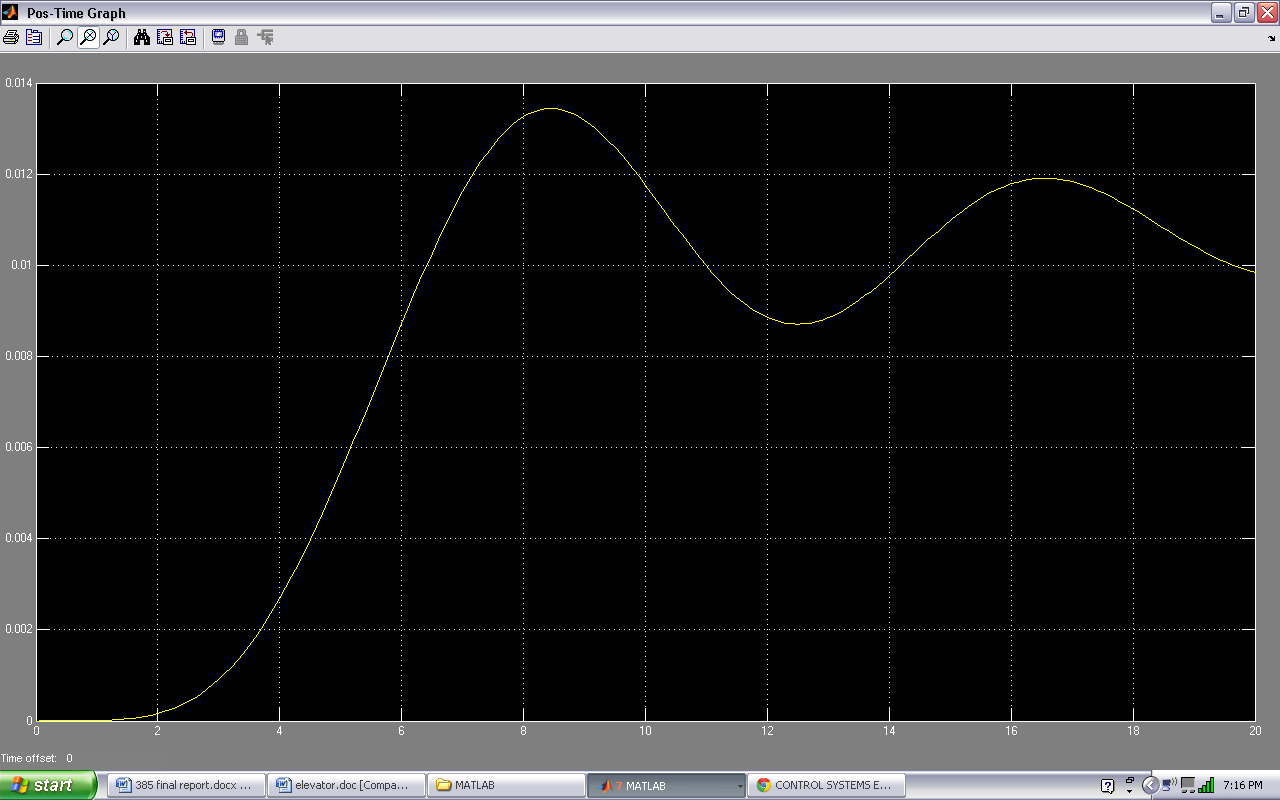


Figure 2 - 15

Rise time <10 sec

Overshoot > 10%

Steady state error > 2%

### Min mass output velocity

Figure 2 - 16

Rise time < 5 sec

Overshoot > 10%

Steady state error > 2%

### Min mass output position

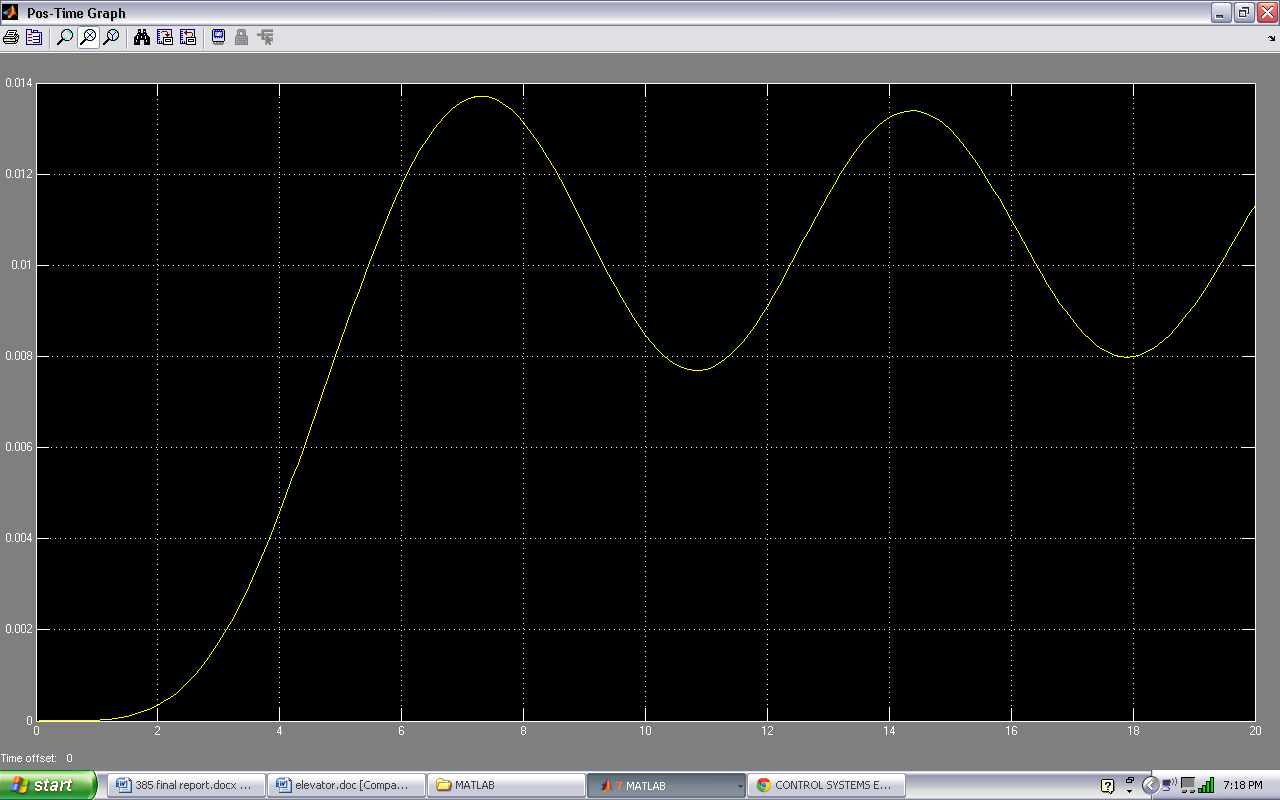


Figure 2 - 17

Rise time > 5 sec

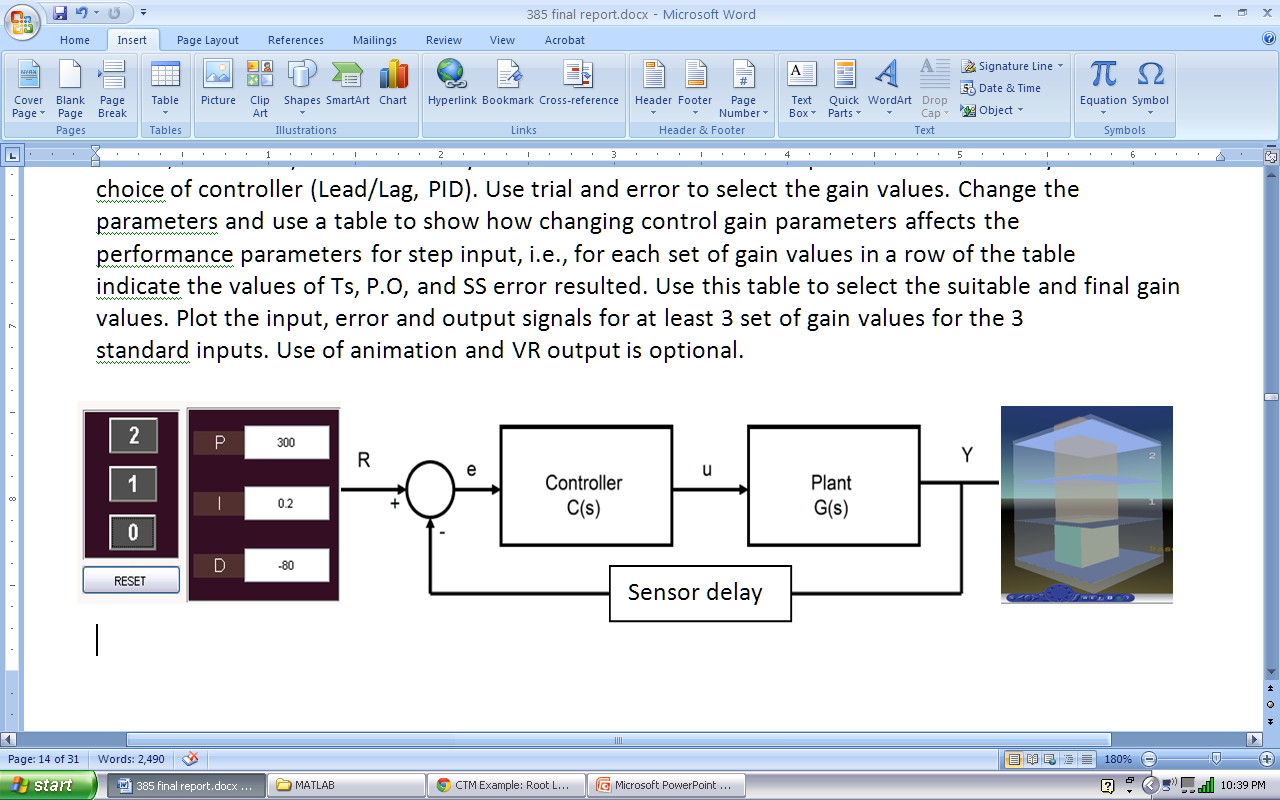
Overshoot > 10%

Steady state error > 2%

Feed-back loop control

## Block diagram

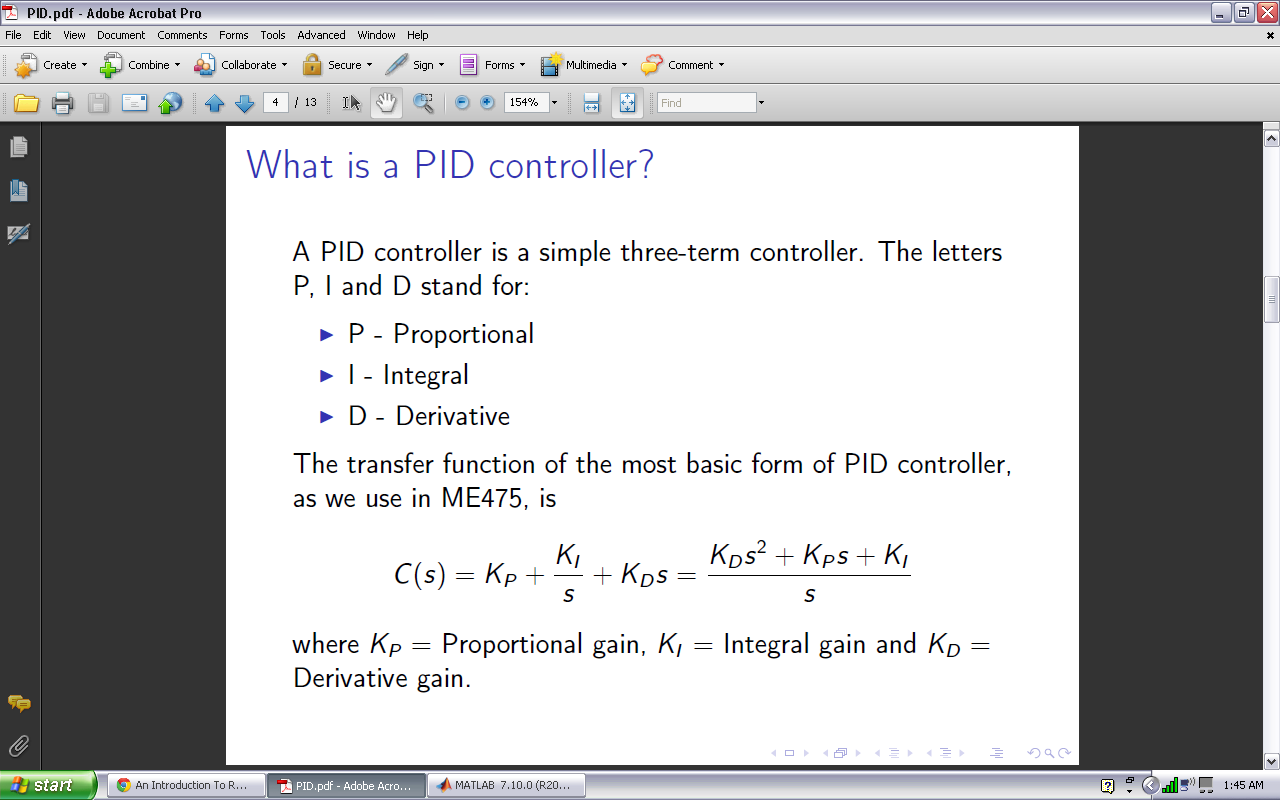
Below, is the block diagram of the proposed system. On the left is the GUI which sets the inputs and generates a force that injects a velocity curve into the feedback loop. In the middle are the controller, transfer function and sensor delay blocks and on the right is the elevator body which moves according to the integrated velocity curve i.e. position.



∫

Figure 3 - 1

### Strategy of Improvement: PID Controller



P-Proportional-Decrease Tr

I- Integral-Decrease OS and Ts

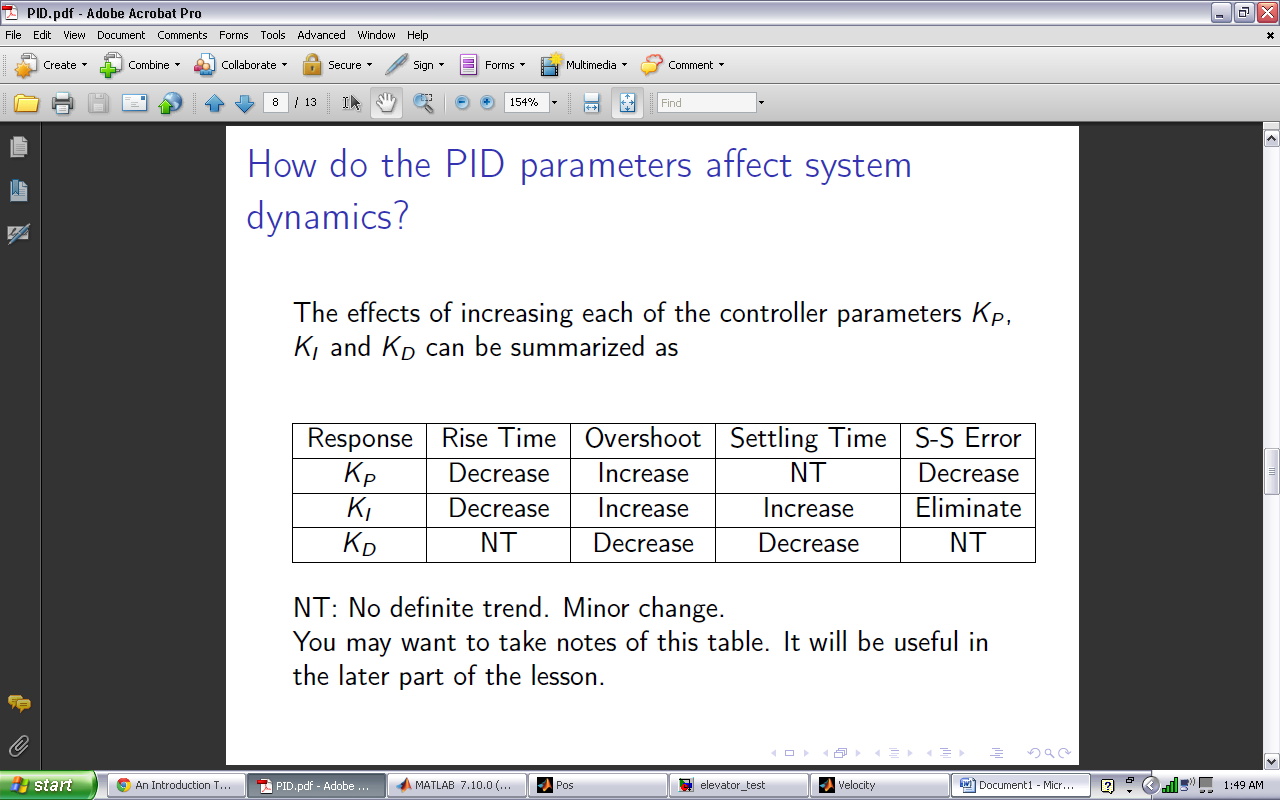
D-Derivative-Eliminate SSE

Benefits

-Used and proven in over 95% of all closed-loop unity feedback control systems

-Tuning is very simple ie very little complexities especially with the use of the Table 1 below.

Table 1



TUNING TABLE:

## Matlab tools

### Elevator World

This is our elevator design. With 3 floors of 5m each, basically a basement, a 1st floor and a 2nd.

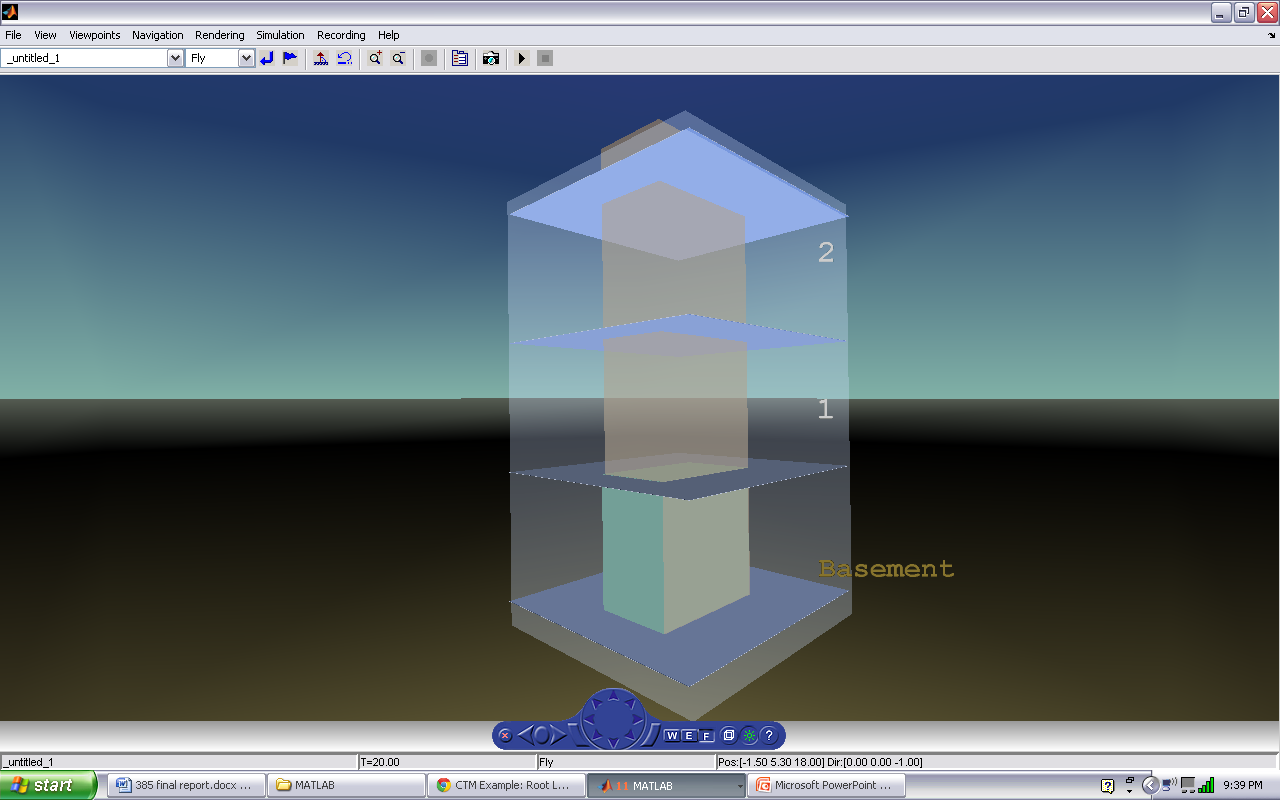


Figure 3 - 2

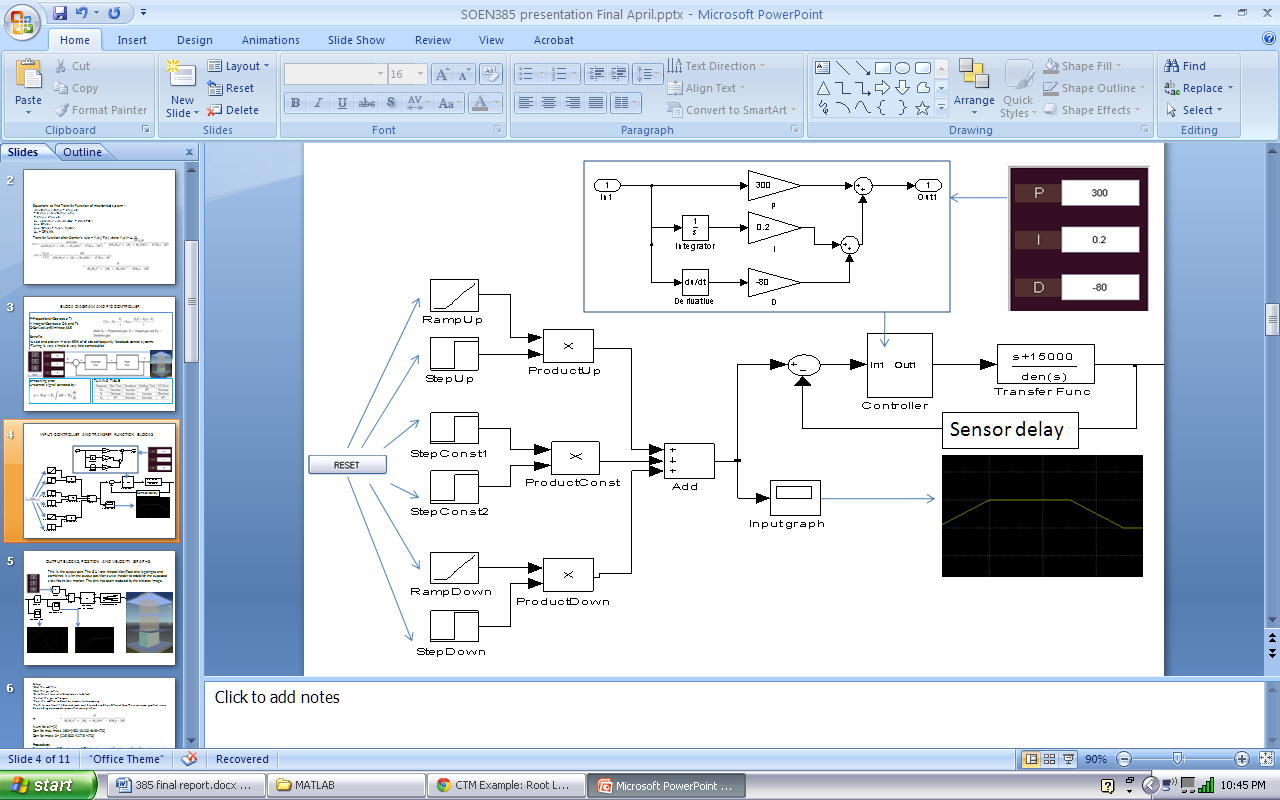
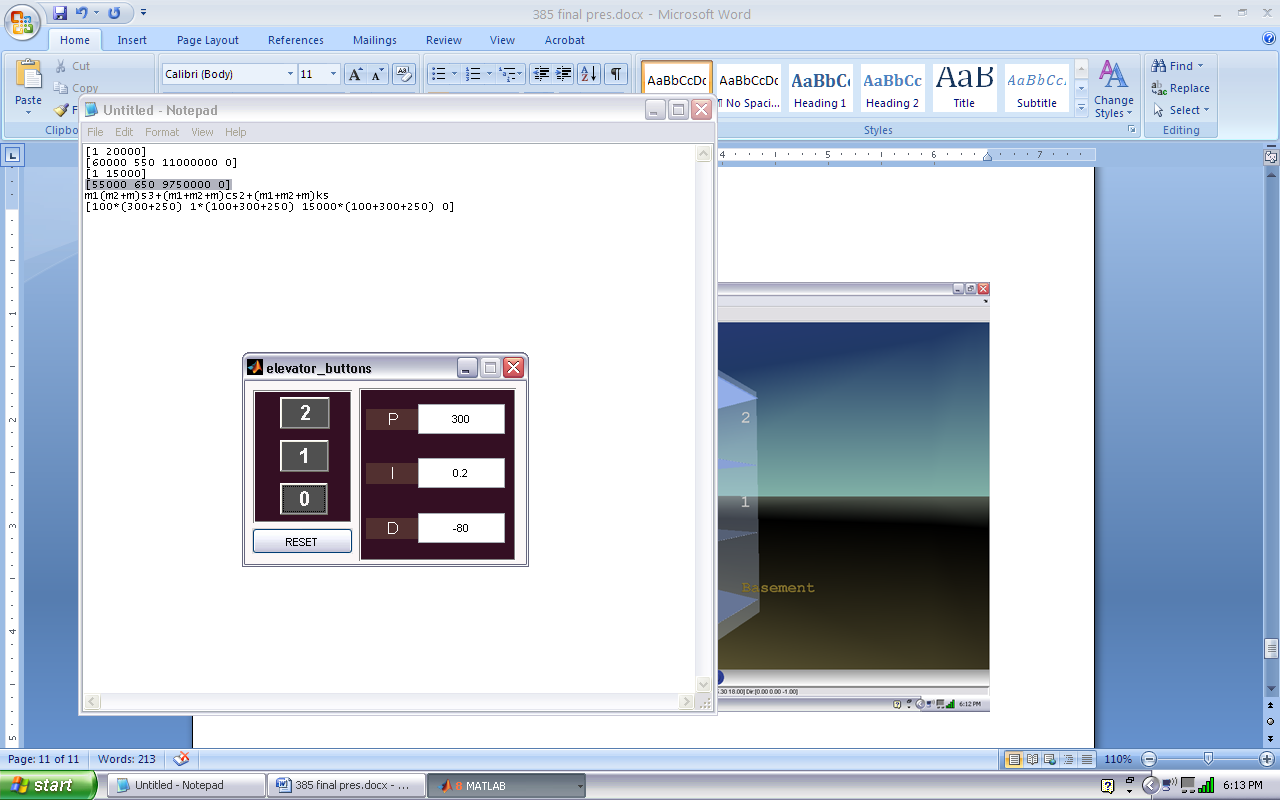
Simulink design

Below is the simulink model of our system. It consists of 4 step and 2 ramp inputs which are combined to one signal as input, a PID controller block, transfer function block, a transport delay with value of 6 as sensor feedback delay and an integrator for the output to convert the output signal to position. There is a position block that is added to this position signal and that block holds the current position the elevator is because the input signal only represents changes in position but has no history of which position is in context. For example a move from floor 1 to floor 2 is similar in input curve to a signal of floor 2 to 3. Both represent a change upwards by 5 meters. But in the end we wish to know which position we are actually moving from and this is kept in this position block and hence if we were in position 0 ie basement(0metres), and we moved to position 1, the output signal would send a position signal of positive 5 which represents the change and our position block holding the current position ie 0metres(basement) will be summed with this signal of positive 5 and hence the elevator will know it is to move 5 metres up from position 0 and not any other position. We only send a y signal to the vr signal expander because this system only deals with vertical motion.

C:\Documents and Settings\nali katana\My Documents\MATLAB\elevator_doc_files\image-005-sl.emfFigure 3 - 3

Sensor delay was added to the feedback loop by request from the report by using a transport delay block with a time delay of 6 in the feedback. There are numerous scope blocks in the model for graphical viewing purposes of the responses.

#### INPUT, CONTROLLER AND TRANSFER FUNCTION BLOCKS

Figure 3 - 4

This is basically the control system. The input ramps and steps below were used to obtain the desired velocity curve. The controller can be set using the GUI and the TF block contains all the coefficients concerened. There is an input scope to view the input response. The controller block is basically a proportional gain, integrator and a derivative block put together. The reset button of the gui is called only to set default parameter values of the input blocks. i.e. the code below:

set\_param('elevator/StepUp', 'Before', num2str(1));%sets initial value(-1=down,1=up)

set\_param('elevator/StepConst2', 'Time', num2str(7));%sets step time

set\_param('elevator/StepConst2', 'Before', num2str(1));%sets initial value(-1=down,1=up)

set\_param('elevator/RampDown', 'start', num2str(5));%sets start time

set\_param('elevator/StepDown', 'Time', num2str(7));%sets step time

set\_param('elevator/StepDown', 'Before', num2str(1));%sets initial value(-1=down,1=up)

set\_param('elevator/Position', 'Value', num2str(0));%sets current position block

These 7 parameters influence the input blocks for the desired amount of signal. When a new floor is pushed for, only 5 of these are affected namely:

increase = 5\*abs(destination\_floor - current\_floor) + 2;

set\_param('elevator/StepUp', 'Before', num2str(direction));

set\_param('elevator/StepConst2', 'Time', num2str(increase));

set\_param('elevator/StepConst2', 'Before', num2str(direction));

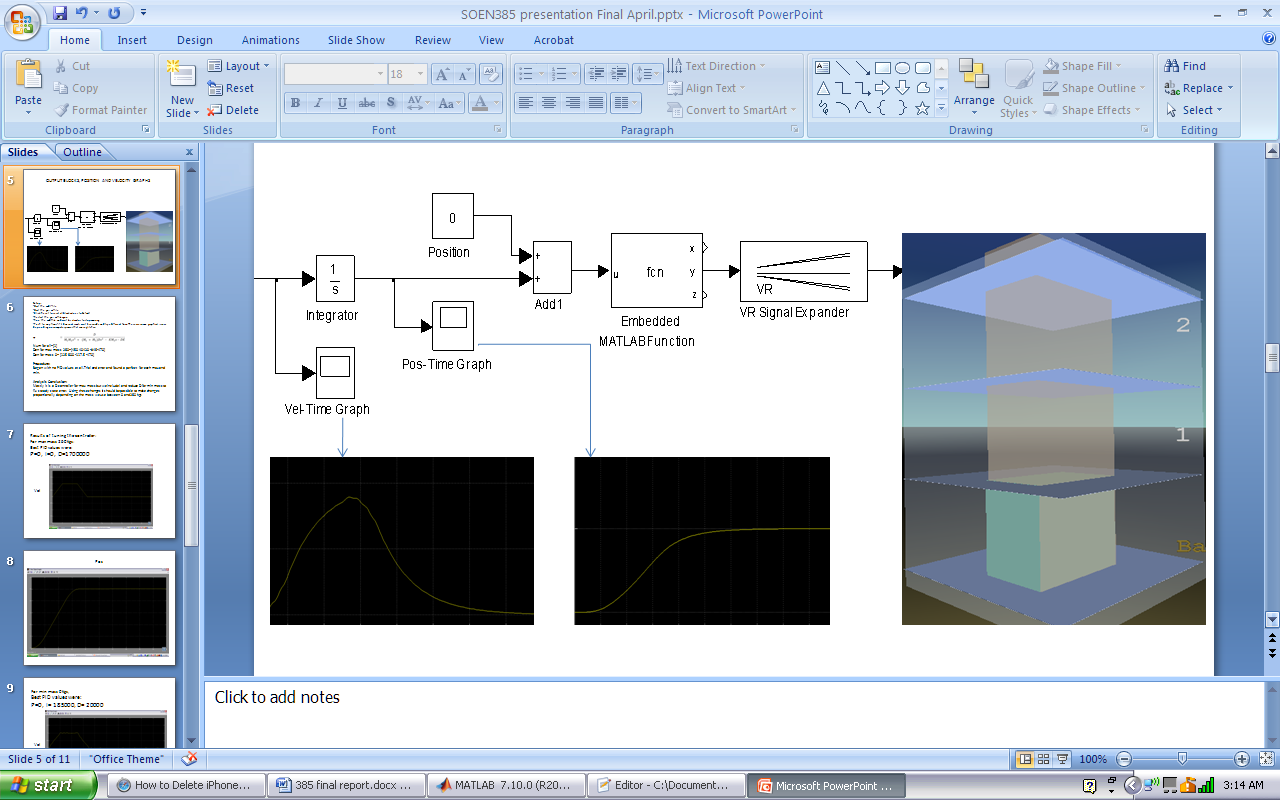
set\_param('elevator/RampDown', 'start', num2str(increase - 2));

set\_param('elevator/StepDown', 'Time', num2str(increase));

set\_param('elevator/StepDown', 'Before', num2str(direction));

Where ‘direction’ is -1 or 1 depending on down or up respectively which result in a graph below or an inverted one above the x axis respectively. Step time is set by multiplying 5(height of one floor) by the number of floors changed. The 2 included in the equation was necessary due to a default step time we had initially set for the upward rise of the velocity.

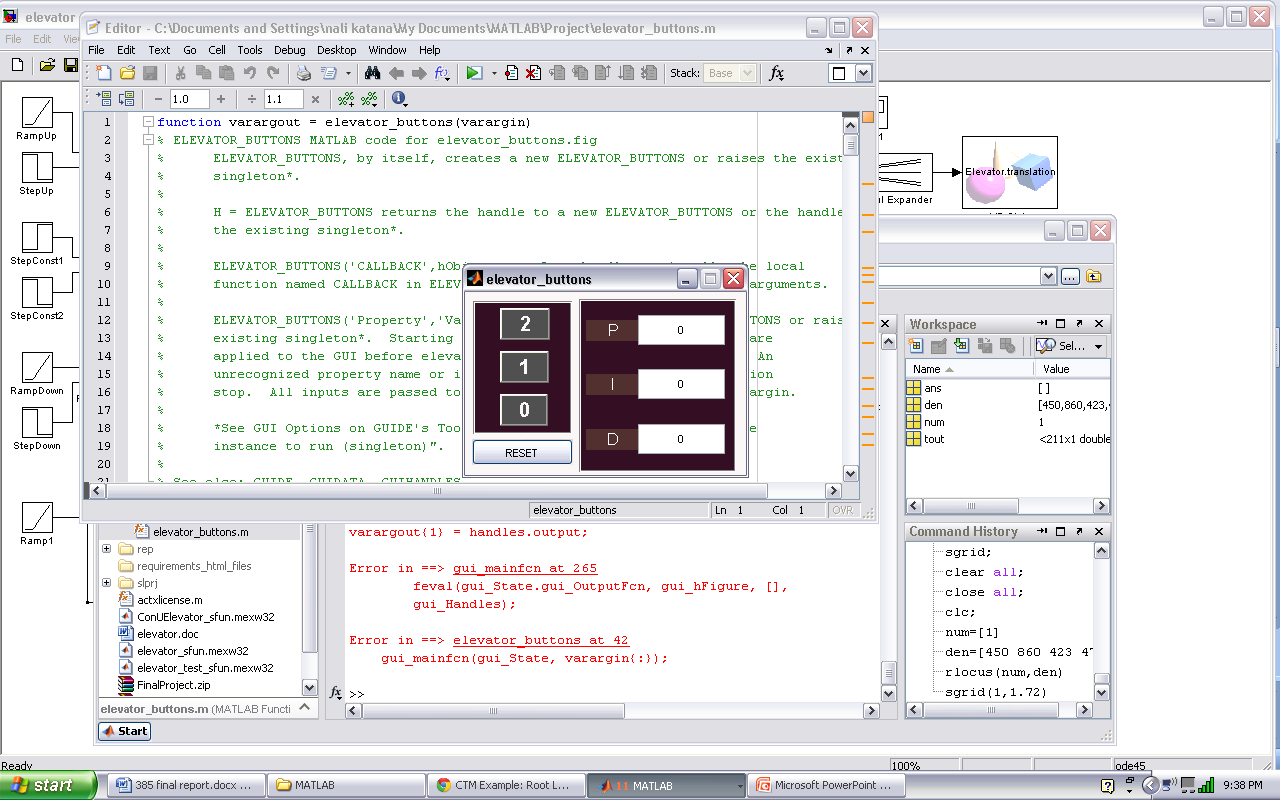
#### OUTPUT BLOCKS, POSITION AND VELOCITY GRAPHS

Figure 3 - 5

This is the output part. The position block as said before contains the current position which is necessary to be added to the change. The integrator converts the output velocity signal to a position signal. The sink has been replaced by the elevator image.

### GUI

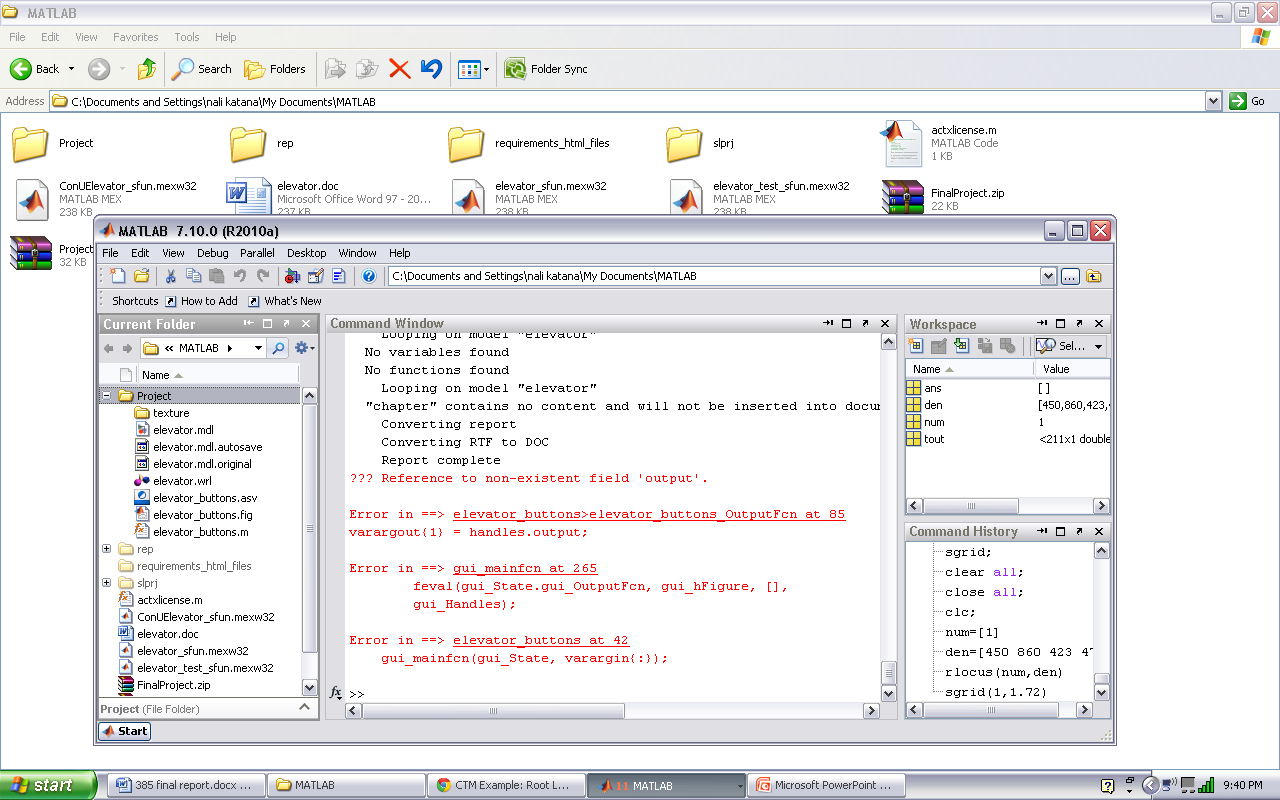
The gui is simple, created using GUIDE and handled by the m-file. Contains 4 buttons, 3 are for floors while 1 is for reset. The text areas are for PID entries the tester may wish to enter. It was to save time going to the mdl and clicking on the controller block each time to change the values. The PID values are set instantly when a button is pushed.

Figure 3 - 6

### Setup of the project files

* Unzip zip package and add paths to matlab
* start the mdl-file and right click the sink block, select mask parameters and set right path of the world wrl-file
* start the m-file and run it for the GUI to display
* click reset on GUI to reset all block values to default
* restart the gui m-file again
* run the mdl-file and wait for elevator in world to stop moving
* push to any floor- if it does not react, wait 5 seconds and try a different floor. There are some graphical issues depending on computer power that cause glitches.

### File directory structure:

Figure 3 - 7

The texture is auto generated folder containing texture information. The mdl is the simulink file. The wrl is the world. The elevator\_buttons files are the m-file and files necessary for the GUI to load and work. The code is contained in the m-file containing call-back functions and necessary logic for our elevator to work. We also implemented queuing just incase two calls were made consecutively before the elevator got to its destinations. Some bugs may occur with that. But its best to wait for the elevator to reach its destination in order to push another floor.

## Data table and graphs

Below is the PID controller tuning. Changes can be noted from one tune to another. We have provided various graphs to show the progress before the final tune.

Using:

-Max Mass of people m=260kg  
-Min Mass of people m= 0kg  
-Mass of elevator body M2= 100kg  
-Mass of elevator shaft M1=500kg  
-Spring K= -470N/m  
-Damping D= 400Ns/m

Note: initially we used k as 470 for our demo but instructor suggested a smaller k inorder to see more oscillations hence we just changed it to -470 and tuned for that. For the data columns in this section, read the values from top to bottom then move to the next column on the right. They are listed as P,I,D respectively. For the data tables, we listed only 6 groups picked at regular intervals from the data columns that we have justified parameters for. Producing table for all of the data would have been strenuous and unnecessary. Up and down graphs i.e. a floor up or a floor down are the same just inverted. Hence the tests below of upward motion account for both directions.

Below are the graphs of the input signals to compare with the outputs:

### Input velocity and position responses

Figure 3 - 8: Input velocity 1 floor up

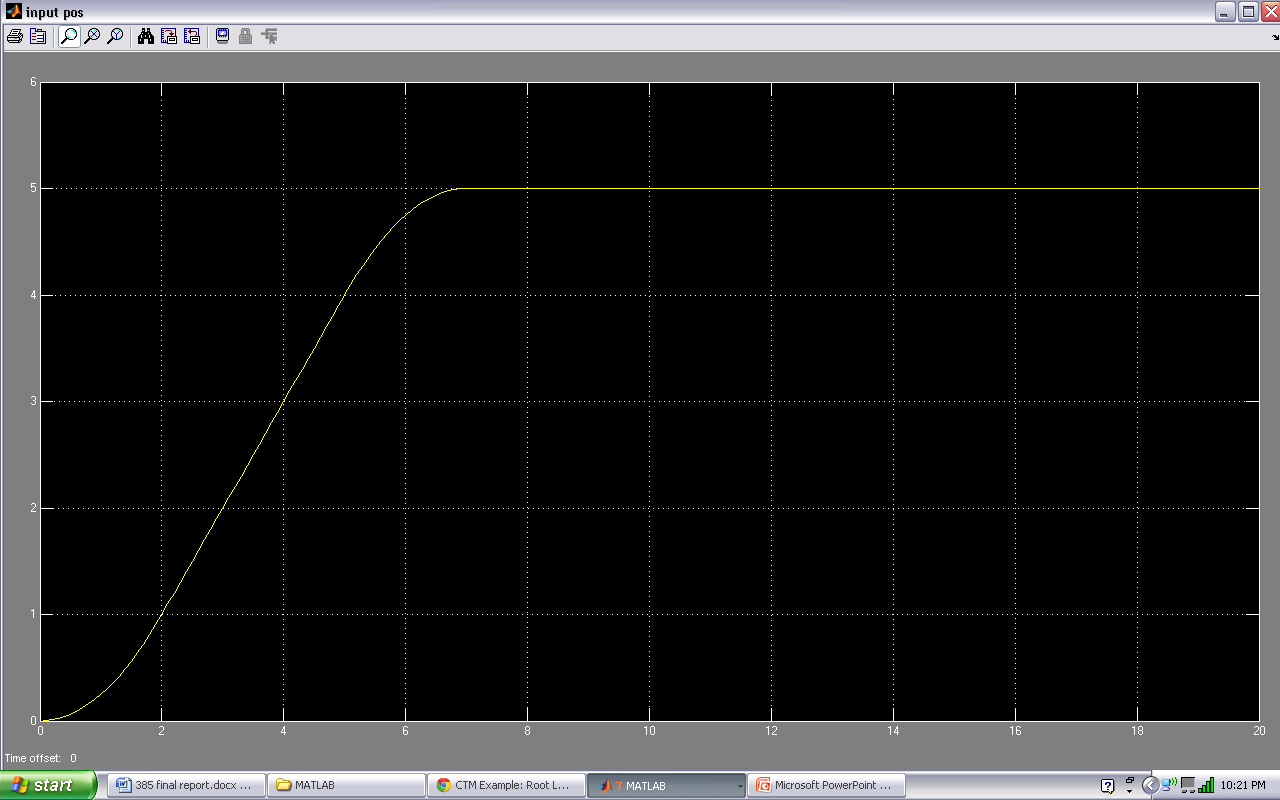
Figure 3 - 9: Input position 1 floor up 

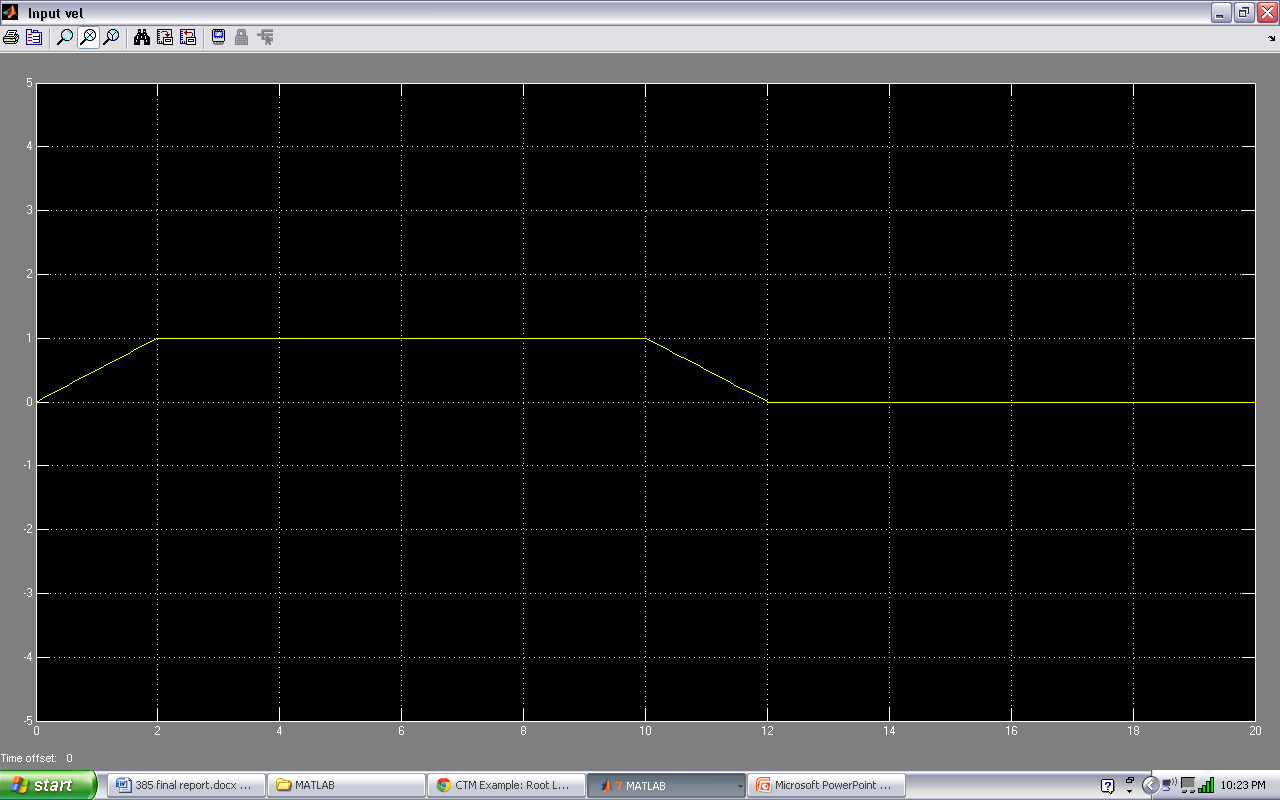
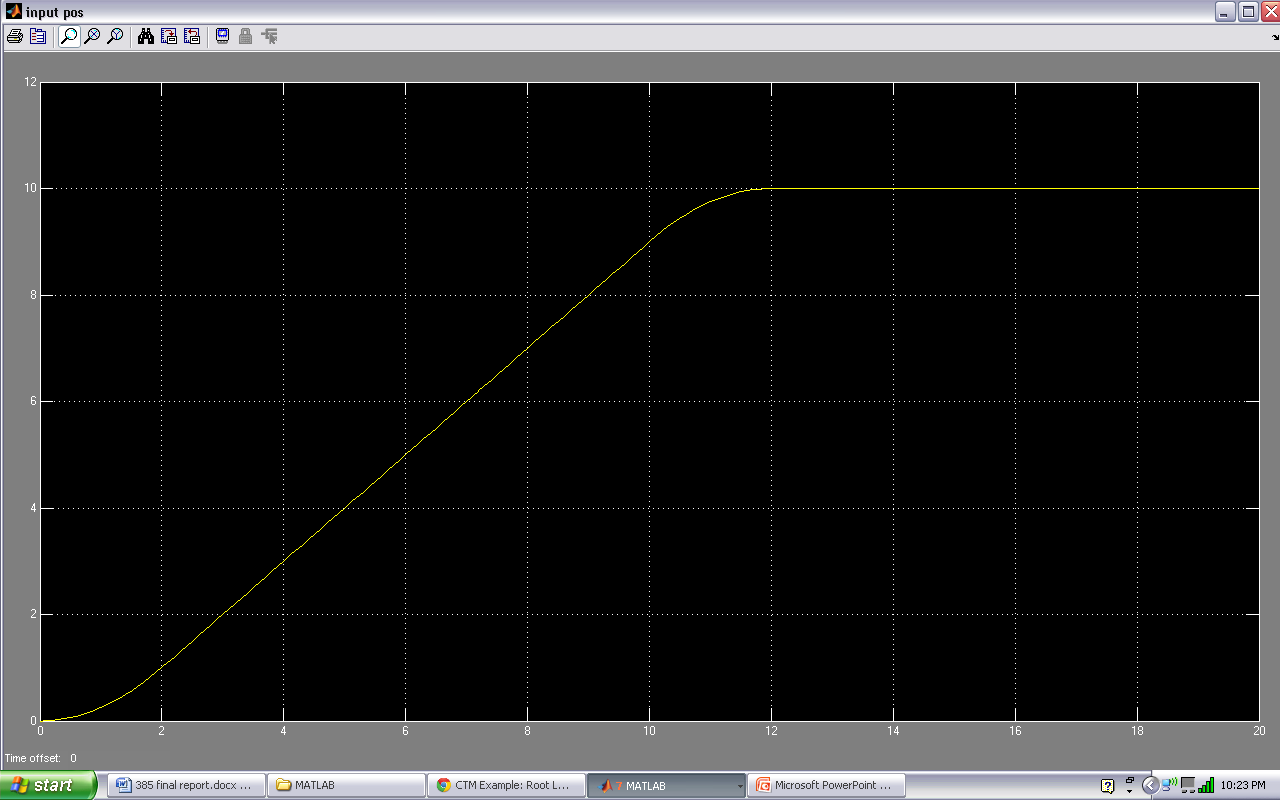
Figure 3 - 10: Input velocity 2 floors up 

Figure 3 - 11: Input position 2 floors up 

### Output velocity and position responses for max mass

Tf=1/[450 860 423 470]

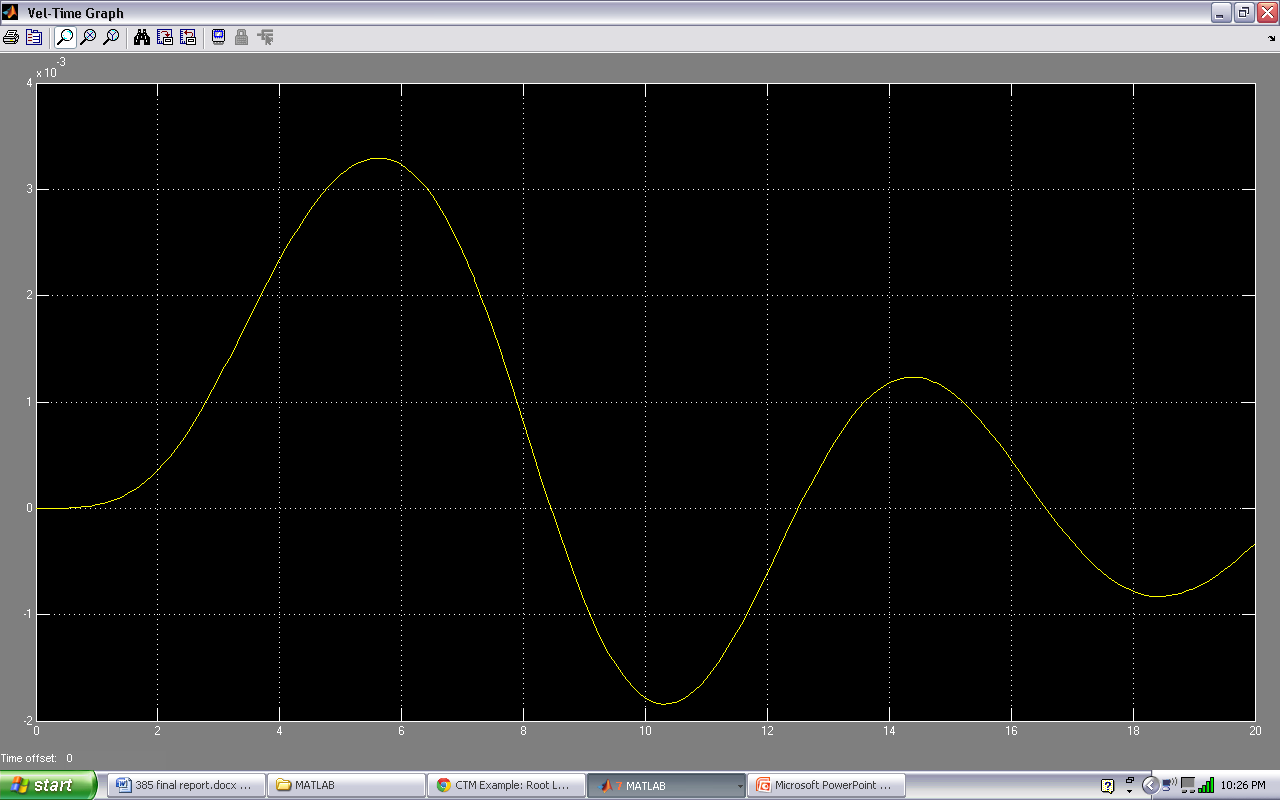
Figure 3 - 12: Uncontrolled output velocity 1 floor up

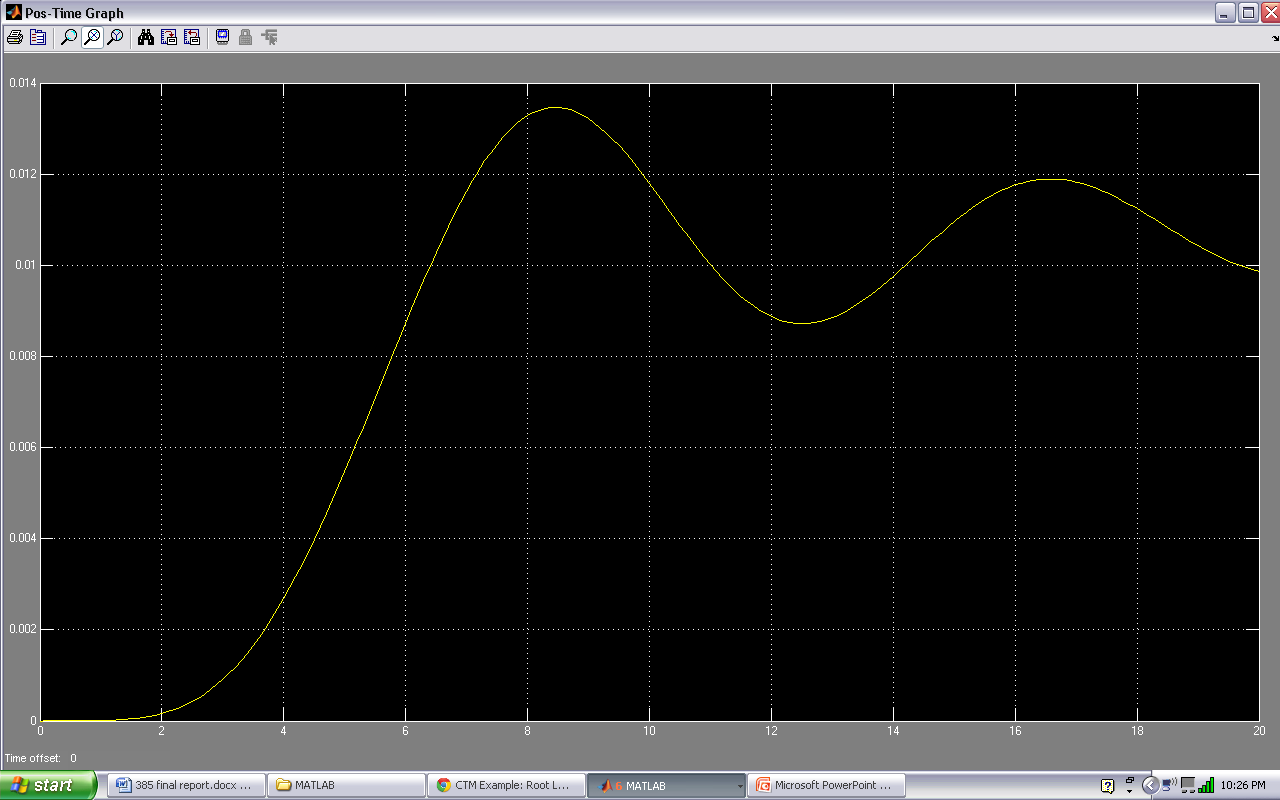
Figure 3 - 13 Uncontrolled output position 1 floor up

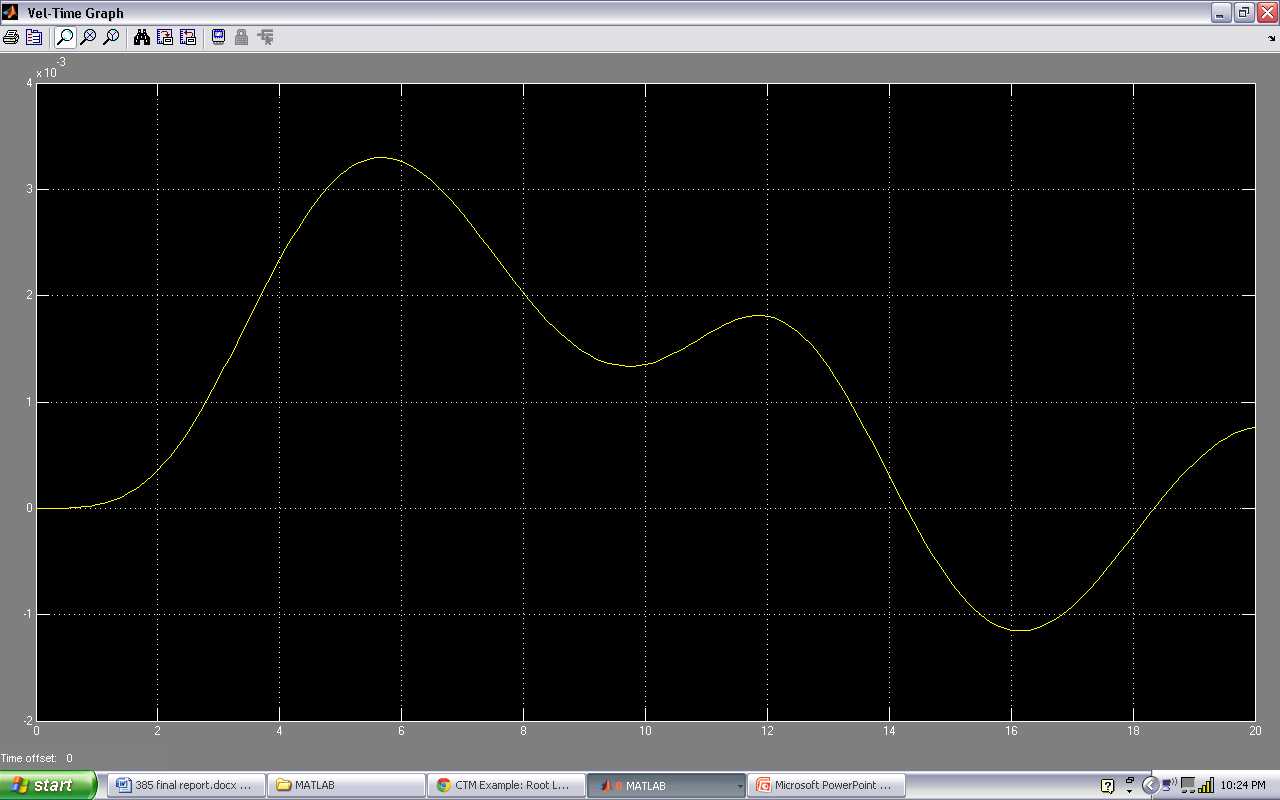
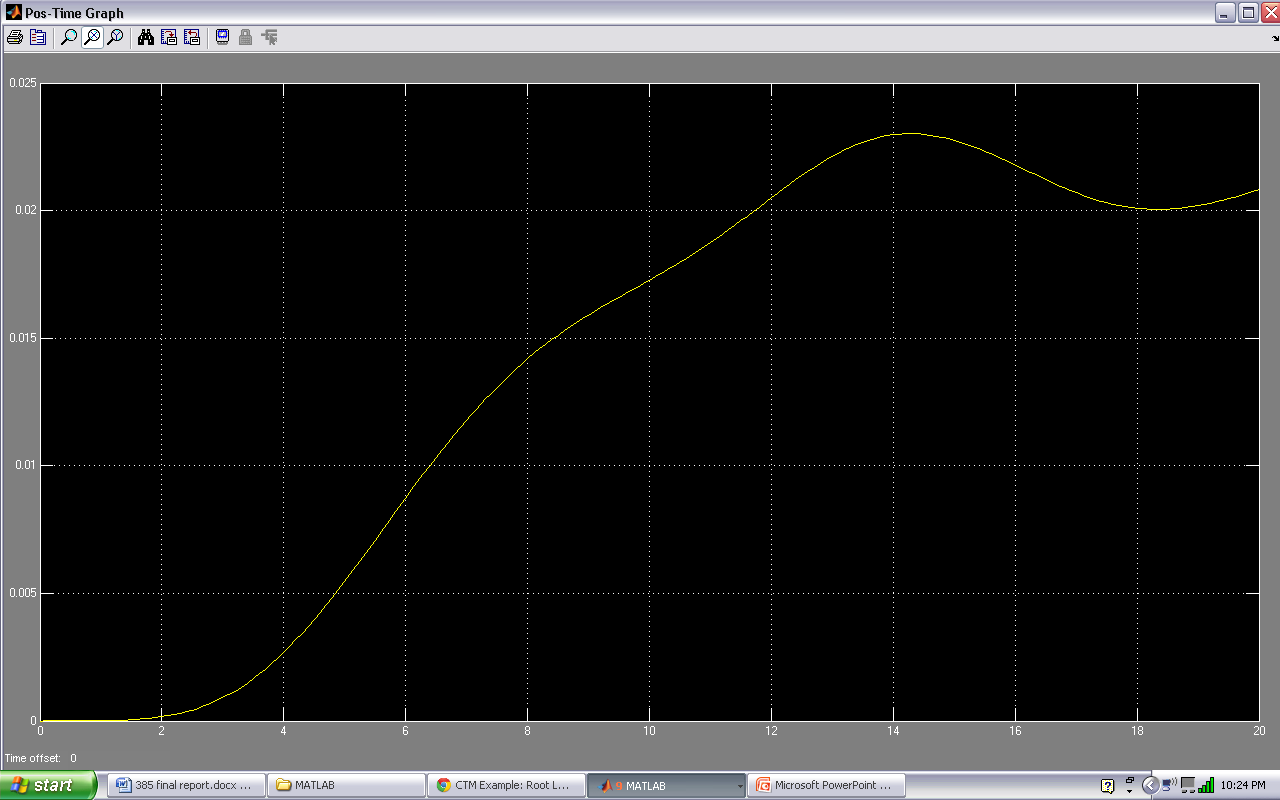
Figure 3 - 14Uncontrolled output velocity 2 floors up 

Figure 3 - 15: Uncontrolled output position 2 floors up 

### Data columns

Best values for max

P=88.5, I=44.8, D=1

240

38

50

140

38

50

90

50

10

95

45.5

10

95

45.5

80

120

45.5

120

120

30

120

150

30

60

150

30

-120

190

42

-120

190

41

-150

190

42

-75

220

42

-75

220

38

-45

100

45

0

100

47

40

100

46.2

51

100

46.2

150

100

45

200

193

53

100

193

48

90

57

41

-28

59

43

-28

64

41.5

-35

60

42

-8

60

43.5

-8

66

43.5

0

71

44

0

82

44.5

0

82

44.5

10

82

44.5

20

83

44.5

-18

83

44.8

10

88.5

44.8

-10

88.5

44.8

-3

88.5

44.8

1

### Data table (for 1 position up)

Rise time < 10 sec

Overshoot < 10%

Steady state error < 2%

Settling time < 20

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| P | 240 | 150 | 100 | 57 | 82 | 88.5 |
| I | 38 | 30 | 45 | 41 | 44.5 | 44.8 |
| D | 50 | 60 | 0 | -28 | 0 | 1 |
| TR | good | good | good | bad | Bad | good |
| TS | good | good | good | good | Good | good |
| OS | good | bad | good | good | Slight | slight |
| SSE | bad | bad | bad | good | Good | good |

Table 2

Tuning samples

240,38,50

#### 

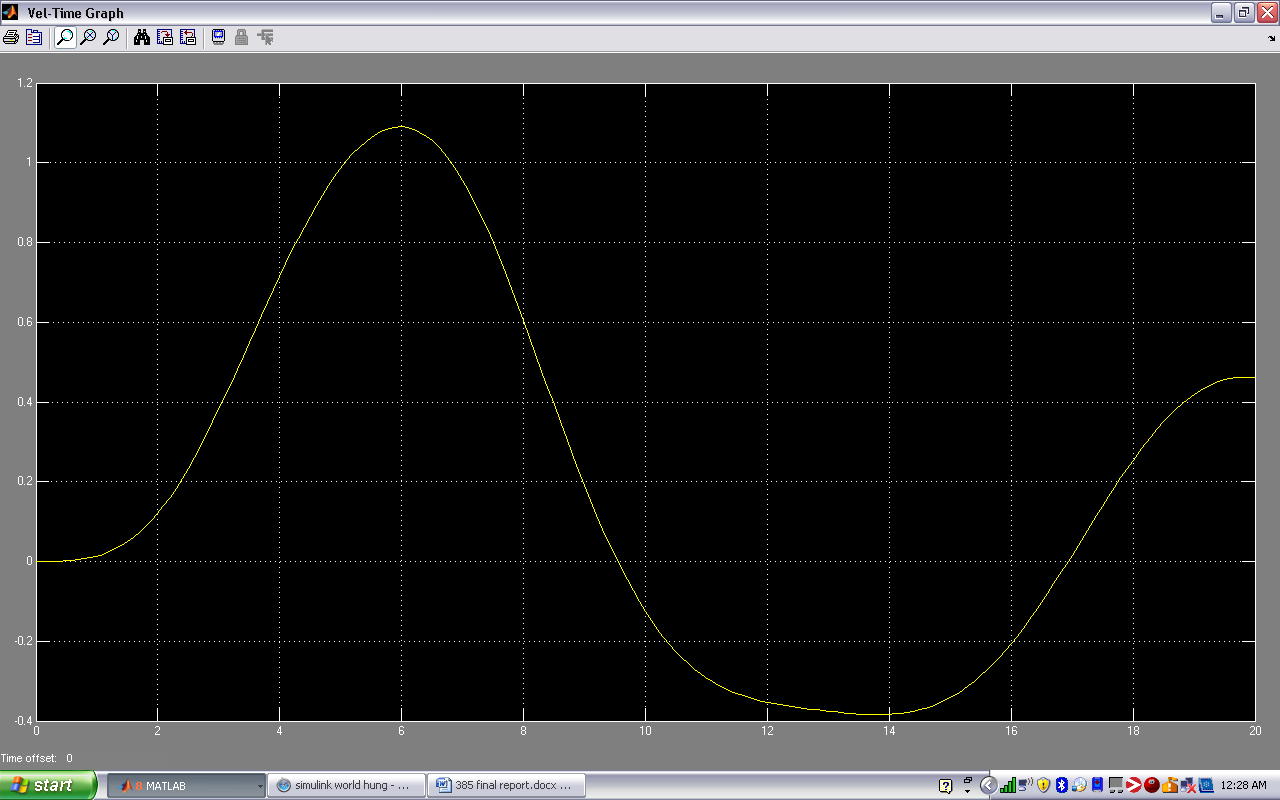
Figure 3 - 16: Controlled Output velocity 1 floor up

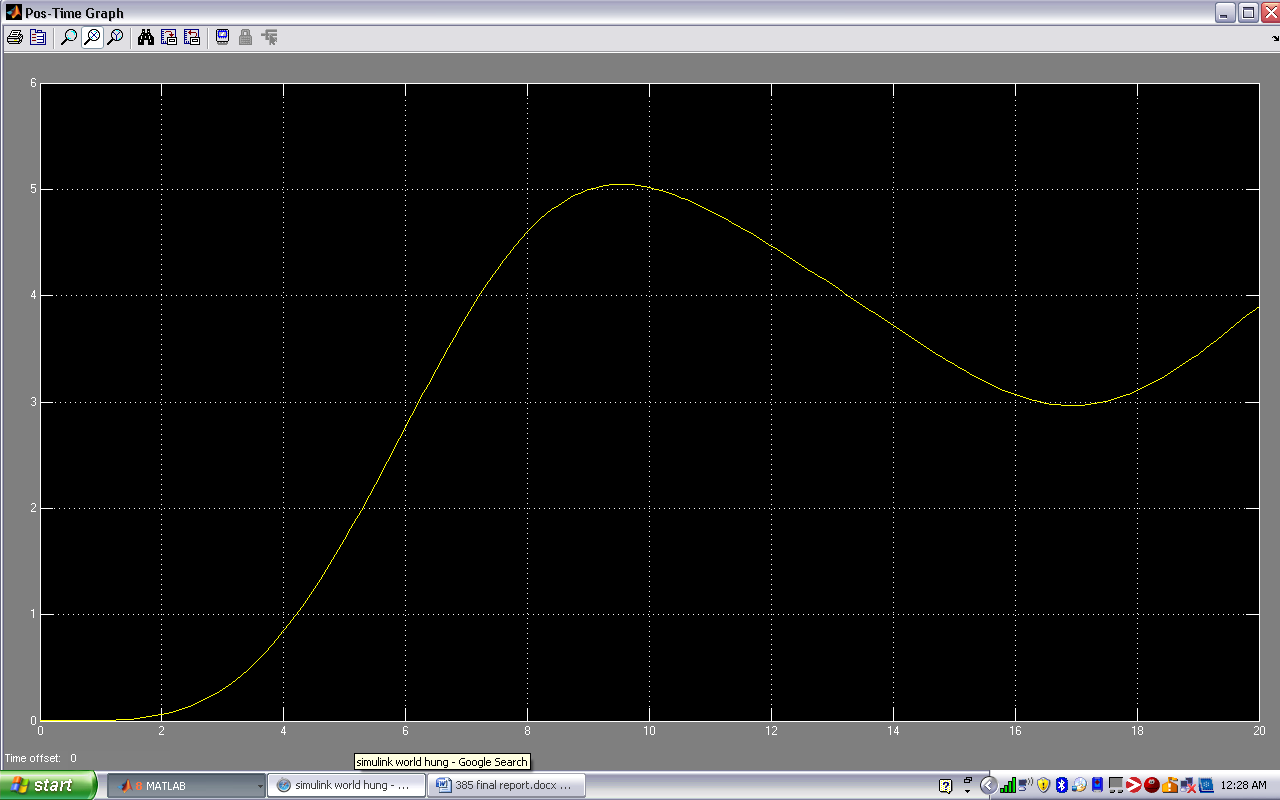
Figure 3 - 17: Controlled Output position 1 floor up 

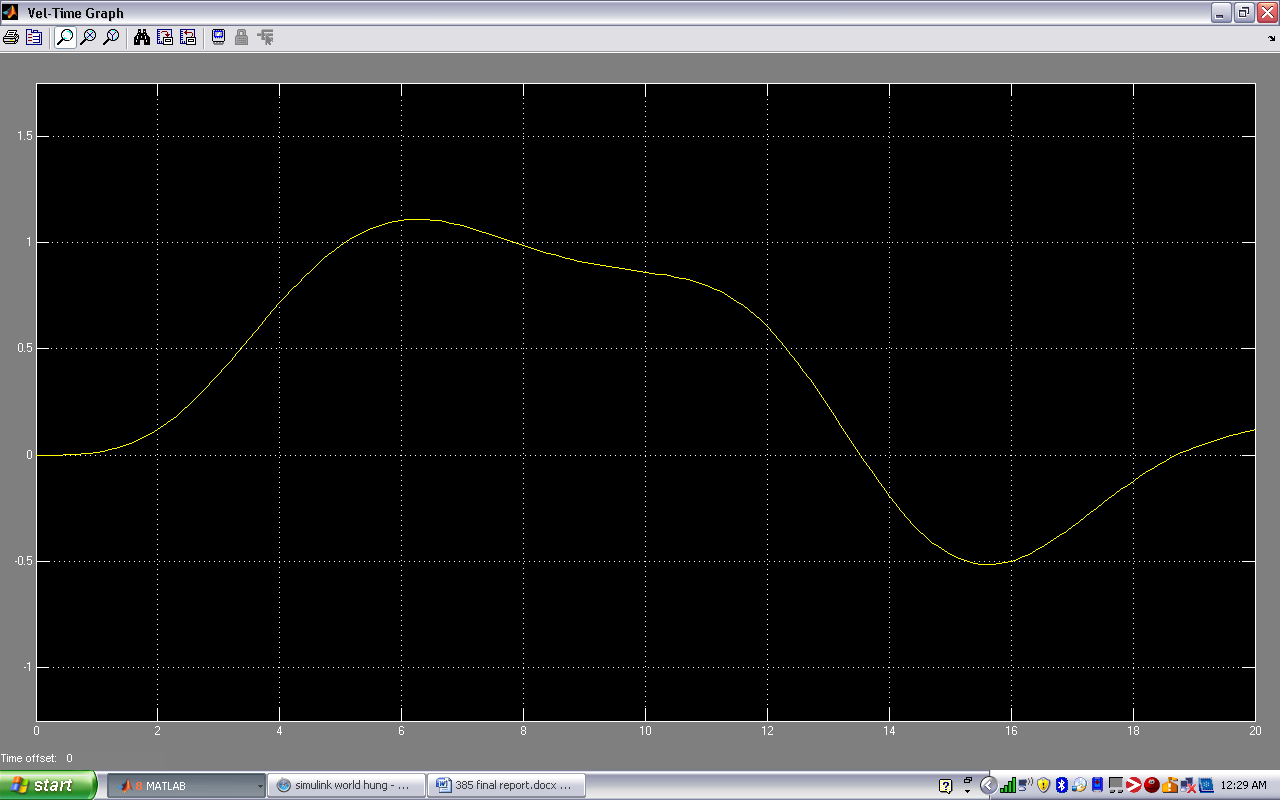
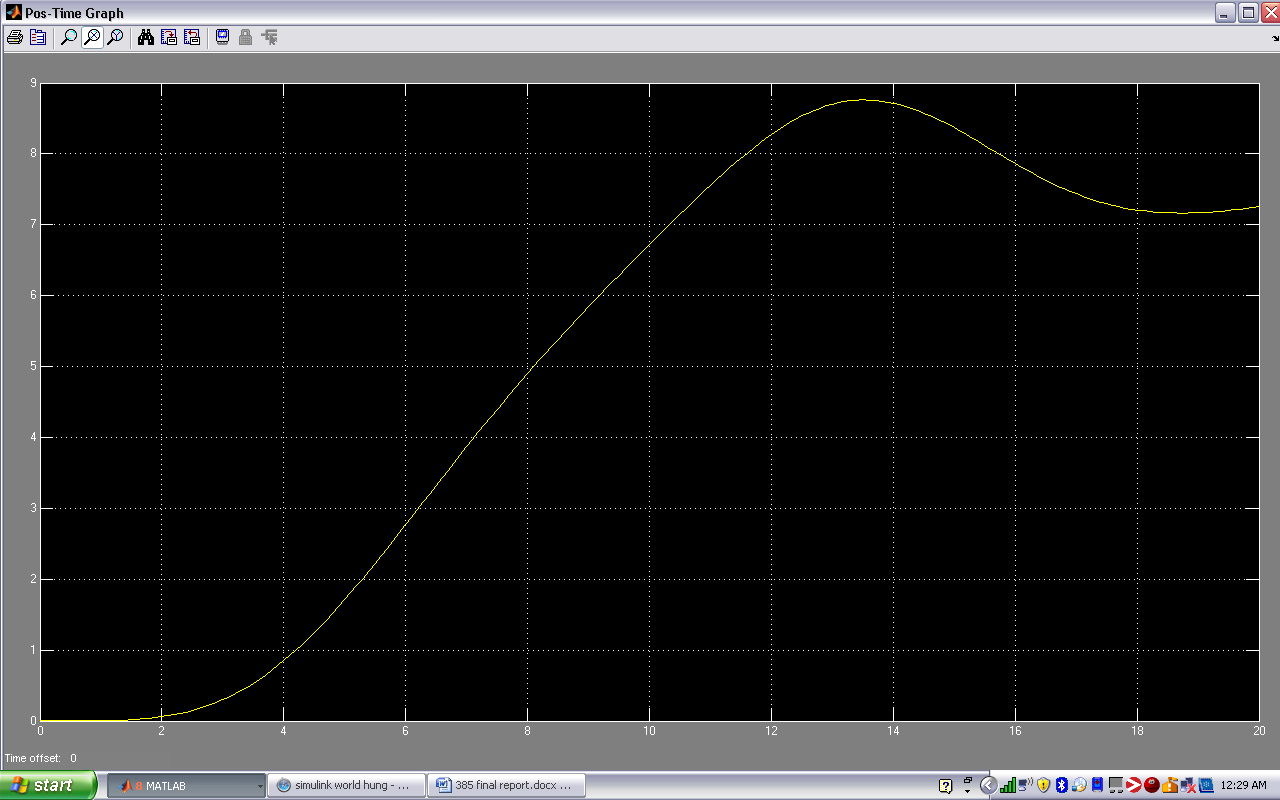
Figure 3 - 18: Controlled Output velocity 2 floors up

Figure 3 - 19: Controlled Output position 2 floors up 

150,30,60

Figure 3 - 20: Controlled Output velocity 1 floor up 

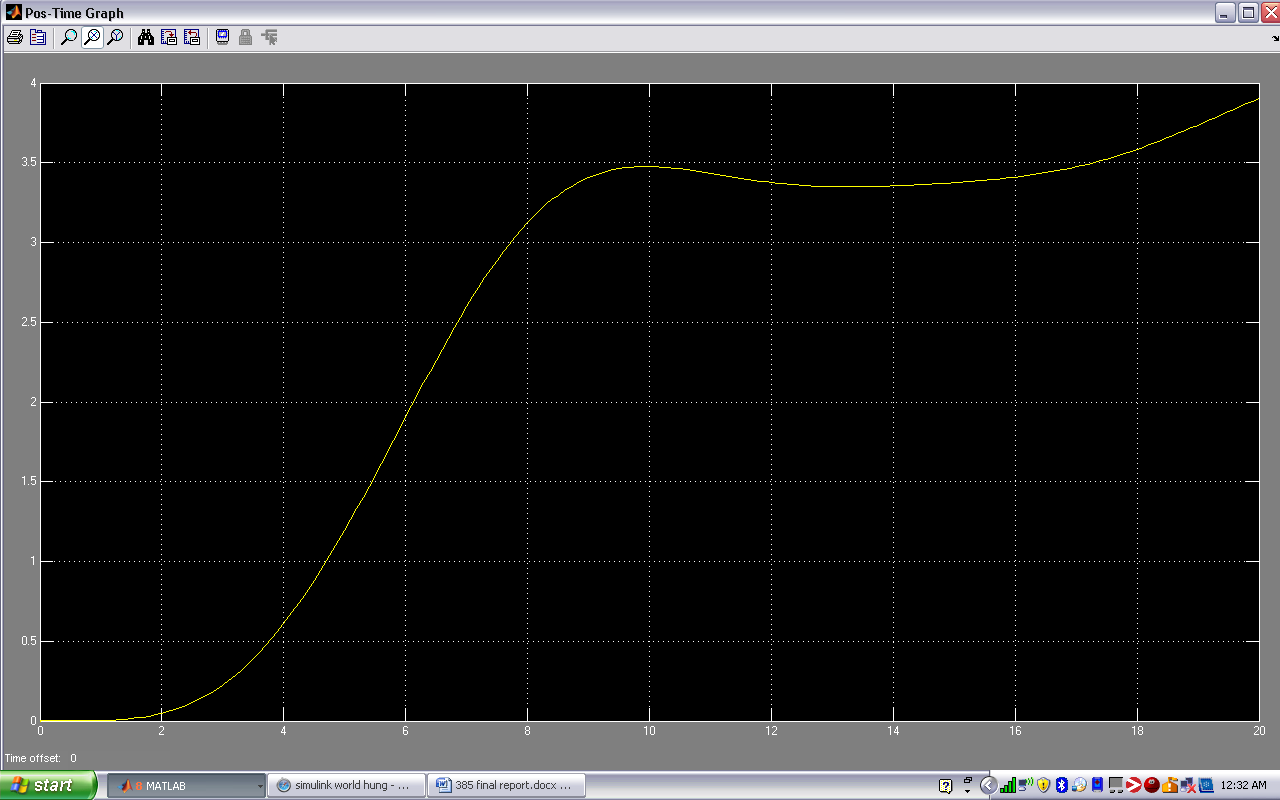
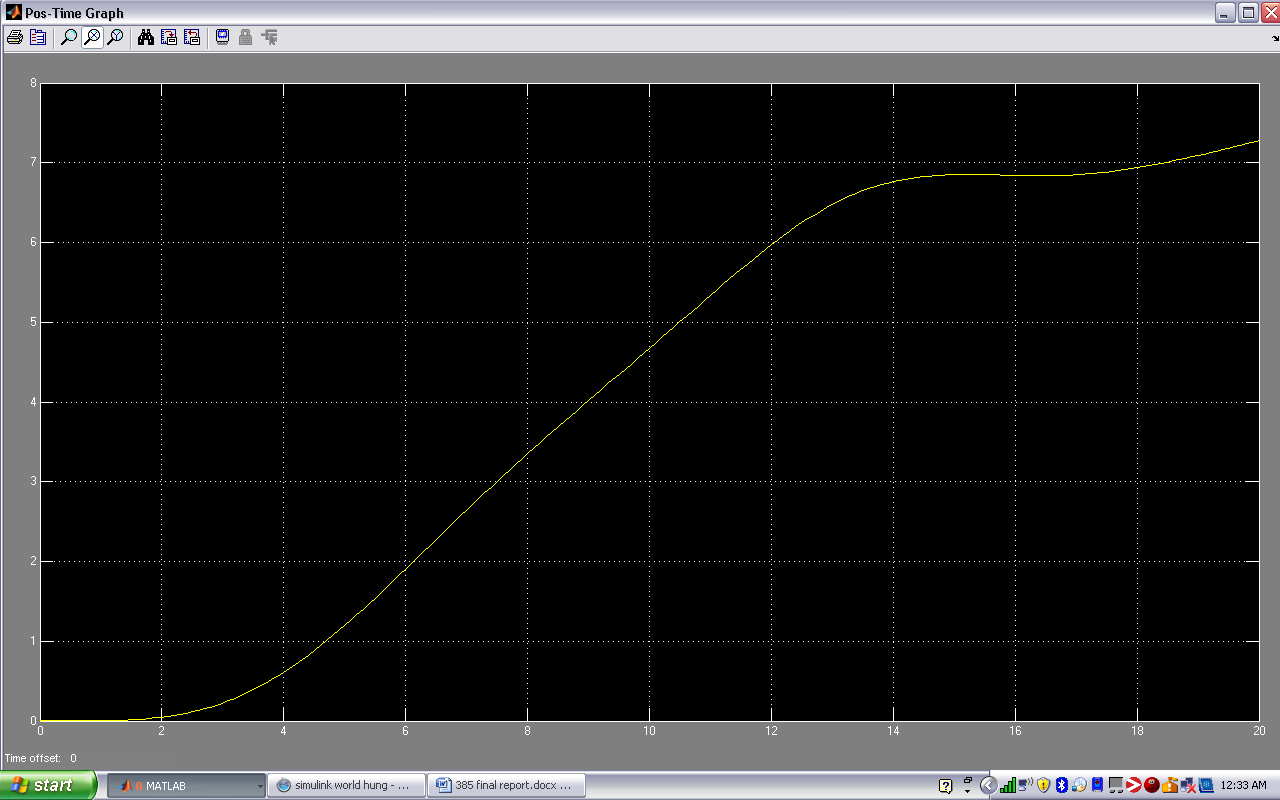
Figure 3 - 21: Controlled Output position 1 floor up 

Figure 3 - 22: Controlled Output velocity 2 floors up

Figure 3 - 23: Controlled Output position 2 floors up

100,45,0

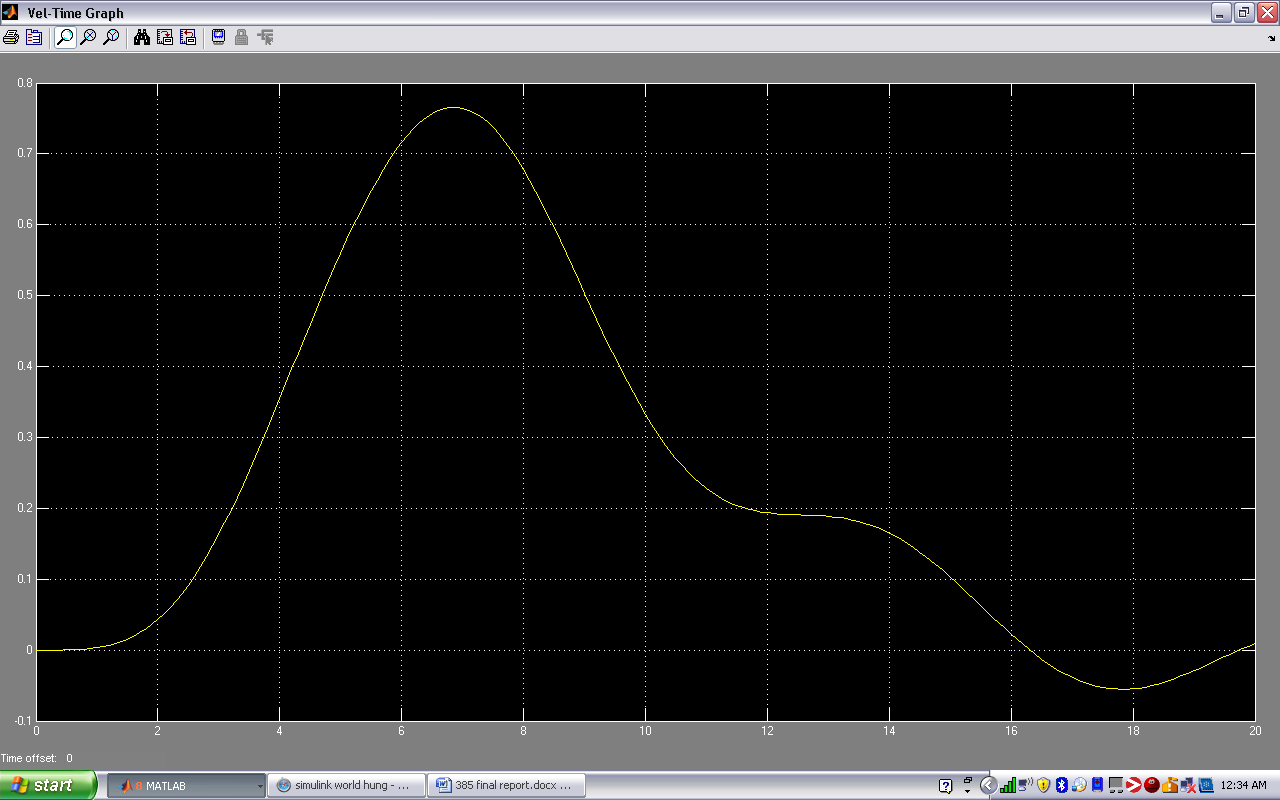
Figure 3 - 24: Controlled Output velocity 1 floor up

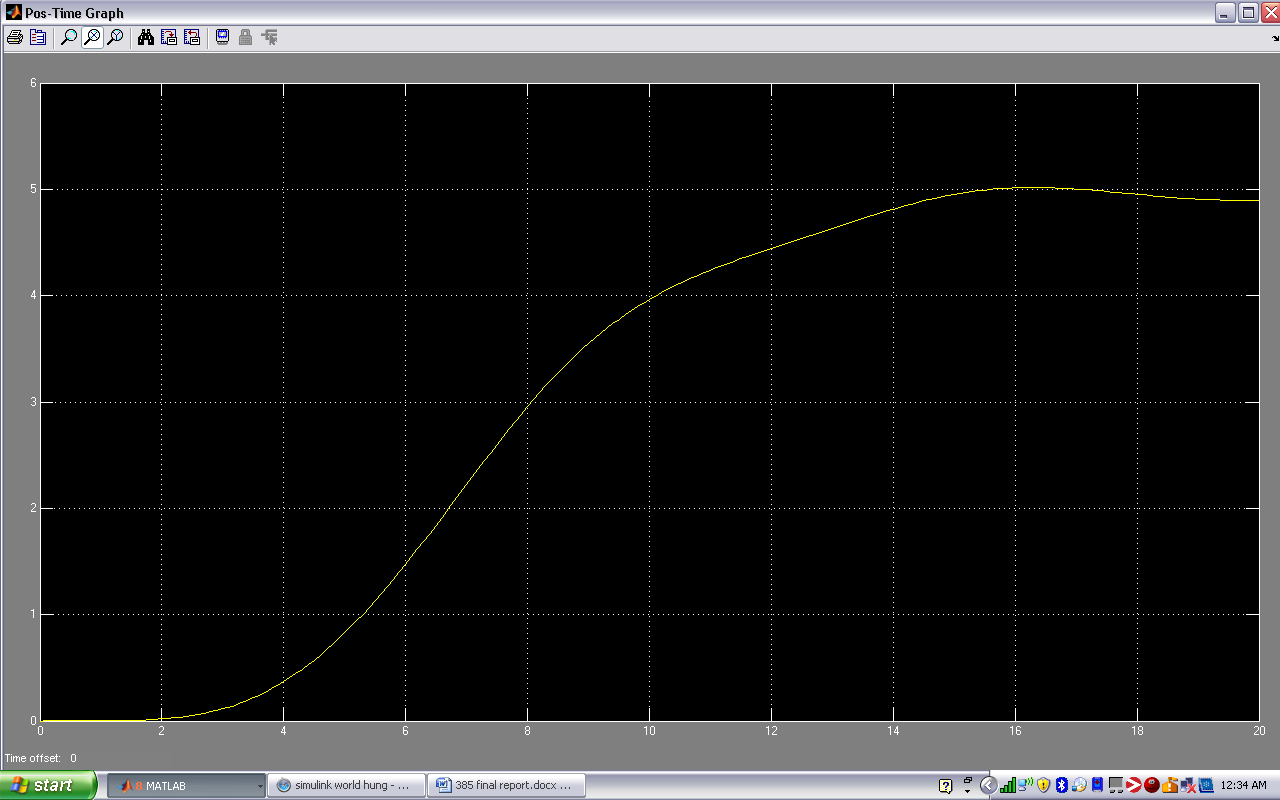
Figure 3 - 25 Controlled Output position 1 floor up

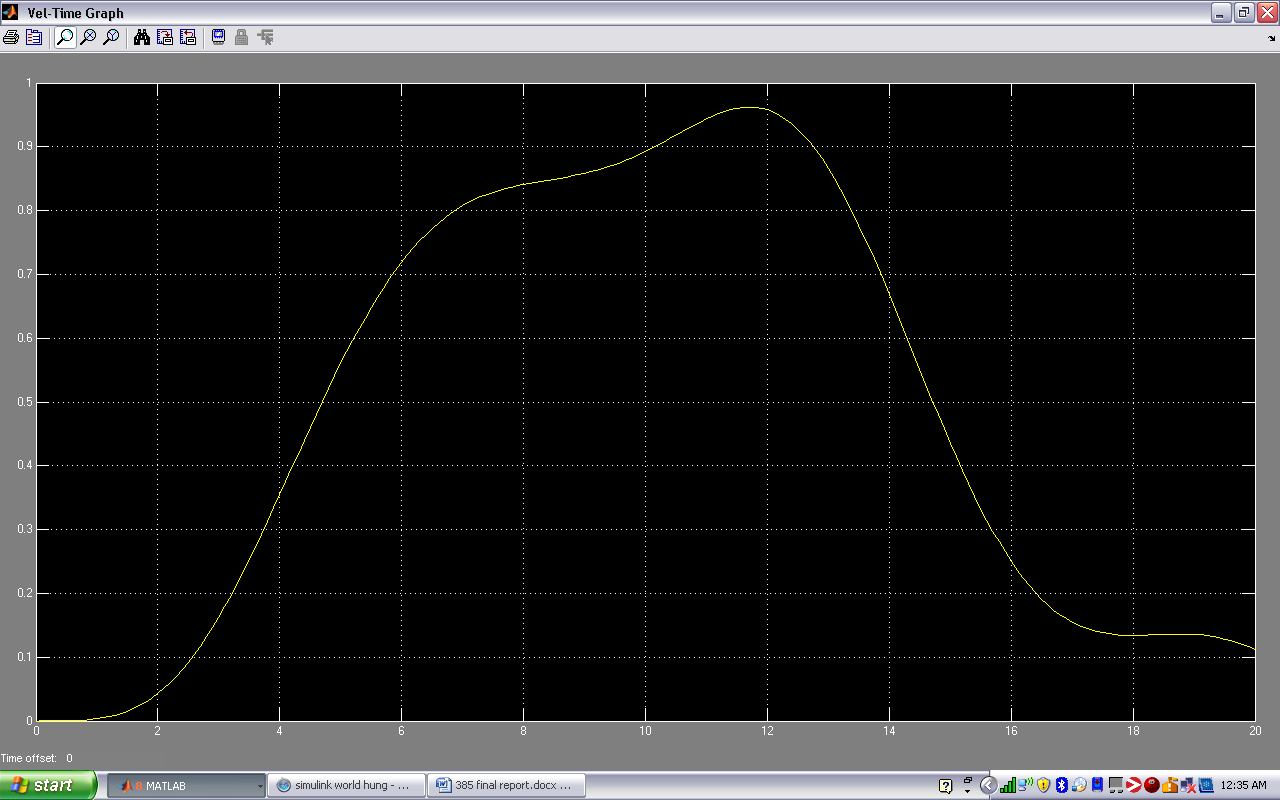
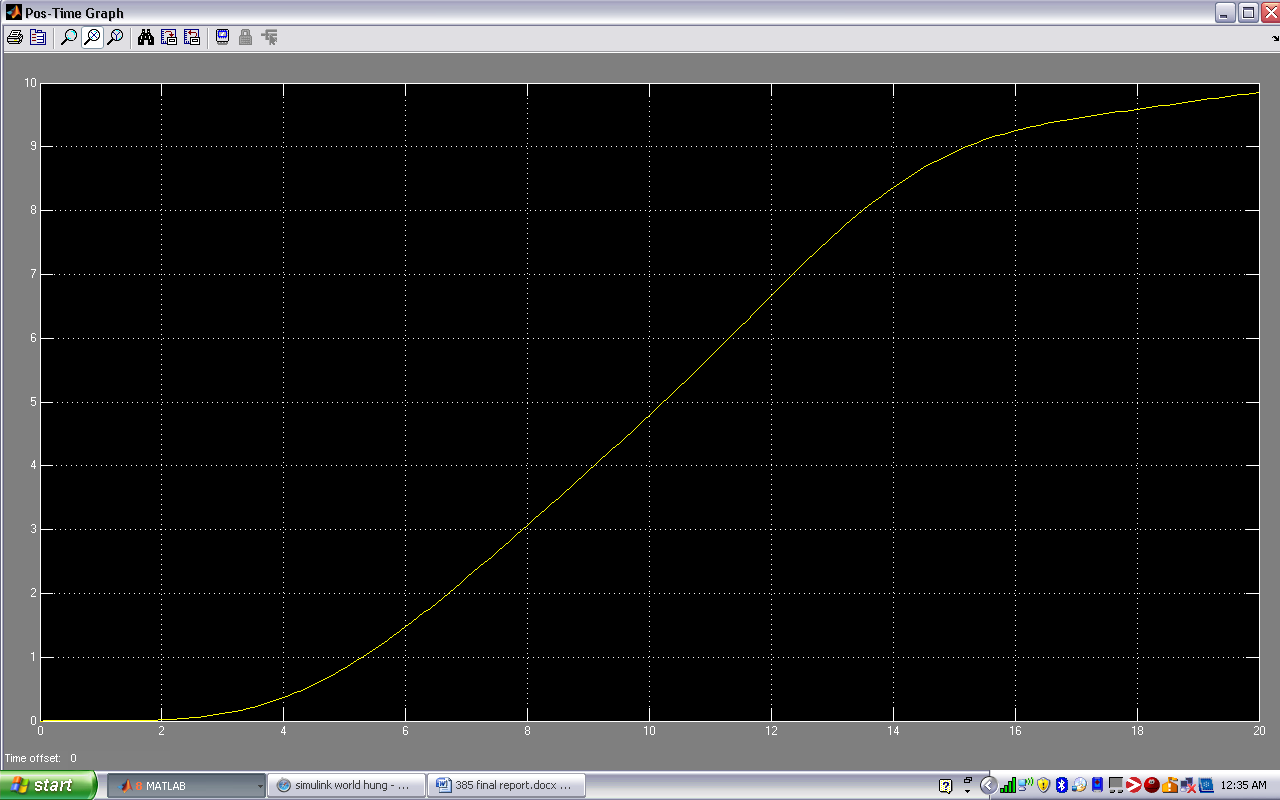
Figure 3 - 26 Controlled Output velocity 2 floors up

Figure 3 - 27 Controlled Output position 2 floors up

57, 41,-28

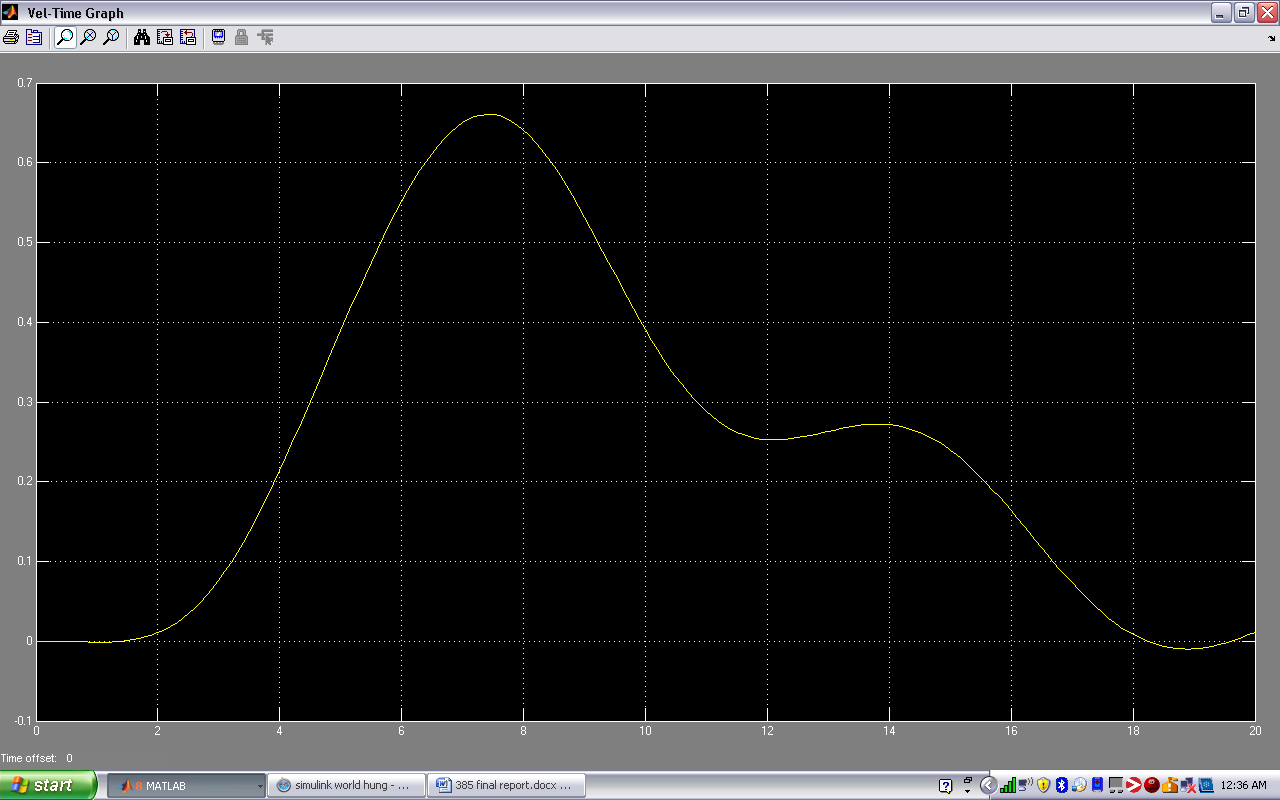
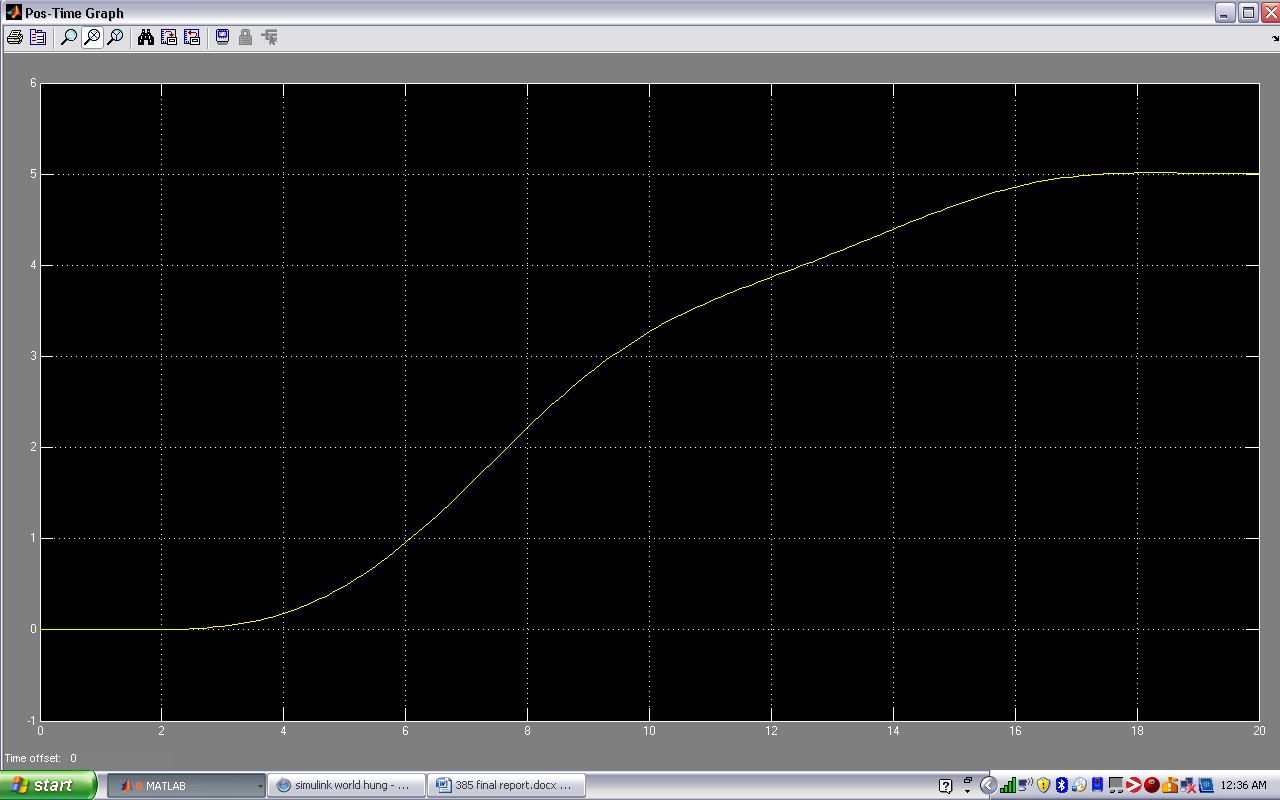
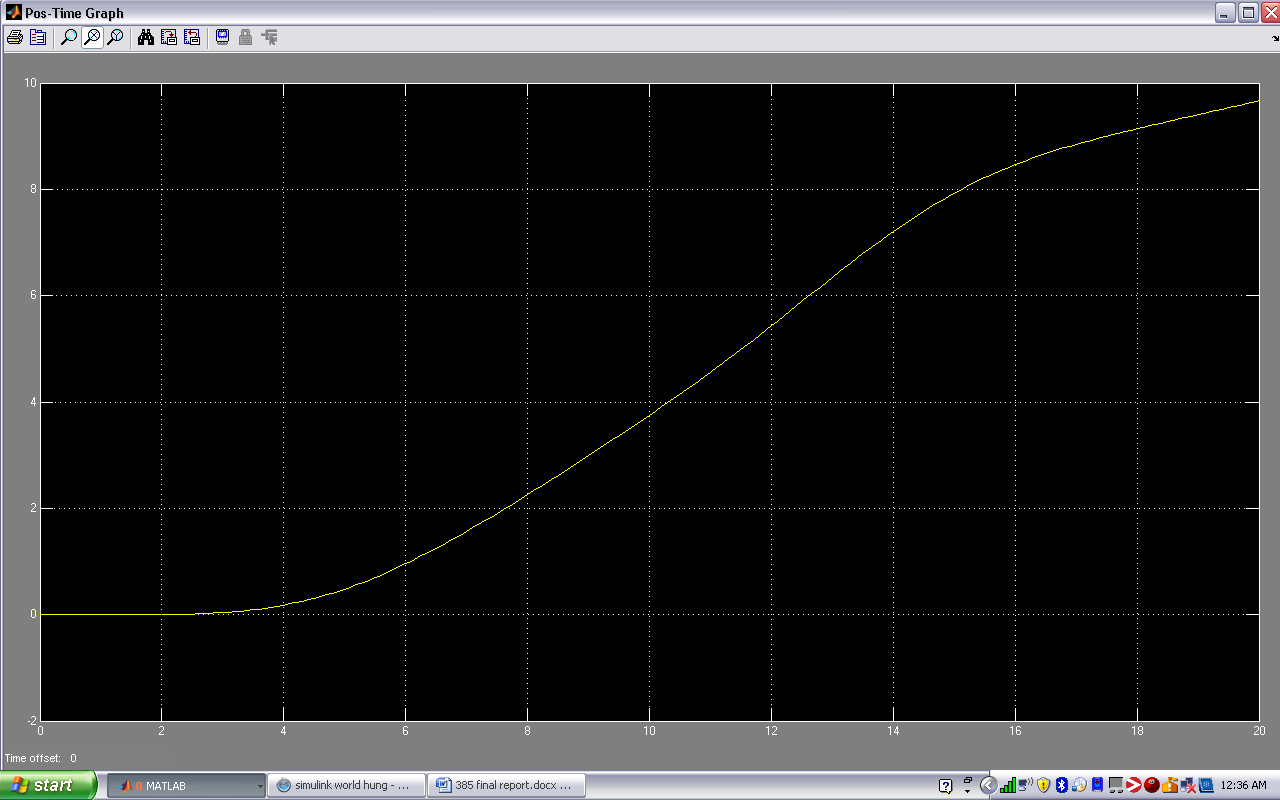
Figure 3 - 28 Controlled Output velocity 1 floor upFigure 3 - 29 Controlled Output position 1 floor up

Figure 3 - 30 Controlled Output velocity 2 floors up

Figure 3 - 31 Controlled Output position 2 floors up

82,44.5,0

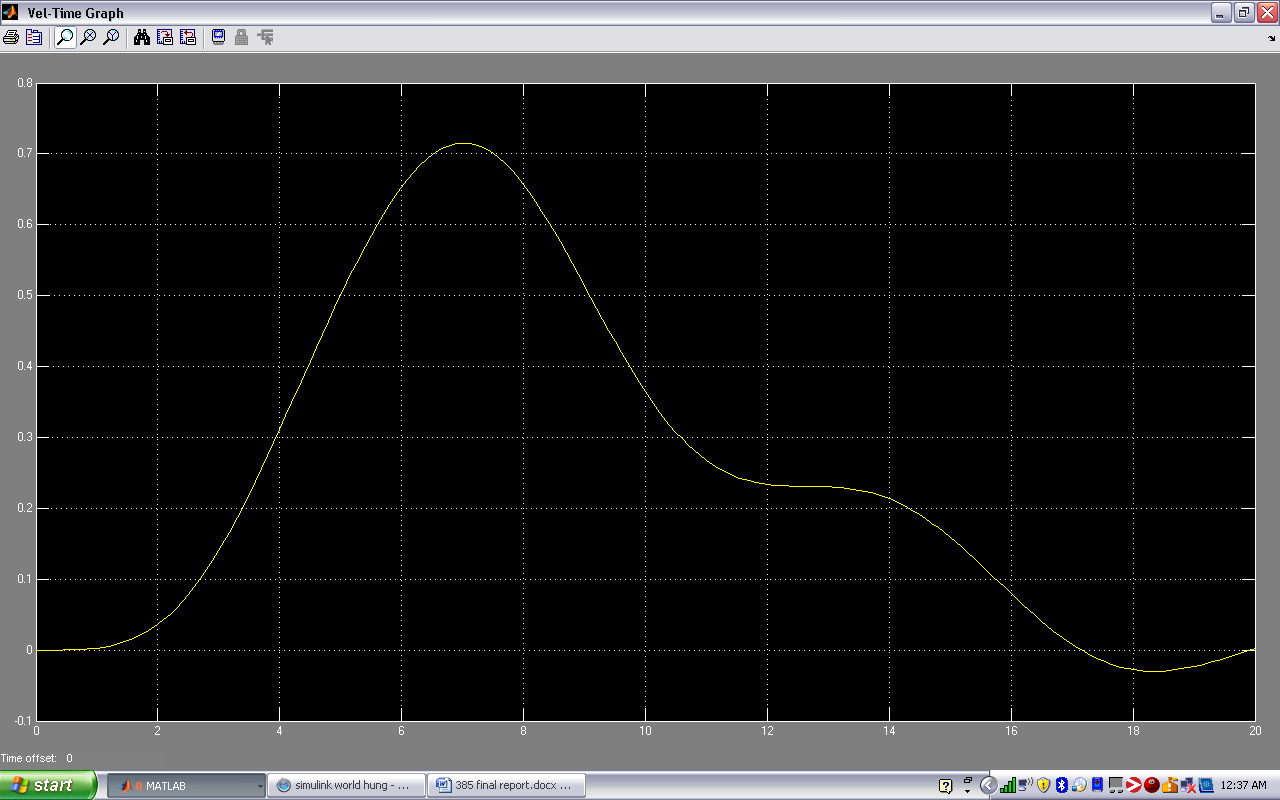
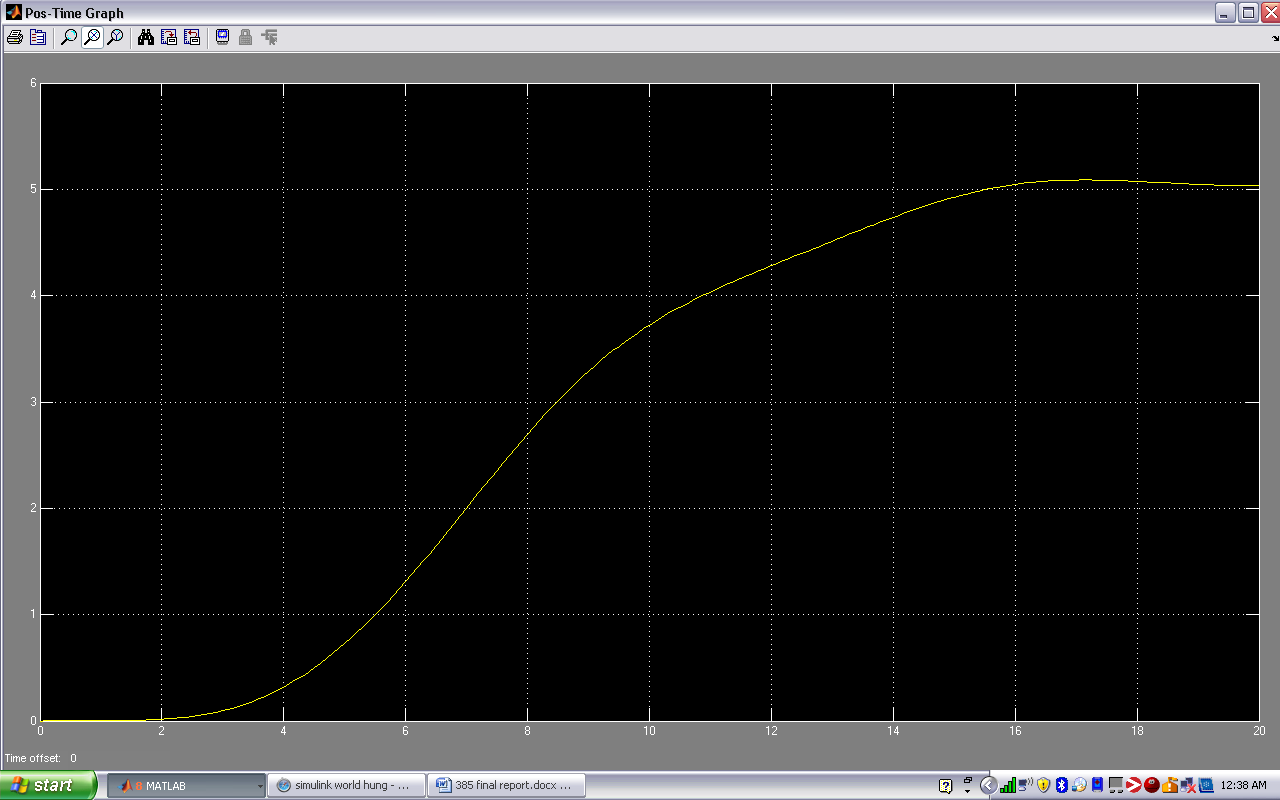
Figure 3 - 32 Controlled Output velocity 1 floor upFigure 3 - 33 Controlled Output position 1 floor up

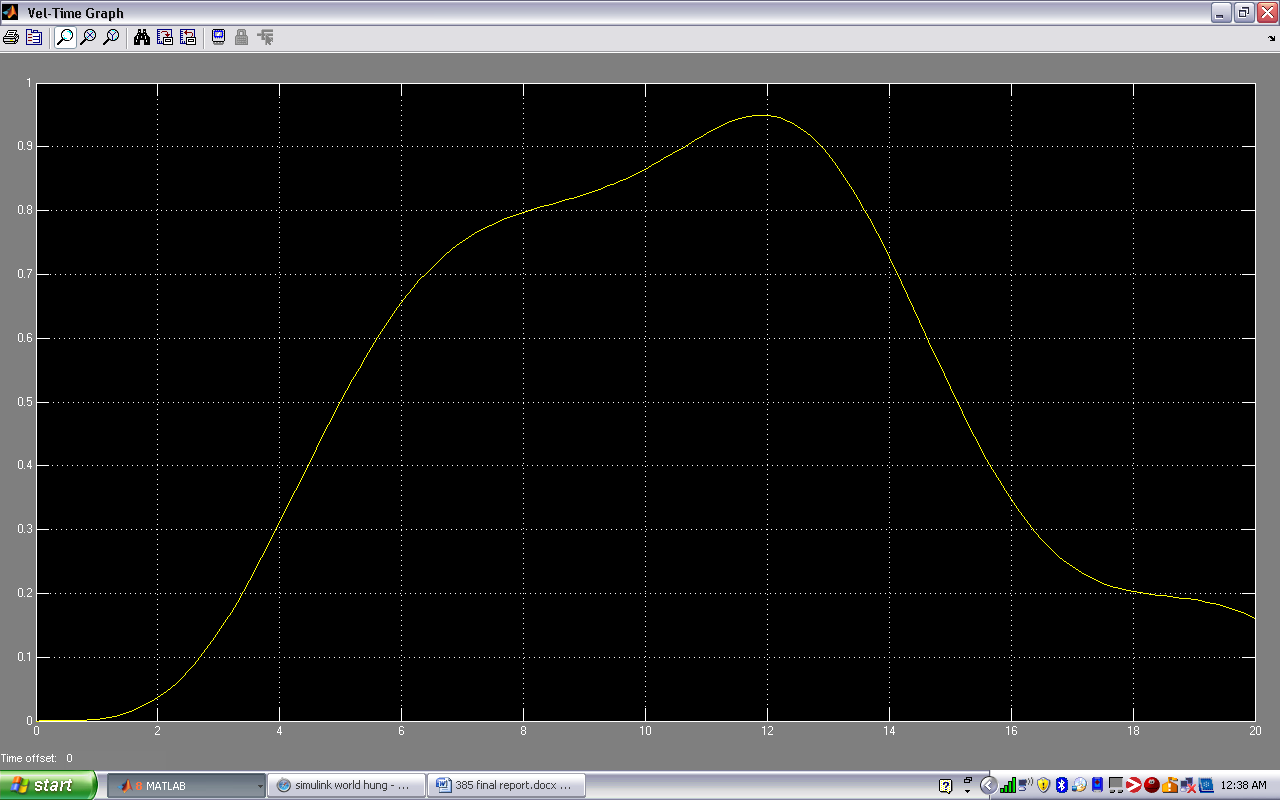
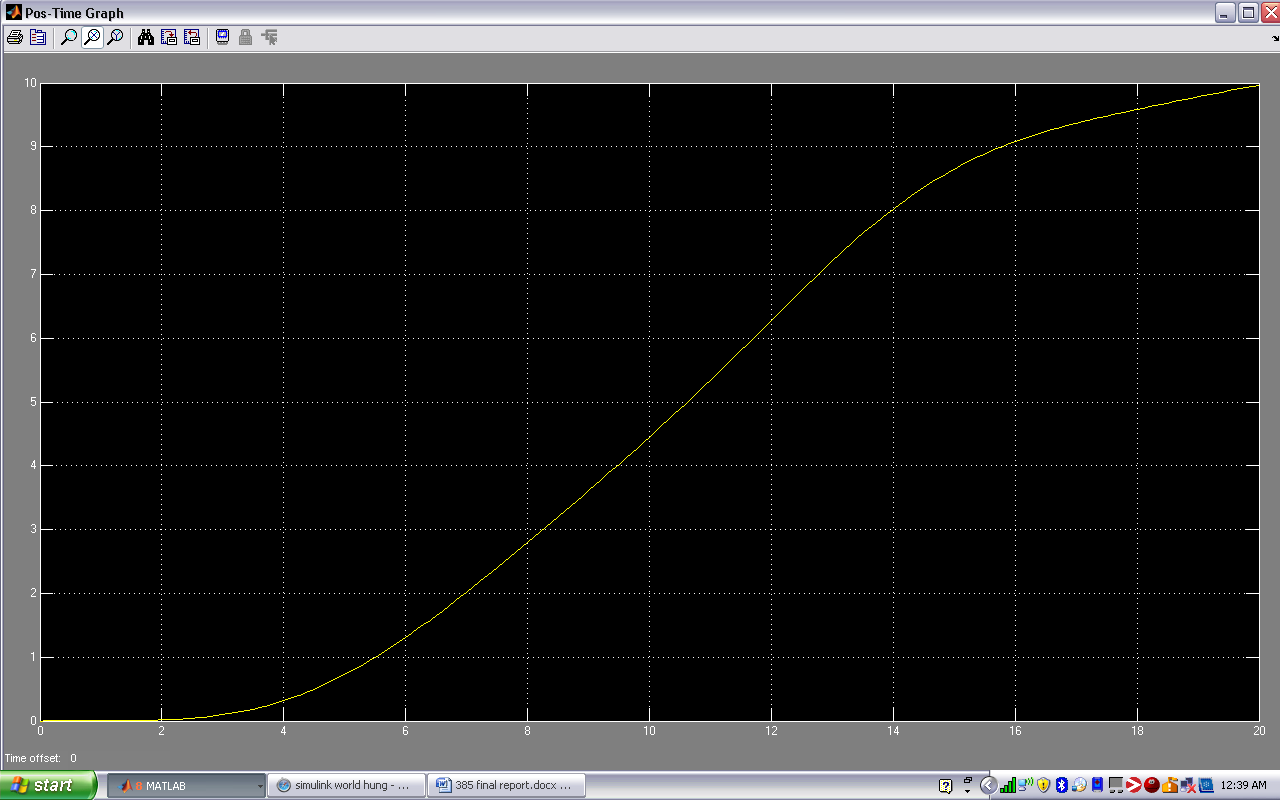
Figure 3 - 34 Controlled Output velocity 2 floors up

Figure 3 - 35 Controlled Output position 2 floors up

**Best** = 88.5,44.8,1

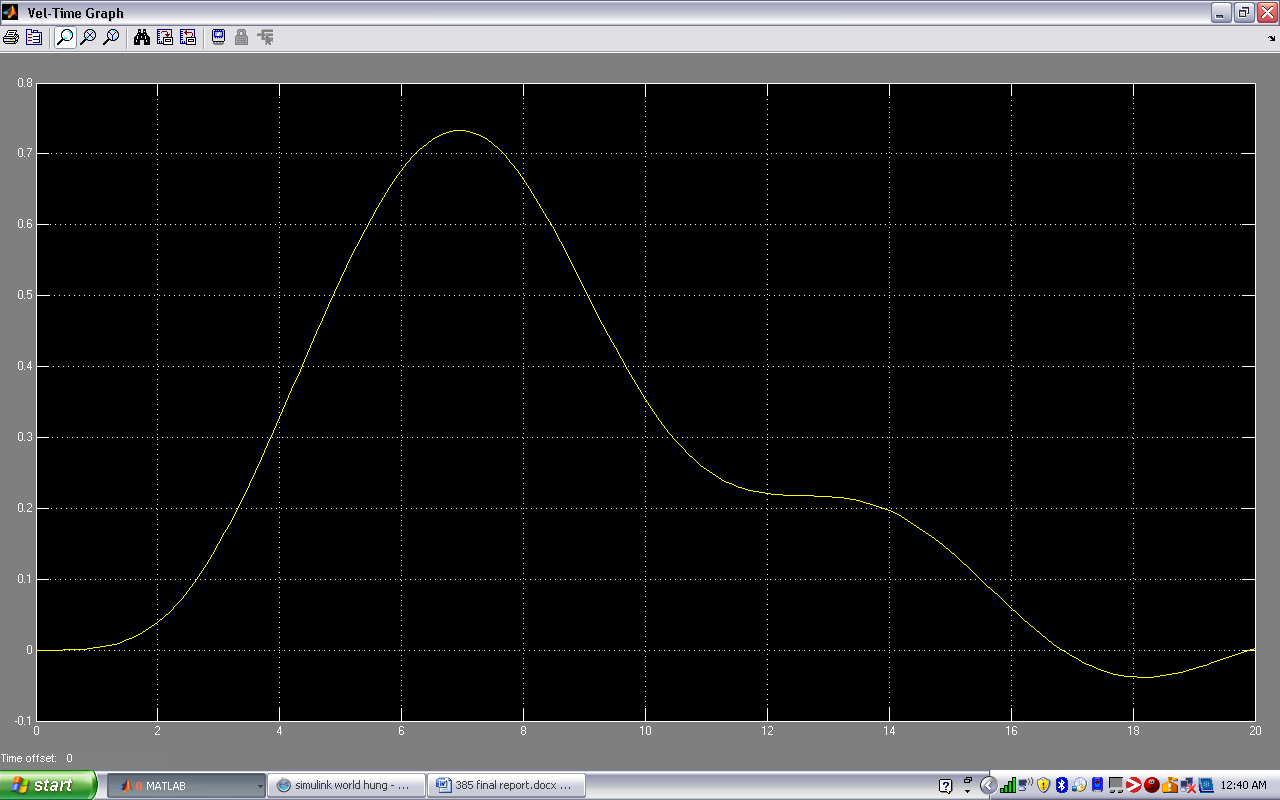
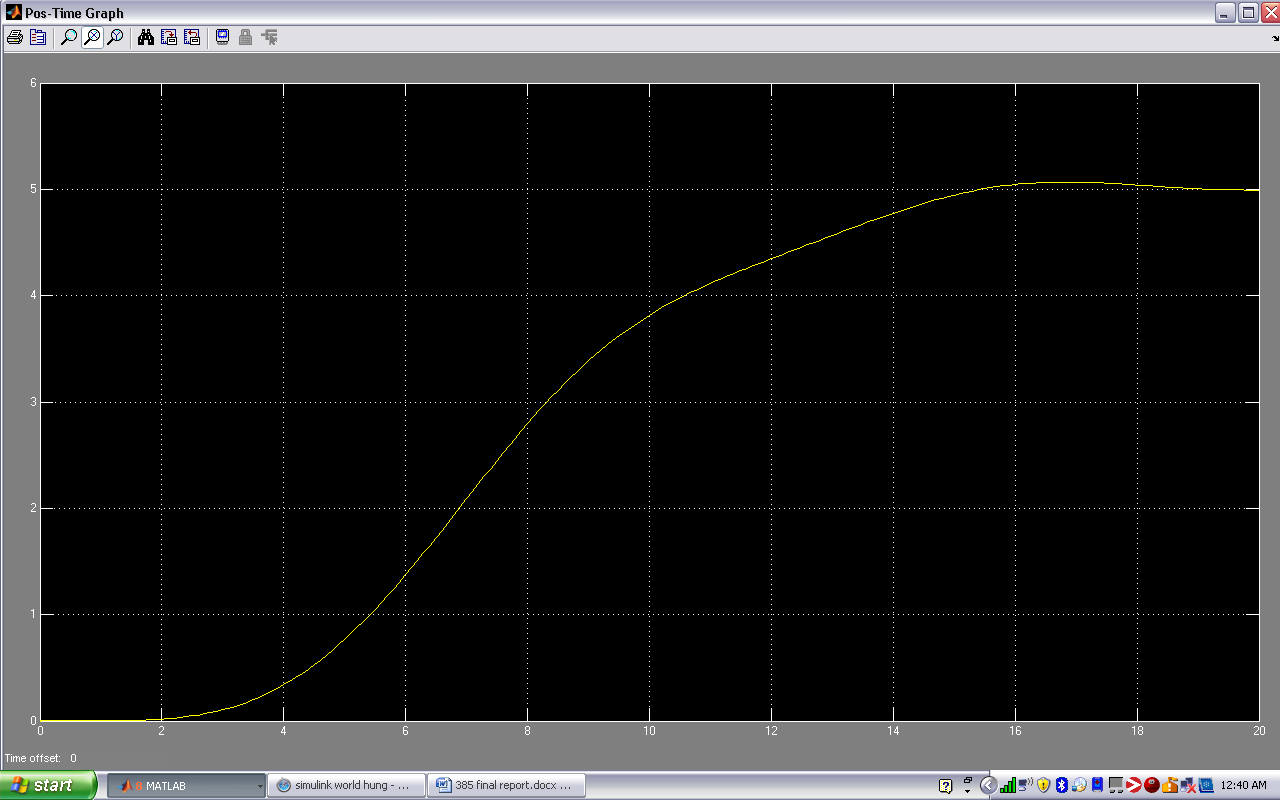
Figure 3 - 36 Controlled Output velocity 1 floor upFigure 3 - 37 Controlled Output position 1 floor up

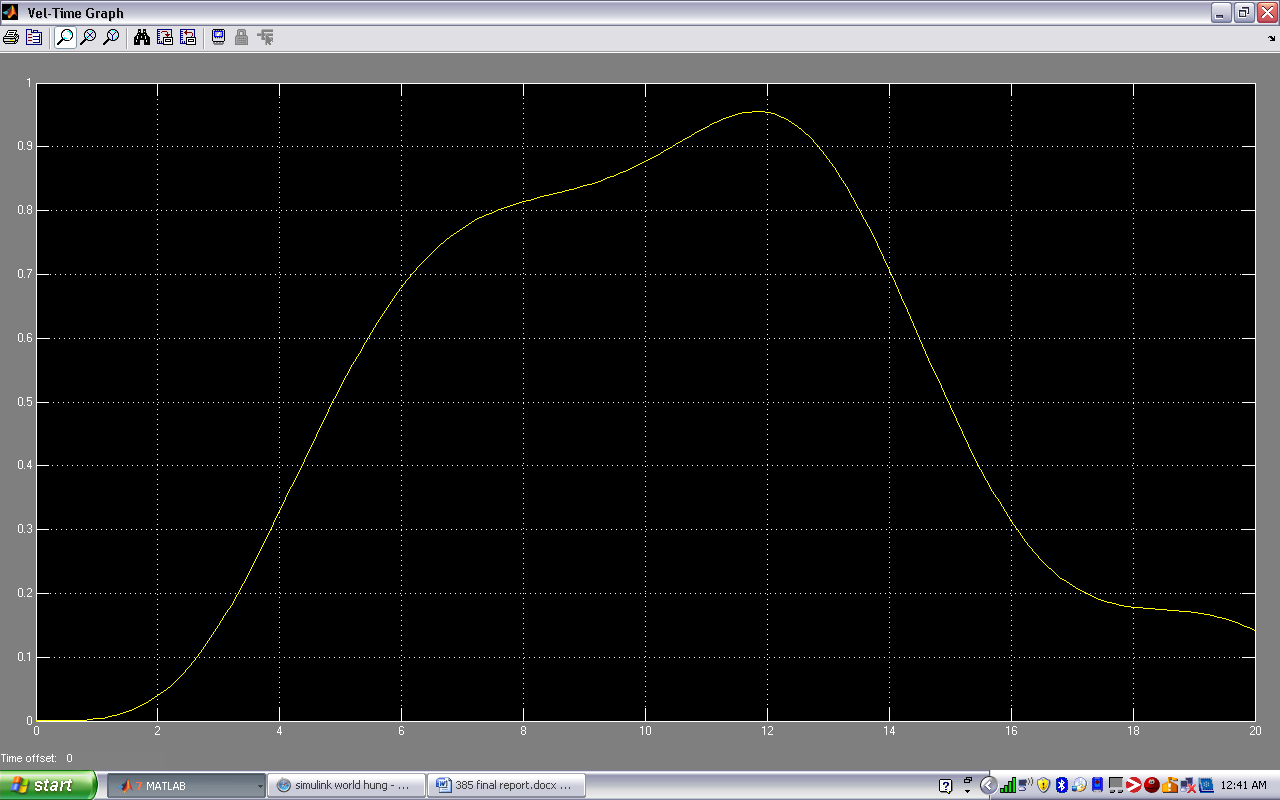
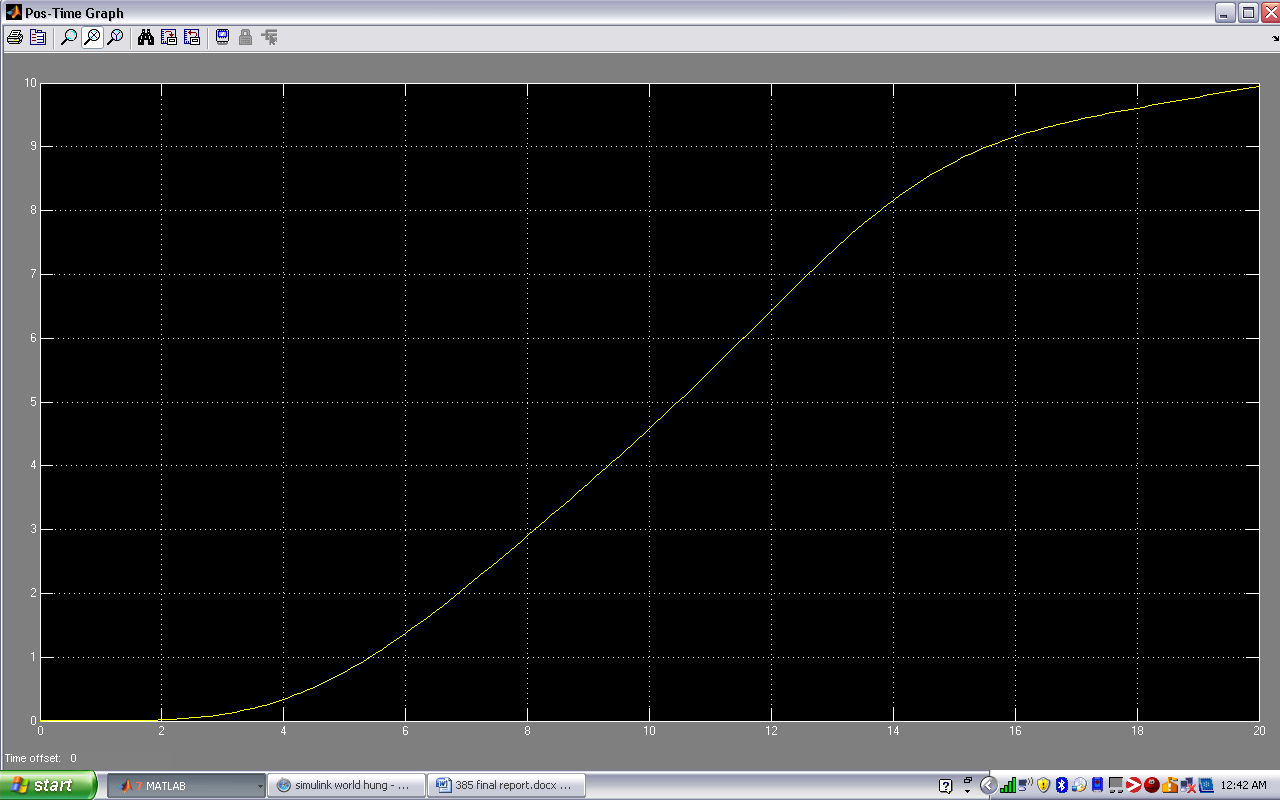
Figure 3 - 38 Controlled Output velocity 2 floors up

Figure 3 - 39 Controlled Output position 2 floors up

### Output velocity and position responses for min mass

Tf=1/[125 600 117 470]

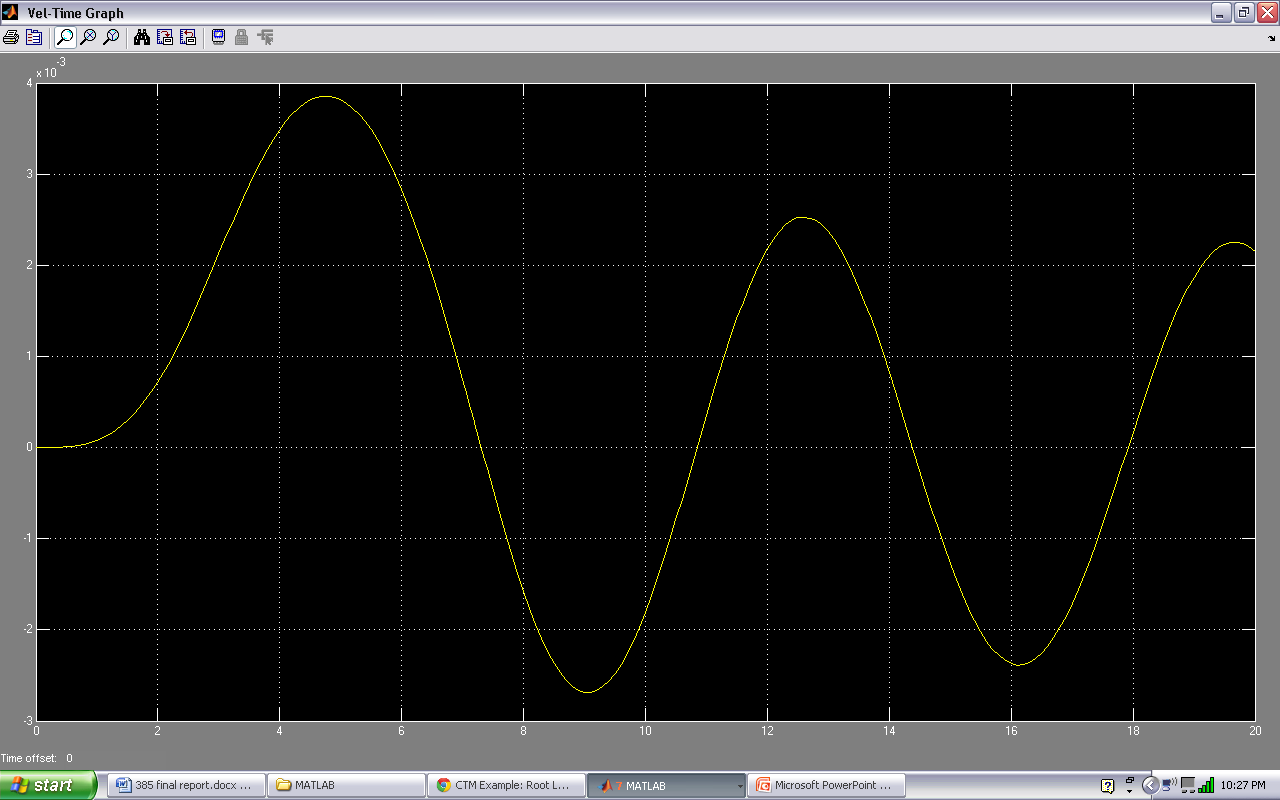
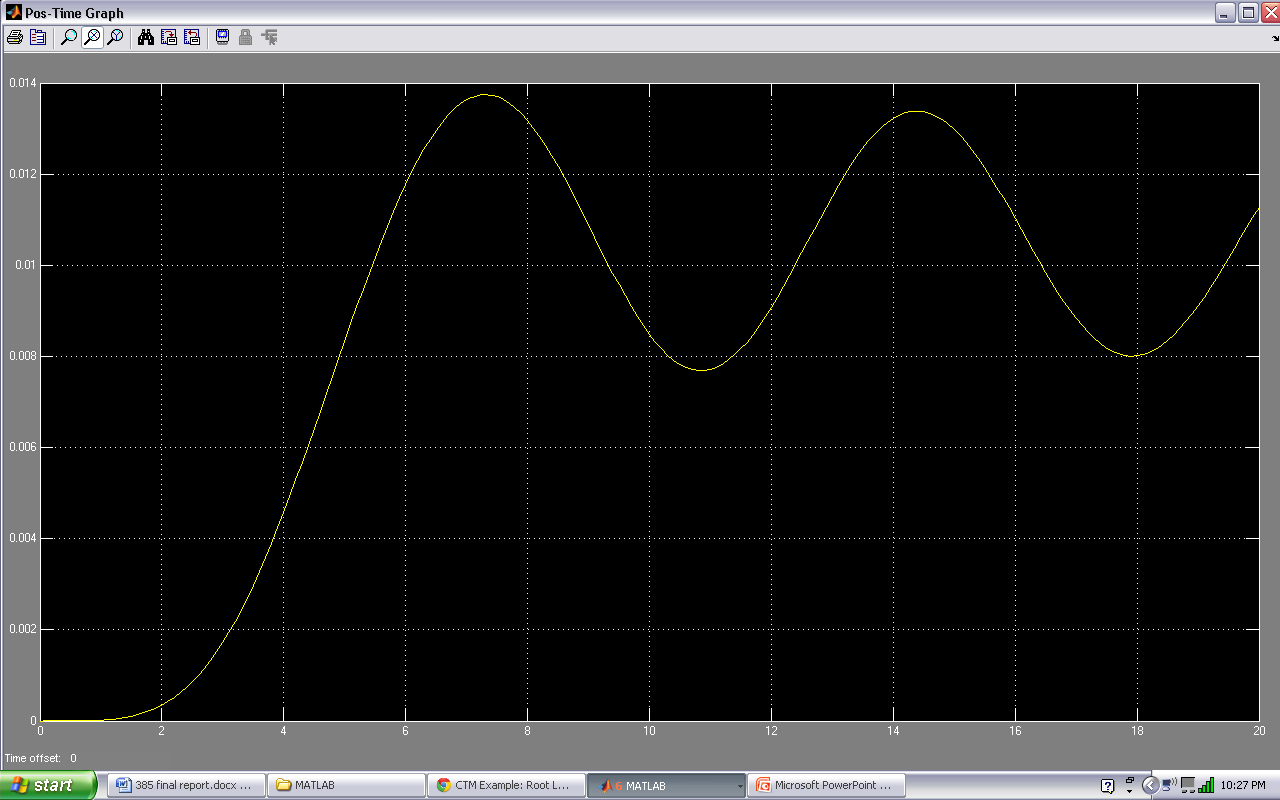
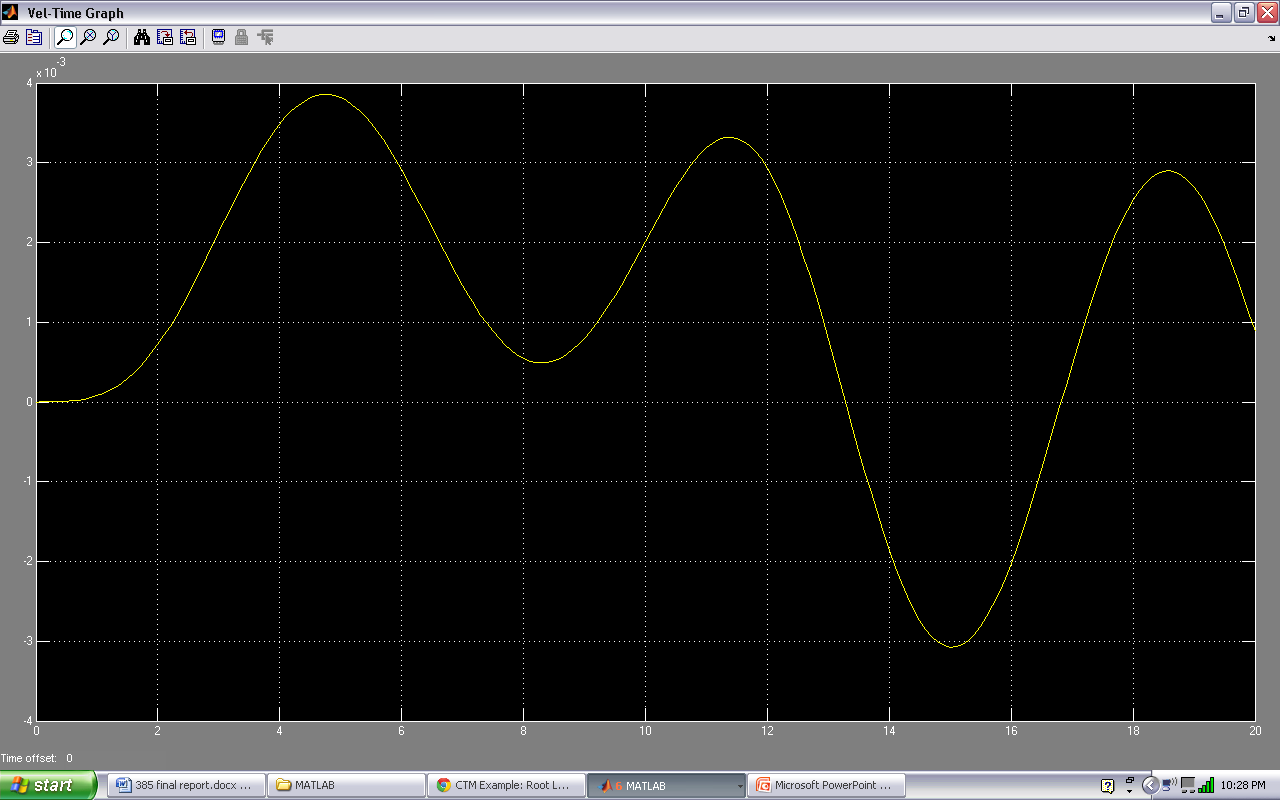
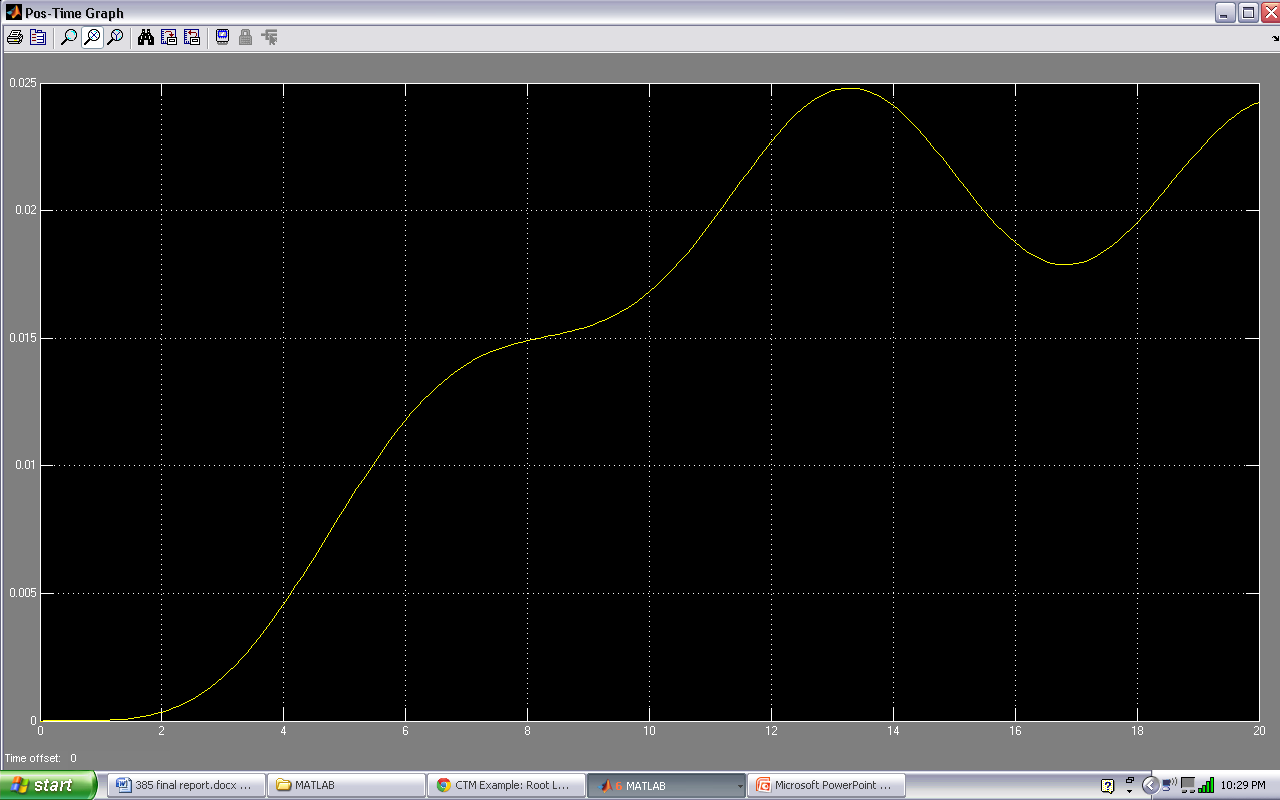
Figure 3 - 40: Uncontrolled output velocity 1 floor upFigure 3 - 41: Uncontrolled output position 1 floor up 

Figure 3 - 42 Uncontrolled output velocity 2 floors upFigure 3 - 43: Uncontrolled output position 2 floors up 

### Data columns

Best values for min mass

P=147.3

I=60.2

D=126.5

We expect a quicker rise and settling time due to the lesser weight

95

50

100

182

57

150

88

65

190

88

50.4

205

88

50.4

185

88

49.7

155

70

45

105

0

45

105

-6

60

105

-6

60

95

30

60

95

30

60

125

30

60

155

90

43

185

60

43

5

60

50

5

40

50

5

40

50

30

40

50

60

40

70

60

40

90

60

90

50

60

90

50

80

90

49.9

60

90

49.9

32

95

49.9

32

105

49.9

32

100

49.9

22

150

49.9

22

170

89.9

22

-80

89.9

22

-80

59.9

22

-50

79.9

120

10

49.9

120

90

49.9

120

120

54

120

120

60

70

120

60

150

150

60

150

150

57

120

170

62

150

170

65

180

167

58

153

145

59.9

113

145

59.9

120

142.6

59.5

110

149.6

57.5

129.5

149.6

59.5

126.5

147.3

59.5

126.5

147.3

60.2

126.5

### Data table (for 1 position up)

Requirements

Rise time < 5 sec

Overshoot < 10%

Steady state error < 2%

Settling time < 15

Table 3

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| P | 95 | 30 | 90 | 10 | 145 | 147.3 |
| I | 50 | 60 | 50 | 49.9 | 59.9 | 60.2 |
| D | 100 | 125 | 80 | 120 | 120 | 126.5 |
| TR | bad | bad | bad | bad | Good | good |
| TS | slight | good | good | bad | Good | good |
| OS | bad | good | slight | bad | Slight | good |
| SSE | slight | good | bad | bad | Slight | slight |

### Tuning samples

95,50,100

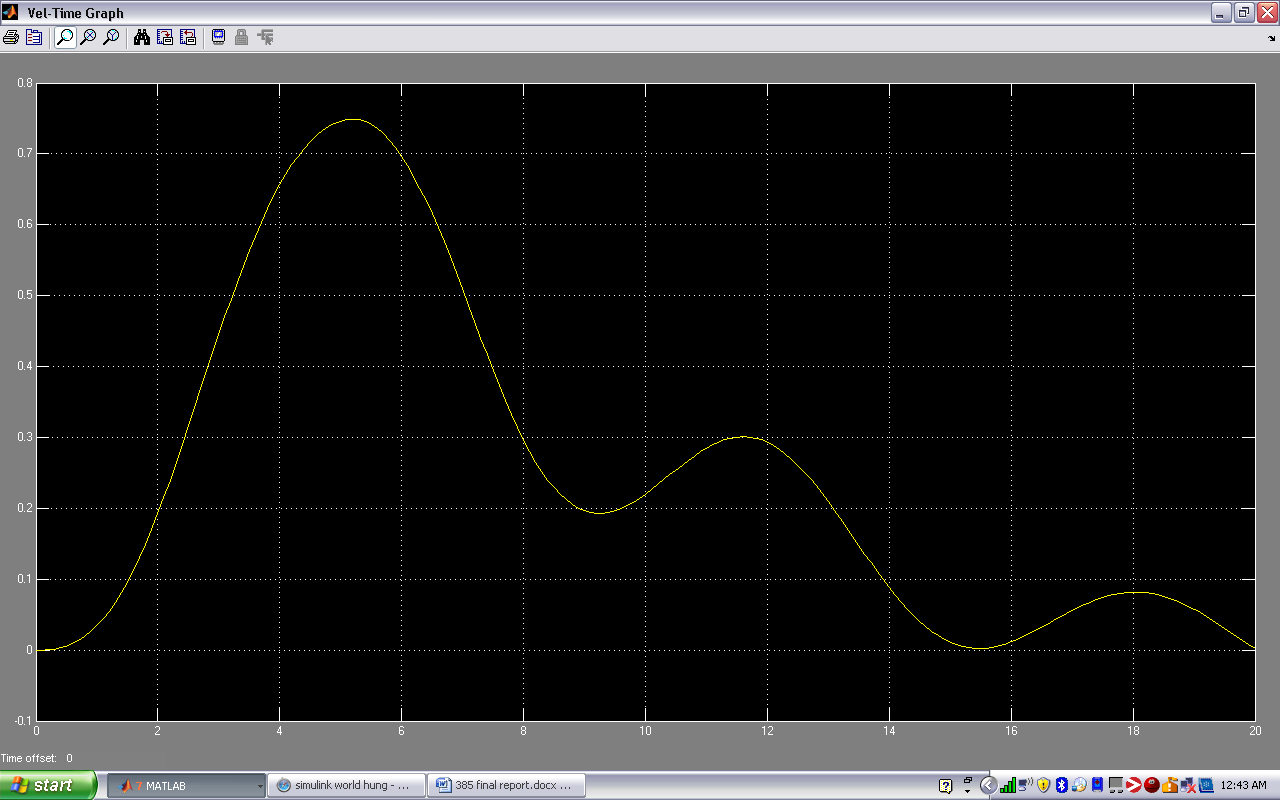
Figure 3 - 44 Controlled Output velocity 1 floor up

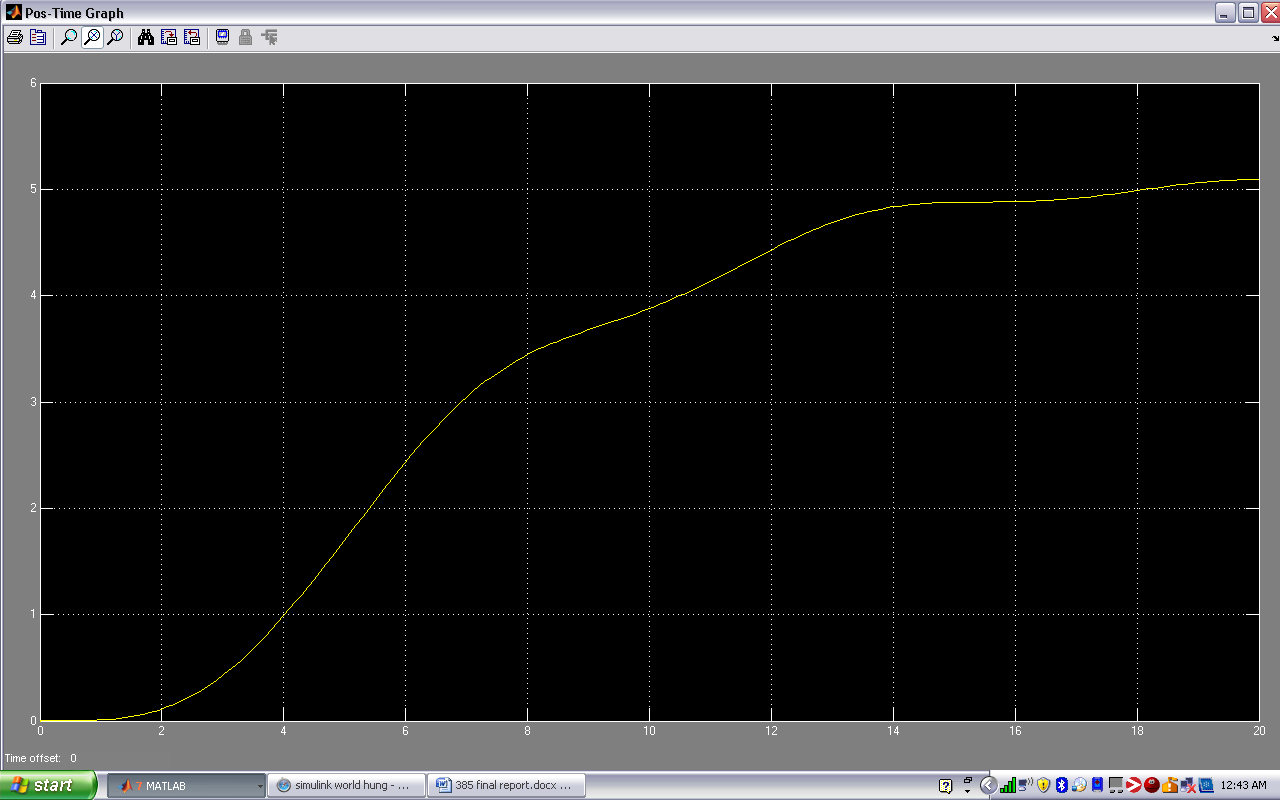
Figure 3 - 45 Controlled Output position 1 floor up

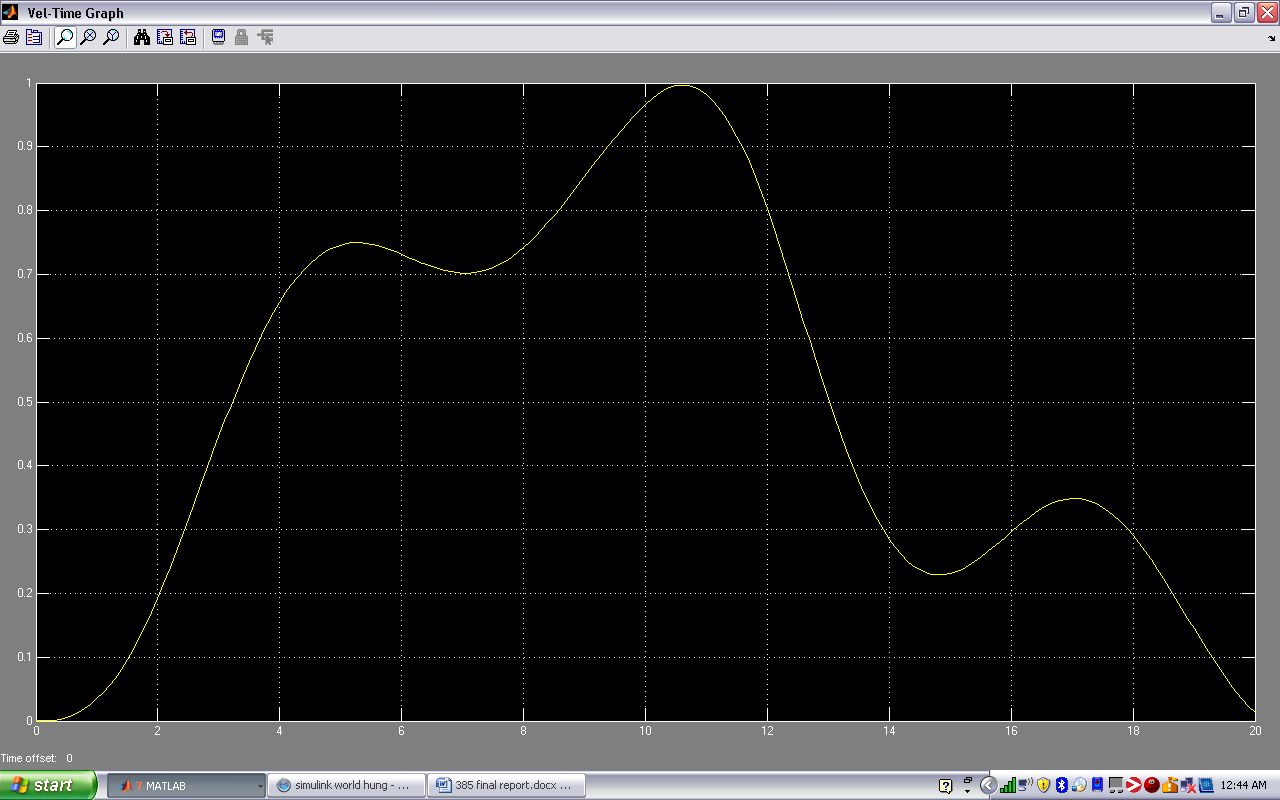
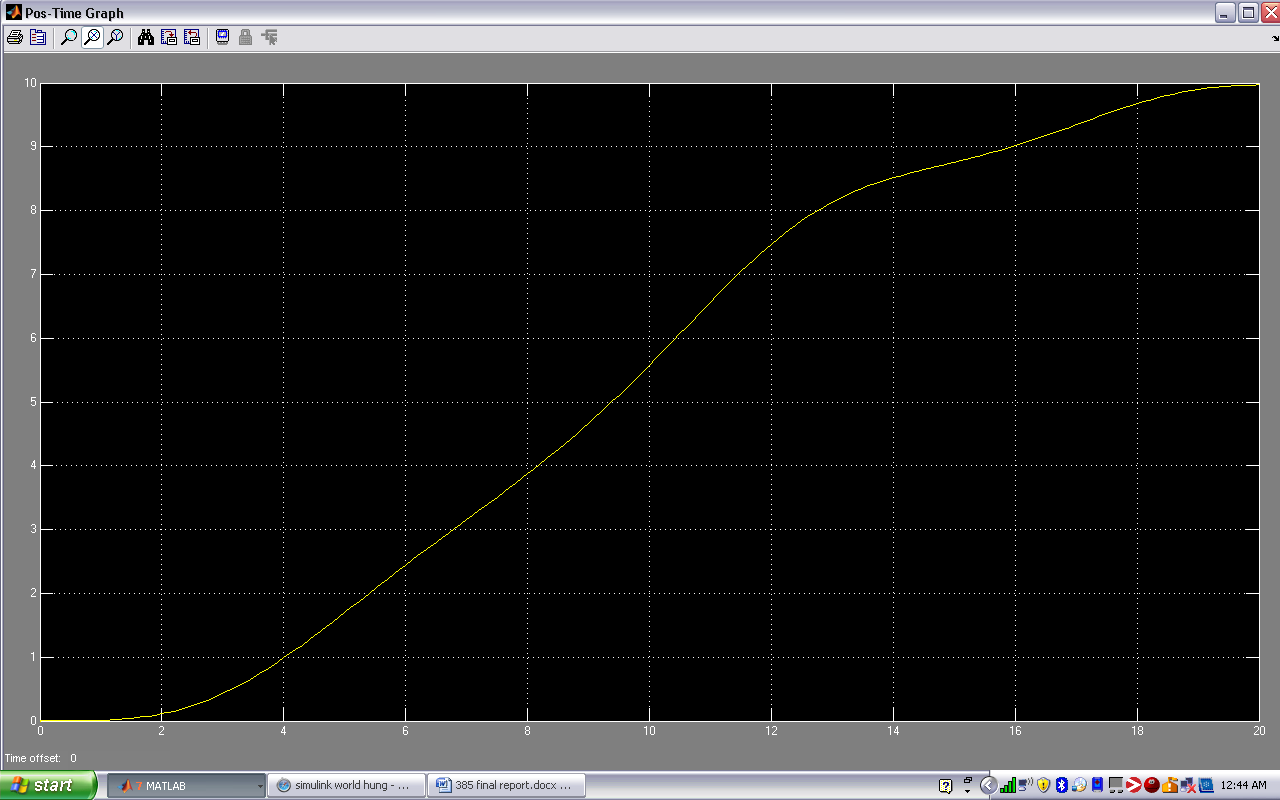
Figure 3 - 46 Controlled Output velocity 2 floors up 

Figure 3 - 47 Controlled Output position 2 floors up

30,60,125

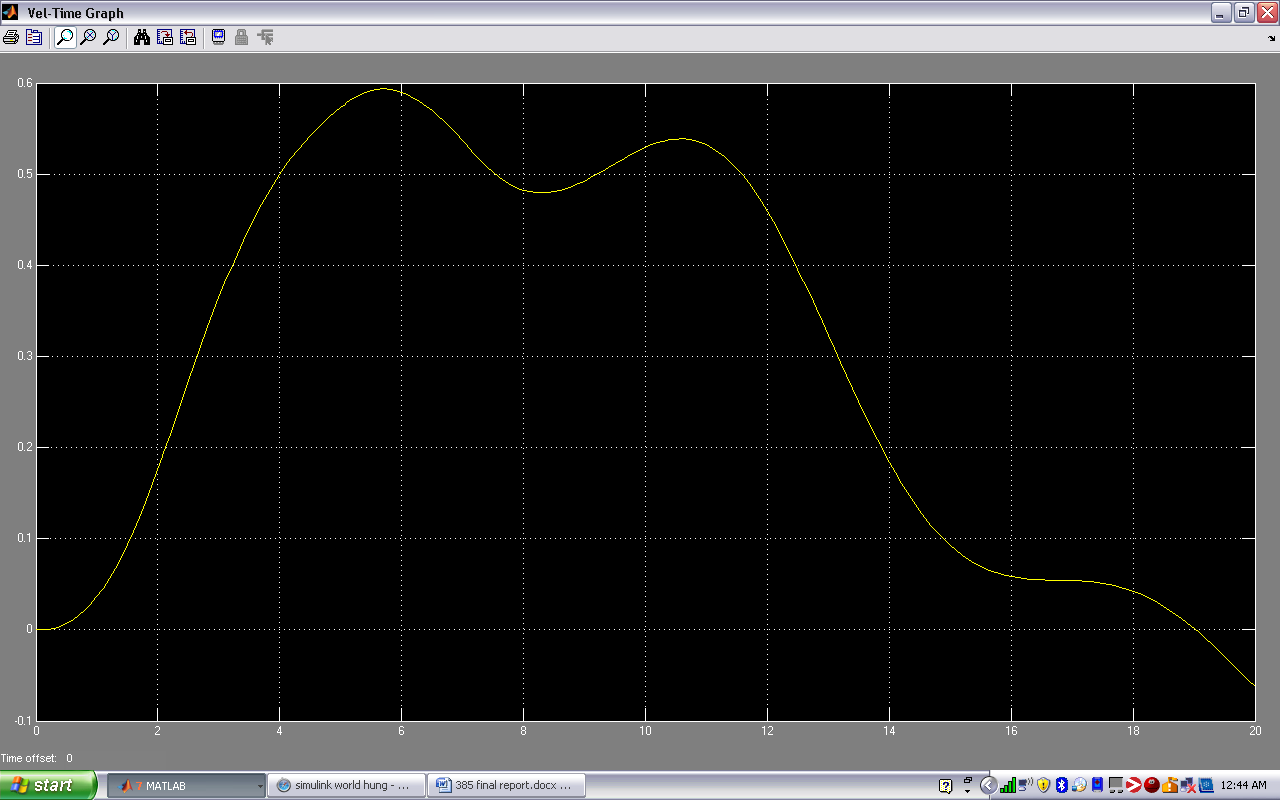
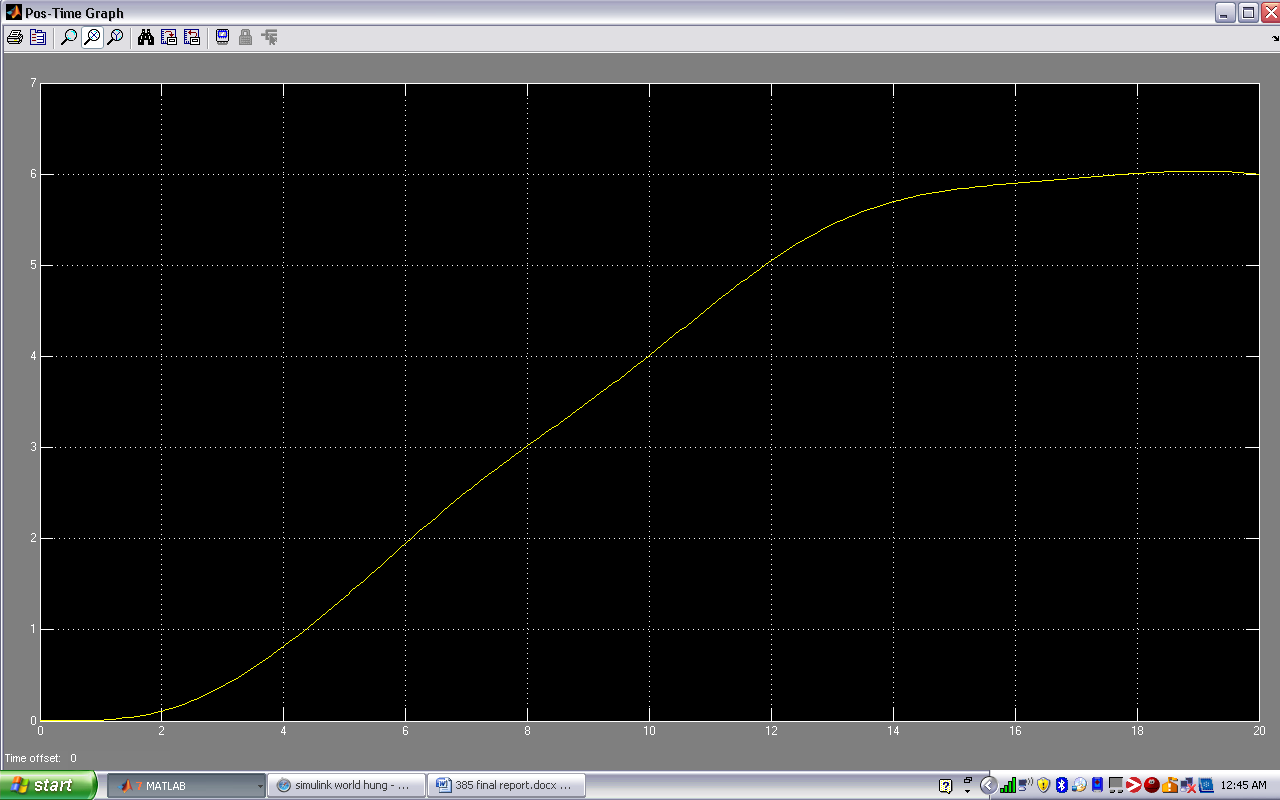
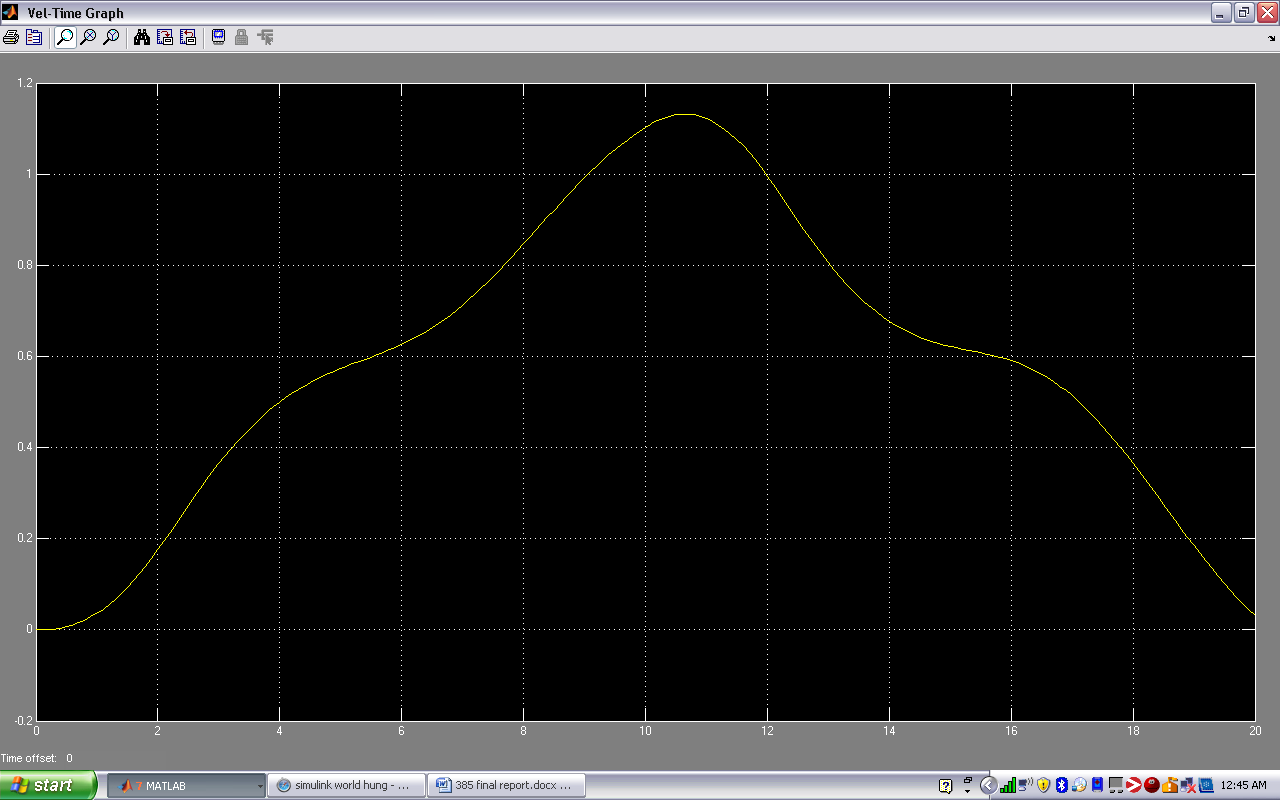
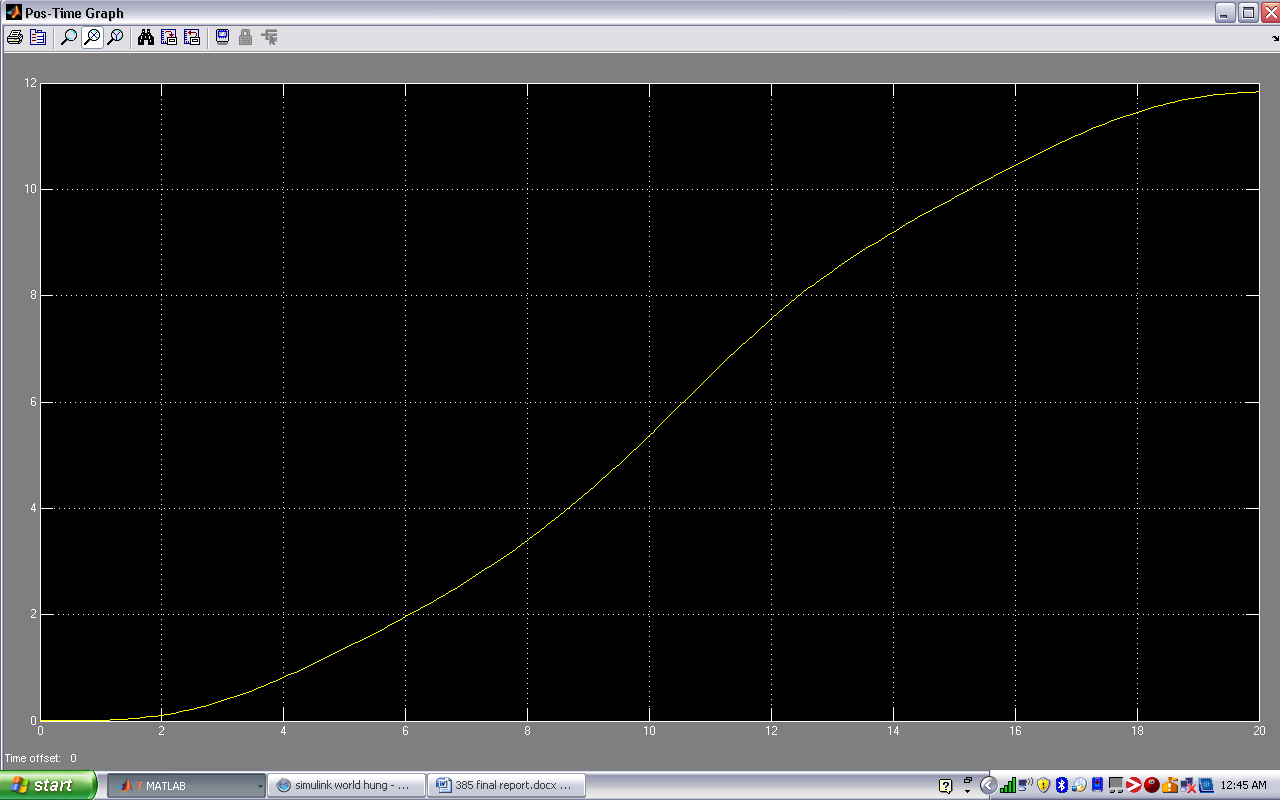
Figure 3 - 48 Controlled Output velocity 1 floor upFigure 3 - 49 Controlled Output position 1 floor upFigure 3 - 50 Controlled Output velocity 2 floors up

Figure 3 - 51 Controlled Output position 2 floors up

90,50,80

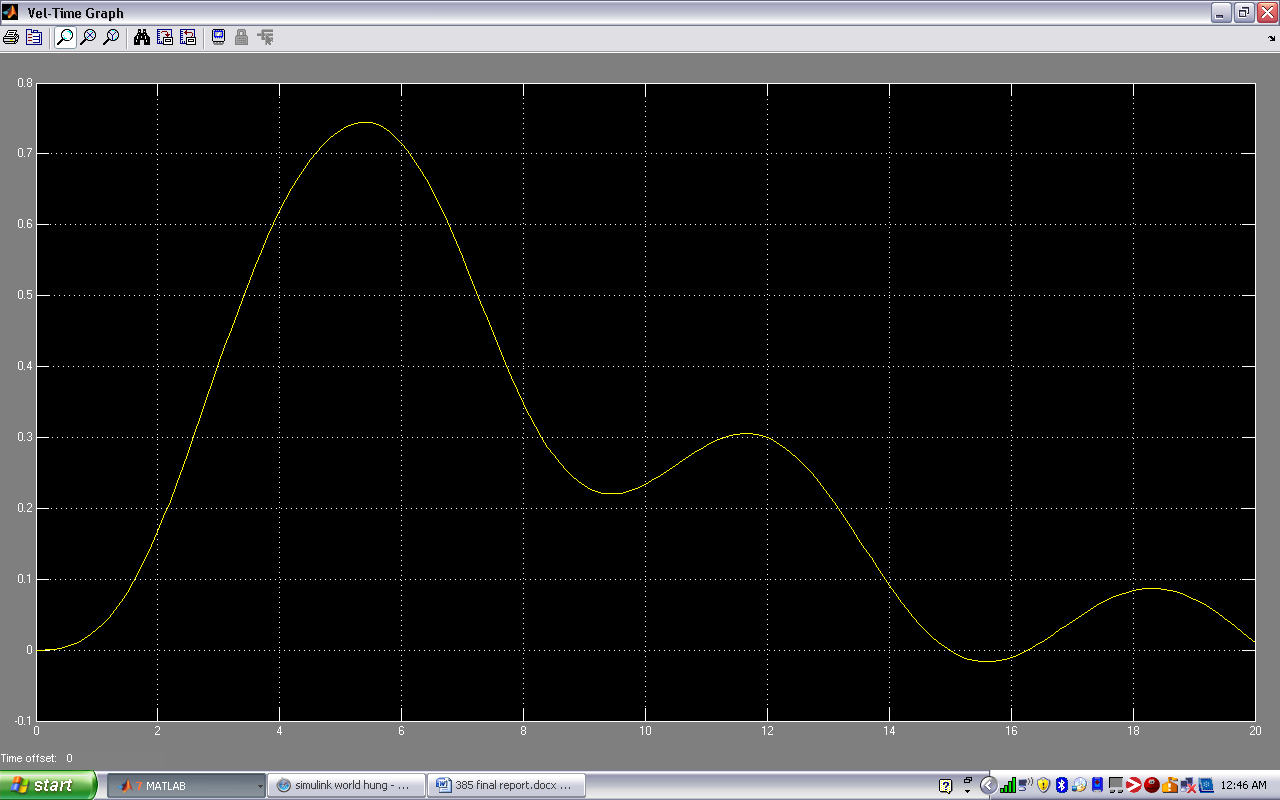
Figure 3 - 52 Controlled Output velocity 1 floor up

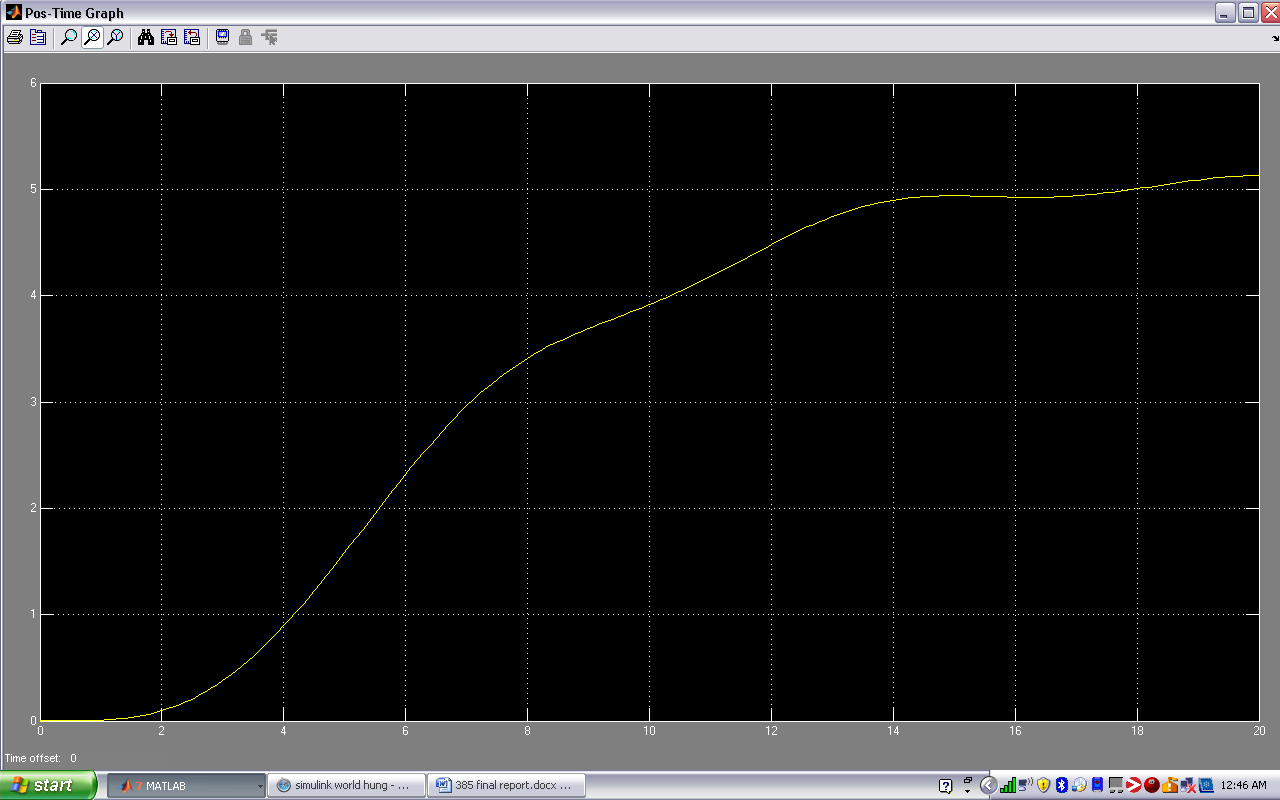
Figure 3 - 53 Controlled Output position 1 floor up

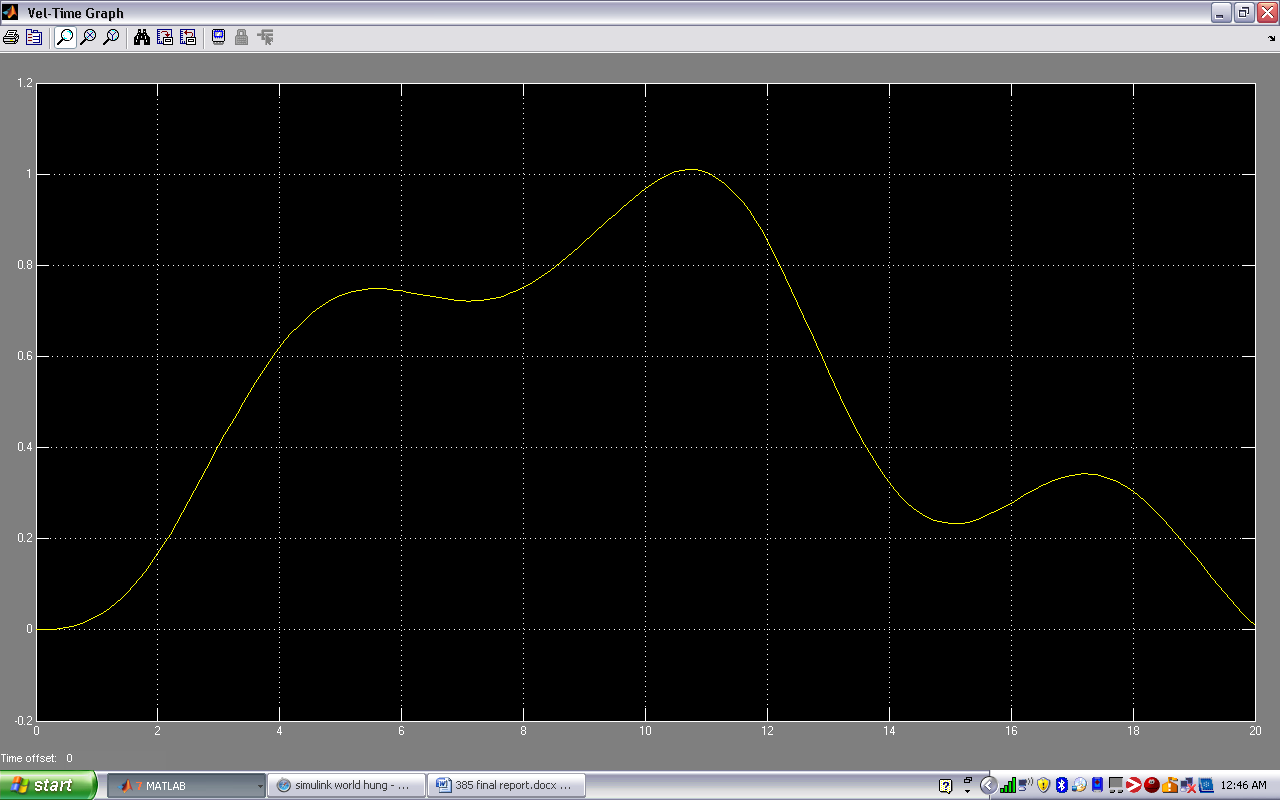
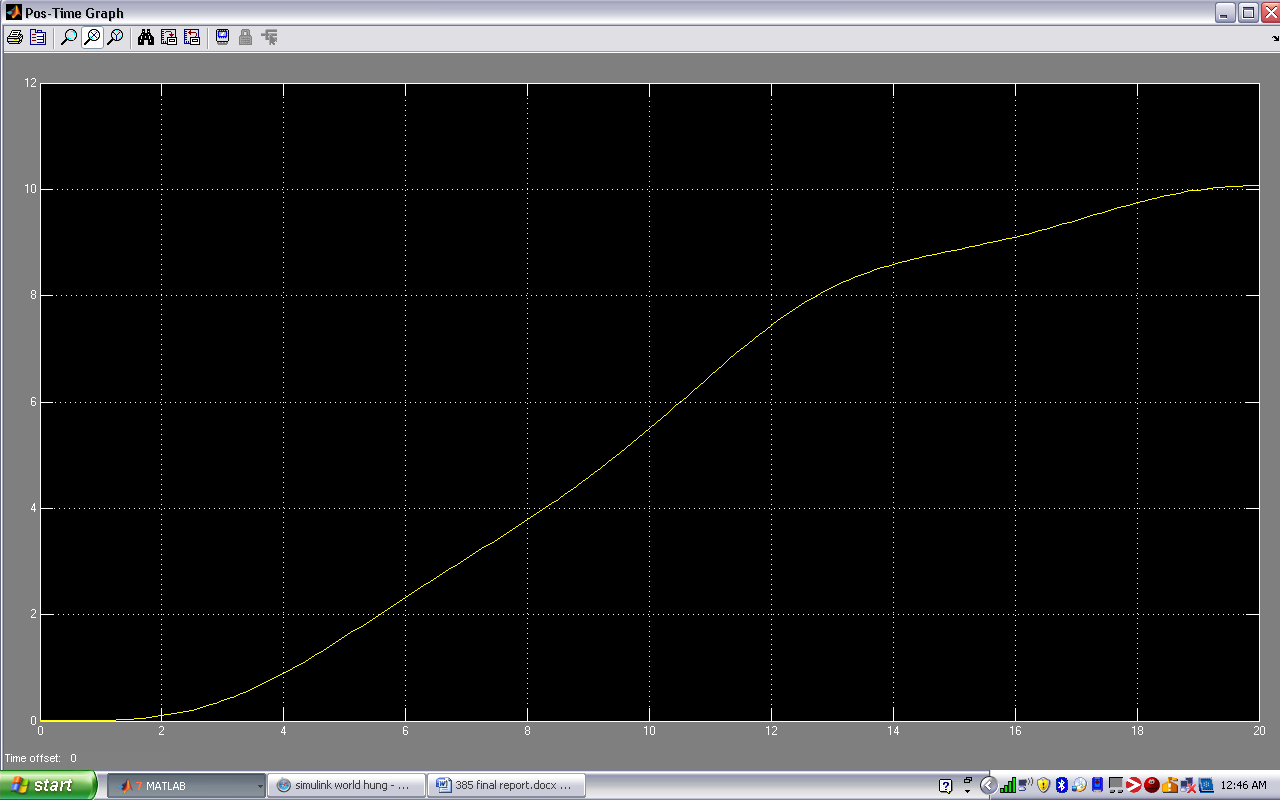
Figure 3 - 54 Controlled Output velocity 2 floors up

Figure 3 - 55 Controlled Output position 2 floors up

10,49.9,120

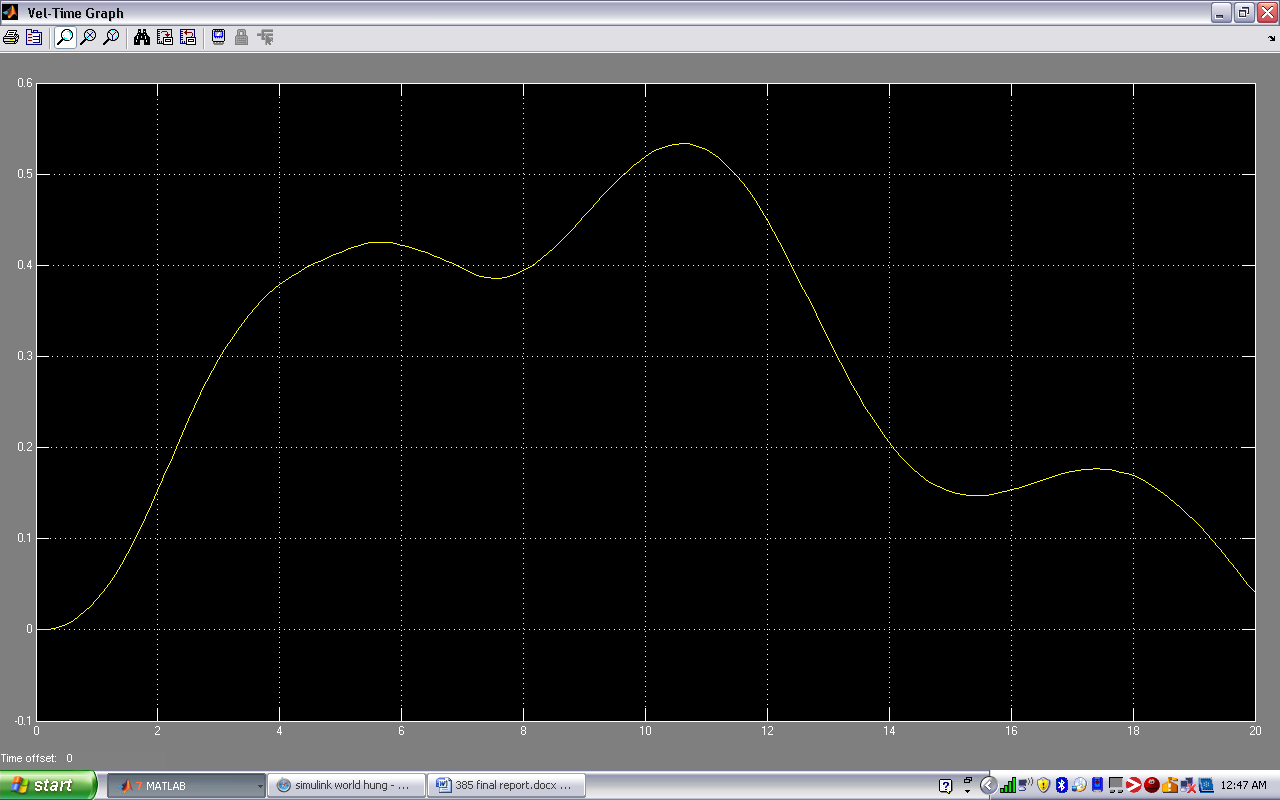
Figure 3 - 56 Controlled Output velocity 1 floor up

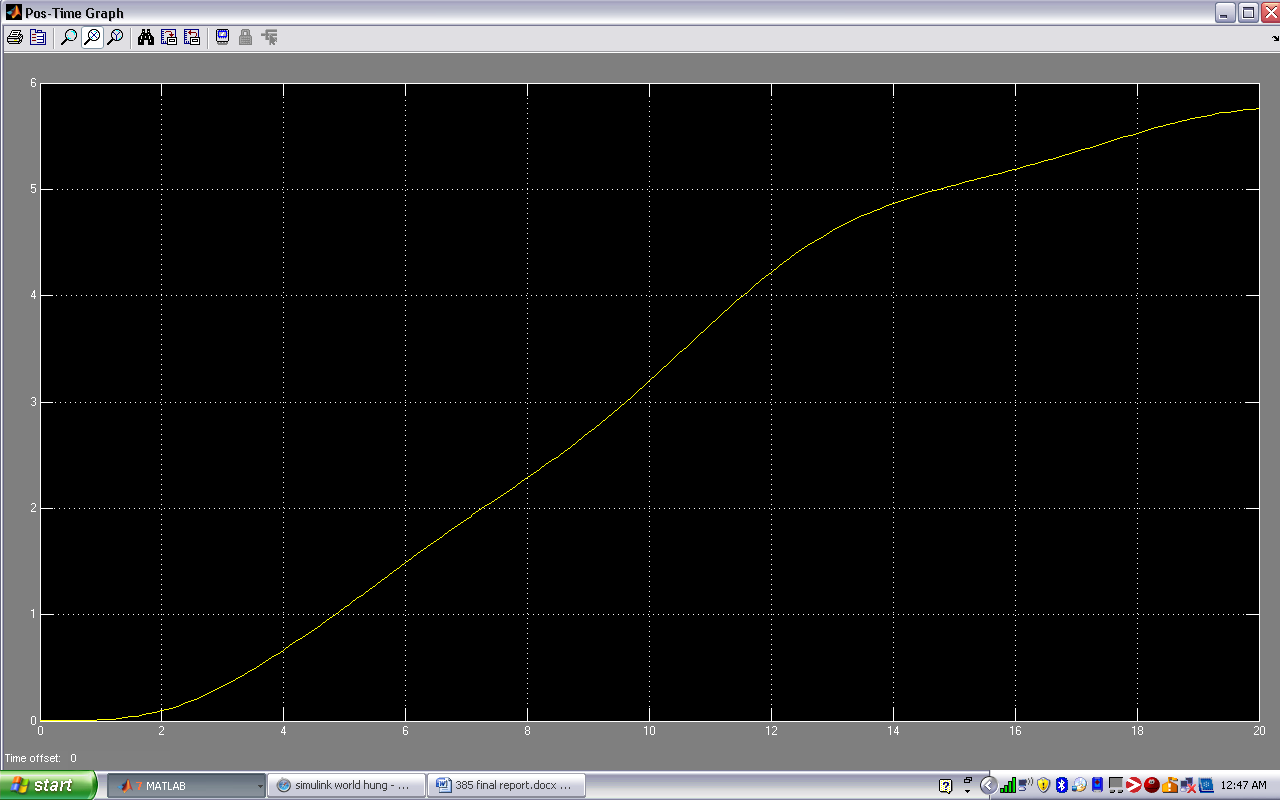
Figure 3 - 57 Controlled Output position 1 floor up

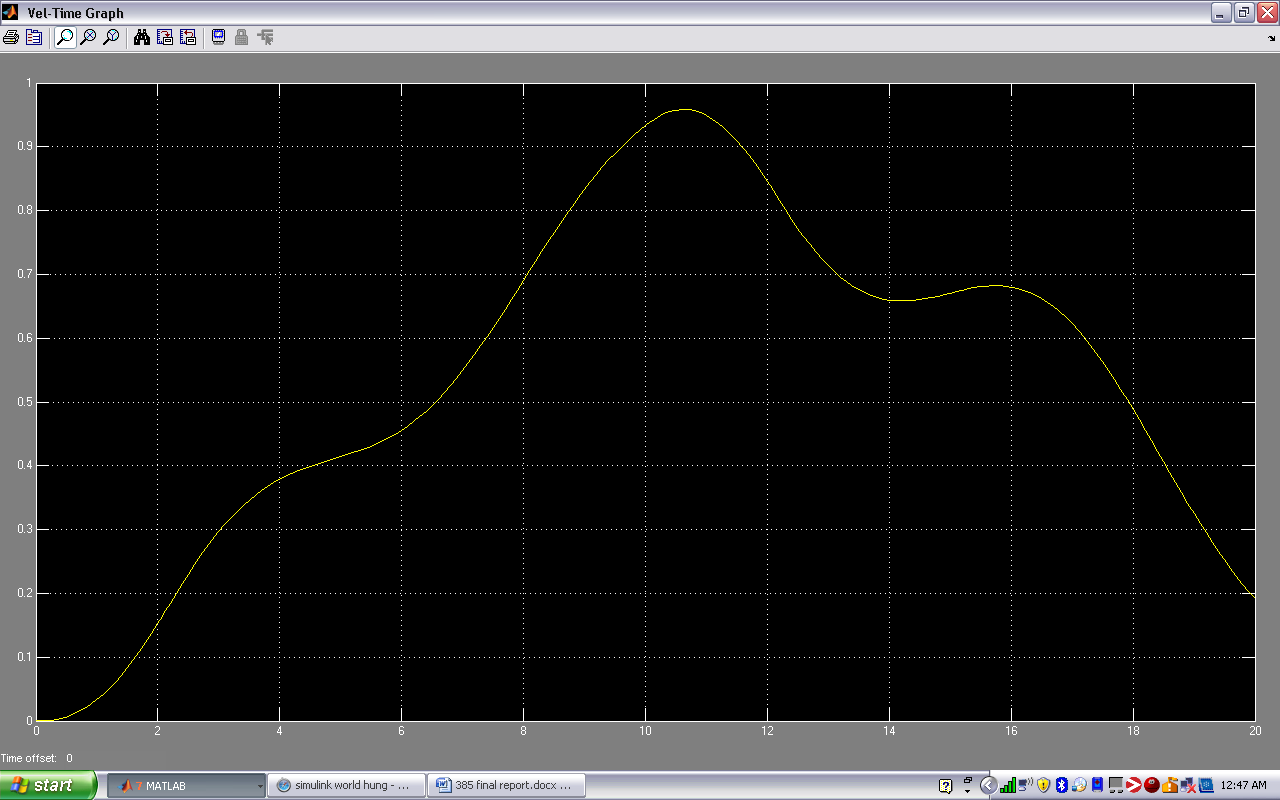
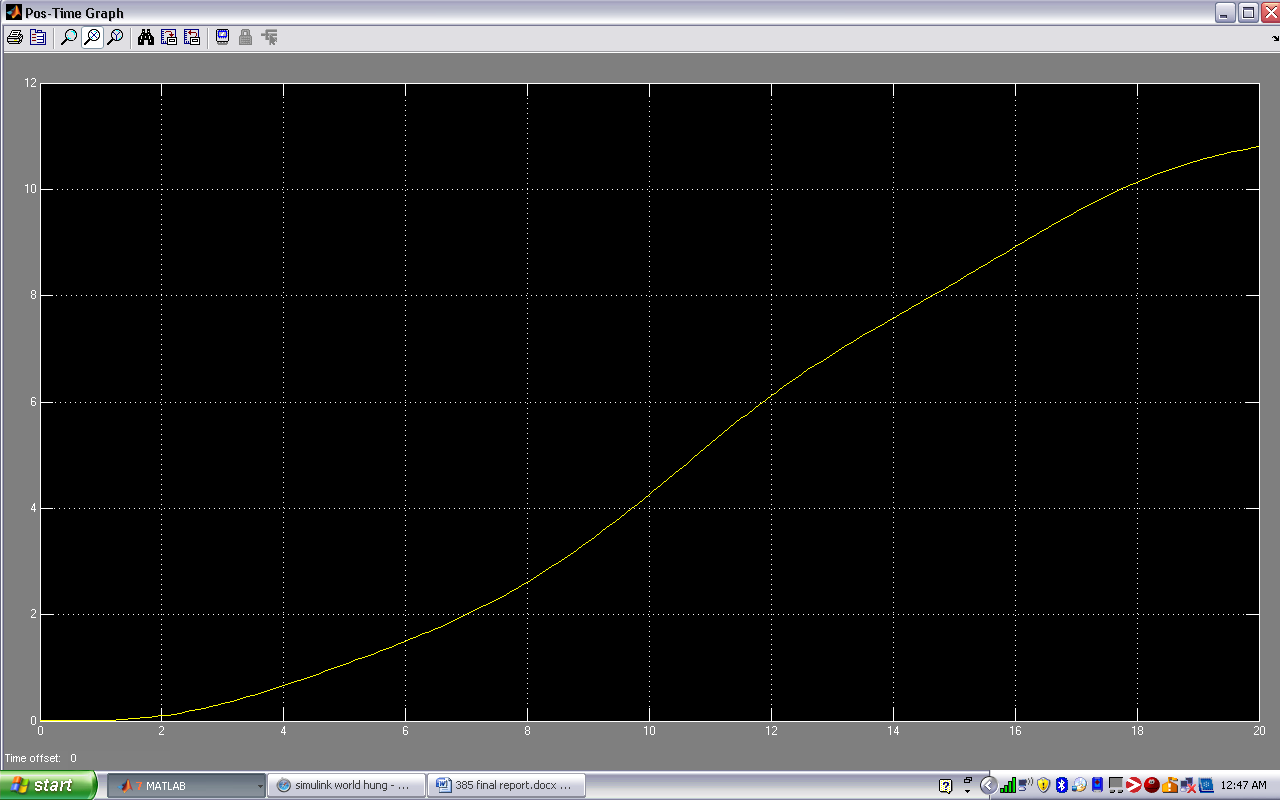
Figure 3 - 58 Controlled Output velocity 2 floors up

Figure 3 - 59 Controlled Output position 2 floors up

145,59.9,120

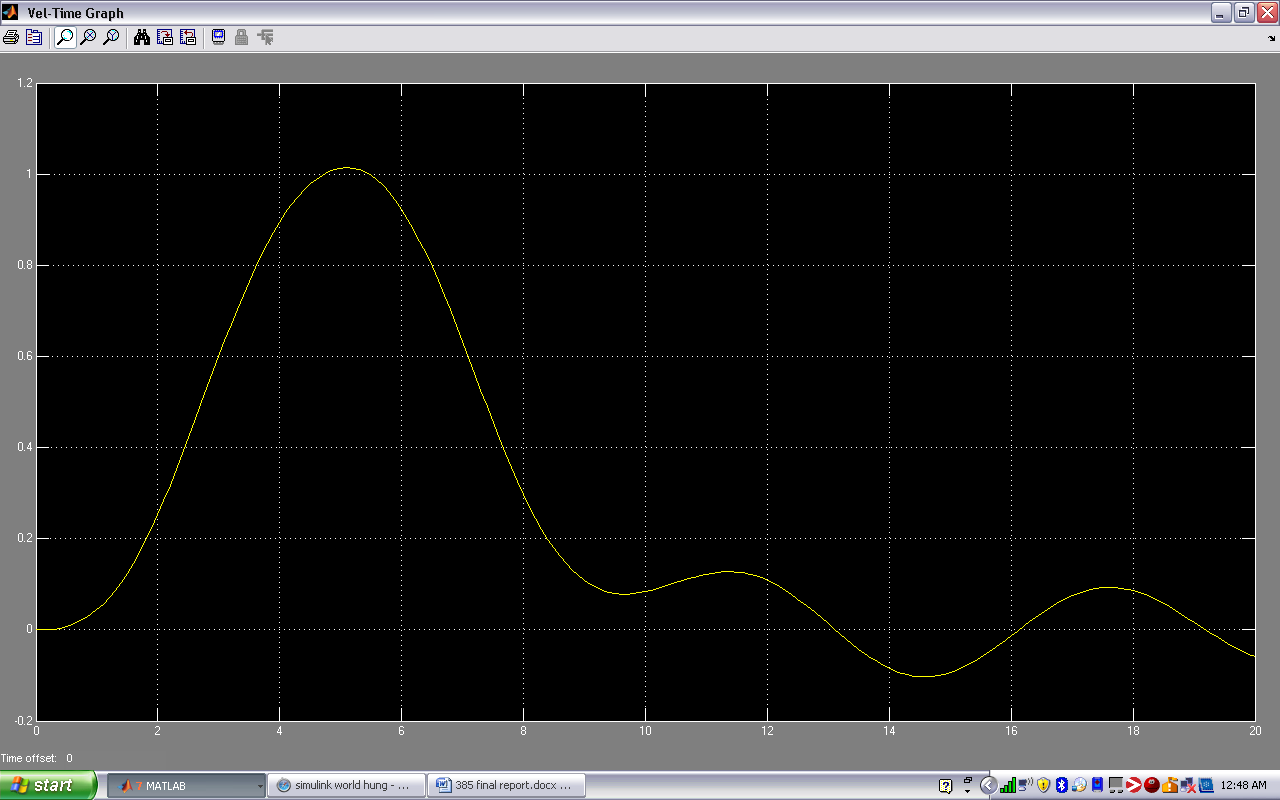
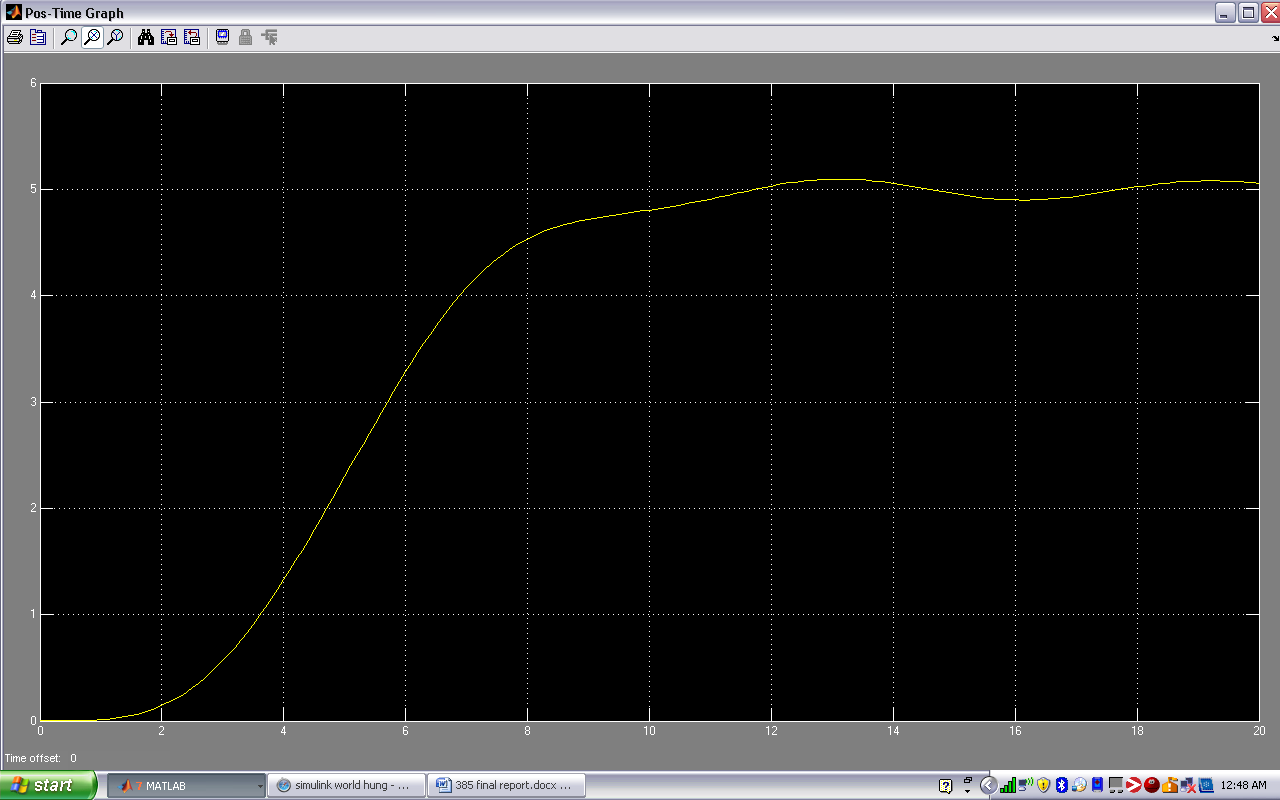
Figure 3 - 60 Controlled Output velocity 1 floor upFigure 3 - 61 Controlled Output position 1 floor up

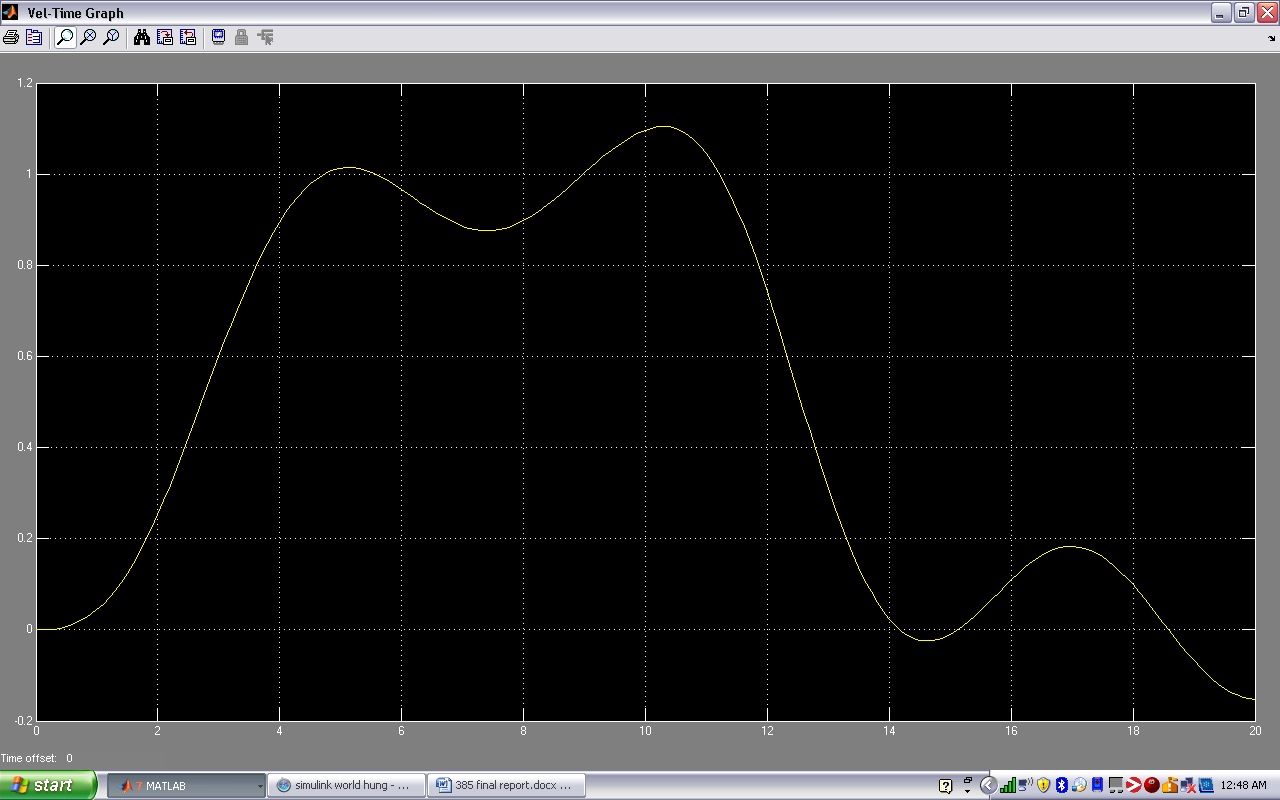
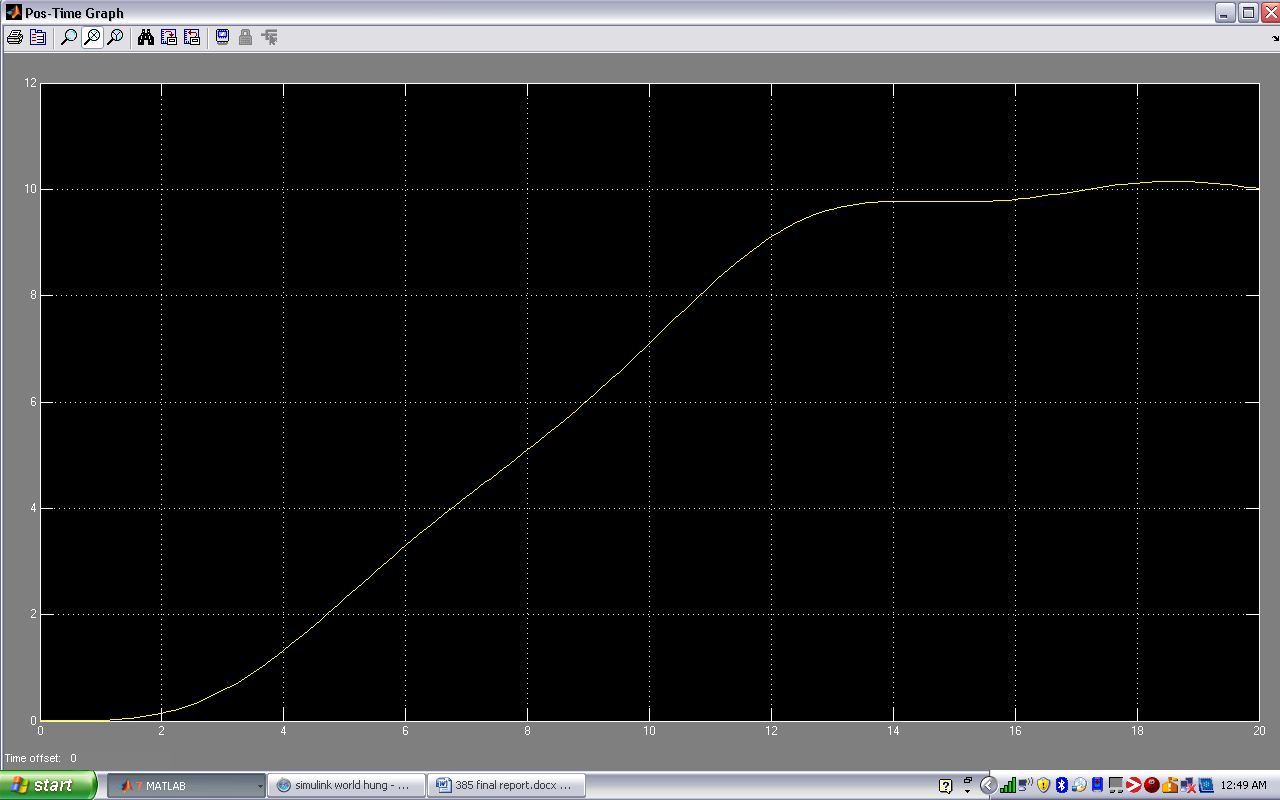
Figure 3 - 62 Controlled Output velocity 2 floors up

Figure 3 - 63 Controlled Output position 2 floors up

Best=147.3,60.2,126.5

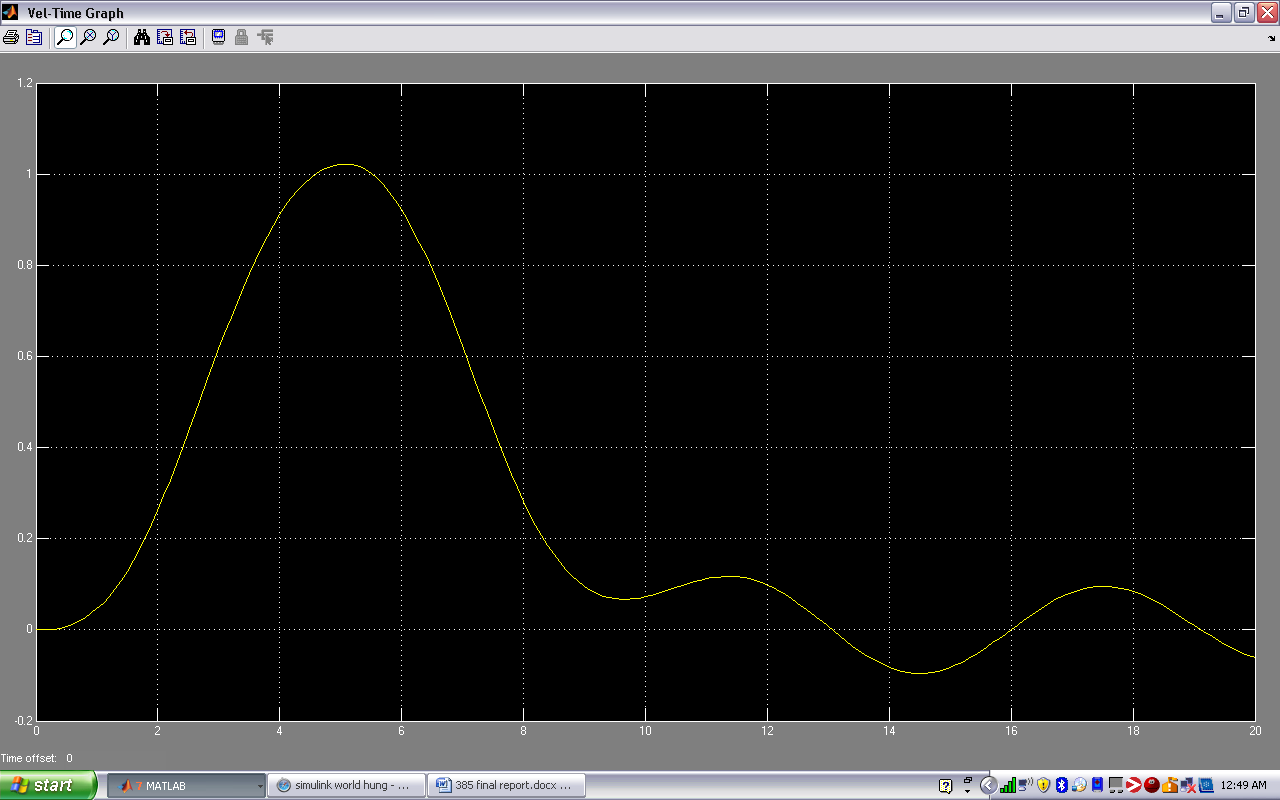
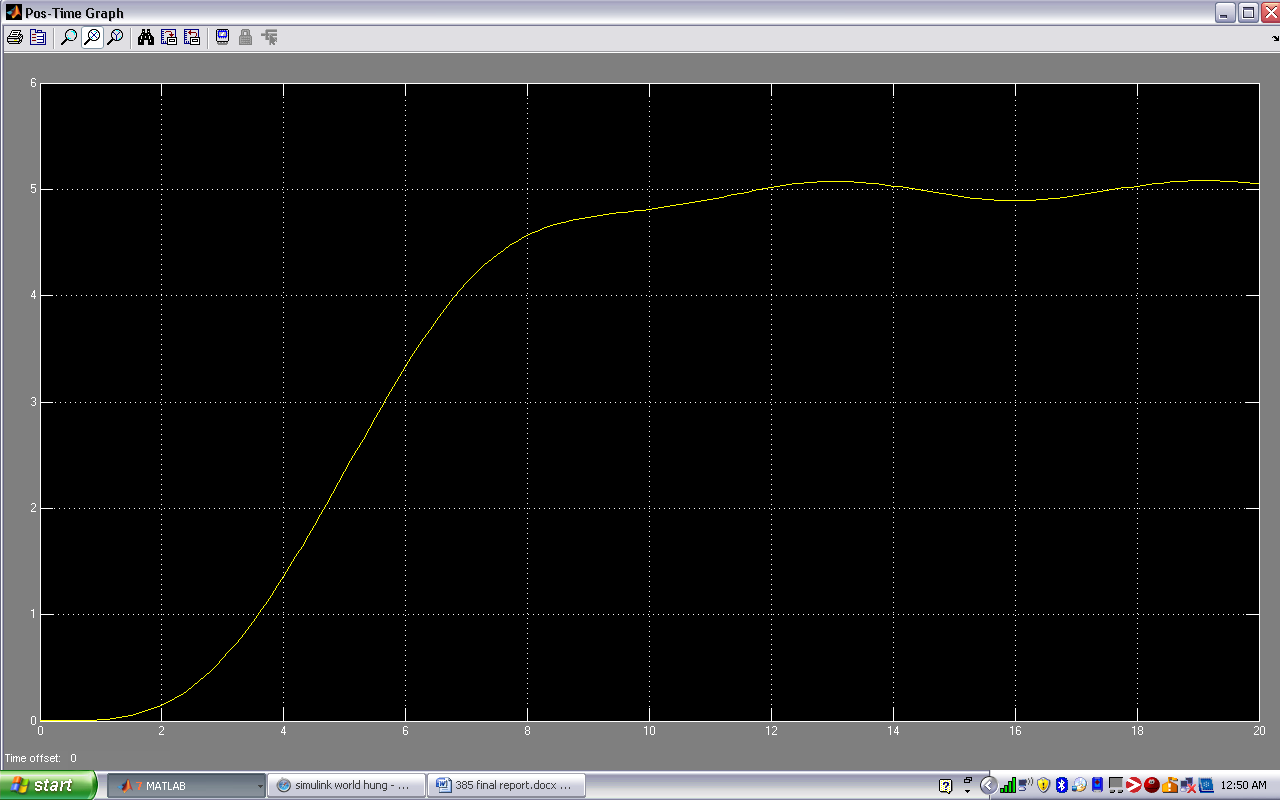
Figure 3 - 64 Controlled Output velocity 1 floor upFigure 3 - 65 Controlled Output position 1 floor up

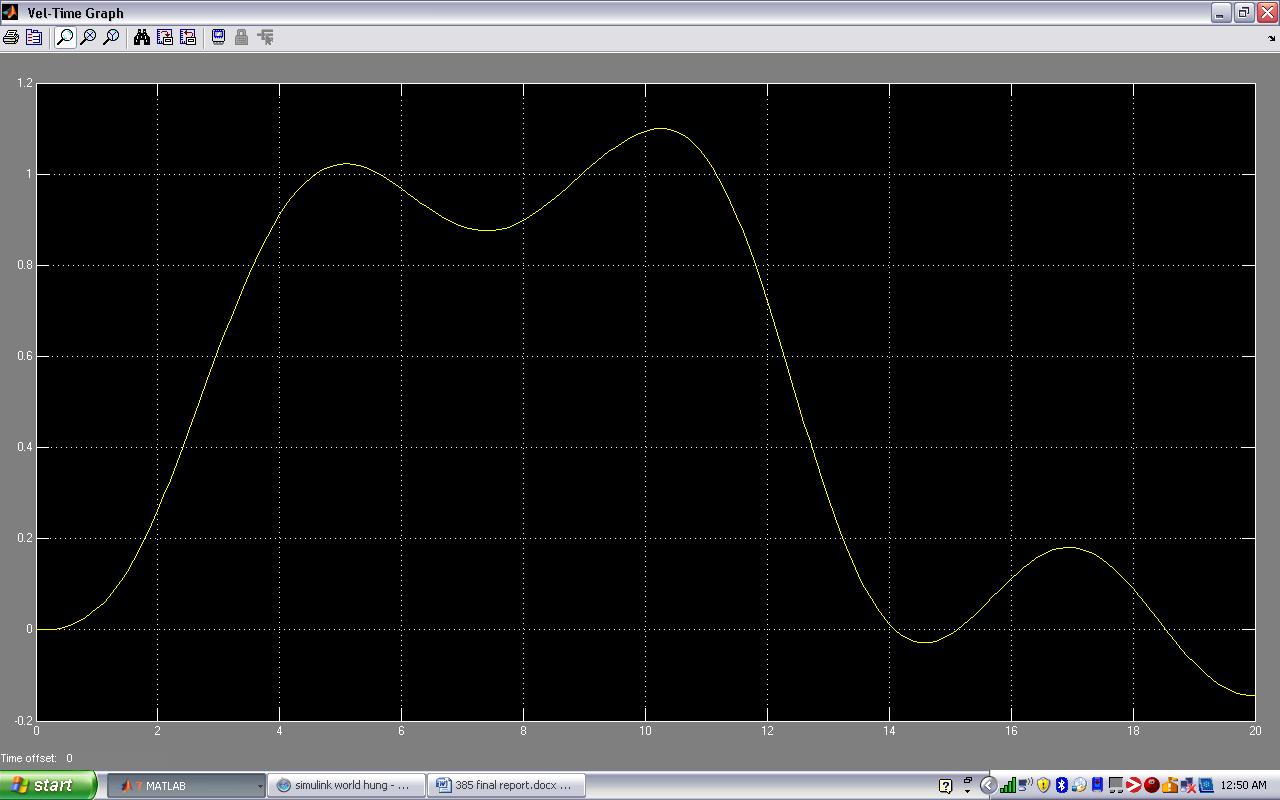
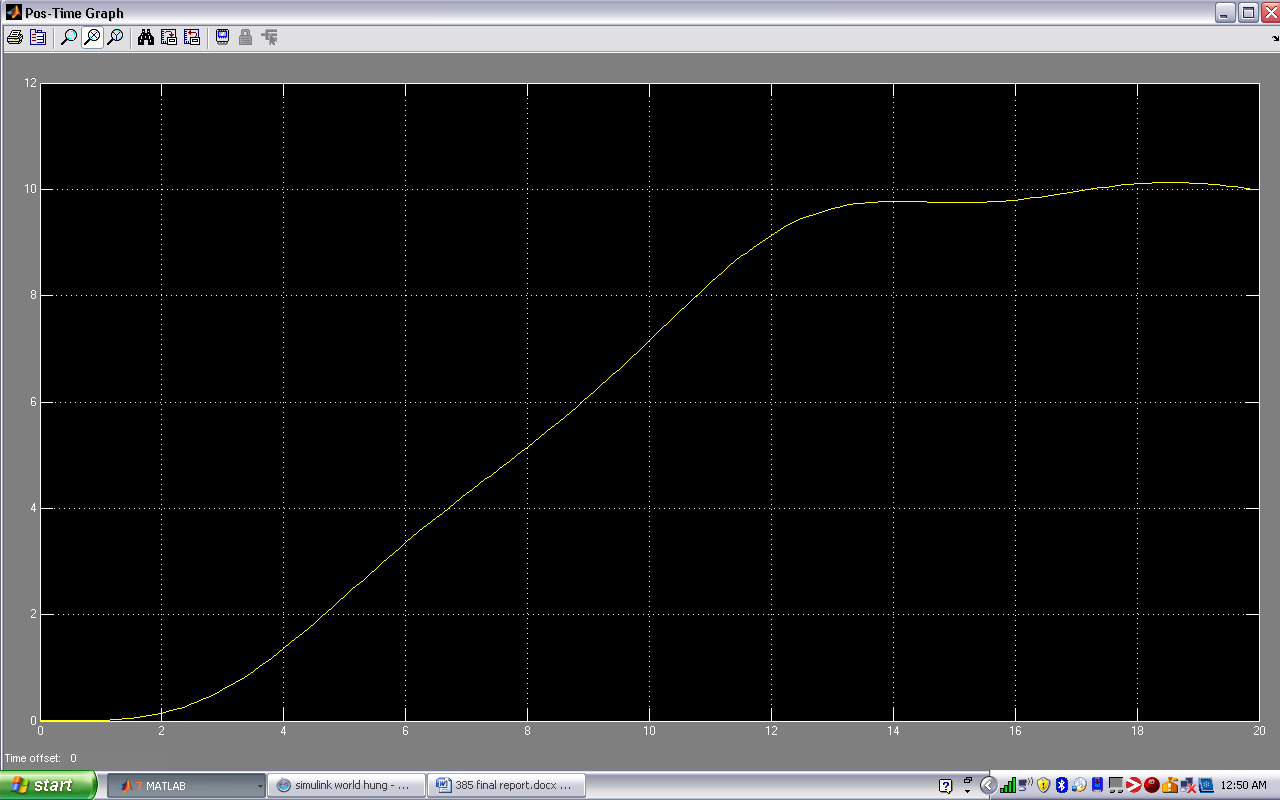
Figure 3 - 66 Controlled Output velocity 2 floors up

Figure 3 - 67 Controlled Output position 2 floors up

# Discussion

## Analysis

Between masses 0 to 260 there is a change in:

P=147.3 to 88.5 proving decrease with mass increase

I=60.2 to 44.8 proving decrease with mass increase

D=126.5 to 1 proving decrease with mass increase

Hence we can conclude an indirect proportion relation such that:

260kgs change =58.8p change hence 0.226 change in p per kg

=15.4i change hence 0.059 change in I per kg

=125.5d change hence 0.483 change in d per kg

We assume that if these controller decrements per increased kg are maintained proportionally and vice-versa, that the elevator will be able to move with quite good stability for any mass between 0 and 260. Hence for example a passenger mass increased by 50 on a certain floor will require the controller to have:

* A decrease in p of 11.3 which is 50 x 0.226
* A decrease in I of 2.95 which is 50 x 0.059
* A decrease in d of 24.15 which is 50 x 0.483

This way, a non-linear model of an elevator can be controlled linearly using a PID controller.

## Conclusions and General Discussion

An automatic elevator is a closed loop control system that is required for most multistory buildings. For elevator systems, an elevator cannot oscillate at the different stops. In general creating an over damped system allows for the lack of oscillation, however, in over damped systems the response time is very slow.

In real life, for elevators using a counter weight, a motor rotates the pulley which has the elevator on one end and the counter weight on the other. In the case of having a hydraulic elevator, there is still a pulley but the elevator mass is lifted using hydraulic ram. Our model accounted for both types by having a hydraulic actuator with passive damping at the bottom. In order to model any control system a lot of time and patience is required as not all systems are stable. To achieve stability in our elevator, the PID controller had to be tuned, along with appropriate values chosen. An elevator appears to be quite simple in concept; however, to tune and design a controller requires a lot more than just understanding how they work. We learned that in order to stabilize a system a lot of trial and error will be needed and our own judgments based on past attempts. A person presses a desired floor in an elevator, this creates an actuating signal for our system to respond.

To improve the system we modeled, for future versions of this project, we could have one spring at the top and one at the bottom and the hydraulic actuator with passive damping at the bottom. This helps such that the hydraulic actuator does not need to work so hard to push the mass upward. The bottom spring would help counter the one at the top hence less force required to move elevator mass up and or down.

# References (outside of class material)

D'Azzo, John Joachim et al. Linear Control System Analysis and Design With Matlab. CRC Press, August 14 2003. Pg 97-99 <<http://books.google.ca/books?id=GiQbrwgRvEYC&pg=PA5&lpg=PA5&dq=designing+an+elevator+control+system+matlab&source=bl&ots=INkwqyFToH&sig=Ctndej9zJkJhmQcLxXSiXdGnUJQ&hl=en&sa=X&ei=BcOET-qoFJDCgAf3rMG7Bw&ved=0CFgQ6AEwBw#v=onepage&q&f=false>> March 17 2012

Harris, Tom. "How Elevators Work” 12 February 2002. HowStuffWorks.com. <http://science.howstuffworks.com/transport/engines-equipment/elevator.htm> 3 April 2012.

Lewis, F. L. “EE 4343/5329 - Control System Design Project LECTURE 6” June 19 2000. Automation & Robotics Institute: The University of Texas at Arlington. <http://arri.uta.edu/acs/ee4314/lectures/RootLocus/pidLL.pdf> 2 April 2012

NASA. “How elevators work” September 2 2008. Nasa eClips. <<http://www.youtube.com/watch?v=6gxFp1x65Vo>> March 12 2012

Trumper , David. “2.003 Problem Archive” Pg 570: Problem 16.18, and pg 173. April 19 2005. Massachusetts Institute of Technology: MIT Open Course Ware. <http://ocw.mit.edu/courses/mechanical-engineering/2-003-modeling-dynamics-and-control-i-spring-2005/assignments/archive.pdf> 28 March 2012

Animated Control Tutorials <<http://140.78.137.200/ifac50/Project_F/content3.html>> 22 March 2012

Henderson, Tom. “Work, Energy, and Power - Lesson 1” The Physics Classroom. 1996 – 2011< http://www.physicsclassroom.com/class/energy/u5l1b.cfm> February 27 2012