Modern Control Systems (EEE F422)

Boeing 747 - Pitch Control

Course Project

By:

Amogh Dabholkar (2016A8PS0393G)

Dhruva Barfiwala (2016A3PS0135G)

Problem Statement and Overview

The equations governing the motion of an aircraft are a very complicated set of six nonlinear coupled differential equations. However, under certain assumptions, they can be decoupled and linearized into longitudinal and lateral equations. Aircraft pitch is governed by the longitudinal dynamics. The equations form a system which is unstable, thus requiring the use of a controller to achieve desired accuracy within a reasonable timeframe in addition to stabilizing the system.

In this project, we have used MATLAB as well as Simulink modeling to design an autopilot that controls the pitch of an aircraft. A PID tuned controller has been designed using MATLAB control designer which stabilizes the system and gives the desired output(which satisfies certain pre-decided design requirements). Finally, a Simulink model of the system is developed and the effect of noise in the external on the system performance is tested.

Assumptions

We will assume that the aircraft is in steady-cruise at constant altitude and velocity; thus, the thrust, drag, weight and lift forces balance each other in the x- and y-directions. We will also assume that a change in pitch angle will not change the speed of the aircraft under any circumstance (unrealistic but simplifies the problem a bit).

Physical System

The linearized equations of motion under the aforementioned assumptions are as follows:

$$\dot{\alpha} = \mu\Omega\sigma[-(C_L + C_D)\alpha + \frac{1}{(\mu - C_L)}q - (C_W \sin\gamma)\theta + C_L]$$

$$\dot{q} = \frac{\mu\Omega}{2i_{yy}}[[C_M - \eta(C_L + C_D)]\alpha + [C_M + \sigma C_M(1 - \mu C_L)]q + (\eta C_W \sin\gamma)\delta]$$

$$\dot{\theta} = \Omega q$$

Where,

α = Angle of attack.
q = Pitch rate.
θ = Pitch angle.
δ = Elevator deflection angle.
$\mu = \frac{\rho S \bar{c}}{4m}$.
ρ = Density of air.
S = Platform area of the wing.
$\bar{c}$ = Average chord length.
m = Mass of the aircraft.
$\Omega = \frac{2U}{\bar{c}}$.
U = Equilibrium flight speed.
C_T = Coefficient of thrust.
C_D = Coefficient of drag.
C_L = Coefficient of lift.
C_W = Coefficient of weight.
C_M = Coefficient of pitch moment.
γ = Flight path angle.
$\sigma = \frac{1}{1+\mu C_L}$ = Constant.
i_{yy} = Normalized moment of inertia.
$\eta = \mu\sigma C_M$ = Constant.

System Modelling

Considering the values taken from the data from one of Boeing's commercial aircrafts, the system equations can be written with numerical coefficients as follows:

$$\dot{\alpha} = -0.313\alpha + 56.7q + 0.232\delta$$
$$\dot{q} = -0.0139\alpha - 0.426q + 0.0203\delta$$
$$\dot{\theta} = 56.7q$$

Subsequently, the state-space model of the system can be made, with the state variables and external input

α=Angle of attack
q=Pitch rate
θ=Pitch angle
δ=Elevator deflection angle

$$\begin{bmatrix} \dot{\alpha} \\ \dot{q} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} -0.313 & 56.7 & 0 \\ -0.0139 & -0.426 & 0 \\ 0 & 56.7 & 0 \end{bmatrix} \begin{bmatrix} \alpha \\ q \\ \theta \end{bmatrix} + \begin{bmatrix} 0.232 \\ 0.0203 \\ 0 \end{bmatrix} [\delta]$$

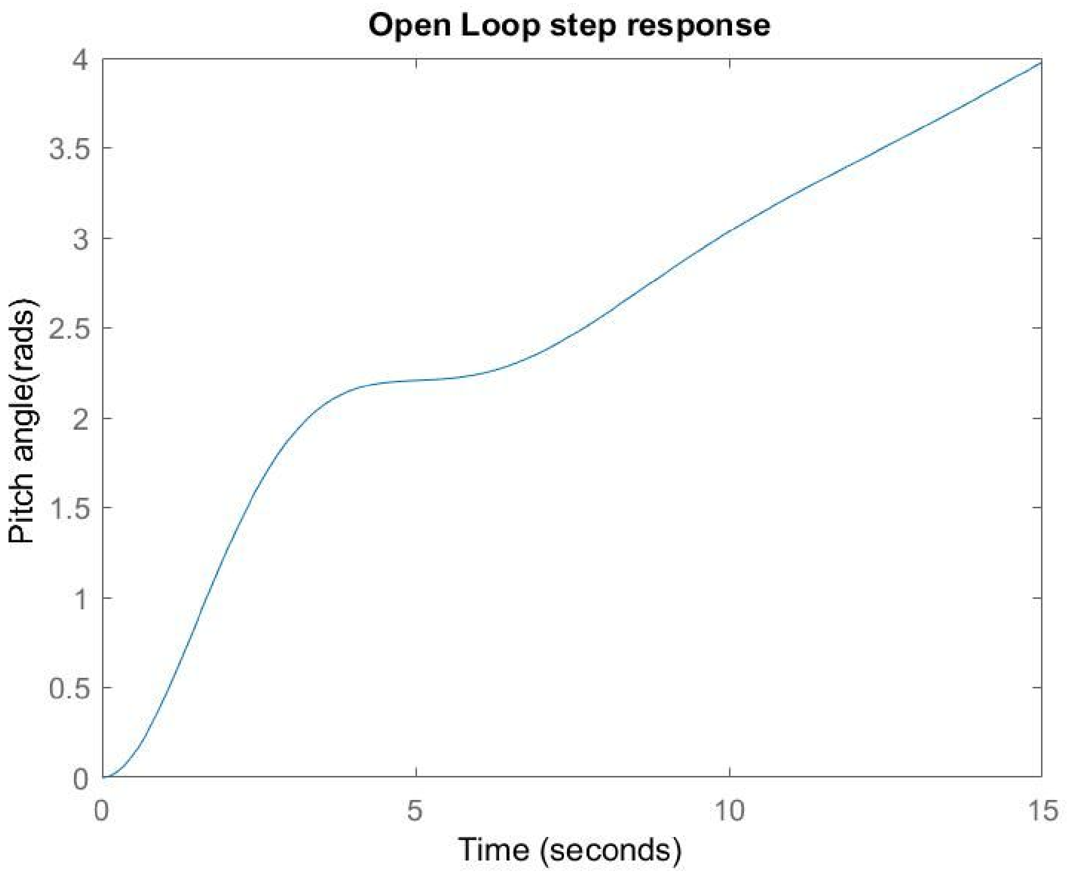
$$y = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \alpha \\ q \\ \theta \end{bmatrix}$$

The output for this system is the pitch angle, thus the open loop transfer function, computed by taking the Laplace transform of the system equations and performing certain mathematical manipulations, is

$$P(s) = \frac{\Theta(s)}{\Delta(s)} = \frac{1.151s + 0.1774}{s^3 + 0.739s^2 + 0.921s}$$

Open Loop Response

The step response of the open loop system is as follows



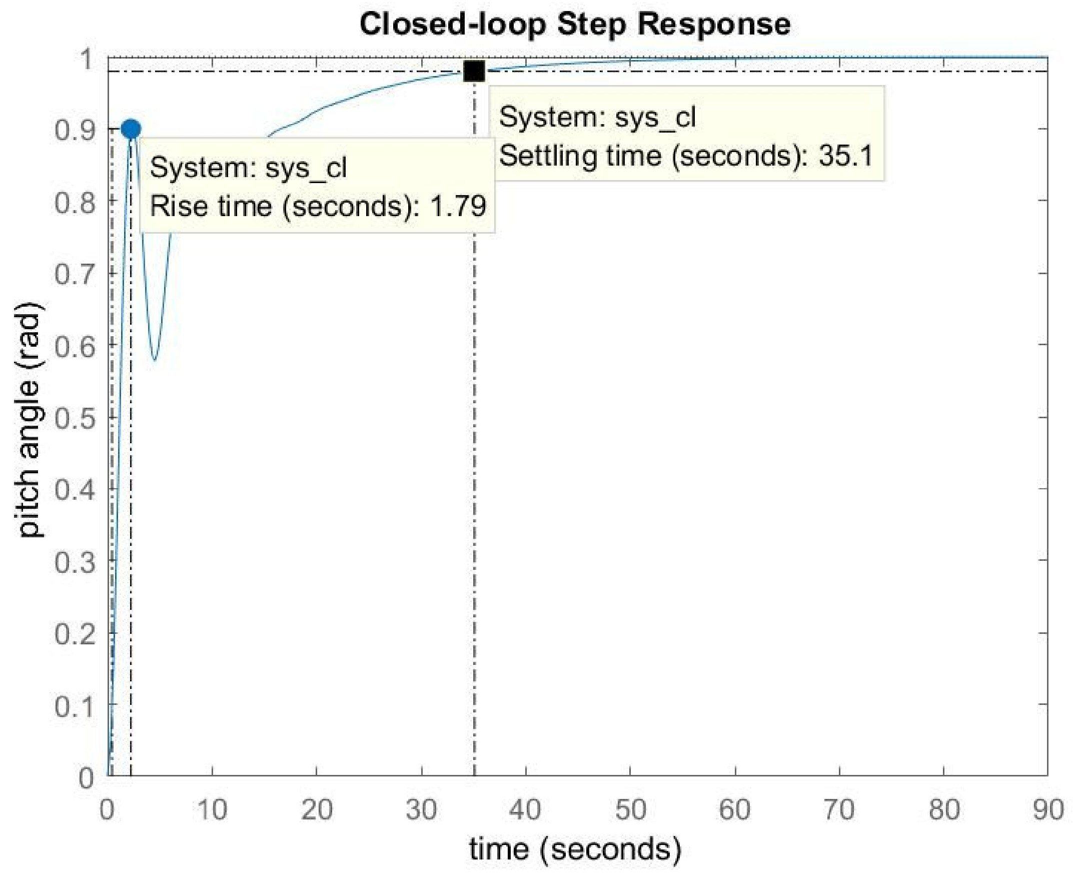
It can clearly be seen that this system is unstable. The system has a pole on the origin, which acts a pure integrator. Therefore, when the system is given a step input its output continues to grow to infinity.

Closed Loop Response

The unity gain closed loop transfer function of the system is as follows

$$Y(s) = \frac{1.151s + 0.1774}{s^3 + 0.739s^2 + 2.072s + 0.1774} R(s)$$

Corresponding to this, the step response of the system is



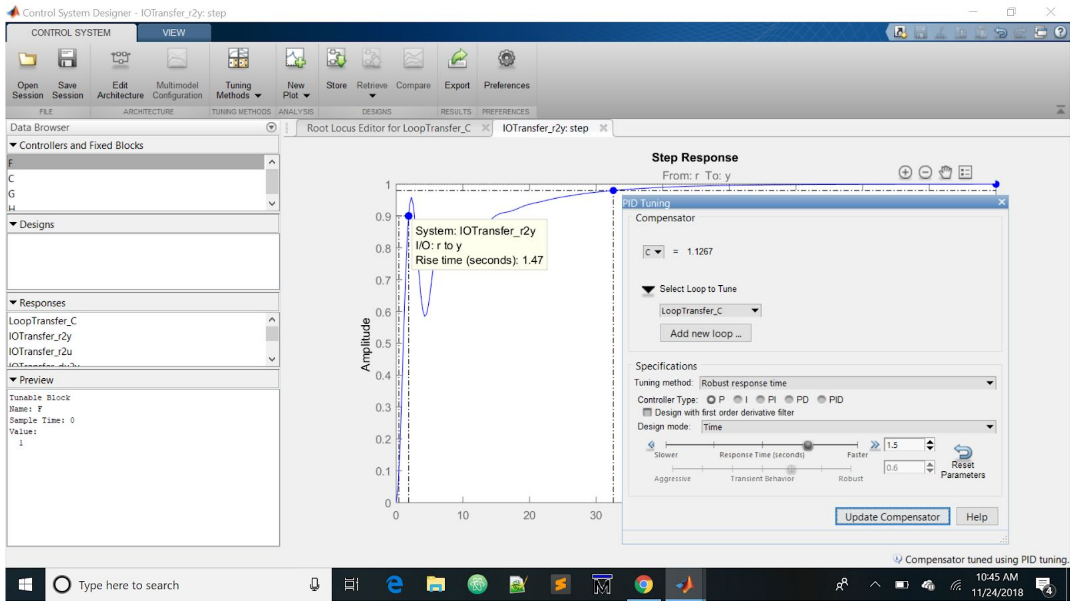
While the closed loop system is stable, it can be seen that this system is very sluggish in terms of the settling time and also has a sharp oscillatory behaviour immediately following its initial rise.
In order to improve the performance of this system, a controller is designed which meets the following design specifications:

- Overshoot less than 10%
- Rise time less than 2 seconds
- Settling time less than 10 seconds
- Steady-state error less than 2%

PID Controller Design

Step 1 - Proportional Control:

We have used the MATLAB Control System Designer tool to tune the controller. We started off by adding a simple proportional controller.


Control System Designer - IOTransfer_r2y: step

Tunable Block
Name: F
Sample Time: 0
Value:
  1

System: IOTransfer_r2y
I/O: r to y
Rise time (seconds): 1.47

PID Tuning

Compensator: C = 1.1267

Select Loop to Tune: LoopTransfer_C

Tuning method: Robust response time

Controller Type: [✓ P] I PI PD PID

Design mode: Time

Response Time (seconds): 1.5

Transient Behavior: 0.6

Compensator tuned using PID tuning.

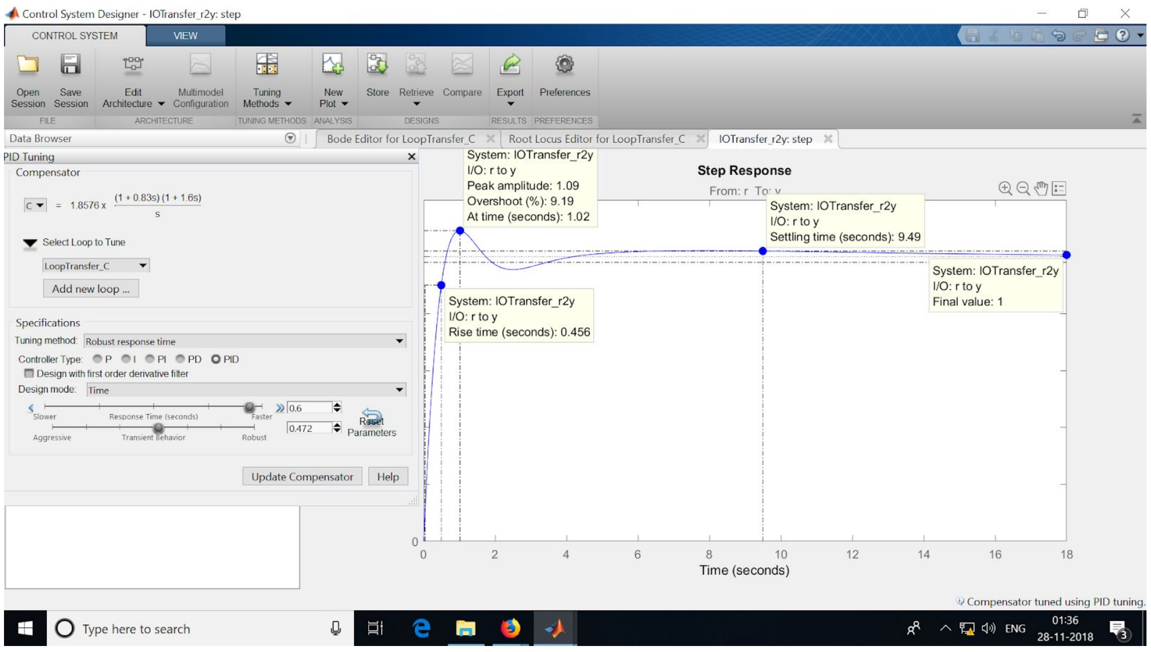
This controller meets the rise time requirement, but the settle time is much too large. A faster response time can be acquired by moving the slider to the right, however, this will result in an increase in overshoot and oscillation. The proportional controller does not provide a sufficient degree of freedom in tuning. Since the steady state requirement is already met, we will directly move to a PID Controller rather than a PI Controller.

Step 2 - PID Control:

By adding derivative control we may be able to reduce the oscillation in the response a sufficient amount that we can then increase the other gains to reduce the settling time.

Transfer Function for PID Controller :

$$1.8576 * \frac{(0.83\ s + 1)(1+1.6\ s)}{s}$$



PID Tuning

Compensator

C = 1.8576 x $\frac{(1 + 0.83s)(1 + 1.6s)}{s}$

Select Loop to Tune: LoopTransfer_C

Specifications

Tuning method: Robust response time

Controller Type: P I PI PD [✓ PID]

Design with first order derivative filter

Design mode: Time

Response Time (seconds): 0.6

Transient Behavior: 0.472

System: IOTransfer_r2y
I/O: r to y
Peak amplitude: 1.09
Overshoot (%): 9.19
At time (seconds): 1.02

System: IOTransfer_r2y
I/O: r to y
Settling time (seconds): 9.49

System: IOTransfer_r2y
I/O: r to y
Rise time (seconds): 0.456

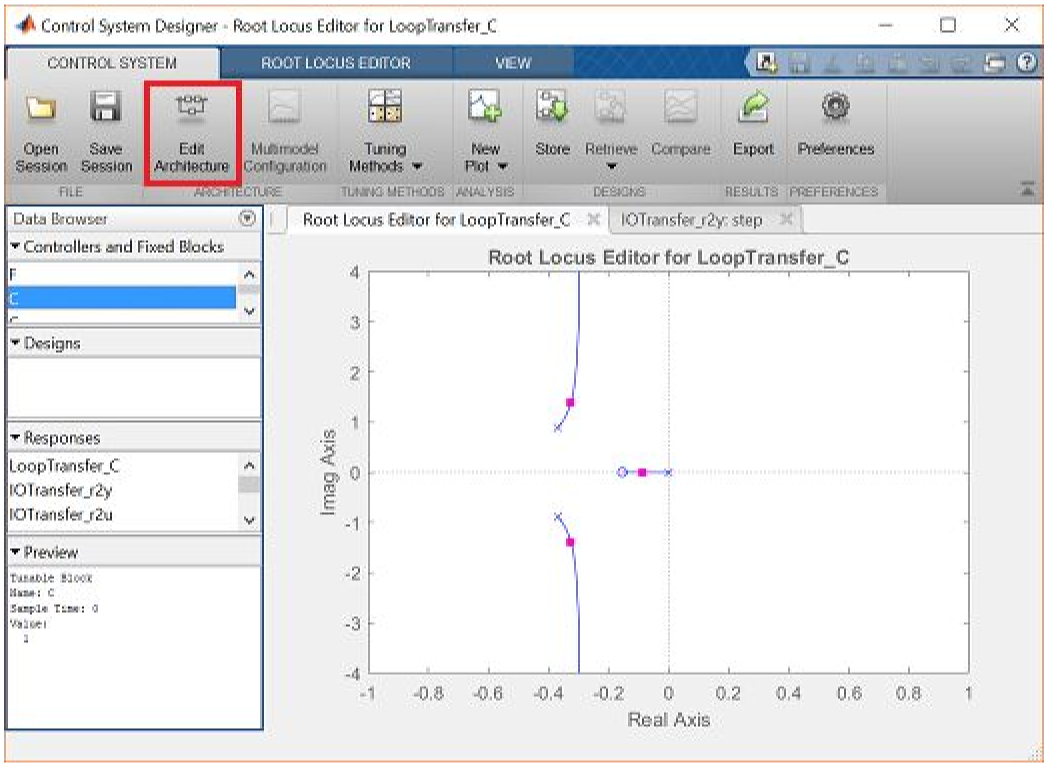
System: IOTransfer_r2y
I/O: r to y
Final value: 1

Compensator tuned using PID tuning.

- Reducing the **Response Time** requirement (moving the slider to the right) will make the response faster.
- Moving the **Transient Behavior** slider towards **Robust** will help reduce the oscillation.
- This response meets all of the requirements. So this is our final PID Controller.

Lead Controller Design

This is the original root locus design :

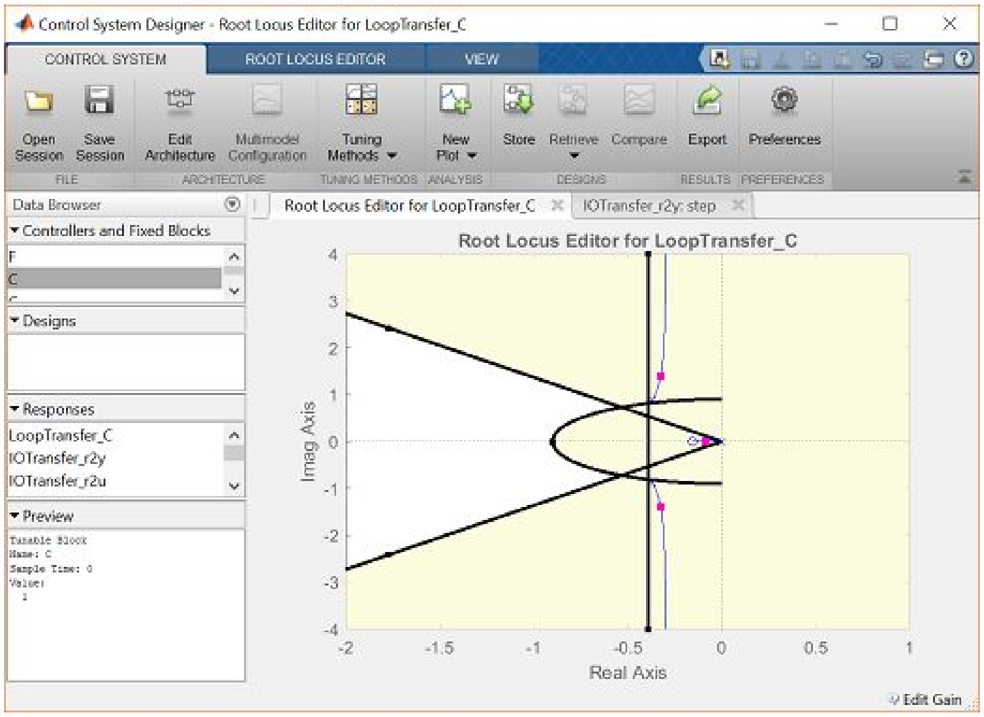


```
Tunable Block
Name: C
Sample Time: 0
Value:
  1
```

Settling time, Percent overshoot can be added directly. Rise time is not explicitly included as one of the drop-down choices, however, we can use the following approximate relationship that relates rise time to natural frequency.

$$\omega_n \approx \frac{1.8}{T_r}$$

Therefore, our requirement that rise time be less than 2 seconds corresponds approximately to a natural frequency of greater than 0.9 rad/sec for a canonical underdamped second-order system. Adding this requirement to the root locus plot in addition to the settle time and overshoot requirements generates the following figure.



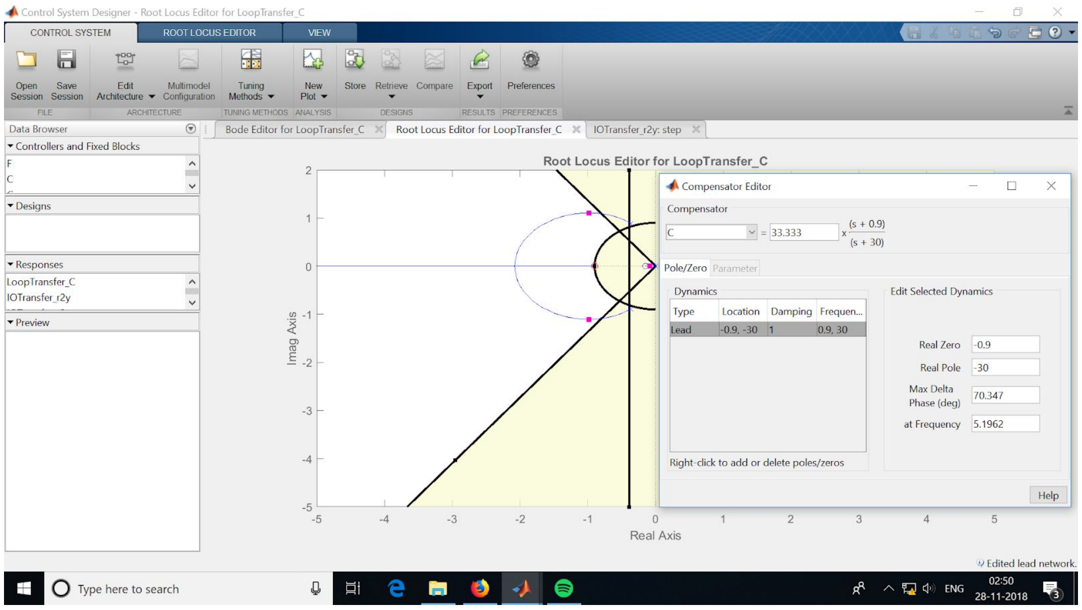
```
Tunable Block
Name: C
Sample Time: 0
Value:
  1
```

Lead Compensation :

We specifically need to shift the root locus more to the left in the complex plane to get it inside our desired region. We can do this is to employ a lead compensator.
The transfer function of a typical lead compensator is the following, where the zero has smaller magnitude than the pole, that is, it is closer to the imaginary axis in the complex plane.

$$C(s) = K\frac{s+z}{s+p}$$

We will place the pole further towards the left as we have to pull the root locus further towards the left.


Control System Designer - Root Locus Editor for LoopTransfer_C
Root Locus Editor for LoopTransfer_C
Real Axis
Imag Axis

Compensator Editor

Compensator

C = 33.333 x (s + 0.9) / (s + 30)

Type	Location	Damping	Frequen...
Lead	-0.9, -30	1	0.9, 30

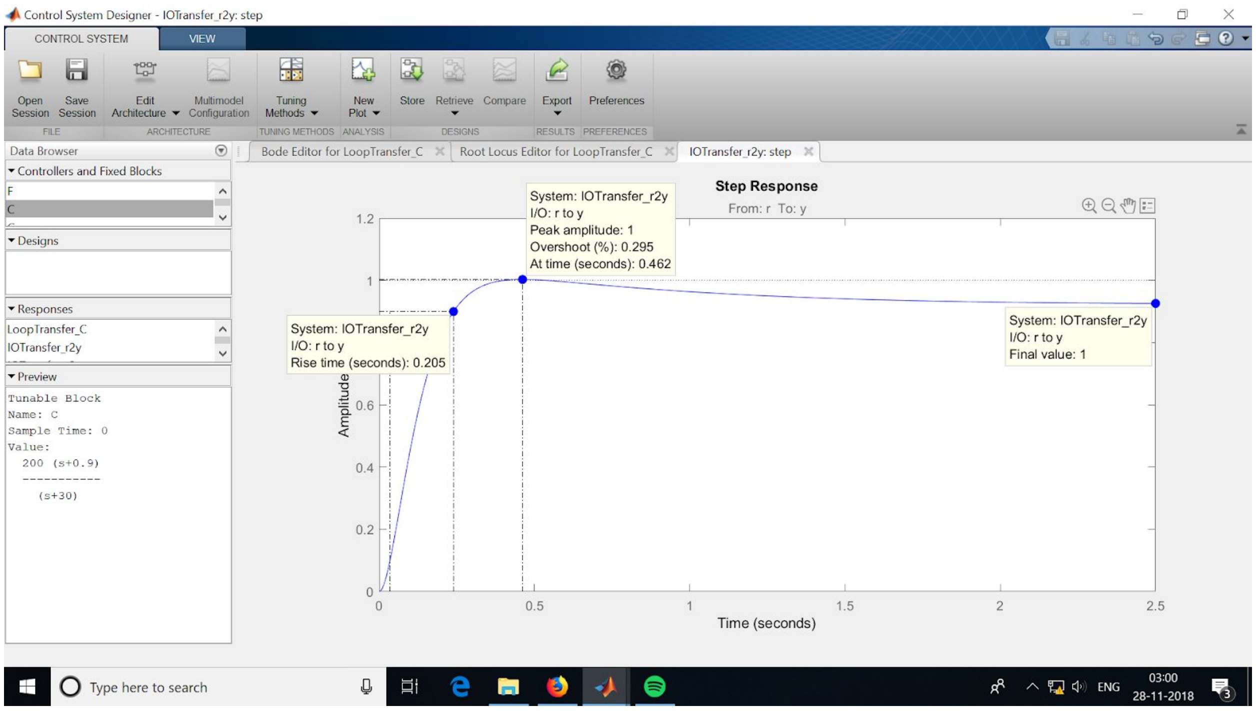
Real Zero: -0.9

Real Pole: -30

Max Delta Phase (deg): 70.347

at Frequency: 5.1962

Now, varying the K value, we change the settling time and rise time to fall within our requirement. After changing the values of K, K=200 satisfies all of our requirements.



System: IOTransfer_r2y
I/O: r to y
Peak amplitude: 1
Overshoot (%): 0.295
At time (seconds): 0.462

System: IOTransfer_r2y
I/O: r to y
Rise time (seconds): 0.205

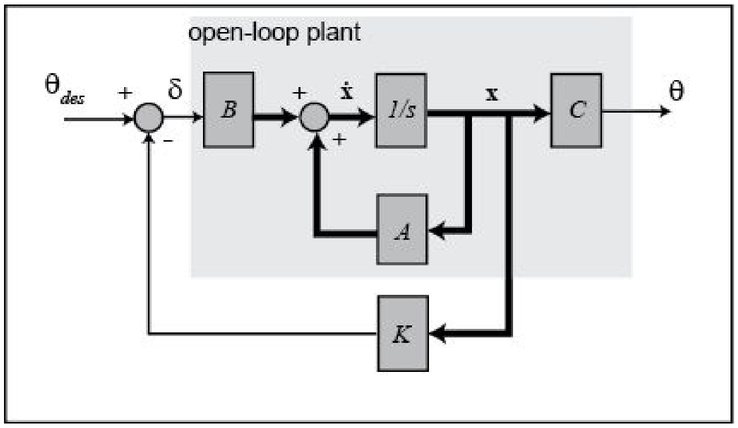
System: IOTransfer_r2y
I/O: r to y
Final value: 1

```
Tunable Block
Name: C
Sample Time: 0
Value:
  200 (s+0.9)
  -----------
    (s+30)
```

All of our design requirements are satisfied by this controller. This is our final Lead Compensator.

Control design via pole placement

The schematic of a full-state feedback control system is shown below (with D = 0).



where,

- K = control gain matrix
- $\mathbf{x}$ = $[\alpha,\ q,\ \theta]'$ = state vector
- θ_{des} = reference (r)
- δ = $(\theta_{des} - K\mathbf{x})$ = control input (u)
- θ = output (y)

Referring back to the state-space equations at the top of the page, we see that substituting the state-feedback law $\delta = (\theta_{des} - K\mathbf{x})$ for δ leads to the following.

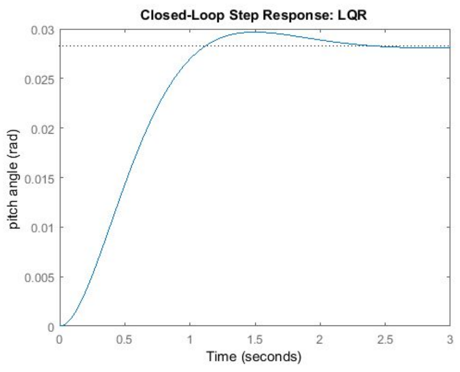
$$\dot{\mathbf{x}} = (A - BK)\mathbf{x} + B\theta_{des}$$

$$\theta = C\mathbf{x}$$

- Based on the above, matrix $A - BK$ determines the closed-loop dynamics of our system.
- This feedback law presumes that all of the state variables in the vector $\mathbf{x}$ are measured, even though θ is our only output. If this is not the case, then an observer needs to be designed to estimate the other state variables.
- This process becomes more difficult if we have a higher-order system or zeros. With a higher-order system, we can take advantage of Dominant Pole Phenomenon. The effect of zeros is more difficult to address using a pole-placement approach to control.
- Another limitation of this pole-placement approach is that it doesn't explicitly take into account other factors like the amount of required control effort.

Linear quadratic regulation(LQR)

- **Linear Quadratic Regulator (LQR)** method to generate the "best" gain matrix K, without explicitly choosing to place the closed-loop poles in particular locations.
- This type of control technique optimally balances the system error and the control effort based on a cost that the designer specifies that defines the relative importance of minimizing errors and minimizing control effort.
- In the case of the regulator problem, it is assumed that the **reference is zero.**
- Therefore, in this case the **magnitude of the error** is equal to the **magnitude of the state.**
- For simplicity, we will choose the **control weighted matrix (R)** equal to 1 and the **state-cost matrix (**Q**)** equal to $pC'C$.
- Employing the vector C from the output equation means that we will only consider those states in the output in defining our cost.
- In this case, θ is the only state variable in the output. The weighting factor (p) will be varied in order to modify the step response.
- In this case, R is a scalar since we have a single input system.
- Now, referring to the closed-loop state equations given above assuming a control law with non-zero reference, $\delta = (\theta_{des} - K\mathbf{x})$, we can then generate the closed-loop step response.
- Varying the value of p depending on the cost, we can vary the closed-loop step response. At p = 50, we get a satisfactory response regarding the settling time and response time. The graph is shown below -

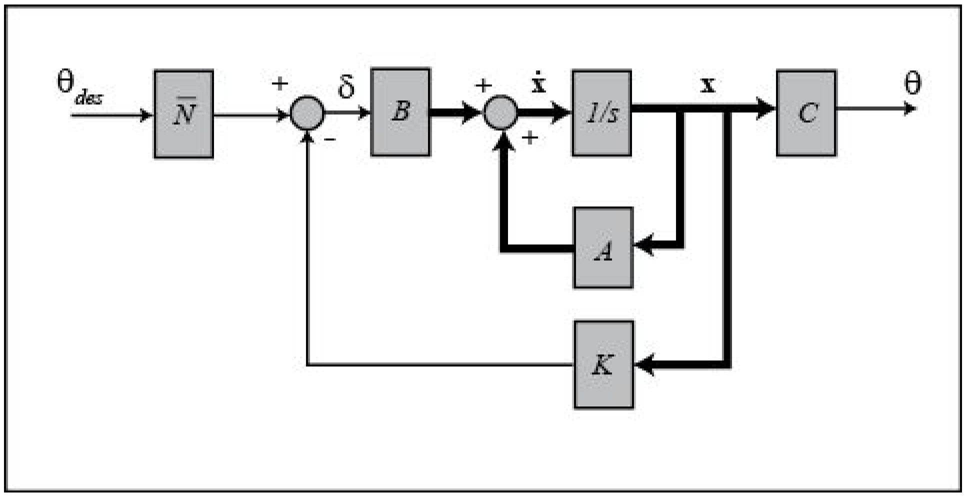


  - Examination of the above demonstrates that the rise time, overshoot, and settling time are satisfactory. However, there is a large steady-state error.

- One way to correct this is by introducing a precompensator ($\bar{N}$) to scale the overall output.

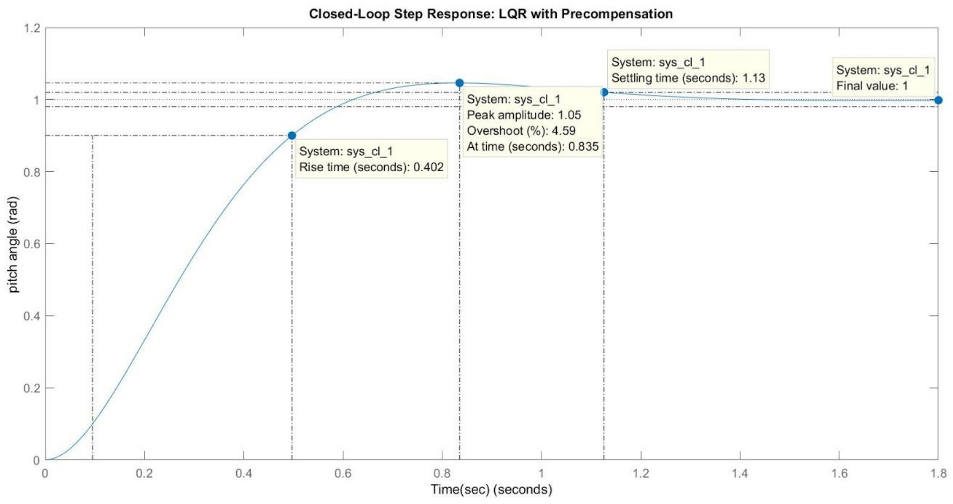
Adding precompensation :

- The full-state feedback system does not compare the output to the reference; instead, it compares all states multiplied by the control matrix ($K\mathbf{x}$) to the reference (see the schematic shown above).
- Thus, we should not expect the output to equal the commanded reference. To obtain the desired output, we can scale the reference input so that the output does equal the reference in steady state.
- This can be done by introducing a precompensator scaling factor called $\bar{N}$.
- The basic schematic of our state-feedback system with scaling factor $\bar{N}$ is shown below.



- $\bar{N}$ has been computed using the MATLAB function rscale.m.

- After including the precompensator in the system, the modified step response looks like :



- Now the steady-state error has been eliminated and all design requirements are satisfied.
- The precompensator $\overline{N}$ employed above is calculated based on the model of the plant and further that the precompensator is located outside of the feedback loop.
- Therefore, if there are errors in the model (or unknown disturbances) the precompensator will not correct for them and there will be steady-state error.
- We could have directly added an Integral Controller instead of a pre-compensator to resolve the error in steady-state value.
- The tradeoff with using integral control is that the error must first develop before it can be corrected for, therefore, the system may be slow to respond.
- The precompensator on the other hand is able to anticipate the steady-state offset using knowledge of the plant model.
- A useful technique is to **combine** the **precompensator** with **integral control** to leverage the advantages of each approach.
- This combined controller can be implemented by adding one more PI Controller or basically also involving this in the primary feedback controller.
- We have implemented this model in the Simulink Model.

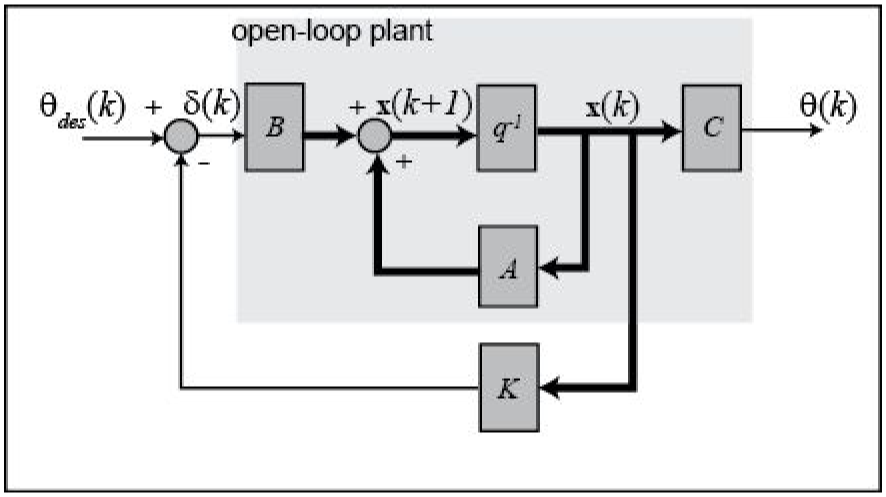
Digital Controller Design

- The first step in the design of a digital control system is to generate a sampled-data model of the plant.
- MATLAB can be used to generate this model from a continuous-time model using the c2d command. The c2d command requires three arguments: a system model, the sampling time (Ts) and the type of hold circuit.
- We will assume a zero-order hold (zoh) circuit.
- In choosing a sample time, it is desired that the sampling frequency be fast compared to the dynamics of the system in order that the sampled output of the system captures the system's full behavior, that is, so that significant inter-sample behavior isn't missed.
- One measure of a system's "speed" is its closed-loop bandwidth.
- A good rule of thumb is that the sampling frequency be at least 30 times larger than the closed-loop bandwidth frequency which can be determined from the closed-loop Bode plot.
- From the closed-loop Bode plot, the bandwidth frequency can be determined to be approximately 2 rad/sec (0.32 Hz).
- Thus, to be sure we have a small enough sampling time, we will use a sampling time of 0.01 sec/sample.
- The discrete-time state-space model shown below:

$$\begin{bmatrix} \alpha(k+1) \\ q(k+1) \\ \theta(k+1) \end{bmatrix} = \begin{bmatrix} 0.9969 & 0.05649 & 0 \\ -0.0001 & 0.9957 & 0 \\ 0 & 0.5658 & 1 \end{bmatrix} \begin{bmatrix} \alpha(k) \\ q(k) \\ \theta(k) \end{bmatrix} + \begin{bmatrix} 0.0024 \\ 0.0002 \\ 0.0001 \end{bmatrix} [\delta(k)]$$

$$y(k) = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \alpha(k) \\ q(k) \\ \theta(k) \end{bmatrix} + [0][\delta(k)]$$

- The schematic of a discrete full-state feedback control system is shown below, where q^{-1} is the delay operator (not the aircraft's pitch rate q). Note that it is assumed that D = 0.



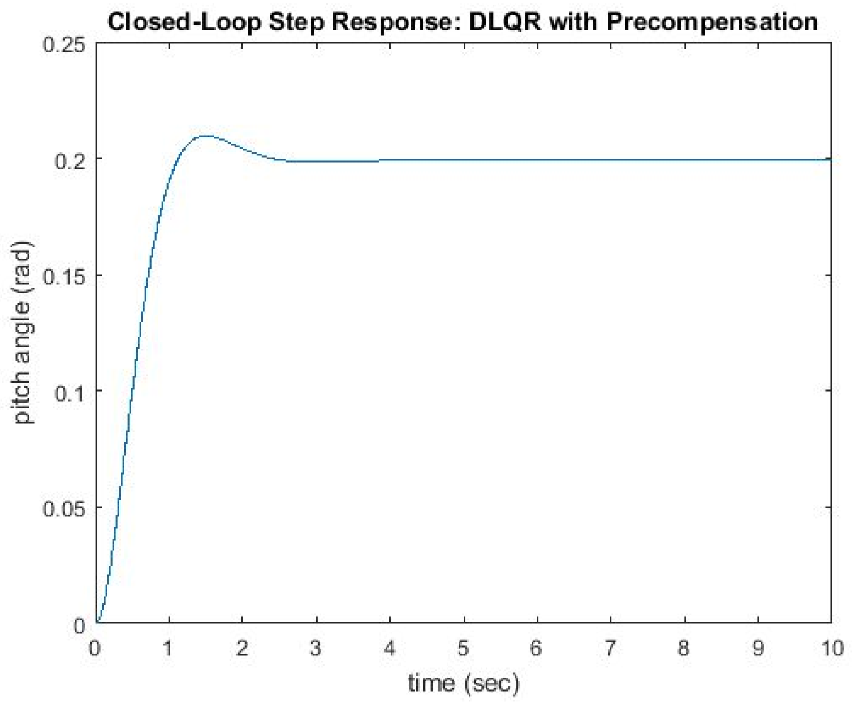
- Discrete time system equations are :

$$\mathbf{x}(k+1) = (A - BK)\mathbf{x}(k) + B\theta_{des}(k)$$
$$\theta(k) = C\mathbf{x}(k)$$

Linear quadratic regulation (LQR)

- In this digital version, we will use the discrete version of the same LQR method.
- This type of control technique optimally balances the system error and the control effort based on a cost that the designer specifies that defines the relative importance of minimizing errors and minimizing control effort.
- Rest of the factors are the same as those in continuous time domain.
- MATLAB Command dlqr which is the discrete-time version of the lqr command has been used. Both the cost matrices have been kept the same as that in continuous time domain.
- Referring to the closed-loop state equations given above assuming a control law with non-zero reference, $\delta(k) = (\theta_{des}(k) - K\mathbf{x}(k))$, we can then generate the closed-loop step response.
- Examining this step response, we will get a large steady-state error.
- We have added a pre-compensator($\overline{N}$) to scale the overall output.
- The user-defined function rscale.m to find $\overline{N}$ because it is only defined for continuous systems.

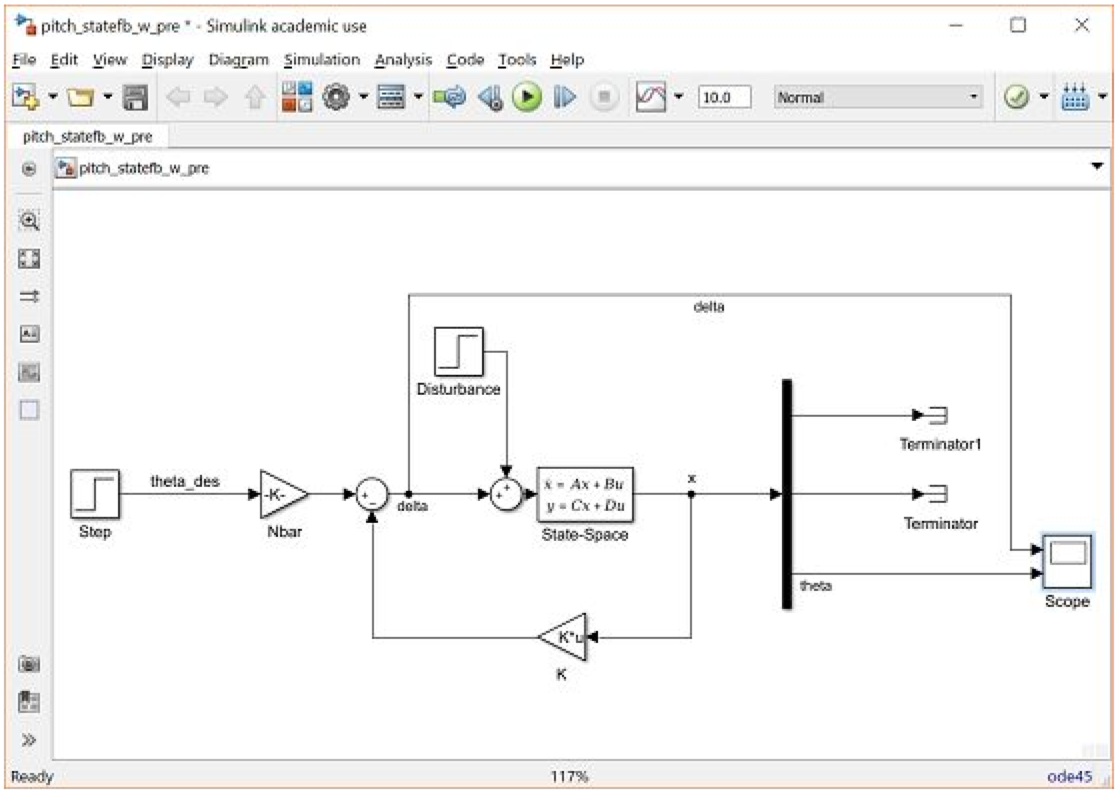
- The correct scaling by trial and error. After several trials, it was found that $\overline{N}$ equal to 6.95 provided a satisfactory response.
- After including this in the system, the modified step response looks like :



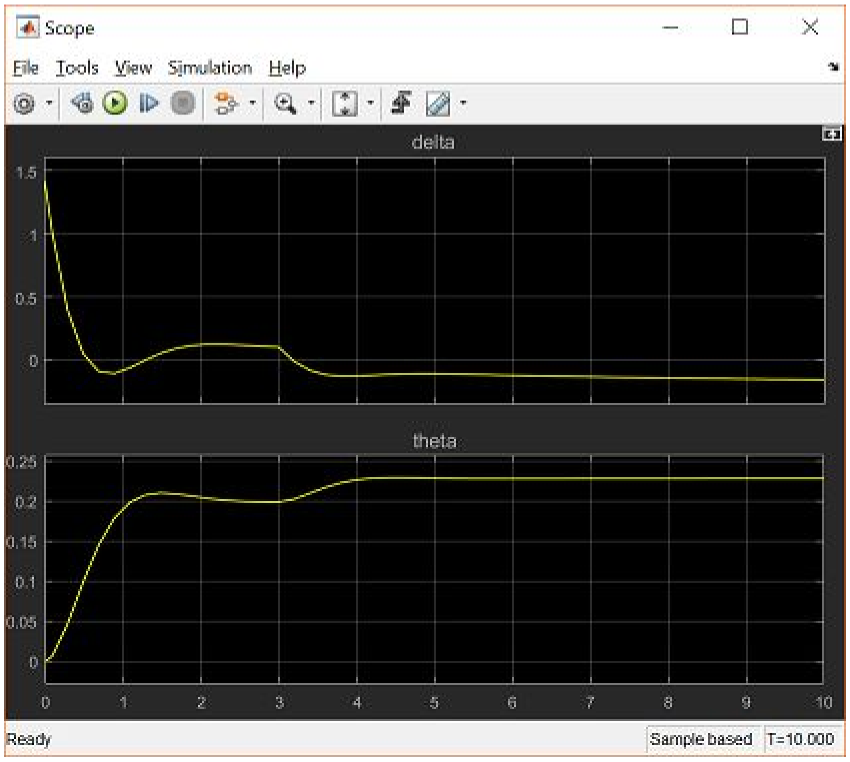
- From this plot, we see that the $\overline{N}$ factor eliminates the steady-state error. Now all design requirements are satisfied.

Simulink Modelling

- Simulink modelling has been done to verify the plots of MATLAB and the results are well in sync.
- But the main purpose of using Simulink was to model a disturbance and check the system robustness, especially with the use of a combined PI Control instead of just a pre-compensator which has been discussed below.
- Continuing from the point of discussion of combining the precompensator and the proposed integral control in LQR, the drawback of using a precompensator like the one implemented above is that it is calculated on the basis of a model of the plant and is located outside of the feedback loop such that the output of the summing junction in the above model is no longer the true error.
- Therefore, if there are errors in the model or an unknown disturbance, the precompensator will not correct for them and there will be steady-state error.
- In order to investigate this phenomenon, we have added a disturbance to our model.
- The disturbance is generated by a Step block with the **Final value** set to "0.2" and the **Step time** set to "3". The disturbance is modeled as entering the system in the same manner as the control input δ.



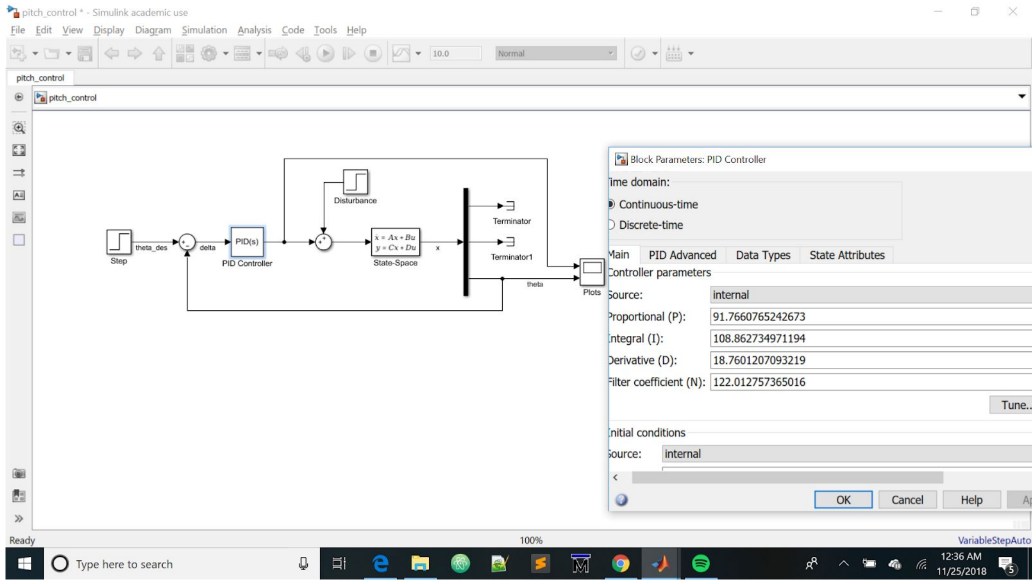
- Running the simulation will generate a response like the one shown below.



- Inspection of the above plot demonstrates how the occurrence of the disturbance at a time of 3 seconds drives the system away from the desired steady-state value of 0.2 radians and the presence of the constant precompensator is not able to correct for the effect of the disturbance.
- So we will add integral control also to it.
- **Model uncertainty** is another source of error that should be considered. For example, if the "actual" system model had a B matrix equal to [0.232*1.4 0.0203*0.6 0]', then the controller K used above would lead to an unstable system.
- This sort of result is not uncommon to controllers designed using a technique like the Linear Quadratic Regulator method.
- This makes some sense in that the control gain K was designed only to minimize the resulting error and the required control effort; other goals, such as robustness to model uncertainty, were not considered in the design.

Automated PID tuning with Simulink :

- As mentioned above, adding integral control to our compensator can help to reduce the steady-state error that arises due to disturbances and model uncertainty.
- We have implemented a PID controller assuming only the output θ is measured. Furthermore, we have used Simulink's built-in capabilities to automatically tune the PID controller.
- The resulting model should appear as follows:



Block Parameters: PID Controller

Time domain: (●) Continuous-time ( ) Discrete-time

Main | PID Advanced | Data Types | State Attributes

Controller parameters

Source: internal

Proportional (P): 91.7660765242673

Integral (I): 108.862734971194

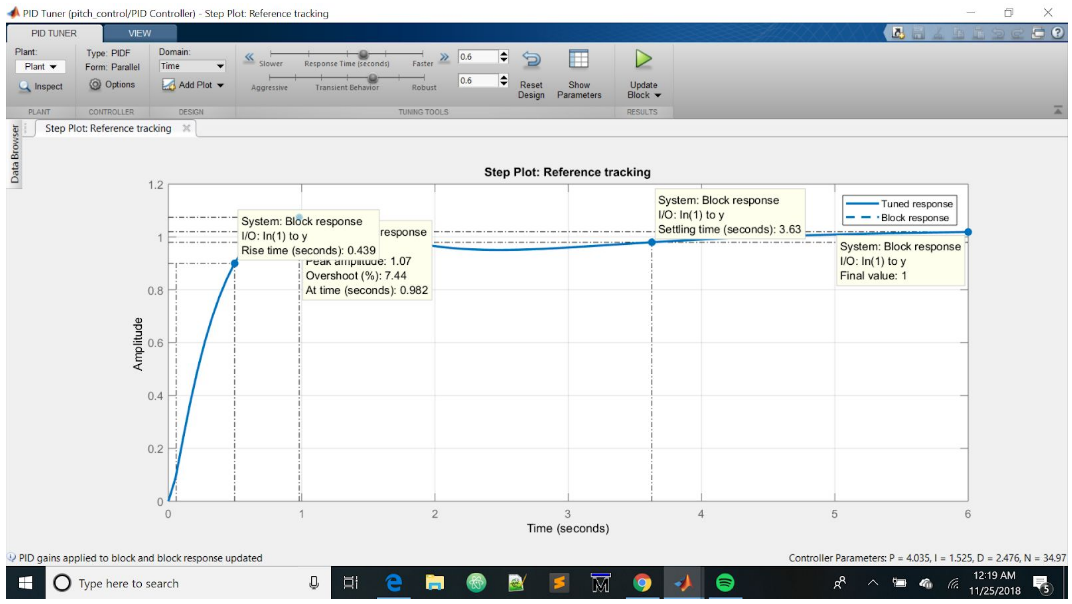
Derivative (D): 18.7601207093219

Filter coefficient (N): 122.012757365016

Initial conditions

Source: internal

- The step plot will look as follows :



System: Block response
I/O: In(1) to y
Rise time (seconds): 0.439

System: Block response
I/O: In(1) to y
Peak amplitude: 1.07
Overshoot (%): 7.44
At time (seconds): 0.982

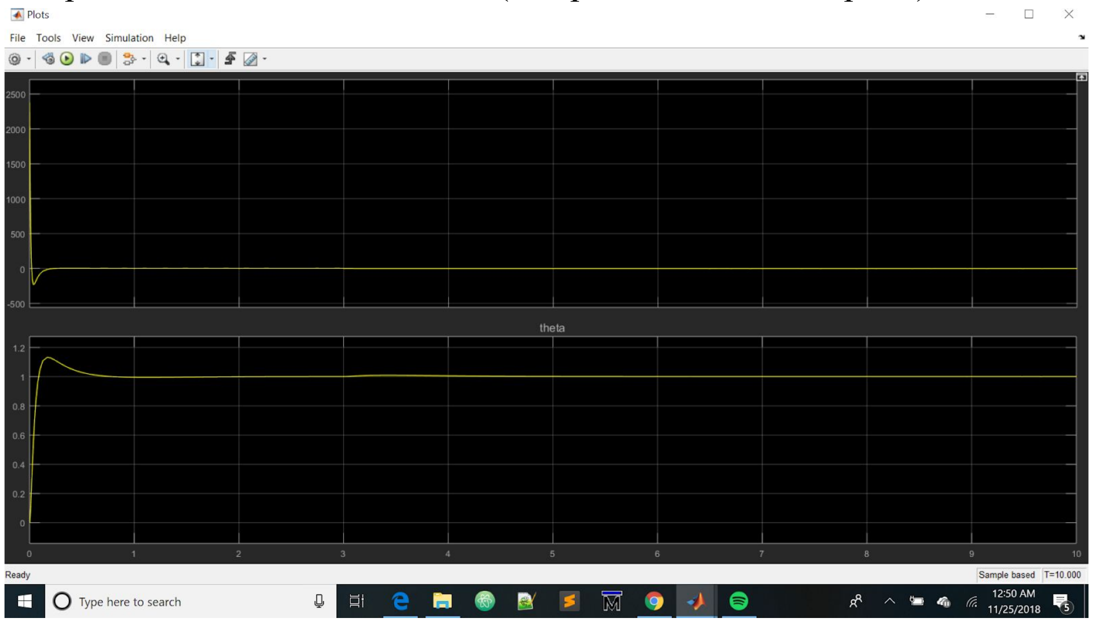
System: Block response
I/O: In(1) to y
Settling time (seconds): 3.63

System: Block response
I/O: In(1) to y
Final value: 1

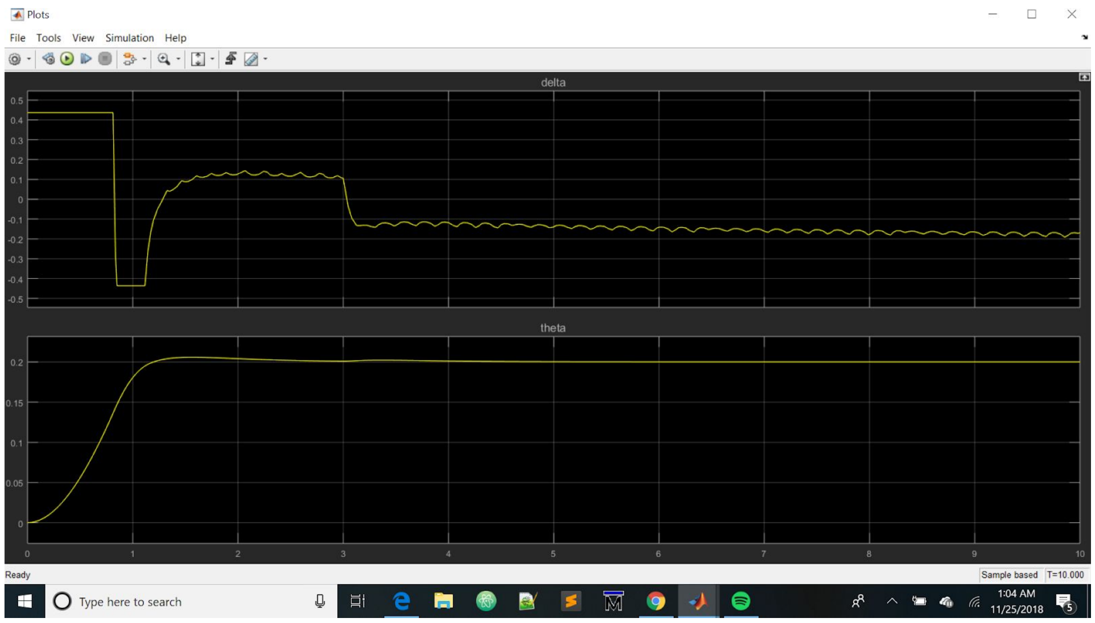
PID gains applied to block and block response updated

Controller Parameters: P = 4.035, I = 1.525, D = 2.476, N = 34.97

- The performance of the controller (comparison of delta vs pitch):



- Thus, we can observe that the disturbance is corrected for in steady-state because the PID controller we are using includes an integral term.
- In practice, it is likely that the elevator angle δ of the aircraft would be limited to something like that -25 degrees (-0.4363 rad) to +25 degrees (0.4363 rad).
- To ensure that we don’t go beyond this limit, we have set the controller to saturate.
- After choosing these options and running the simulation generates the following results which meet our requirements, even in the presence of the disturbance as shown below.



Conclusions

- We have designed a PID Controller and a Lead Controller for controlling the pitch of an airplane.
- We have used LQR to find the appropriate gain matrix.
- We have discretized the system and then have performed all the analysis for the same.
- We have worked on improving some of the common drawbacks of using pre-compensators in LQR, while also pointing out another drawback of this control technique, by using the Simulink model and during that, making our controller more robust (by testing it with the step-disturbance).

References

- http://ctms.engin.umich.edu/CTMS/index.php?example=AircraftPitch&section=SystemModeling

- https://www.youtube.com/playlist?reload=9&list=PLMrJAkhIeNNR20Mz-VpzgfQs5zrYi085m

- http://ctms.engin.umich.edu/CTMS/index.php?aux=Extras_rscale