Flight Stability and Control
Term Project Star
F-104 "Starfighter"

Efe Ozan Bozkurt
110170018

January 2, 2021

Contents

1 Introduction

In this course, as a project, every student is responsible for studying an aircraft according to the preference. This study includes creating state-space equations matrices (A and B) for longitudinal and lateral cases. With these matrices, one must create the control surface's transfer functions. Aftermath, project wants one to process these functions to create an autopilot for the aircraft, research the responses for specific inputs.

In my project, I preferred the famous F-104 Starfighter. This military type aircraft has single jet engine and capable of supersonic flight. Its development goes far back to the cold war between United States of America (U.S.A.) and Soviet Union. To be exact, first flight of this old fighter-jet is February 20, 1954. It's design requirements are shaped on current enemy fighter-jet at 1950's MiG's. USA wanted a light-weight aircraft with capable of maximum altitude and climbing performance. Photo of F-104 is below for reader to see:

In this homework, most of the mathematical process and creation of the control algorithms are studied on MATLAB. To be spesific, I used MATLAB's build-in tools, MATLAB Script, Simulink and the add-on for Simulink, DSP System Toolbox (Design and simulate streaming signal processing systems).

2 First Question

For start, question wants me to determine state-space equations of the aircraft using geometric, mass, aerodynamic characteristics. To do that, first I used course main book which is Nelson 2nd Edition. I also used National Aeronautics and Space Administration(NASA)'s technical notes.(NASA TN D-6943) I take the data as sea-level flight.

State-space system can be presented as:

$$\dot{x} = Ax + Bu$$

$$y = Cx$$

Longitudinal and Lateral state-space equations must be found seperately.

2.1 Matrices of Longitudinal Motion

According to the Nelson, longitudinal equations can be represented as:

$$\begin{bmatrix} \Delta\dot{u} \\ \Delta\dot{w} \\ \Delta\dot{q} \\ \Delta\dot{\theta} \end{bmatrix} = \begin{bmatrix} X_u & X_w & 0 & -g \\ Z_u & Z_w & u_0 & 0 \\ M_u + M_{\dot{w}}Z_u & M_w + M_{\dot{w}}Z_w & M_q + M_{\dot{w}}u_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \Delta u \\ \Delta w \\ \Delta q \\ \Delta\theta \end{bmatrix} + \begin{bmatrix} X_\delta & X_{\delta_T} \\ Z_\delta & Z_{\delta_T} \\ M_\delta + M_{\dot{w}}Z_\delta & M_{\delta_T} + M_{\dot{w}}Z_{\delta_T} \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \Delta\delta \\ \Delta\delta_T \end{bmatrix}$$

This equations are written for the inputs of elevator and thrust. But for simpilicity of the lecture and project, δ_T will be taken as zero. Therefore starfighter will show no change in its thrust. In longitudinal equations, the values to consider are change of x-direction velocity, change of z-direction velocity, change of pitch rate and lastly change of pitch angle. Values of the coefficents in the A and B matrices can be determined using Nelson's summary of derivative formulas. Can be seen below:

TABLE 4.2
Summary of longitudinal derivatives

$X_u = \frac{-(C_{D_u} + 2C_{D_0})QS}{mu_0}$	$X_w = \frac{-(C_{D_\alpha} - C_{L_0})QS}{mu_0}$
$Z_u = \frac{-(C_{L_u} + 2C_{L_0})QS}{mu_0}$	
$Z_w = \frac{-(C_{L_\alpha} + C_{D_0})QS}{mu_0}$	$Z_{\dot{w}} = C_{L_{\dot{\alpha}}} \frac{\bar{c}}{2u_0} QS/(u_0 m)$
$Z_\alpha = u_0 Z_w$	$Z_{\dot{\alpha}} = u_0 Z_{\dot{w}}$
$Z_q = C_{Z_q} \frac{\bar{c}}{2u_0} QS/m$	$Z_{\delta_e} = C_{Z_{\delta_e}} QS/m$
$M_u = C_{m_u} \frac{(QS\bar{c})}{u_0 I_y}$	
$M_w = C_{m_\alpha} \frac{(QS\bar{c})}{u_0 I_y}$	$M_{\dot{w}} = C_{m_{\dot{\alpha}}} \frac{\bar{c}}{2u_0} \frac{QS\bar{c}}{u_0 I_y}$
$M_\alpha = u_0 M_w$	$M_{\dot{\alpha}} = u_0 M_{\dot{w}}$
$M_q = C_{m_q} \frac{\bar{c}}{2u_0} (QS\bar{c})/I_y$	$M_{\delta_e} = C_{m_{\delta_e}} (QS\bar{c})/I_y$

Using the information given above, A and B matrices can be created easily. Using these definitions, coefficents are calculated in MATLAB. The coefficents are calculated as:

$X_u = -0.0683$	$X_w = 0.037$	$Z_u = -0.1909$	$Z_w = -0.4809$
$Z_{\dot{w}} = 0$	$Z_\alpha = -137.9857$	$Z_{\dot{\alpha}} = 0$	$Z_q = 0$
$Z_{\delta_e} = -25.339$	$M_u = 0$	$M_w = -0.0069$	$M_{\dot{w}} = -0.00028525$
$M_\alpha = -1.9671$	$M_{\dot{\alpha}} = -0.0818$	$M_q = -0.2967$	$M_{\delta_E} = -4.487$

Now, these numbers must used to create A and B matrices. Characteristic matrices can be written as:

$$A = \begin{bmatrix} -0.0683 & 0.037 & 0 & -32.2 \\ -0.1919 & -0.4809 & 286.9148 & 0 \\ 5.4459.10^{-5} & -0.0067 & -0.3785 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

$$B = \begin{bmatrix} 0 \\ -25.3990 \\ -4.4801 \\ 0 \end{bmatrix}$$

Therefore, longitudinal state-space equation has been determined. It can be used to calculate response of the plane for certain input of elevator. As I mentioned before, thrust is invalid in this case.

2.2 Matrices of Lateral Motion

Now, for Starfighter, state-space for lateral motion must be determined to achieve full control. For this, I used Nelson's book as a source. A and B matrices for lateral motion can be presented as:

$$\begin{bmatrix} \Delta\dot{\beta} \\ \Delta\dot{p} \\ \Delta\dot{r} \\ \Delta\dot{\phi} \end{bmatrix} = \begin{bmatrix} \frac{Y_\beta}{u_0} & \frac{Y_p}{u_0} & -\left(1 - \frac{Y_r}{u_0}\right) & \frac{g\cos\theta_0}{u_0} \\ L_\beta & L_p & L_r & 0 \\ N_\beta & N_p & N_r & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} \Delta\beta \\ \Delta p \\ \Delta r \\ \Delta\phi \end{bmatrix} + \begin{bmatrix} 0 & \frac{Y_{\delta_r}}{u_0} \\ L_{\delta_a} & L_{\delta_r} \\ N_{\delta_a} & N_{\delta_r} \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \Delta\delta_a \\ \Delta\delta_r \end{bmatrix}$$

In these matrices, sideslip angle, yaw rate, rolling rate and roll angle are taken as outputs. To control these quantities, pilot must use aileron and rudder. Coefficents for the matrices are calculated using the formulas Nelson described:

TABLE 5.1
Summary of lateral directional derivatives

$Y_\beta = \frac{QSC_{y_\beta}}{m}$ (ft/s² or m/s²) $N_\beta = \frac{QSbC_{n_\beta}}{I_z}$ (s^{-2}) $L_\beta = \frac{QSbC_{l_\beta}}{I_x}$ (s^{-2})

$Y_p = \frac{QSbC_{y_p}}{2mu_0}$ (ft/s) or (m/s) $N_p = \frac{QSb^2C_{n_p}}{2I_zu_0}$ (s^{-1})

$L_p = \frac{QSb^2C_{l_p}}{2I_xu_0}$ (s^{-1})

$Y_r = \frac{QSbC_{y_r}}{2mu_0}$ (ft/s) or (m/s) $N_r = \frac{QSb^2C_{n_r}}{2I_zu_0}$ (s^{-1})

$L_r = \frac{QSb^2C_{l_r}}{2I_xu_0}$ (s^{-1})

$Y_{\delta a} = \frac{QSC_{y_{\delta_a}}}{m}$ (ft/s²) or (m/s²) $Y_{\delta_r} = \frac{QSC_{y_{\delta_r}}}{m}$ (ft/s²) or (m/s²)

$N_{\delta a} = \frac{QSbC_{n_{\delta_a}}}{I_z}$ (s^{-2}) $N_{\delta r} = \frac{QSbC_{n_{\delta_r}}}{I_z}$ (s^{-2})

$L_{\delta a} = \frac{QSbC_{l_{\delta_a}}}{I_x}$ (s^{-2}) $L_{\delta r} = \frac{QSbC_{l_{\delta_r}}}{I_x}$ (s^{-2})

I calculated all the quantities in MATLAB. Quantities for the lateral motion can be seen in the table below:

$Y_\beta = -43.5980$	$Y_p = 0$	$N_\beta = 3.4679$	$N_p = -0.0371$	$L_\beta = -20.4071$
$L_p = -1.2707$	$Y_r = 0$	$N_r = -0.1989$	$L_r = 1.1815$	$Y_{\delta_a} = 0$
$Y_{\delta_r} = 7.7508$	$N_{\delta_a} = 0.0291$	$N_{\delta_r} = -1.1097$	$L_{\delta_a} = 4.5479$	$L_{\delta_r} = 5.2475$

After the finding the values of these quantities, A and B matrices for lateral motion can be created:

$$A = \begin{bmatrix} -0.1520 & 0 & -1 & 0.1122 \\ -20.4071 & -1.2707 & 1.1815 & 0 \\ 3.4679 & -0.0371 & -0.1989 & 0 \\ 0 & 1 & & 0 \end{bmatrix}$$

$$B = \begin{bmatrix} 0 & 0.0270 \\ 4.5479 & 5.2475 \\ 0.0291 & -1.1097 \\ 0 & 0 \end{bmatrix}$$

2.3 Modes of the Motions

This study introduced all the motion matrix for the Starfighter so far. Now eigenvalues of the matrices must be found for the mode determination. It will be done in two parts seperately: longitudinal motion, lateral motion.

2.3.1 Longitudinal

For longitudinal motion, eigenvalues can be found with the simple calculation for A matrix. But there is two approximation methods to use. These methods are described below:
–Long Approximation(Phugoid): a gradual interchange of potential and kinetic energy about the altitude and airspeed(Nelson p.152). In this approximation change in the angle of attack is taken as ratio of change in the z-direction velocity and flight velocity. As change in the angle of attack goes to zero, change in the z-direction velocity also must go to zero.

$$\Delta\alpha = \frac{\Delta w}{u_0}$$

Therefore the state-space system changes character:

$$\begin{bmatrix} \Delta\dot{u} \\ \Delta\dot{\theta} \end{bmatrix} = \begin{bmatrix} X_u & -g \\ \frac{-Z_u}{u_0} & 0 \end{bmatrix} \begin{bmatrix} \Delta u \\ \Delta\theta \end{bmatrix}$$

– Short Mode: This approximation takes place when change in the x-direction velocity is taken as zero. Therefore the matrix system changes character:

$$\begin{bmatrix} \Delta\dot{w} \\ \Delta\dot{q} \end{bmatrix} = \begin{bmatrix} Z_w & u_0 \\ M_w + M_{\dot{w}}Z_w & M_q + M_{\dot{w}}u_0 \end{bmatrix} \begin{bmatrix} \Delta w \\ \Delta q \end{bmatrix}$$

Firstly, simple calculation of the eigenvalues will take place. To do this operation, we must use the equation below:

$$|\lambda I - A| = 0$$

In MATLAB, there is a function called eig() which do this operation automatically. With use of this function, eigenvalues of the A matrix can be found as:

$$\lambda_{phugoid} = -0.0311 \pm 0.1382i$$

$$\lambda_{short} = -0.4328 \pm 1.3836i$$

Now, to do a comparision, approximations will be used to find eigenvalues of the A matrix. In MATLAB, I created the matrices given above in the definitions. Calculating the eigenvalues of these matrices with the function eig() gave me the answer for the λ's.

$$\lambda_{phugoid} = -0.0342 \pm 0.1423i$$

$$\lambda_{short} = -0.4297 \pm 1.3875i$$

There is a considerable similarity between exact solution and the approximated solution for the eigenvalues. With these eigenvalues I can determine the natural frequency and damping ratio. For phugoid mode, natural frequency and daming ratio are formulated as:

$$\omega_{n_p} = \sqrt{\frac{-Z_u g}{u_0}}$$

$$\zeta_p = \frac{-Xu}{2\omega_{n_p}}$$

Using these formulas, phugoid mode natural frequency and damping is determined as:

$$\omega_{n_p} = 0.1464(rad/s)$$

$$\zeta_p = 0.005$$

For short period, natural frequency and damping ratio is formulated as:

$$\omega_{n_{sp}} = \sqrt{\frac{Z_\alpha M_q}{u0} - M_\alpha}$$

$$\zeta_{sp} = -\frac{M_q + M_{\dot{\alpha}} + \frac{Z_\alpha}{u0}}{2\omega_{n_{sp}}}$$

Using these formulas, short period mode natural frequency and damping ratio is determined as:

$$\omega_{n_{sp}} = 1.4525(rad/s)$$

$$\zeta_{sp} = 0.2959$$

2.3.2 Lateral

In lateral motion of an aircraft, there is three modes. First one is spiral mode. This mode is a slowly convergent/divergent motion. Characteristic root for this mode can be found with:

$$\lambda_{spiral} = \frac{L_\beta N_r - L_r N_\beta}{L_\beta}$$

Stability condition for this root is to be greater than zero. Which this statement can be produced: "Increasing the L_β (Dihedral Effect) have an effect of stabilization on the aircraft."

For the Starfigter, spiral mode root is found as:

$$\lambda_{spiral} = 0.0019$$

Next mode is the roll mode. This mode is a highly convergent motion. Approximation is made with single degree of freedom rolling motion. Therefore root of the motion is:

$$\lambda_{roll} = -\frac{1}{\tau} = L_p$$

And for the Starfighter, $L_p = -1.2707$. Therefore

$$\lambda_{roll} = -1.2707$$

The last mode of the lateral motion is dutch roll. This roll is under effect of primarily yawing motion and sideslip. Therefore we configure the A matrix of the lateral motion for this motion. A matrix becomes:

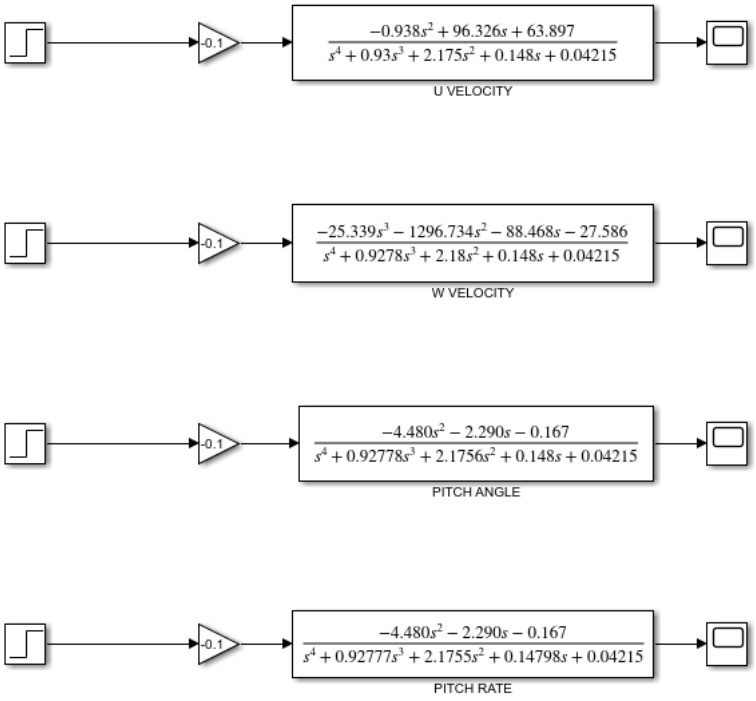
$$\begin{bmatrix} \Delta\dot{\beta} \\ \Delta\dot{r} \end{bmatrix} = \begin{bmatrix} \frac{Y_\beta}{u0} & -(1 - \frac{Y_r}{u0}) \\ N_\beta & N_r \end{bmatrix} \begin{bmatrix} \Delta\beta \\ \Delta r \end{bmatrix}$$

Eigenvalues of this matrix will give me wanted roots for the dutch roll. Eigenvalues which are the roots of the Dutch roll are:

$$\lambda_{dutch} = -0.1754 \pm 1.82621i$$

2.4 Input of δ_e=-0.1

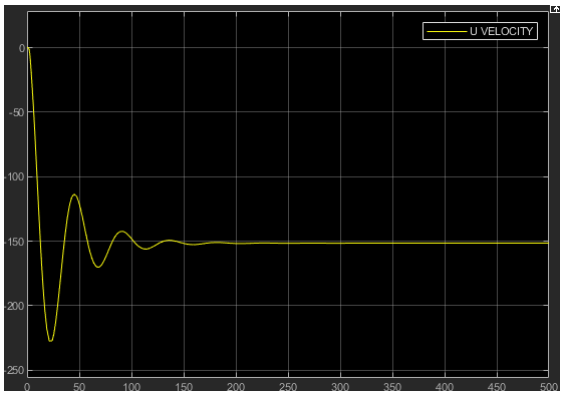
In this section of the question, there is a input of elevator into the Starfighter. Value of the input is -0.1. I have to create the transfer functions for the longitudinal equations from state-space system and after that I have to simply give an input signal to the transfer functions. This process is done by SİMULİNK. Transfer functions are found by the build-in MATLAB function ss2tf(). Code can be investigated at the Appendix A. After the investigation of the transfer function, I created a simulation for the transfer functions of Longitudinal motion quantities which are U velocity, W velocity, pitch rate and pitch angle. I simply added a step input and multiplied it with the gain of -0.1 to get a step input which has the value of -0.1. The simulation process can be seen below:



I ran the simulation for 500 seconds. I can see the responses on the "scope" element of the simulation. These responses are natural responses therefore there is no other effect of control. This study will investigate each response of the longitudinal motion element one by one under the subsection of this section.

2.4.1 Response of U Velocity

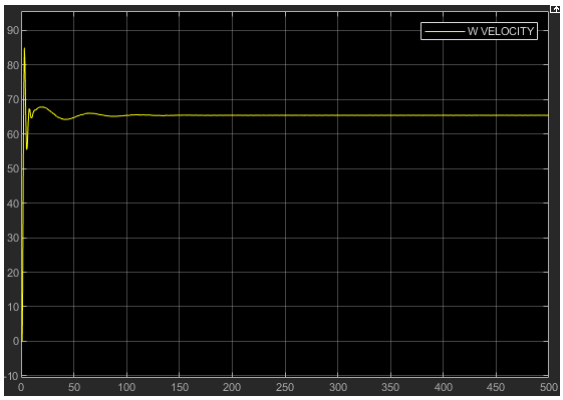
Scope of the simulation for U velocity is presented below:



After the elevator input, x-direction velocity had a significant drop. After the first drop, the damping effect did a job to settle the velocity down at the y-axis value of approximately -150. However the settling time is actually pretty high for a military type aircraft "Starfighter". It takes approximately 180 seconds to settle.

2.4.2 Response of W Velocity

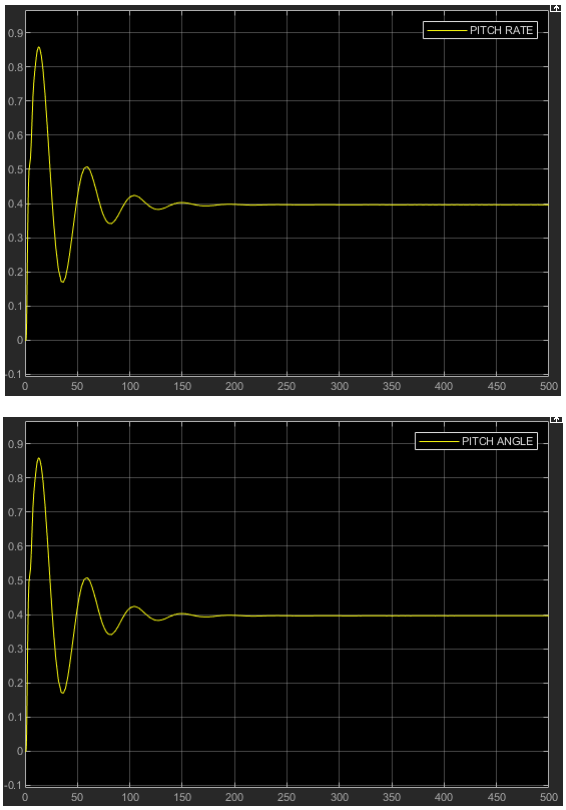
Scope of the simulation for W Velocity is presented below:



This scope has pretty high potential for Starfighter. From this scope, I can say that the settling time is pretty low after the of elevator compared to the U velocity.

2.4.3 Response of Pitching

Scope of the simulation for pitch rate and pitch angle are presented below:

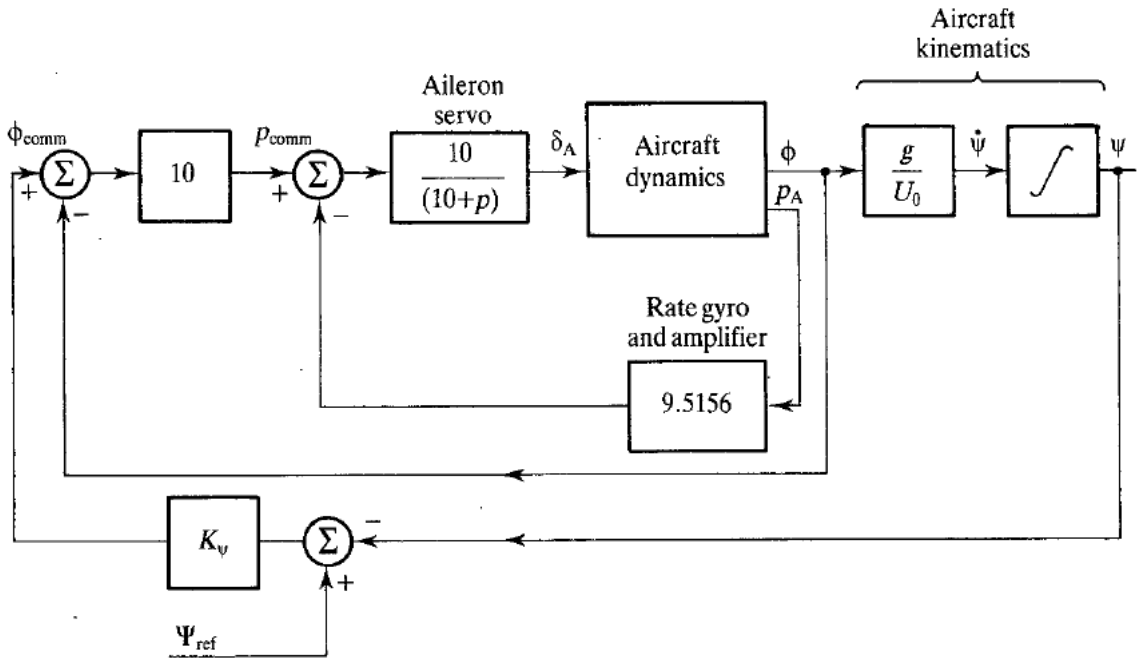


Because of the perturbation that created by the input of elevator which has the value of -0.1 along the y-axis, pitch rate and pitch angle are important. Without any control mechanicsm, pitch rate and pitch angle disturbation can be seen above. They are highly similar.

3 Second Question

3.1 Design of Autopilot

In this part of the study, I have to develop an autopilot for the Starfighter. Homework gave me the permission for the choosing the motion type. I chose the longitudinal motion to create an autopilot. This autopilot have the capacity of stabilizing the u-velocity, angle of attack, pitch rate and pitch angle. Homework also gave me a source to learn how to create an autopilot. This source is McLean's Automatic Flight Control Systems. In the source, autopilot developing method is done by inner-outer loops. This method can be shown as:



As you can see in the figure, this control system includes two loops. Inner loop has the servo controller. Its gain value is determined by root locus of the transfer function multiplied with servo. After the determining the gain for inner loop, outer loop gain can be found with the same method. This solution method is applied to every element of the longitudinal motion. To create longitudinal elements transfer functions, Nelson's approach is used. This approach is simply a approximation for the coefficients of transfer function which can be seen below:

TABLE 8.2
Short-period transfer function approximations

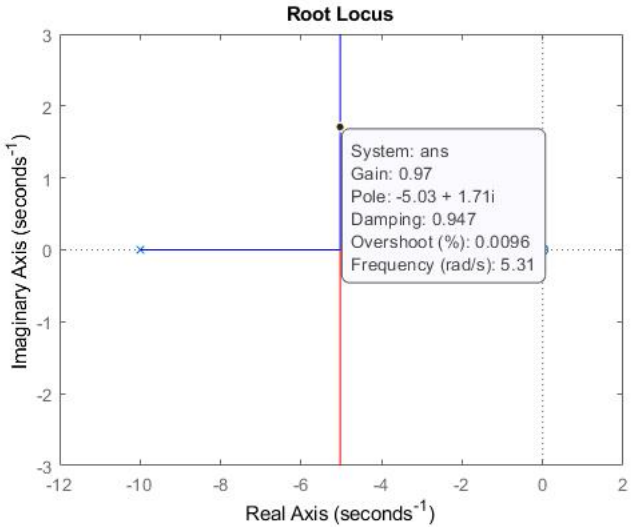
	A, A_α, or A_q	B, B_α, or B_q	C
$\Delta_{sp}(s)$	1	$-(M_q + M_{\dot{\alpha}} + Z_\alpha/u_0)$	$Z_u M_q/u_0 - M_\alpha$
$N^q_{\delta_e}(s)$	Z_{δ_e}/u_0	$M_{\delta_e} - M_q Z_{\delta_e}/u_0$	
$N^q_{\delta_e}(s)$	$M_{\delta_e} + M_{\dot{\alpha}} Z_{\delta_e}/u_0$	$M_\alpha Z_{\delta_e}/u_0 - M_{\delta_e} Z_\alpha/u_0$	

TABLE 8.3
Long-period transfer function approximations

	A, A_u, or A_θ	B, B_u, or B_θ	C
$\Delta_p(s)$	1	$-X_u$	$-Z_u g/u_0$
$N^u_{\delta_e}(s)$	X_{δ_e}	gZ_{δ_e}/u_0	
$N^\theta_{\delta_e}(s)$	$-Z_{\delta_e}/u_0$	$X_u Z_{\delta_e}/u_0 - Z_u X_{\delta_e}/u_0$	

3.1.1 U-Velocity IO Loop

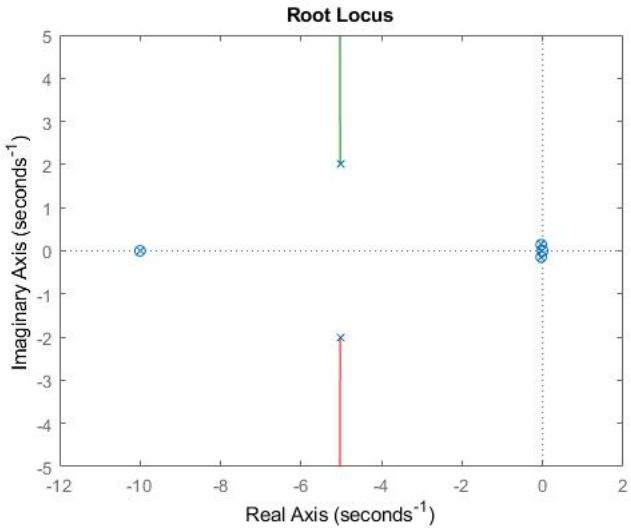
First case for my autopilot is x-directional velocity. First thing to do is plot the root locus of the transfer function of the transfer function of u-velocity multiplied by controller. Transfer function of the U-velocity is created with the approximation in the table 8.3. Plot is shown below:



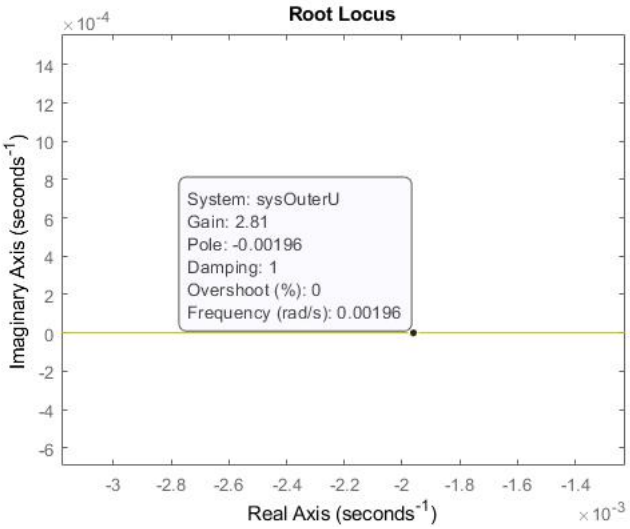
There is a marked point in the plot. This point is a stable point with the gain of 0.97. This gain value is taken for the inner loop gain. Not only because root-locus, I also tried experimental studies on the transfer function with the investigation of the response. Therefore $K_{inner} = 0.97$. Second case is the outer loop gain. This cases transfer function will be found with the feedback of the innerloop. Therefore:

$$tf_{outer} = \frac{tf_{inner}}{1 + K_{inner} tf_{inner}}$$

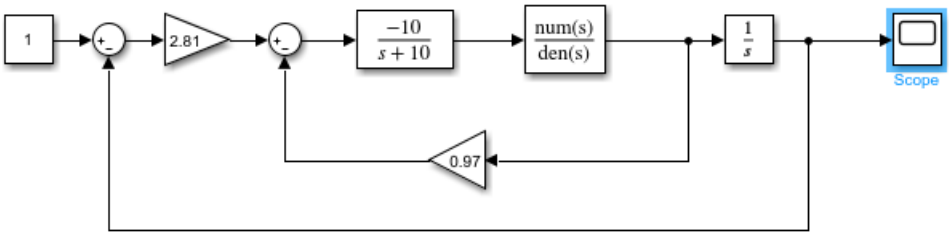
With this method, root locus for the outer loop became:



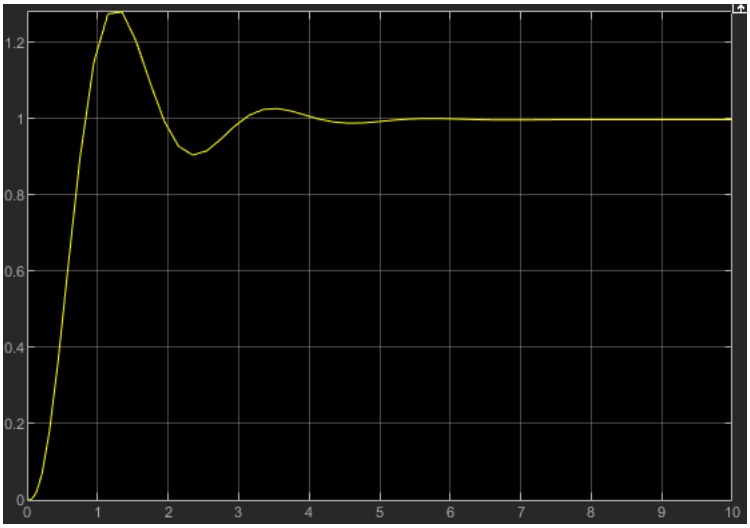
To find the gain, I have to investigate the part near the zero. In this part there is stable gain value which is 2.81. Its zoomed version is shown below:



Thus, I found every gain value I need to develop an autopilot. You can see the simulink version of the autopilot below:



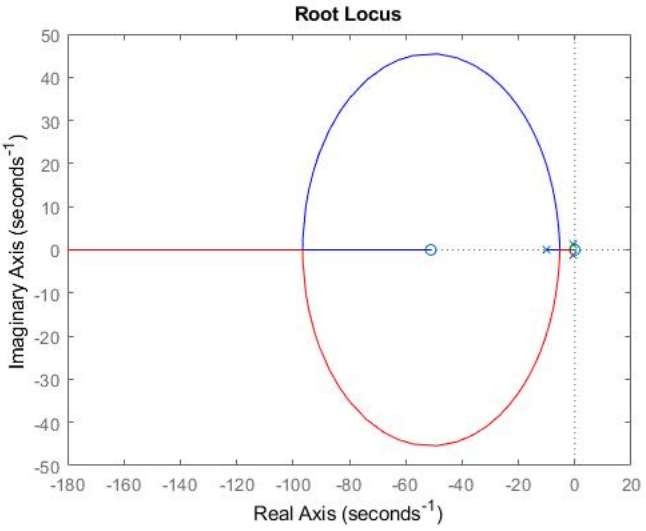
Lastly, scope can be seen below after the execution of the simulation given above:



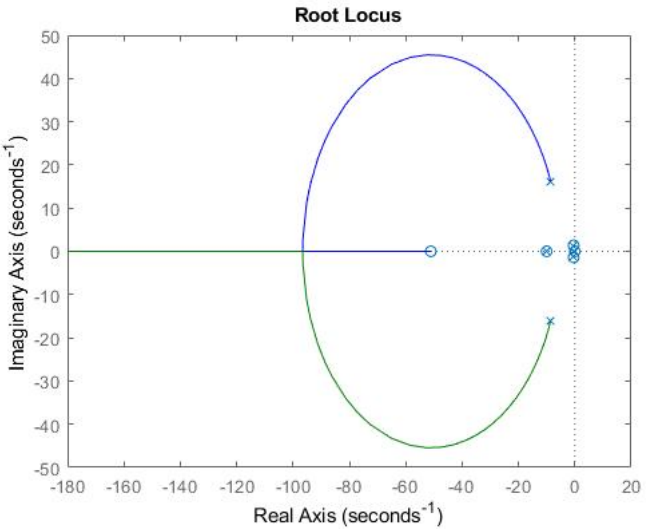
Just because F-104 Starfighter is the under the scope of this study, autopilot must give a short time stabilization. Overshoot is not the primary condition to satisfy. As can be seen in the scope, its x-directional velocity stabilizes at the 5.5 seconds mark. It is not the best autopilot result but under the hard situations, still it can help the pilot to survive.

3.1.2 Angle of Attack IO Loop

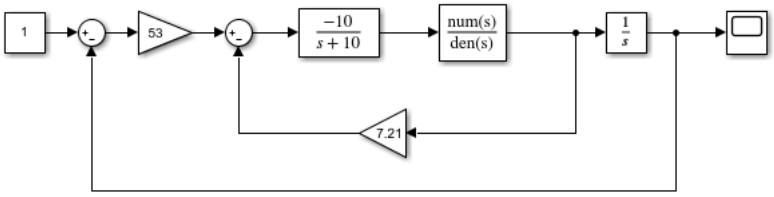
Now, the same process that done with the u-velocity must be done for angle of attack. I created the transfer function with the approximation of the Nelson. First I have investigate for the inner gain value, investigation starts at the root locus of the controller*system. Root locus of the inner loop can be seen below:



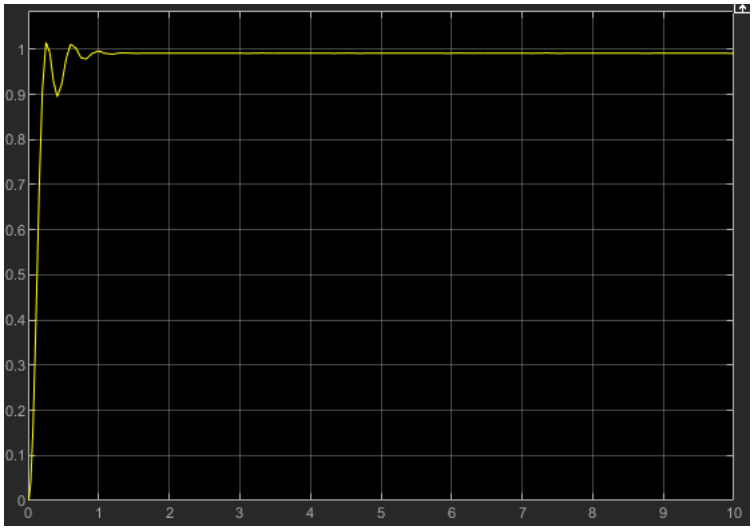
From this plot, I can choose the gain value for my inner loop. I chose K_{inner} = 7.21 which has approximately damping = 0.47. After the choosing the K_{inner}, I can use this value to investigate outer gain value. This will be found with the root locus of the outer loop. This methods are all the same with the methods I used in U velocity autopilot process. Root locus for the outer loop of the angle of attack can be seen below:



From this plot, I choose the K_{outer} as 58 which has the damping value as 0.63. All the gain values are chosen therefore I can create the simulation with the chosen gain values and create an inner-outer loop to control angle of attack. Created simulation can be seen below:



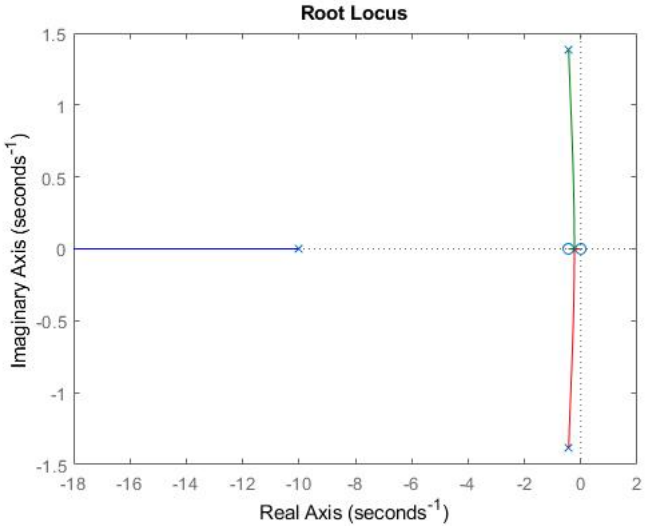
When I run the simulation for 10 seconds, from scope, I can see the response of my autopilot. Plot can be seen below:



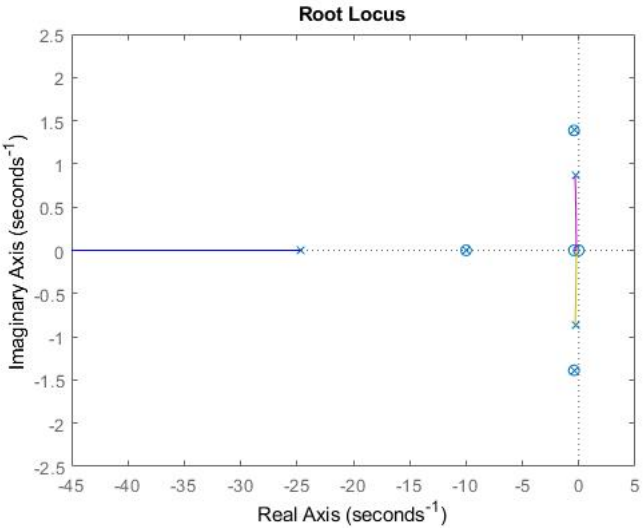
Plot actually came out good. For an military type aircraft like Starfighter, requirement for the any control is to be quick as possible. Angle of attack becomes stable in 1.5 seconds approximately. So I can say that, this autopilot works well.

3.1.3 Pitch Rate IO Loop

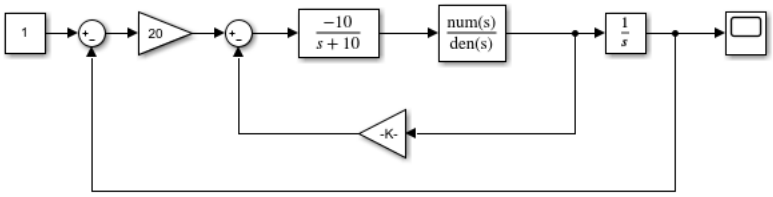
Transfer function is created by the approximation of the Nelson. After the multiplication with the controller transfer function, root locus was investigated for the K_{inner}. Root locus of the inner loop is shown below:



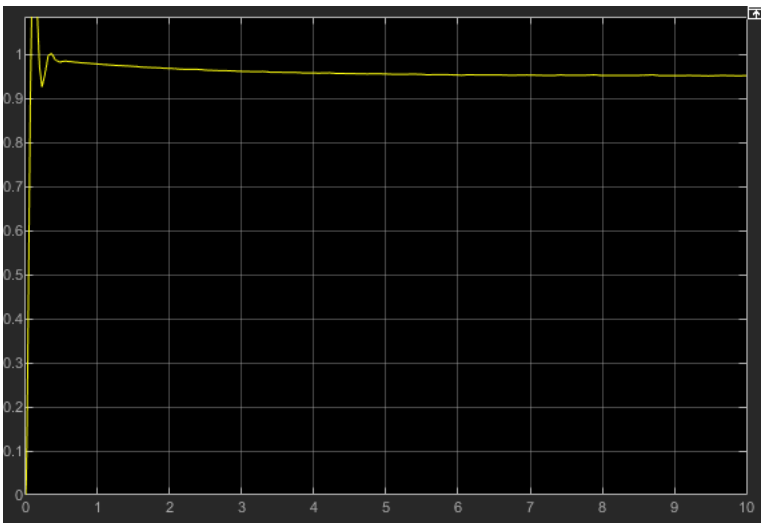
From this root locus, I took the gain as 0.303. It is in the stable area. Therefore I designed outer loop transfer function and plotted the outer loop root locus as shown below:



From this loop, I took the outer loop gain K_{outer} as the value 20. Therefore my autopilot design is completed. Here is the simulation:



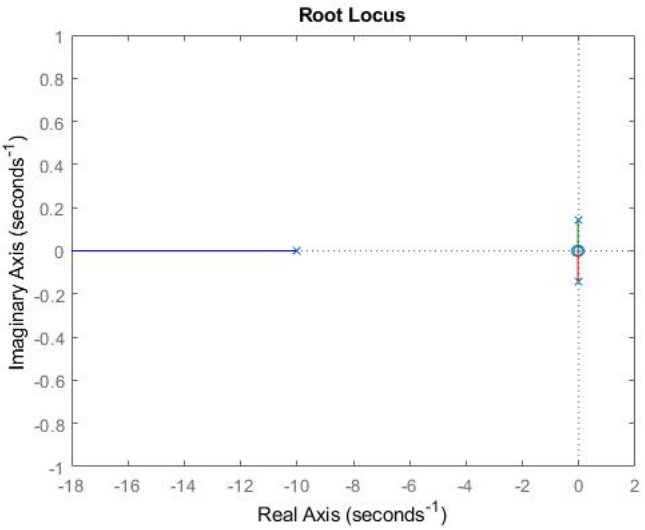
And after the execution for 10 seconds, scope from the simulation is shown below:



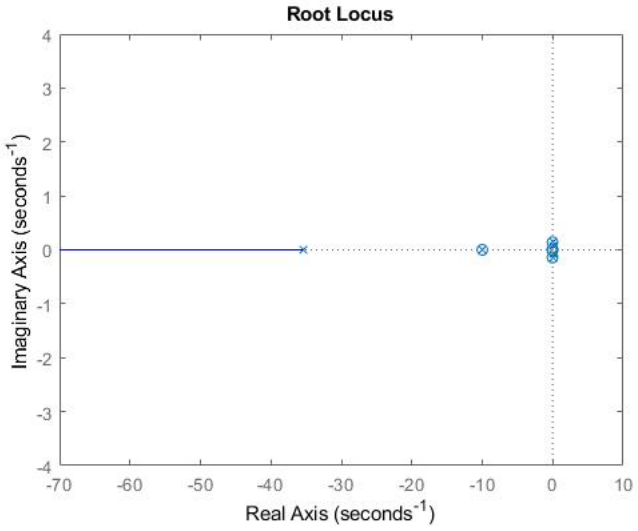
It has pretty quick response but steady state error is high. It is not the perfect autopilot but it has quick reflexes, therefore in the tough situations, It might do the job well.

3.1.4 Pitch Angle "θ" IO Loop

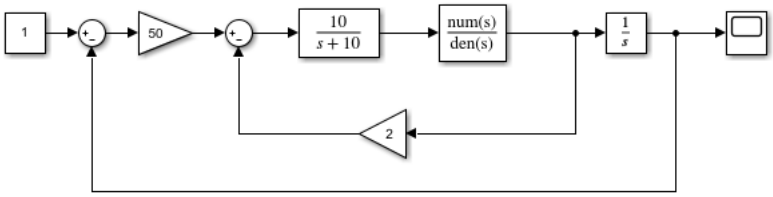
After the creation of the transfer functions with the approximation of the Nelson's, I created the root locus to investigate K_{inner} for the pitch angle.



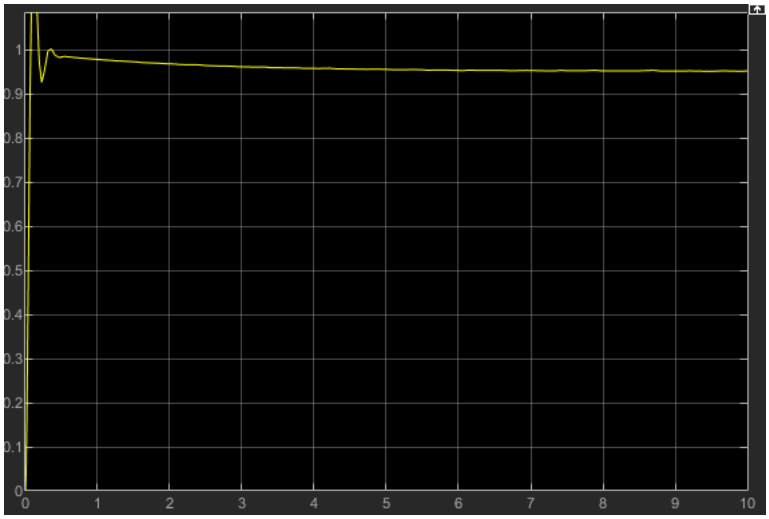
This was the hardest root locus to research because of the limited stable lines. But after the investigation, I decided the value of K_{inner} as 2. Putting this value into the outer loop system to plot root locus and than plotting the root locus:



In the small area in the origin has a potential to find the K_{outer} value. After the investigation of this area, I came up with the value of $K_{outer} = 50$. Now, I can build an autopilot IO Loop for the θ angle. Here is the built simulation: [1].



Running the simulation for 10 seconds will give me output of the pitch rate which is shown below:



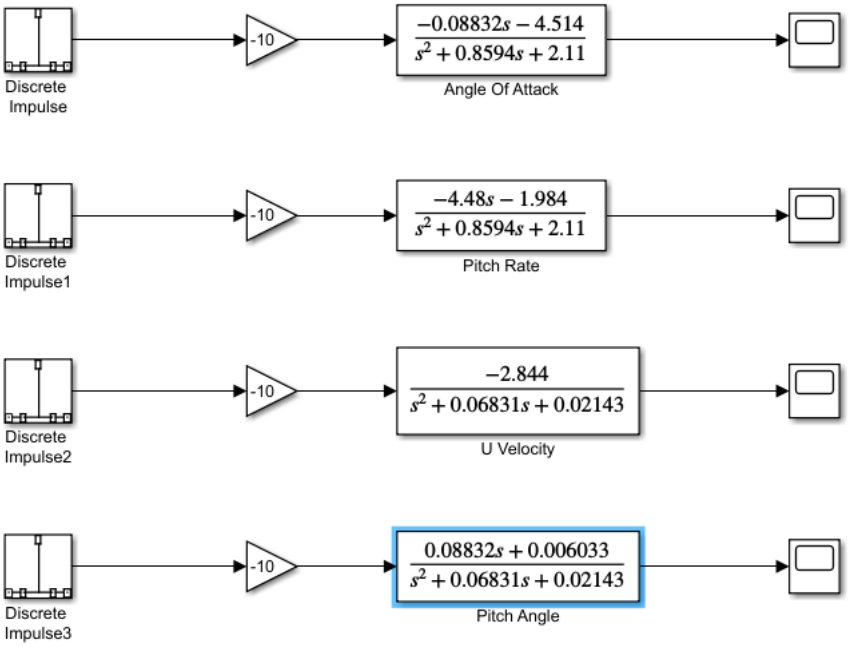
It has quick response with large steady-state error. It is not the best autopilot, but it can work in hard/tough situations.

In this part of the study, I designed an autopilot for the F-104 Starfighter. Generally, my autopilot conclusions are pretty good but can be improved with experimental studies. These experimental studies can be for the data for aerodynamic coefficents, stability derivatives etc.

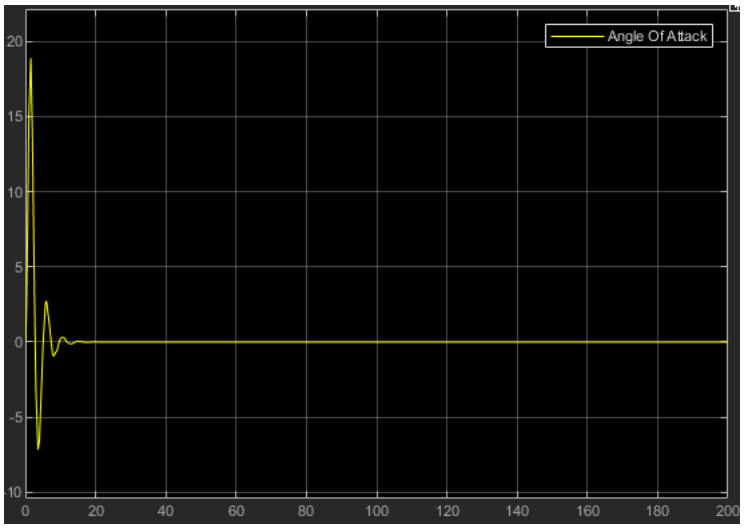
[1] In the other controllers, I used the servo as $\frac{-10}{s+10}$. But in this case this controller gave me uncontrollable output. Therefore I used servo as $\frac{10}{s+10}$

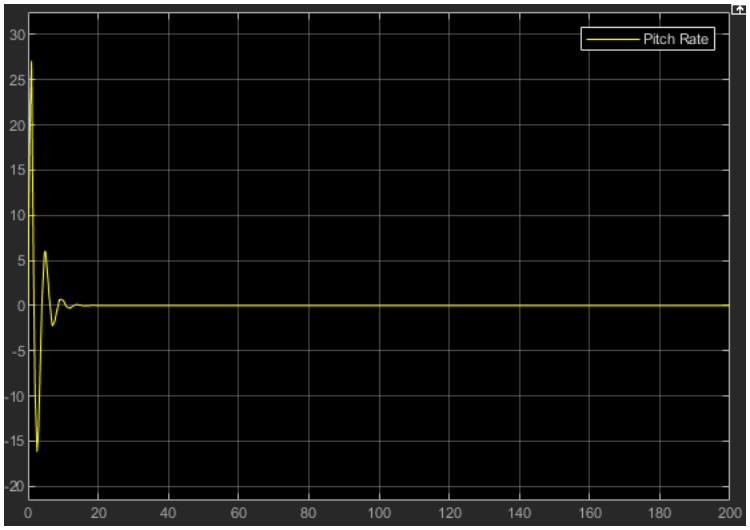
3.2 Time Responses for -10°

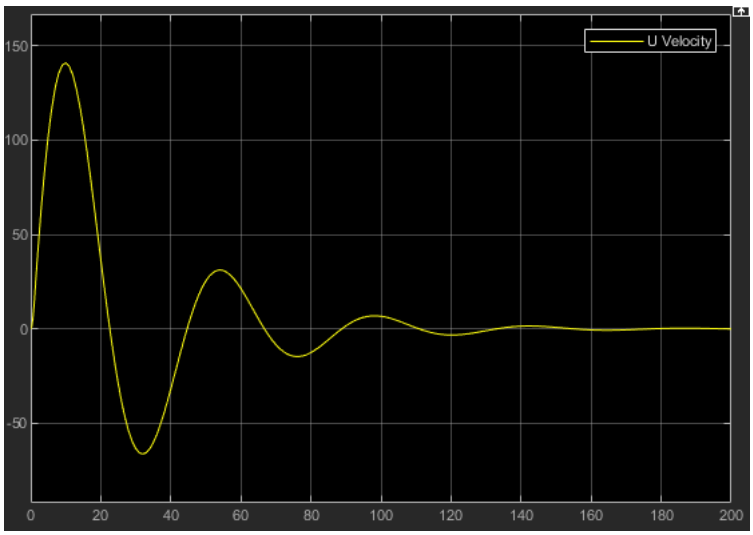
In the last part of the study, I have to create an impulse of −10° and insert it to the longitudinal system functions. First I created a simulation with the transfer functions and impulse. It can be seen below:

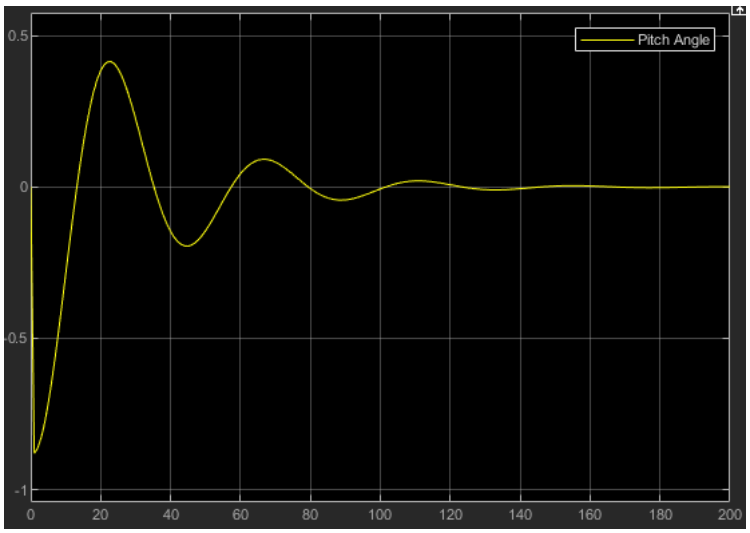


Transfer functions of the longitudinal elements can be seen. These transfer functions are created with Nelson's approximations and used in the previous part of the study. After the execution of the simulation for 200 seconds. My plots are in the scope part. Here are the plots for the elements:









In the first plot which is the angle of attack plot, angle of attack stabilizes really quick. There is no observable steady-state error. It took approximately 15 seconds to reduce the effect of perturbation. In the pitch rate plot, it took approximately 15 seconds to reduce the effects and there is no visual steady-state error. In the plots for U velocity and Pitch angle, the effects are long-timed compared to the others. It took 130 seconds to reduces effectively. There is no visual steady-state error but it takes too long for a military type aircraft.

4 Conclusion

In this study, I worked with the famous aircraft F-104 Starfighter. I created the longitudinal and lateral state-space equations and matrices. I investigate to roots for the certain modes of the motions. I used the data from the sources to create an autopilot for the Starfighter. Lastly, I tested the design for an given perturbation. The results are not the perfect for the industry to use but it can be improved to that goal. In the aviation industry, controls have significant role. They must be integrated to the aircraft perfectly for an safe flight(for civil aviation) and high maneover ability(for military aviation). Because of my study area, which is F104, responses that I got as a result are quick but as I mentioned above it can be quicker. As the technology of aviation improves, there will be a new control types. For example control supported with artificial intelligence is a new technology that engineers and scientists are trying to integrate. It will be a huge relief for pilots. Maybe even it can make them to lose their jobs.

5 Appendix A
MATLAB Code of Creation of Matrices/TFs

```
%% TRANSFER FUNCTION CREATION

    W = 16300; %lb
    Ix = 3549; %slug.ft^2
    Iy = 58611; %slug.ft^2
    Iz = 59669; %slug.ft^2
    Ixz = 0;

    %LONGITUDINAL(Sea Level)
    g = 32.2;
    Mach = 0.257  ;
    ro = 0.002337;

    Cl = 0.735;
    Cd = 0.263;
    Cd_u = 0;
    Cl_a = 3.44;
    Cd_a = 0.45;
    Cm_a = -0.64;
    Cl_adot = 0;
    Cm_adot = -1.6;
    Cl_q = 0;
    Cm_q = -5.8;
    Cl_M = 0;
    Cd_M = 0;
    Cm_M = 0;
    Cl_deltaE = 0.68;
    Cm_deltaE = -1.46;

    %LATERAL(Sea Level)
    Cy_B = -1.17;
    Cl_B = -0.175;
    Cn_B = 0.5;
    Cl_p = -0.285;
    Cn_p = -0.14;
    Cl_r = 0.265;
    Cn_r = -0.75;
    Cl_deltaa =0.039;
    Cn_deltaa =0.0042;
    Cy_deltar =0.208;
    Cl_deltar =0.045;
    Cn_deltar =-0.16;

    S = 196.1;
    b = 21.94;
    c = 9.55;
    u0 = Mach * 1116.4;
    m = W/g;
    Q = 1/2 * ro * u0^2;

    %MATRIX VALUE CALCULATION
    %LONGITUDINAL
    Xu = -(Cd_u + 2*Cd)*Q*S / (u0*m);
    Xw = -(Cd_a - Cl)*Q*S / (u0*m);
    Xq = 0;
    X_deltae = 0;

    Zu = -(0 + 2*Cl)*Q*S / (u0*m);
    Zw = -(Cl_a + Cd)*Q*S / (u0*m);
    Za = u0*Zw;
    Zadot =0;
    Zq = 0;
    Zwdot = 0;
    Z_deltae = -(Cl_deltaE*Q*S)/m;

    Mu = 0;
    Mw = (Cm_a*Q*S*c)/(u0*Iy);
    Mwdot = Cm_adot*c*Q*S*c/(2*u0*u0*Iy);
    Ma = u0*Mw;
    Madot = u0*Mwdot;
    Mq = (Cm_q*c*Q*S*c)/(2*u0*Iy);
    MdeltaE= Cm_deltaE*(Q*S*c)/Iy;
```

```
    %LATERAL
    YB = Q*S*Cy_B/m;
    Yp = 0;
    Yr = 0;
    Y_deltar = Q*S*Cy_deltar/m;
    Y_deltaa = 0;

    N_deltaa = Cn_deltaa*Q*S*b/Iz;
    N_deltar = Cn_deltar*Q*S*b/Iz;
    NB = Q*S*b*Cn_B/Iz;
    Np = Q*S*b^2*Cn_p/(2*Iz*u0);
    Nr = Q*S*b^2*Cn_r/(2*Iz*u0);

    L_deltaa = Q*S*b*Cl_deltaa/Ix;
    L_deltar = Q*S*b*Cl_deltar/Ix;
    LB = Q*S*b*Cl_B/Ix;
    Lp = Q*S*b^2*Cl_p/(2*Ix*u0);
    Lr = Q*S*b^2*Cl_r/(2*Ix*u0);


    %LONGITUDINAL
     A_long = [Xu,Xw,0,-g;Zu,Zw,u0,0;(Mu+Mwdot*Zu),(Mw+Mwdot*Zw),(Mq + Mwdot*u0),0;0,0,1,0]
     B_long = [X_deltae;Z_deltae;MdeltaE+Mwdot*Z_deltae;0]
    %LATERAL
     A_lat = [YB/u0,Yp/u0,-(1-Yr/u0), g*cos(0)/u0; LB,Lp,Lr,0;NB,Np,Nr,0;0,1,0,0]
     B_lat = [0,Y_deltar/u0; L_deltaa,L_deltar;N_deltaa,N_deltar;0,0]
    %%LONGITUDINAL APPROXIMATIONS
    %SHORT PERIOD
    A_short = [Za/u0, 1; Ma + Madot*Za/u0, Mq + Madot];
    %LONG PERIOD
    A_longit = [Xu, -g; -Zu/u0, 0];
    %%LATERAL APPROXIMATIONS
    %DUTCH ROLL
    A_dutch = [YB/u0 , -(1- Yr/u0); NB,Nr];
    %SPIRAL
    spiralRoot = (LB*Nr - Lr*NB)/LB;
    %ROLL
    rollRoot = Lp;

%LONGITUDINAL
C_longU = [1,0,0,0]; %U VELOCITY
C_longW = [0,1,0,0]; %W VELOCITY
C_longQ = [0,0,1,0]; %PITCH RATE
C_longBeta = [0,0,0,1]; %PITCH ANGLE
D_long = 0;

[A_elevatorU,B_elevatorU] = ss2tf(A_long,B_long,C_longU,D_long); %U VELOCITY
[A_elevatorW,B_elevatorW] = ss2tf(A_long,B_long,C_longW,D_long); %W VELOCITY
[A_elevatorPitchRate,B_elevatorPitchRate] = ss2tf(A_long,B_long,C_longQ,D_long); %PITCH RATE
[A_elevatorPitchAngle,B_elevatorPitchAngle] = ss2tf(A_long,B_long,C_longBeta,D_long); %PITCH ANGLE

%LATERAL
C_latB = [1 0 0 0];%SIDESLIP
C_latP = [0 1 0 0];%ROLL RATE
C_latR = [0 0 1 0];%YAW RATE
C_latFi = [0 0 0 1];%ROLL ANGLE
D_lat = 0;

[A_aileronSideslip,B_aileronSideslip] = ss2tf(A_lat,B_lat(:,1),C_latB,D_lat); %SIDESLIP(AILERON)
[A_aileronRollRate,B_aileronRollRate] = ss2tf(A_lat,B_lat(:,1),C_latP,D_lat); % ROLL RATE(AILERON)
[A_aileronYawRate,B_aileronYawRate] = ss2tf(A_lat,B_lat(:,1),C_latR,D_lat); % YAW RATE(AILERON)
[A_aileronRollAngle,B_aileronRollAngle] = ss2tf(A_lat,B_lat(:,1),C_latFi,D_lat); % ROLL ANGLE(AILERON)

[A_rudderSideslip,B_rudderSideslip] = ss2tf(A_lat,B_lat(:,2),C_latB,D_lat); %SIDESLIP(RUDDER)
[A_rudderRollRate,B_rudderRollRate] = ss2tf(A_lat,B_lat(:,2),C_latP,D_lat); %ROLL RATE(RUDDER)
[A_rudderYawRate,B_rudderYawRate] = ss2tf(A_lat,B_lat(:,2),C_latR,D_lat); %YAW RATE(RUDDER)
[A_rudderRollAngle,B_rudderRollAngle] = ss2tf(A_lat,B_lat(:,2),C_latFi,D_lat); %ROLL ANGLE(RUDDER)


%% ROOT CALCULATIONS

%For Longitudinal Equations
%Eigenvalues of A
rootsLong = eig(A_long);
phugoidRoots = rootsLong([3:4]);
shortPeriodRoots = rootsLong([1:2]);

%Approximations
%Short-Period
rootsShort = eig(A_short);
%Long-Term
rootsLong = eig(A_longit);
%% For Laterla Equations
%Eigenvalues of A
rootsLat = eig(A_lat);
dutchRollRoots = rootsLat([1:2]);
rollRoots = rootsLat(3);
spiralRoots = rootsLat(4);

%Approximations
%Dutch Roll
rootsDutch = eig(A_dutch);
%Spiral
rootsSpiral = spiralRoot;
%Roll
rootsRoll = rollRoot;
```

6 Appendix B
Autopilot Creation

```
%FOR LONGITUDINAL
s = tf("s");
controller =-10/(s+10);
%% SHORT MODE
DA = 1;
DB = -(Mq + Madot + (Za/u0));
DC = (Za * Mq / u0) - Ma;

%Angle of Attack
NAa = Z_deltae/u0;
NBa = MdeltaE - (Mq*Z_deltae/u0);
AoAOverElevator = s*(NAa*s + NBa)/(DA * s^2 + DB * s + DC);
sysAoA = controller*AoAOverElevator;
%figure(1)
%title("Root Locus of Inner Loop")
%rlocus(sysAoA)
KinnerAoA = 7.21;
sysOuterAoA = sysAoA/(1+KinnerAoA*sysAoA);
%figure(2)
%rlocus(sysOuterAoA)
%Pitch Rate
NAq = MdeltaE + (Madot*Z_deltae/u0);
NBq = (Ma*Z_deltae-MdeltaE*Za)/u0;
PitchRateOverElevator = s*(NAq*s + NBq)/(DA * s^2 + DB * s + DC);

sysPitch = controller*PitchRateOverElevator;
%figure(3)
%title("Root Locus of Inner Loop")
%rlocus(sysPitch)
KinnerPitch = 0.323;
sysOuterPitch = sysPitch/(1+KinnerPitch*sysPitch);
%figure(4)
title("Root Locus of Outer Loop")
%rlocus(sysOuterPitch)
%% PHUGOID MODE
DA = 1;
DB = -Xu;
DC = -Zu*g/u0;

%U Velocity
NAu = X_deltae;
NBu = g*Z_deltae/u0;
UOverElevator = s*(NAu*s + NBu)/(DA*s^2 + DB*s + DC);
sysU = controller*UOverElevator;
%figure(5)
%rlocus(sysU)
KinnerU = 1.01;
sysOuterU = sysU/(1+KinnerU*sysU);
%figure(6)
%rlocus(sysOuterU)

%Pitch Angle
NAt = -Z_deltae/u0;
NBt = (Xu*Z_deltae-Zu*X_deltae)/u0;
TetaOverElevator = (s)*(NAt*s + NBt)/(DA*s^2 + DB*s + DC);

sysT = -controller*TetaOverElevator;
%figure(7)
%rlocus(sysT)
KinnerT = 28.8;
sysOuterT = sysT/(1+KinnerT*sysT);
%figure(8)
%rlocus(sysOuterT)
```