Miniproject

Carson Sager

ECEN5713

Dr. Martin Hagan

1 April 2025

Step 1: Find the system poles.

In order to find the system poles, we needed to find the eigenvalues of the given matrix A. This was done in Matlab with the following command:

```
[M, EVal] = eig(A);
```

where M is the modal matrix that contains the eigenvectors and EVal is a diagonal matrix containing the eigenvalues on the diagonal. Therefore, the eigenvalues were found to be the following:

```
The system eigenvalues are:
   0.0000 + 0.0000i
 -10.0000 + 0.0000i
  -0.7000 + 3.0000i
  -0.7000 - 3.0000i
```

Step 2: Find the modes of the system, and put the state model in normal form.

The normal form equation is expressed as:

$$\boldsymbol{q}' = T\boldsymbol{q} + B_n\boldsymbol{u}$$

In normal form, the matrix that multiplies the q vector is $T=M^{-1}AM$, or simply the EVal matrix that we received from the previous equation with the eigenvalues on the diagonal. The T matrix will provide us with the modes of the system. For multiplying the **u** vector, $B_n=M^{-1}B$, which is easy to solve since we have already calculated M in the previous question and B is given. Therefore, we have the following normal form:

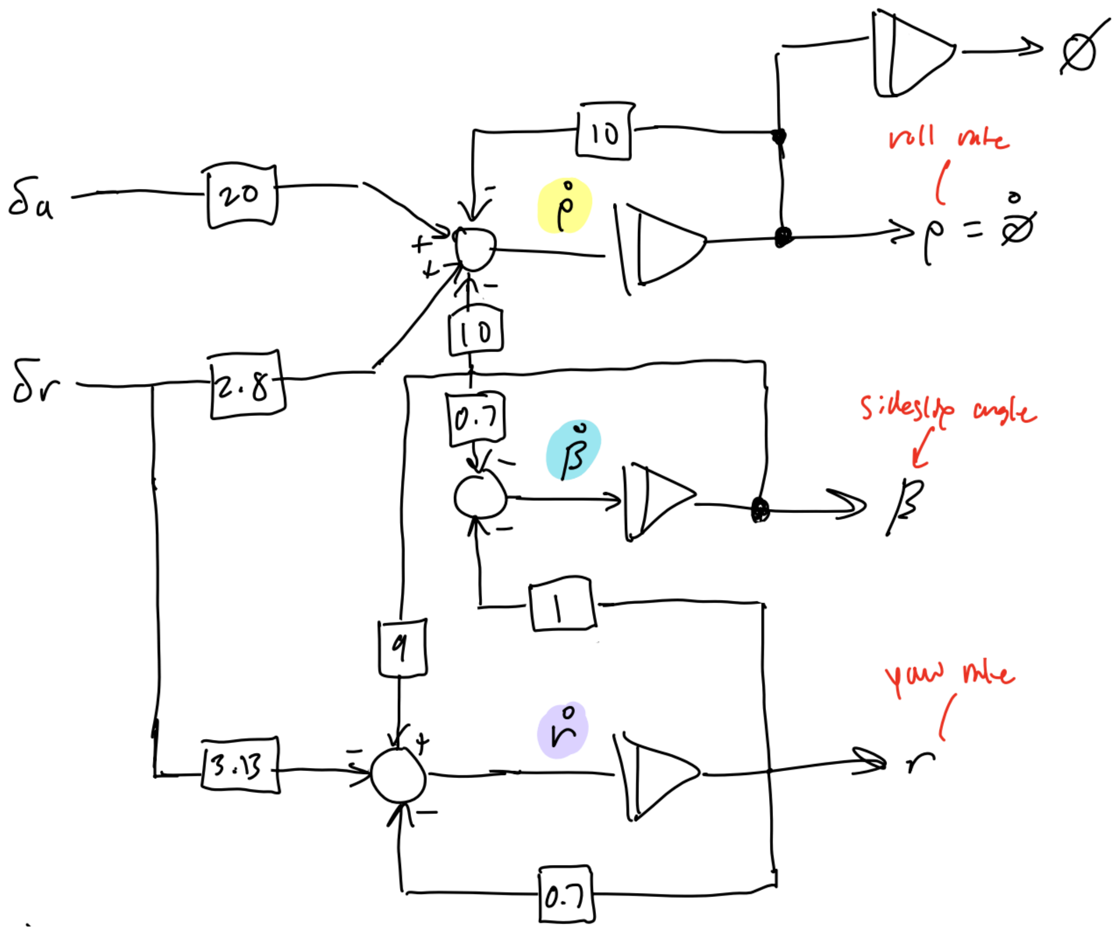
$$\dot{q} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & -10 & 0 & 0 \\ 0 & 0 & -7+3i & 0 \\ 0 & 0 & 0 & -7-3i \end{bmatrix} q + \begin{bmatrix} 2 & -0.0498 \\ 20.0998 & 2.4845 \\ 0 & -1.7425 \\ 0 & -1.7425 \end{bmatrix} u$$

Step 3: Indicate any unstable mode of the system. Explain the physical meaning of any unstable mode in terms of the aircraft operation.

The 2nd, 3rd, and 4th modes are all stable. The first node corresponding with eigenvalue=0, however, is marginally stable. Although this is not technically unstable, it is definitely not ideal for our system as there will be no decay or growth and there is higher potential for a transition to instability.

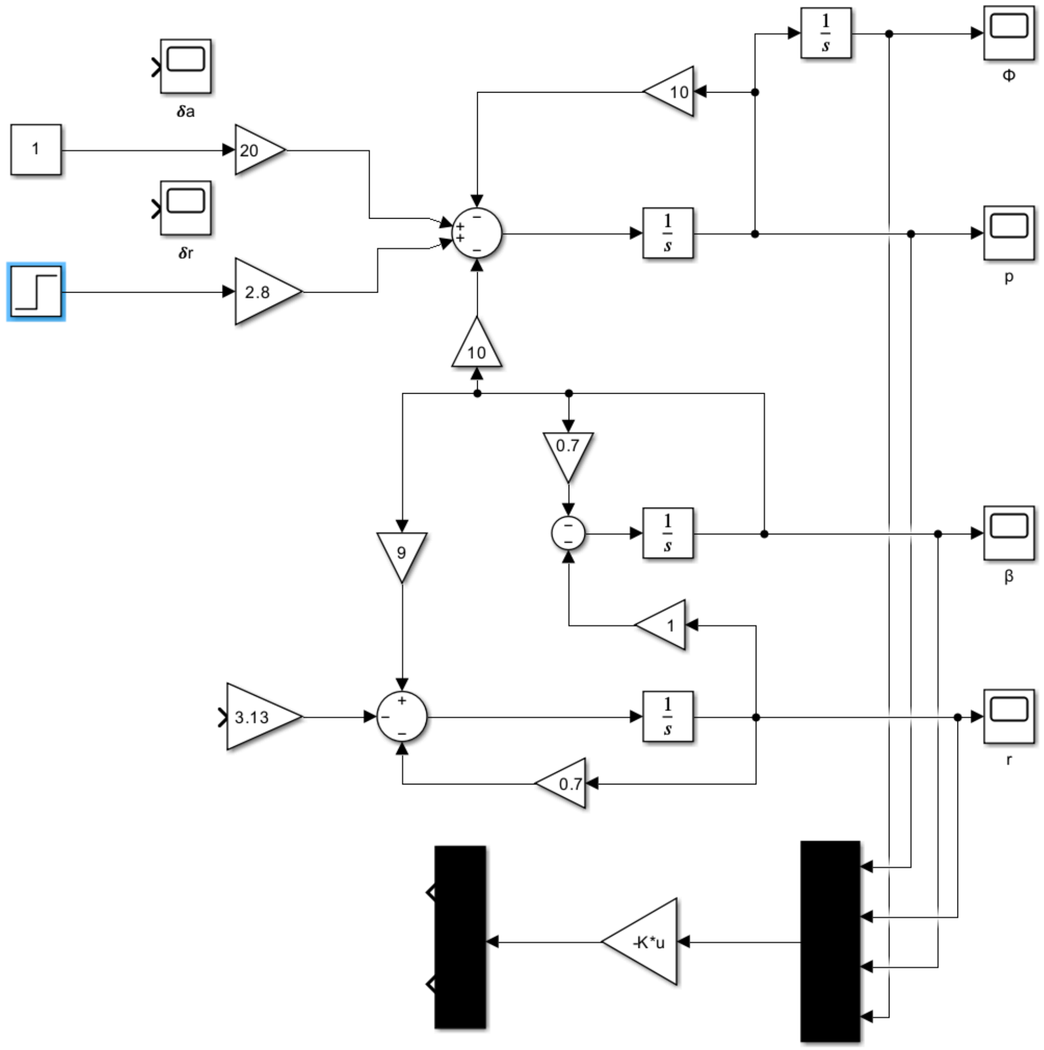
The physical meaning of any unstable node in terms of the aircraft system would be excessive pitch-up/pitch-down, spirals and uncontrollable oscillations, and potentially unstable responses to the control inputs.

Step 4: Make a simulation diagram for the system.



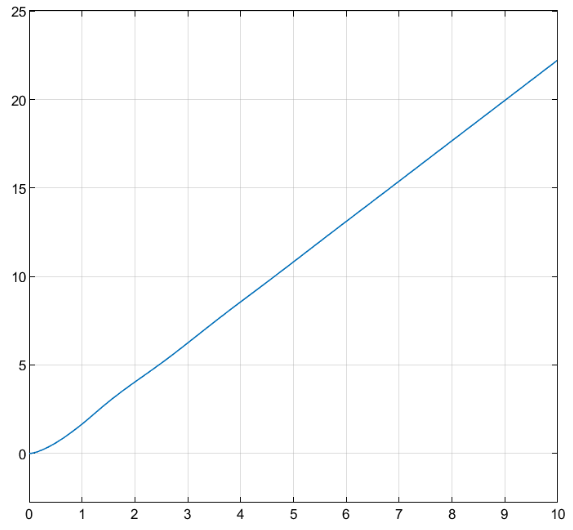
Step 5: Implement the simulation diagram in Simulink.

This model was produced following the addition of the closed- loop feedback (which does not connect back to input for now).

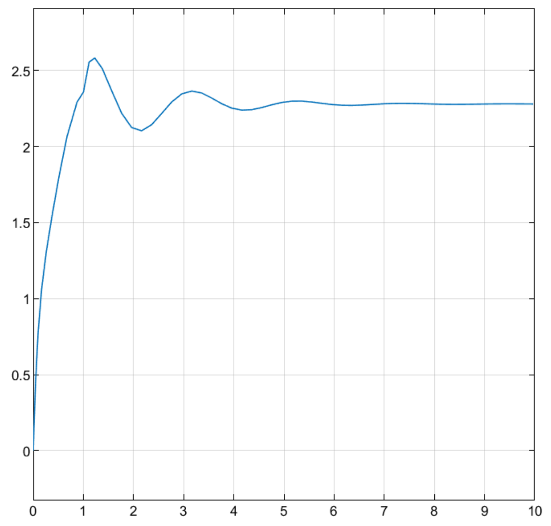


Step 6: Use Simulink to find the two open-loop step responses (apply a step function for each input individually, while setting the other input to zero). For each step response provides plots of all state variables.

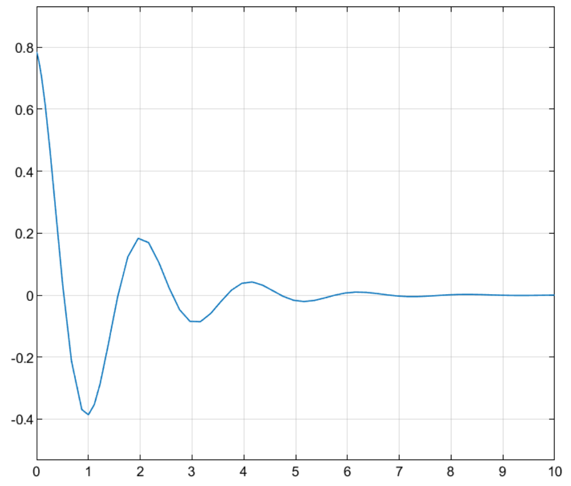
First plots: Setting $\boldsymbol{\delta}_a$ to 1 and $\boldsymbol{\delta}_r$ to the step response:



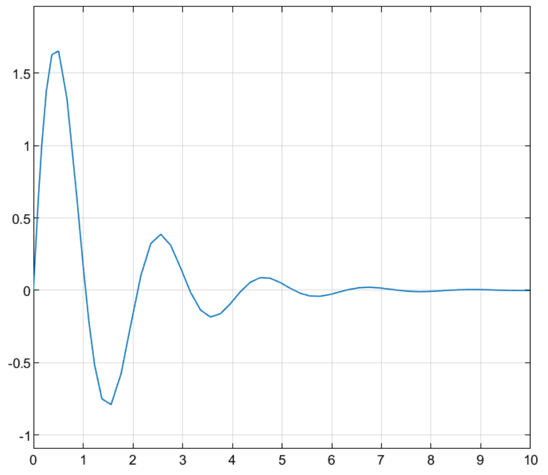
Φ vs. time



p vs. time

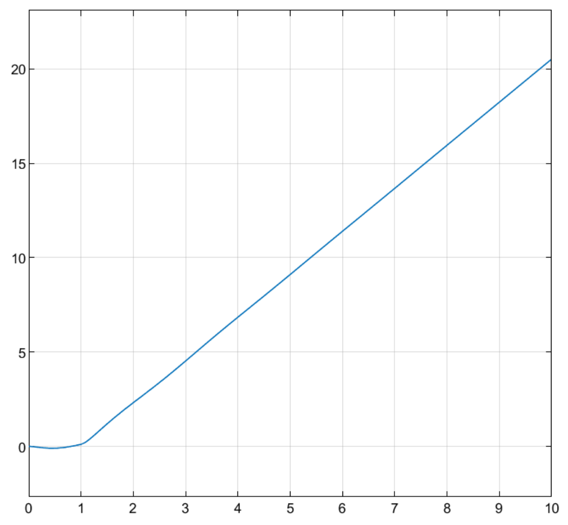


β vs. time

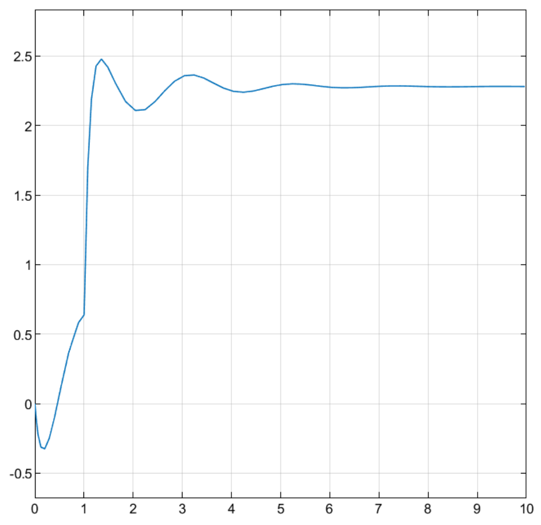


r vs. time

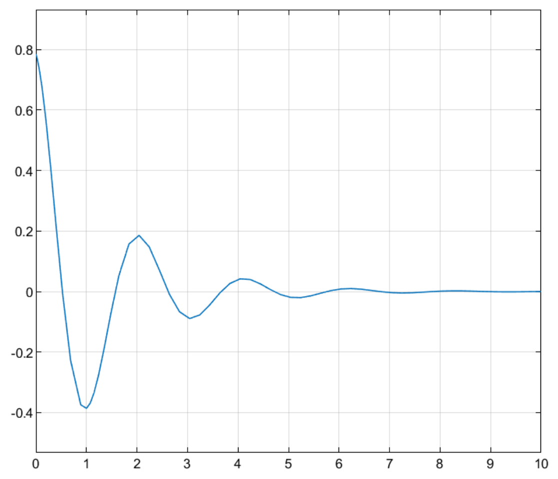
Second plots: Setting $\boldsymbol{\delta}_a$ to the step response and $\boldsymbol{\delta}_r$ to 1:



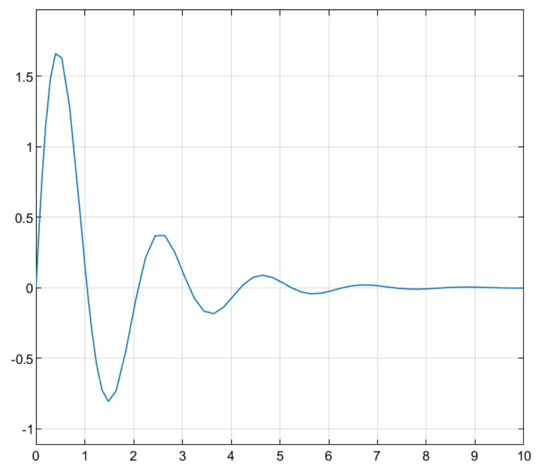
Φ vs. time



p vs. time



β vs. time



r vs. time

Step 7: Test for controllability of the system

In order to test for controllability, I used the B_n matrix that was previously calculated and, since there were no zero rows, the system is completely controllable. Additionally, I calculated the P matrix using Criterion 2 for controllability in the textbook.[1] If the P matrix is full rank (rank(P) = 4 in our case), then the system is completely controllable. The results are shown:

```
Bn matrix:
    2.0000   -0.0498
   20.0998    2.4845
         0   -1.7425
         0   -1.7425

P_matrix:
   1.0e+04 *

    0.0020    0.0003   -0.0200   -0.0028    0.2000    0.0249   -2.0000   -0.2443
         0   -0.0003         0    0.0002         0    0.0027         0   -0.0058
         0         0         0    0.0003         0   -0.0004         0   -0.0024
         0         0    0.0020    0.0003   -0.0200   -0.0028    0.2000    0.0249

Rank of P
     4

The system is controllable (no 0 rows of Bn and rank(P)=4)
```

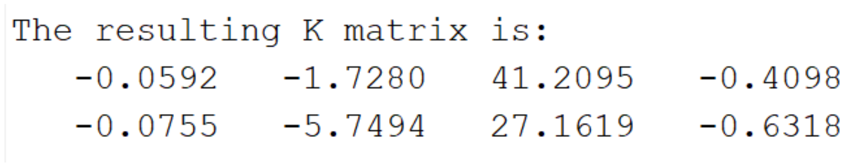
Step 8: Choose a set of desired closed-loop poles. Explain and justify your choice.

The closed-loop poles that I chose were -1, -11, -8+3i, and -8-3i. I chose these specific positions because they are each one unit to the left of the original poles, and all fall within the region of stability to the left of the imaginary axis. Also, it is assumed that we will need some tuning to these values as we proceed since we typically do not have the best choice of closed-loop poles to start with.

[1] Brogan, W. L. (1991). *Modern control theory*. Pearson.

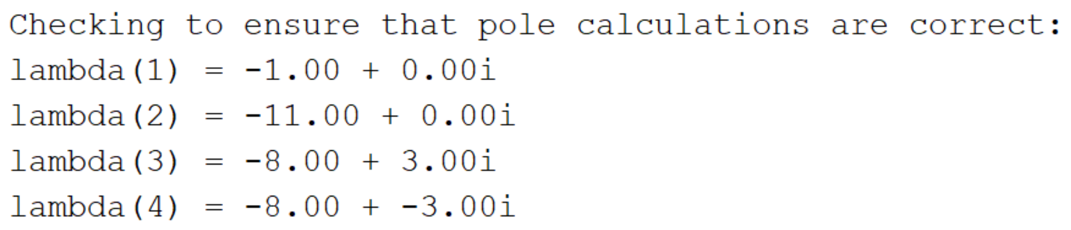
Step 9: Using the method of Section 13.4 (write your own matlab program to implement this method) Find the state variable feedback controller that will produce the desired closed-loop poles.

Code for calculating that feedback controller (K) is included in the zip attached in submission. Resulting K matrix of original pole choices:



```
The resulting K matrix is:
   -0.0592   -1.7280   41.2095   -0.4098
   -0.0755   -5.7494   27.1619   -0.6318
```

Additionally, in order to ensure that our program to calculate K was working properly, the eigenvalues of $A_c = A - B*K$ were calculated. This calculation correctly results in the eigenvalues being the values of the desired closed-loop poles.



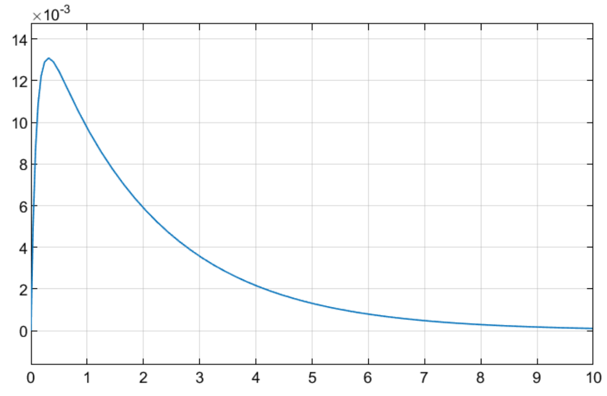
```
Checking to ensure that pole calculations are correct:
lambda(1) = -1.00 + 0.00i
lambda(2) = -11.00 + 0.00i
lambda(3) = -8.00 + 3.00i
lambda(4) = -8.00 + -3.00i
```

Step 10: Implement the controller in Simulink, and test the closed-loop system response to a variety of initial states. Provide plots of all state variables for each response. Also, on a separate graph from the state variables, plot the two input variables.

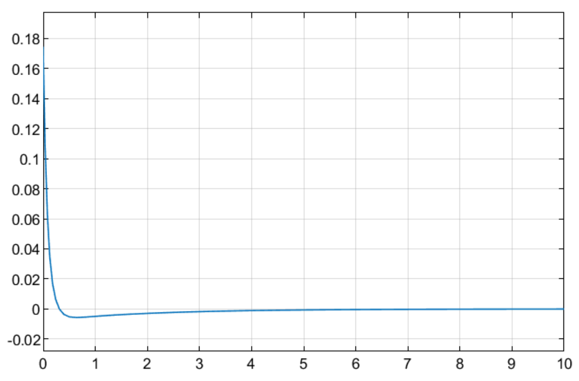
We will be doing four separate tests in order ensure that the closed-loop system provides optimal control so that the angles of the control input ($\boldsymbol{\delta}_a$ and $\boldsymbol{\delta}_r$) do not exceed unreasonable values. Our system will attempt to maintain these inputs below 25° (0.436332 radians). For these tests, the estimated maximum initial approximation of each state in commercial aircrafts was used. This includes heavy banking on turns and crosswind landings, for example. Only one of the states in each test is non-zero as this helped to isolate each state and recognize if the feedback and closed-loop poles that were chosen were working correctly for each state individually.

Test #1

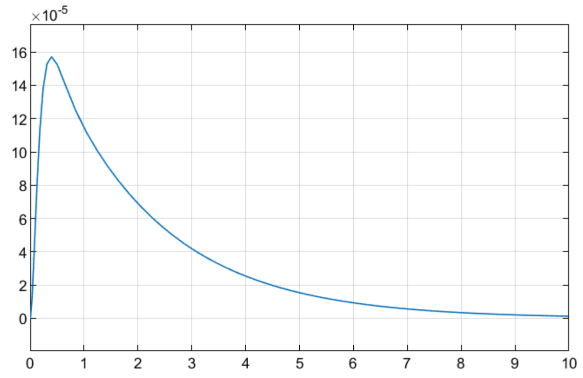
Initial States: p = 10°/sec (0.174533 rad/sec), r = 0°/sec, β = 0°, Φ = 0°



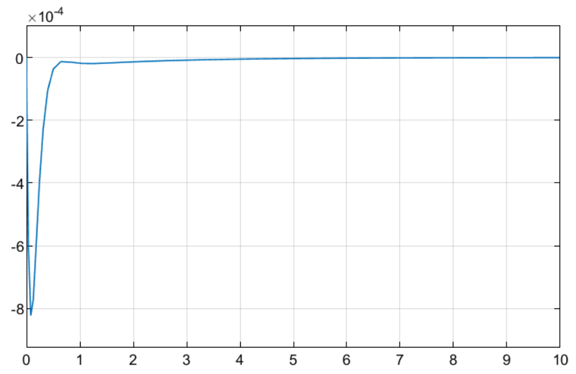
Φ vs. time



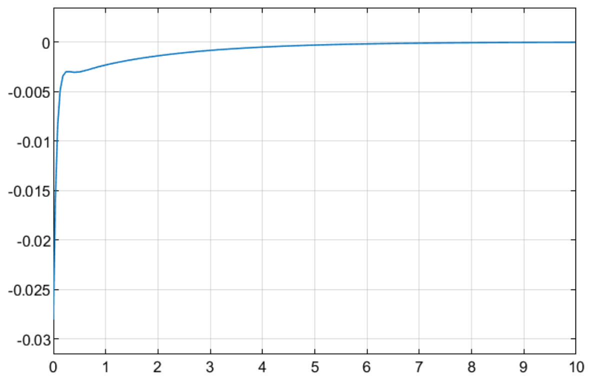
p vs. time



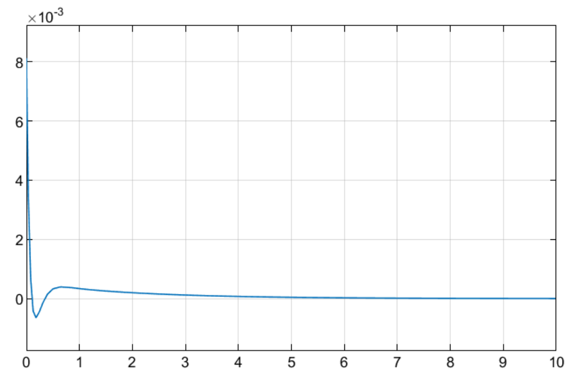
β vs. time



r vs. time



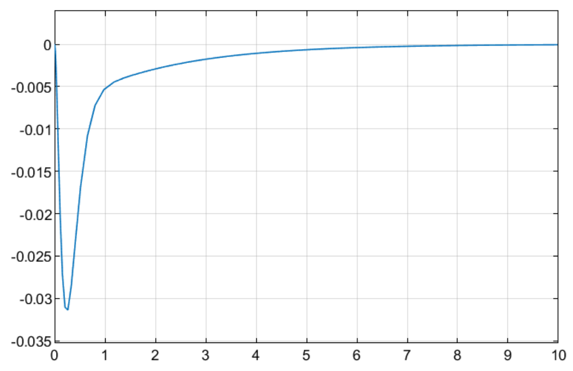
δ_a vs. time



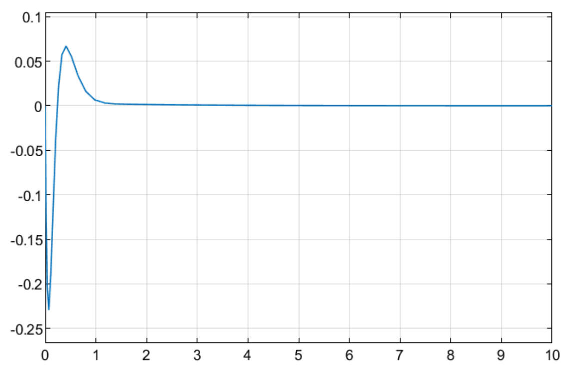
δ_r vs. time

Test #2

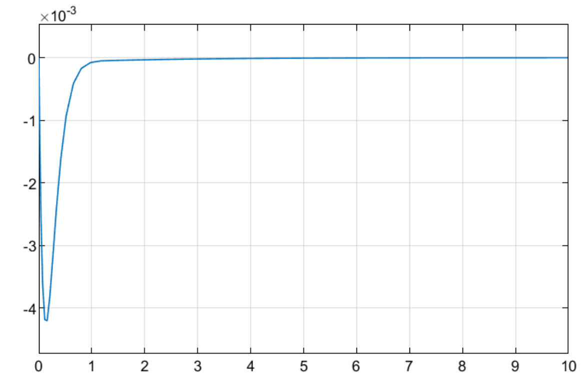
Initial States: p = 0°/sec, r = 5°/sec (0.0872665 rad/sec), β = 0°, Φ = 0°



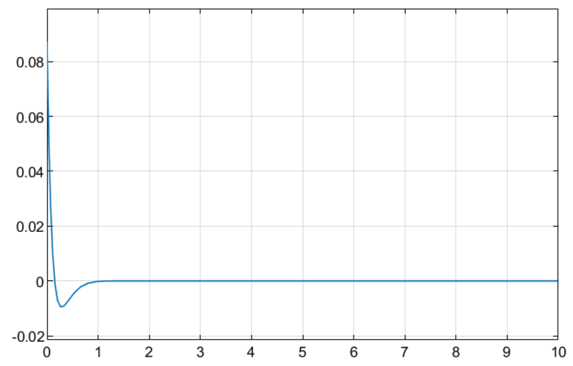
Φ vs. time



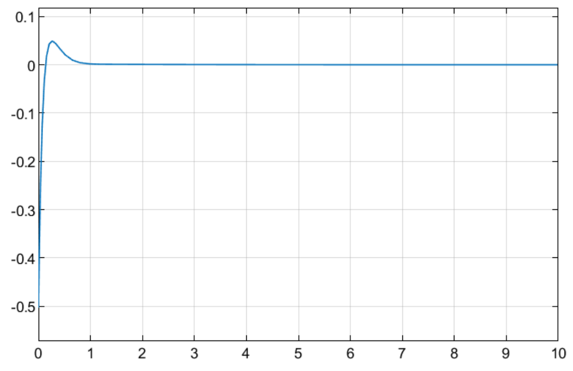
p vs. time



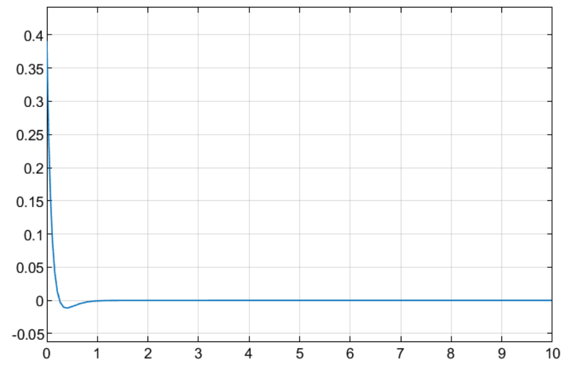
β vs. time



r vs. time



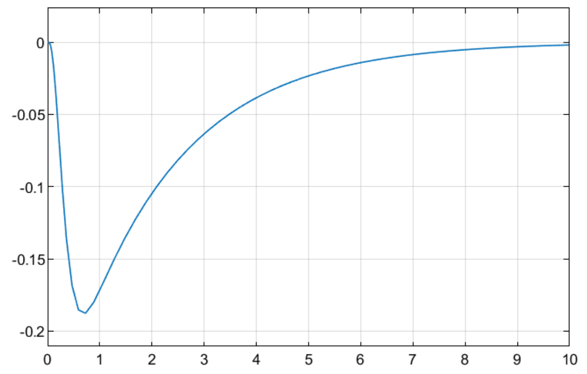
δ_a vs. time



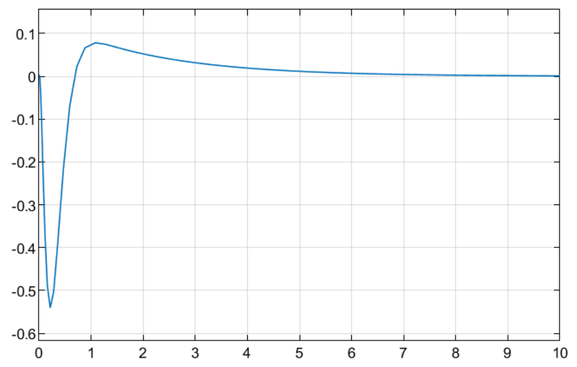
δ_r vs. time

Test #3

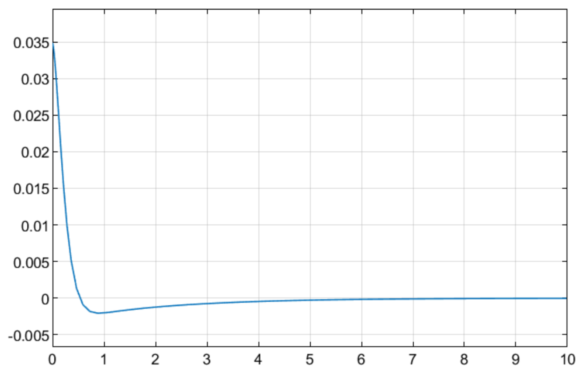
Initial States: p = 0°/sec, r = 0°/sec, β = 2° (0.0349066 rad), Φ = 0°



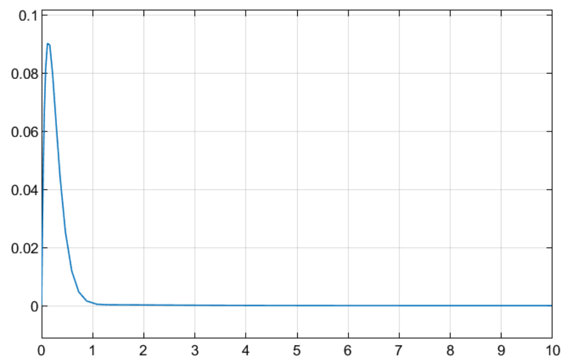
Φ vs. time



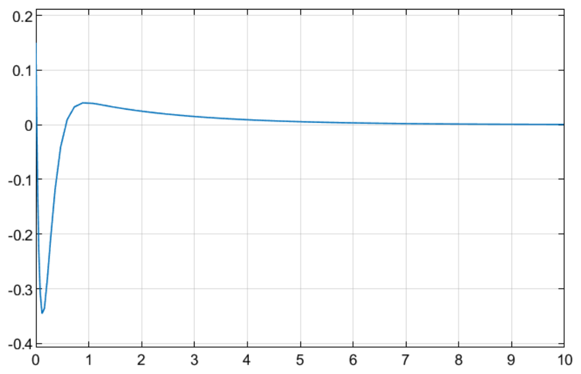
p vs. time



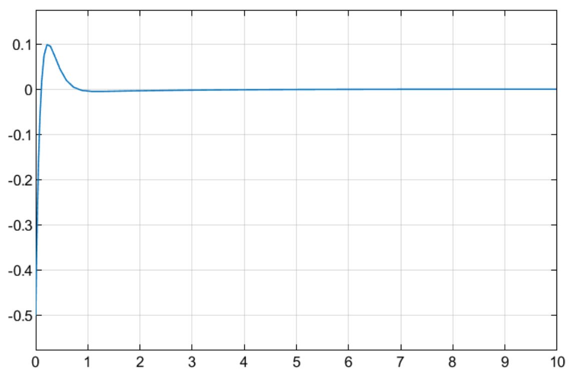
β vs. time



r vs. time



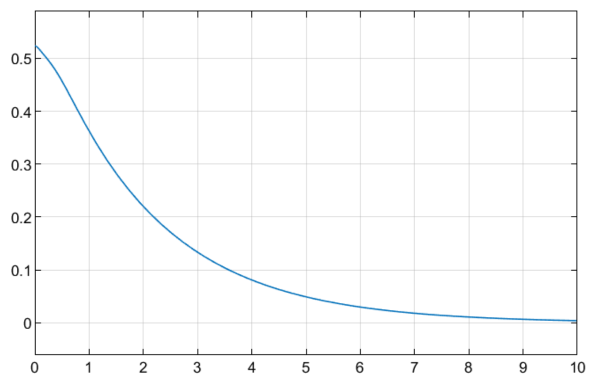
δ_a vs. time



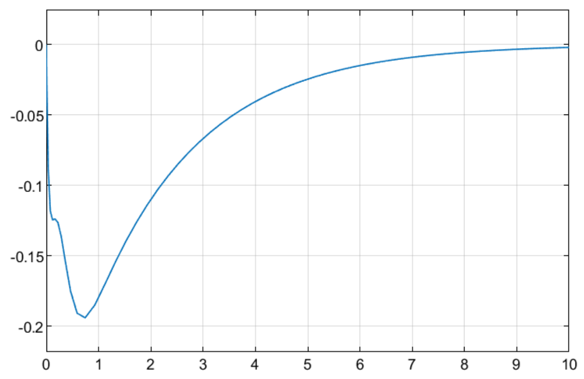
δ_r vs. time

Test #4

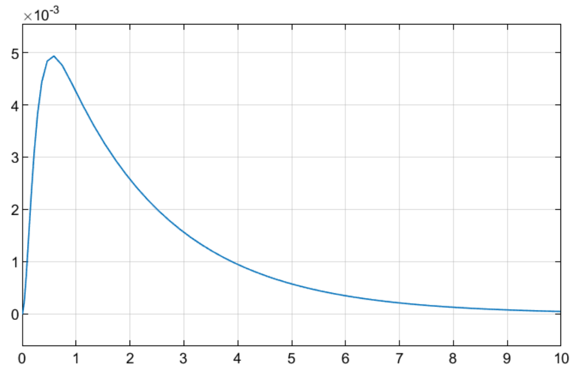
Initial States: p = 0°/sec, r = 0°/sec, β = 0°, Φ = 30° (0.5236 rad.)



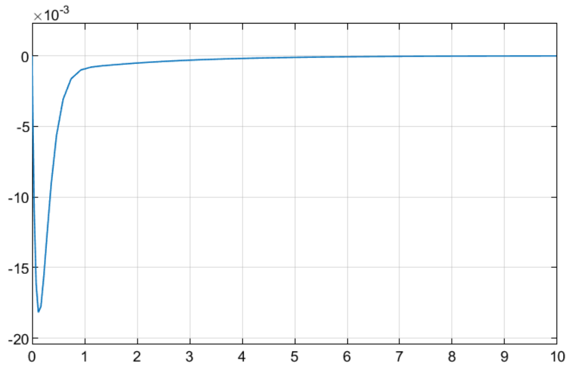
Φ vs. time



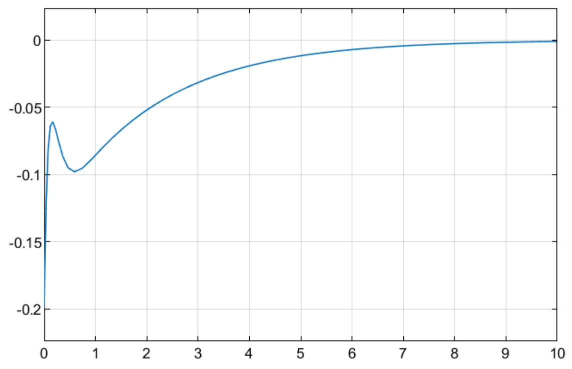
p vs. time



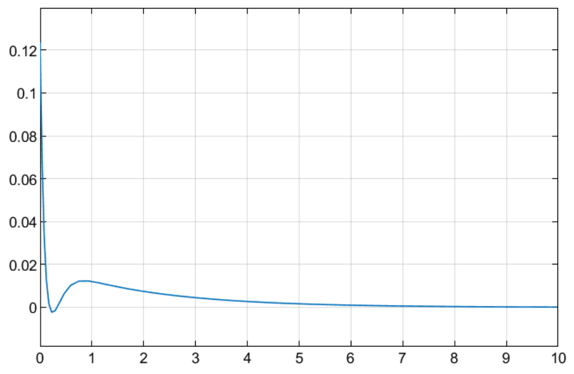
β vs. time



r vs. time



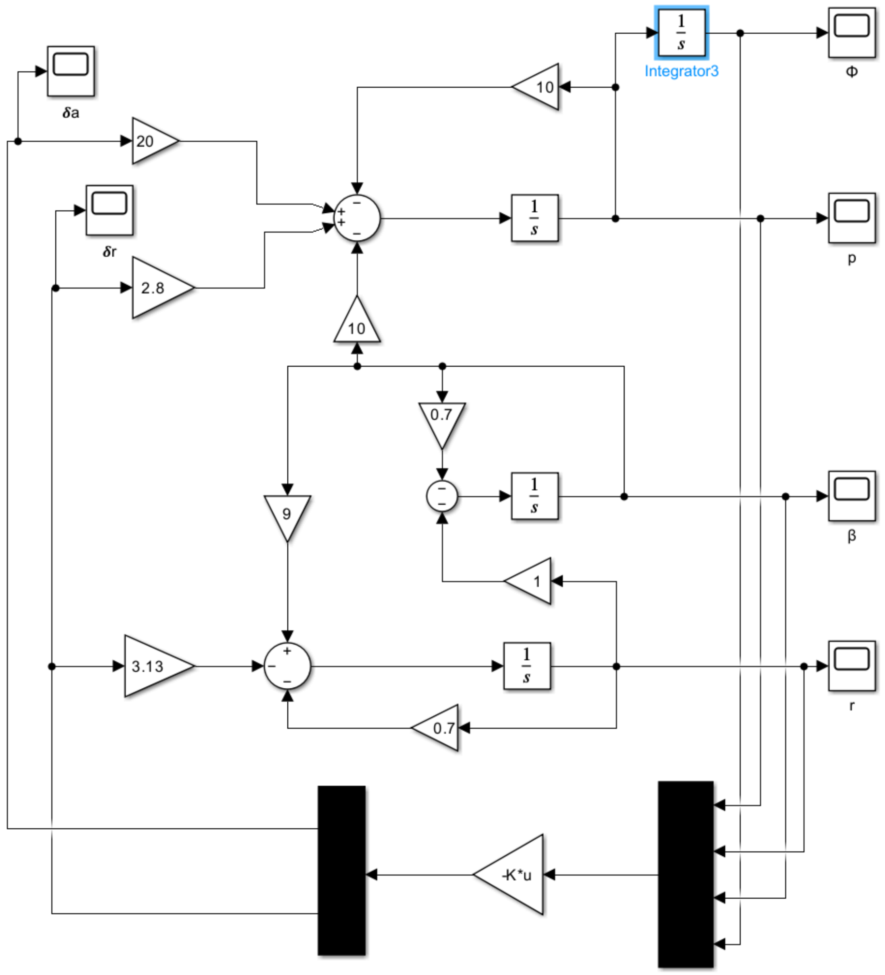
δ_a vs. time



δ_r vs. time

Step 11: Discuss your final design. In particular, comment on the choice of closed-loop pole locations, the speed and oscillation of the closed-loop response, the magnitude of input variables (aileron angle and rudder angle) required to achieve the closed-loop response. Also, discuss practical considerations of implementing the controller in hardware.

Final Simulink diagram:



The final closed loop pole locations I decided to use are the following:

```
NewPoleList = [-0.5 -15 -8+3i -5-3i];
```

with the K matrix for these poles being:

```
The resulting K matrix is:
     0.1606      5.7680    -4.2982     0.3806
    -0.0465     -4.4785    14.3814    -0.2360
```

This is after plenty of iterations of tuning these values until the inputs were within a reasonable range (less than 25° for both inputs) and after changing the initial approximations of the states to realistically be able to control the aircraft. Most of the states settle to 0 within about 6-8 seconds and there is not much oscillation present in the closed-loop response. There were only 1-2 oscillations (if any) per signal. Only two of the elements have imaginary components, so this could be part of the explanation as to the lack of oscillations. As previously mentioned, the magnitude of the input variables were held within 25° or 0.436332 radians if looking at the axis values on the graphs. Although it would be ideal to not turn the controllers even 25°, this appeared to be a realistic goal in terms of the realistic implementation of the control system. It is important to note that multiplexores (muxes) and demuxes were used in the simulink model in order to properly retrieve the inputs from the state outputs ($\mathbf{u} = -K\mathbf{x}$). Practical considerations of implementing the controller in hardware would include the need for ADCs to convert the sensor signals from analog to digital values (and DACs to convert the computed numbers back to analog signals), the need for actuators (whether they be electrical or hydraulic), and the need to for amplifiers in order to provide enough power to actually get the actuators to turn. Additionally, we are assuming that this would be a linear system, but there are various qualities of controlling an aircraft that are generally nonlinear. These include, but are not limited to: aerodynamic nonlinearities, control surface saturation, aircraft geometry, mass distribution, etc.