Aircraft Autopilot Design

TTK4190 Guidance and Control of Vehicles, Assignment 2

Fall 2015

Objective

In this assignment you will apply linear theory to the design of SISO and MIMO autopilots for an F16 fighter aircraft. You will mostly refer to the lecture notes from Beard and McLain (2012).
The code you will produce is the most important result and has the greatest weight on the grade. Whether your code works fine or not, please include inline comments that can help me understand your intentions and figure out possible mistakes.

Deadline and Delivery Details

The answer must be handed in **by 19:00 on Thursday, October 15th**. This assignment accounts for 10% of the overall course grade. Team work is forbidden and you must deliver individual reports and Matlab/Simulink files. If you use Simulink, the simulation should be started via an m-file and not the Simulink window. Matlab code and Simulink models should not be included in the report. The code you will have to hand in must work as is, so I should just run it once in order to obtain *all* the required results. Further information regarding the required Matlab code is given in the corresponding tasks. The report together with all the required code has to be handed in **via Itslearning**. You may write the report with your favourite editor (LaTeX, Word...) or write it by hand and scan it afterwards, but in the end it has to be a **pdf** document. Paper will not be accepted.

F16 fighter aircraft

The linear lateral model of the *F-16* fighter aircraft is given by Stevens and Lewis (pages 370–371, 1992):

$$\begin{aligned} \dot{\boldsymbol{x}} &= \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{u} && (1) \\ \boldsymbol{y} &= \boldsymbol{C}\boldsymbol{x} && (2) \end{aligned}$$

where

$$\boldsymbol{x} = \begin{bmatrix} \beta \text{ (rad)} \\ \phi \text{ (rad)} \\ p \text{ (rad/s)} \\ r \text{ (rad/s)} \\ \delta_a \text{ (rad)} \\ \delta_r \text{ (rad)} \end{bmatrix}, \boldsymbol{u} = \begin{bmatrix} \delta_a^c \text{ (rad)} \\ \delta_r^c \text{ (rad)} \end{bmatrix}, \boldsymbol{y} = \begin{bmatrix} r \text{ (deg/s)} \\ p \text{ (deg/s)} \\ \beta \text{ (deg)} \\ \phi \text{ (deg)} \end{bmatrix} \qquad (3)$$

The actuator dynamics of the aileron and rudder are augmented to the lateral equations of motion as two states δ_a and δ_r, which both are modeled as low-pass filters (LP) with time constants $T_a = 1/20.2$ s. Moreover,

$$H_a(s) = \frac{1}{T_a s + 1} = \frac{20.2}{s + 20.2} \tag{4}$$

The control inputs for the augmented model are denoted δ_a^c and δ_r^c, respectively.
To avoid that steady-state values of the yaw rate r are fed back to the autopilot, a *washout filter* can be used:

$$H_w(s) = \frac{r_w}{r}(s) = \frac{T_w s}{T_w s + 1} = \frac{s}{s+1} \tag{5}$$

The washout filter is chosen as a high-pass (HP) filter with time constant $T_w = 1.0$ s.
During a coordinated turn the course angle χ satisfies the *bank-to-turn* equation:

$$\dot{\chi} = \frac{g}{V_a} \tan(\phi) \tag{6}$$

In the absence of wind or sideslip, $\dot{\beta} = \beta = 0$. Consequently, $\psi = \chi$ and $V_a = V_g$. The state-space matrices and trim condition are given below:

Matlab state-space matrices and trim condition

Speed	$V_a = 502$ ft/s $= 552$ km/h
Mach-number	$M = 0.45$

```
A = [
    -0.3220    0.0640    0.0364   -0.9917    0.0003    0.0008
     0              0         1   -0.0037         0         0
   -30.6492         0   -3.6784    0.6646   -0.7333    0.1315
     8.5396         0   -0.0254   -0.4764   -0.0319   -0.0620
          0         0         0         0     -20.2         0
          0         0         0         0         0     -20.2 ];

B = [
          0         0
          0         0
          0         0
          0         0
       20.2         0
          0      20.2 ];

C = [
          0         0         0   57.2958         0         0
          0         0   57.2958         0         0         0
    57.2958         0         0         0         0         0
          0   57.2958         0         0         0         0  ];
```

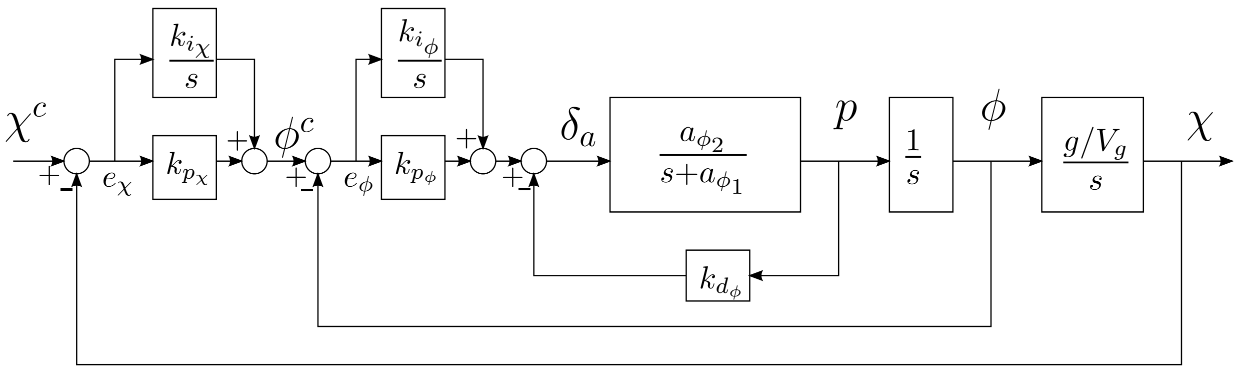

Figure 1: Successive loop closure for autopilot design.

1 Open-loop analysis

Compute the Dutch roll natural frequency and relative damping ratio. Explain what the Dutch roll mode is. Compute the spiral-divergence and roll modes. Comment on the results.

2 Autopilot for course hold using successive loop closure

Figure 1 from Chapter 6 in Beard and McLain (2012) shows the block diagram for a lateral autopilot using successive loop closure. This represents an autopilot for course hold using only aileron δ_a as control input. Moreover, the rudder δ_r is not used for feedback, and you may ignore the actuators dynamics $H_a(s)$ (this simplifies the state-space model).

1. Find numerical values for a_{ϕ_1} and a_{ϕ_2} based on the state-space model (1)–(2).

2. Find numerical values for the five gains in Figure 1 associated with the lateral autopilot and present the open-loop and closed-loop transfer functions from χ^c and χ in a Bode plot.

   (Hint: when designing the roll loop, choose $\delta_a^{max} = 45°$, $e_\phi^{max} = 15°$, $\zeta_\phi = 0.707$)

   Code hand-in: *Hand in an m-file named* `run_task_22.m` *that will initialize the simulation, run the simulation, and generate the necessary plots. Include any other files necessary to execute* `run_task_22.m`. *Place all Matlab files in a folder entitled "task_22".*

3. Present simulation results for course hold and course changing maneuvers: for course hold choose a fixed set-point, whereas for course change choose a series of steps (you may choose yourself the desired values). The F16 aircraft model must be the complete state-space model (1)–(2), including the actuators dynamics, and the autopilot must use the decoupled model given by a_{ϕ_1} and a_{ϕ_2} (in other words, use Fig. 1 but replace the block for p/δ_a with the complete state-space model).

   Code hand-in: *Hand in an m-file named* `run_task_23.m` *that will initialize the simulation, run the simulation, and generate the necessary plots. Include any other files necessary to execute* `run_task_23.m`. *Place all Matlab files in a folder entitled "task_23".*

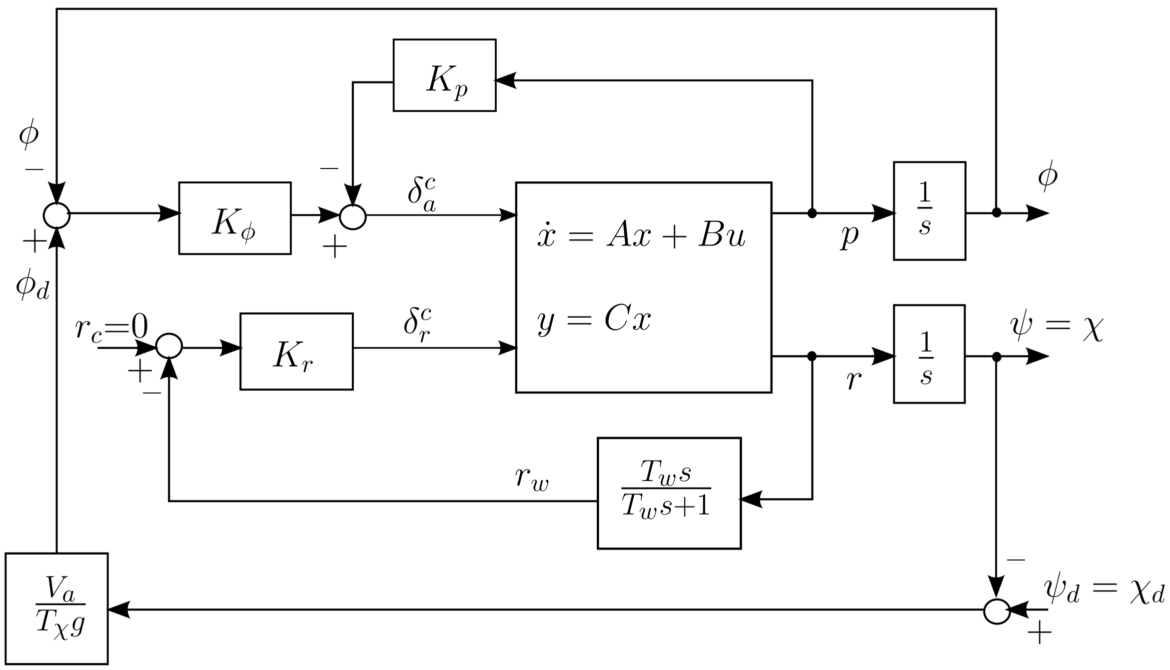

Figure 2: MIMO lateral autopilot using inner-loop yaw damper.

3 MIMO linear autopilot for course hold

We are going to improve the autopilot design by designing a MIMO controller, represented in Fig. 2. The idea is to design a yaw damper that uses feedback from the washout filter instead of the yaw rate such that only high-frequency rudder commands δ_r^c contribute to damping, while the aileron control δ_a^c will be designed for course hold. A yaw damper is a system that attempts to remove the yaw rate that inevitably results from a turn, as a consequence of the Dutch roll mode.

1. Consider the yaw damper (inner loop in Figure 2):

$$\delta_r^c = K_r(r_c - H_w(s)r) = K_r(r_c - r_w) \tag{7}$$

   where $r_c = 0$ is the yaw rate setpoint. Find a feedback gain K_r using the root locus method, and include a plot showing r for a step response $r_c = 5$ deg/s. Notice that only feedback from r_w is accepted.

   Code hand-in: *Hand in an m-file named* `run_task_31.m` *that will initialize the simulation, run the simulation, and generate the necessary plots. Include any other files necessary to execute* `run_task_31.m`*. Place all Matlab files in a folder entitled "task_31".*

2. The next step is to bank the aircraft to obtain a coordinated turn. For a given reference signal χ_d we want χ to follow. Let,

$$T_\chi \dot{\chi} + \chi = \chi_d \tag{8}$$

where $T_\chi = 10$ s is a user specified time constant. Use the *bank-to-turn* equation (6) and show that the desired roll angle ϕ_d under the assumption of small roll angles ϕ satisfies:

$$\phi_d = \frac{V_a}{T_\chi g}(\chi_d - \chi) \tag{9}$$

Find numerical values for K_p and K_ϕ such that δ_a^c turns the aircraft with satisfactory performance, and present simulation results for course hold and course changing maneuvers using the same values you used in task 2.3. Also include two plots that compare δ_a with δ_a^c, and δ_r with δ_r^c. Simulate the F16 aircraft using the complete state-space model (1)–(2).

Code hand-in: *Hand in an m-file named* `run_task_32.m` *that will initialize the simulation, run the simulation, and generate the necessary plots. Include any other files necessary to execute* `run_task_32.m`. *Place all Matlab files in a folder entitled "task_32".*

3. Compare the autopilot simulation results (Sections 2 and 3) and comment the results: which autopilot shows the best performance? Why?

References

R. W. Beard and T. W. McLain (2012). *Small Unmanned Aircraft: Theory and Practice.* Princeton University Press.

B. L. Stevens and F. L. Lewis (1992). *Aircraft Control and Simulation* John Wiley & Sons Ltd.