TTK4190 Guidance, Navigation and Control of Vehicles
Norwegian University of Science and Technology

Take-Home Project

Aircraft Autopilot Design by Successive-Loop Closure

Maximum score 100 points

Figure 1: The F16 Fighting Falcon.

Objectives

The take-home project is a Matlab m-file coding exercise with a focus on aircraft dynamics and autopilot design. In this assignment, you will apply linear theory and design lateral autopilots for the *F16 Fighting Falcon* using successive-loop closure. The assignment covers the design of three aircraft autopilot systems:

- Sideslip hold
- Roll attitude hold
- Course hold

The recommended reading material is the book and lecture notes of Beard & McLain (2012), and the ”Aircraft Notes” by Fossen (2022), which can be found on the ”Lecture plan and lecture notes” course page on Blackboard.

Score

You need to obtain at least 60 of a total of 100 points to pass the project.

Deadline

The take-home project should be handed in by Thursday 26th of September at 23:59.

Delivery Details

This is an individual assignment. You are supposed to upload a single zip file to Blackboard containing the following 7 files:

- **One PDF-file containing your written answers:**
  Fill in your answers in the *empty fields* of this document. Save the file with your comments and keep the file name:
  - assignment.pdf

  You can save the file and modify your answers as many times as you like before uploading the final version of the PDF file to Blackboard.
- **Two Matlab plots (graphics files):**
  - rollAndSideslip.png
  - course.png

  You are welcome to use the provided plotting function plotResults(simdata,n) to create the figures with all necessary subplots. You can save/export the figures from Matlab (File `->` Save as `->` filename.png) or just use screen dump/clipping. Please follow the above naming conventions when uploading the graphics files to Blackboard.
- **Four Matlab functions (m-files):**

  You are supposed to add your code inside the files:
  - windTriangle.m
  - sideslipHold.m
  - rollAttitudeHold.m
  - courseHold.m

  All functions should be m-files submitted in the following format:

```
function [a,b] = myfunction(x)
% [a, b]  = myfunction(x)
a = x + 1;
b = x + 2;
end
```

  The example above shows the Matlab file `myfunction.m` where a comment is added to show the function call (syntax to call the function).

Aircraft Model

The lateral model of the *F16 Fighting Falcon* is given by the state-space model

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

where

$$\mathbf{x} = \begin{bmatrix} \beta \text{ (rad)} \\ p \text{ (rad/s)} \\ r \text{ (rad/s)} \\ \phi \text{ (rad)} \\ \psi \text{ (rad)} \end{bmatrix}, \quad \mathbf{u} = \begin{bmatrix} \delta_a \text{ (rad)} \\ \delta_r \text{ (rad)} \end{bmatrix}, \quad \mathbf{y} = \begin{bmatrix} \beta \text{ (rad)} \\ p \text{ (rad/s)} \\ r \text{ (rad/s)} \\ \phi \text{ (rad)} \\ \psi \text{ (rad)} \end{bmatrix} \tag{2}$$

The aileron δ_a has a maximum deflection of ± 21 degrees and the maximum rudder angle δ_r is ± 30 degrees.

During a coordinated turn the course angle χ satisfies the *bank-to-turn* equation

$$\dot{\chi} = \frac{g}{V_g} \tan(\phi) \cos(\chi - \psi) \approx \frac{g}{V_g} \phi \tag{3}$$

The linear approximation is accurate for small sideslip angles β and small roll angles ϕ implying that $\chi = \psi$. This is a good assumption since the sideslip angle β is kept small by using the rudder δ_r. At the same time the ailerons δ_a are used to hold a constant course $\chi = \chi_c$, which again produces small roll angles ϕ. This implies that three different autopilot systems must be designed:

- **Sideslip hold:** Regulate the sideslip angle β to zero by means of the rudder δ_r
- **Roll attitude hold (inner loop):** Regulate the roll angle ϕ to a constant value ϕ_c by means of the ailerons δ_a
- **Course hold (outer loop):** Regulate the course angle χ to a constant value χ_c by generating the roll angle command ϕ_c

Note that all autopilots are designed for setpoint regulation and not trajectory tracking. This will give overshoots and low-performance tracking capabilities when changing the setpoints.

Continuous-time state-space matrices and trim condition

Ground speed $\quad V_g = 928.4 \text{ km/h} = 257.9 \text{ m/s}$

$$\mathbf{A} = \begin{bmatrix} -0.1203 & 0.1479 & -0.9860 & 0.0636 & 0 \\ -17.5184 & -1.2523 & 0.5484 & 0 & 0 \\ 3.5318 & -0.0274 & -0.1788 & 0 & 0 \\ 0 & 1 & 0.1489 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

$$\mathbf{B} = \begin{bmatrix} 0.0001 & 0.0003 \\ -0.2560 & -0.0044 \\ -0.0133 & -0.0292 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}, \qquad \mathbf{C} = \boldsymbol{I}_5 \tag{4}$$

Problem 1: Open-loop Stability Analysis

The purpose of Problem 1 is to analyze the aircraft in open loop to understand the capabilities of the aircraft. The stability properties of the aircraft will be analyzed using the characteristic equation and eigenvalues. Recommend reading is Beard & McLain (2012, Section 5.6) and Fossen (2022). The tasks are as follows:

1a (9 points)

Run the script `openLoop.m` to load the aircraft model into the workspace. The script loads the system matrices and presents some useful results. Analyze these results and provide numerical answers to the following questions:

Based on the bode plot generated by the `openLoop.m` script, what is the roll angle cross-over frequency? (numerical answer)

Answer: 0.202

What is the course angle cross-over frequency? (numerical answer)

Answer: 0.0901

Based on this, what value would you choose for the bandwidth of the inner roll loop when designing a bank-to-turn autopilot system?

Answer: The inner roll loop bandwidth should be 5-10 times higher than the outer loop bandwidth, when designing a bank-turn-autopilot.

1b (16 points)

The Matlab commands `eig(A)` and `damp(A)` are used in `openLoop.m` to provide analysis and visualization of the lateral aircraft modes. Use this analysis to give numerical answers to the following questions:

What is the Dutch-roll natural frequency? (numerical answer)

Answer: 2.44

What is the Dutch-roll relative damping ratio? (numerical answer)

Answer: 0.107

Can you, very briefly, in your own words, describe how the Dutch roll mode affects the yaw and roll motion? How would the motion change with an increased relative damping ratio?

Answer: An aircraft in Dutch roll mode is moving around both the yaw and roll axis at the same time. A yaw movement induces a roll in the opposite direction, and a roll movement induces a yaw in the opposite direction. By increasing the relative damping ratio, the stabilization of the roll and yaw motions happens quicker, because the oscillations becomes more damped when increasing the relative damping ratio.

What is the eigenvalue of the spiral-divergence mode, and is the mode stable/unstable?

Answer: e = - 0.00486, stable because it is in the left half plane

What is the eigenvalue of the roll mode? Is it faster or slower than the spiral-divergence mode?

Answer: f = -1.02, faster than the spiral-divergence mode because the negative real part is lagrer

1c (10 points)

Write the m-file function: `[alpha,beta,Va,Vg,Vw] = windTriangle(v_g,v_w)` where `alpha` is the angle of attack (deg), `beta` is the sideslip angle (deg), `Va` is the airspeed (m/s), `Vg` is the ground speed (m/s), and `Vw` is the wind speed (m/s). Before uploading the m-file windTriangle.m to Blackboard you can test the function using:

```
v_g = [257.9 0 0]';       % ground velocity vector expressed in BODY [m/s]
v_w = [14.3 0.5 -0.3]';  % wind velocity vector expressed in BODY [m/s]
[alpha,beta,Va,Vg,Vw] = windTriangle(v_g,v_w)
alpha =
    0.0706
beta =
   -0.1176
Va =
  243.6007
Vg =
  257.9000
Vw =
   14.3119
```

1d (5 points)

Explain briefly the physical meaning of the outputs `alpha_0` and `beta_0` below. What do these values tell us about the state of the aircraft?

```
v_w = [0 0 0]';
[alpha_0,beta_0,Va,Vg,Vw] = windTriangle( v_g, v_w )
```

Answer: Alpha_0 (angle of attack) and beta_0 (side-slip angle) defines the relative orientation of the body, stability and wind coordinate frames.

When v_w = [0 0 0]´ indicating no wind, it can be seen from the wind-triangle that Va and Vg coincides. Then, there is no difference in air speed and ground speed. We also have that v_g only have one x-component, meaning that the aircraft is flying straight. The outputs from the windtriangle when using these vectors, yields alpha_0 = 0, and beta_0 = 0. This emphasizes that the plane is flying straight and has no movement either upwards or sideways.

Problem 2: Lateral Autopilot Design for Simultaneous Sideslip and Roll Attitude Hold

The decoupled model for the roll angle ϕ with aileron δ_a as input is obtained from the state-space model according to (Beard & McLain 2012, Section 5.4)

$$\dot{\phi} = p \tag{5}$$
$$\dot{p} = a_{21}\beta + a_{22}p + a_{23}r + b_{21}\delta_a + b_{22}\delta_r \tag{6}$$

where a_{ij} and b_{ij} correspond to the elements ij in the matrices $\mathbf{A}$ and $\mathbf{B}$, respectively. By assuming that $\beta = r = 0$ and $\delta_r = 0$, the roll rate differential equation becomes

$$\dot{p} \approx a_{22}p + b_{21}\delta_a \tag{7}$$

The corresponding transfer function is

$$h_{\phi\delta_a}(s) = \frac{\phi}{\delta_a}(s) = \frac{a_{\phi_2}}{s(s + a_{\phi_1})} \tag{8}$$

where

$$a_{\phi_1} = -a_{22} = 1.2523 \tag{9}$$
$$a_{\phi_2} = b_{21} = -0.2560 \tag{10}$$

The Matlab script `closedLoop.m` will be used to run the aircraft simulator. The main simulation loop is shown below:

```
%% MAIN LOOP
x = [0 0 0 0 0]';                             % initial states and commands
phi_int = 0; chi_int = 0; beta_int = 0;
phi_c = 0; chi_c = 0;

simdata = zeros(N+1,13);                      % table of simulation data

for i = 1:N+1

    t = (i-1) * h;                            % time (s)

    % Measurements: y[k]
    beta = x(1); p = x(2); r = x(3); phi = x(4); psi = x(5);
    chi = psi;        % assumes that beta = 0

    % Autopilot test cases
    if n == 0

        delta_a_c = 25 * sin(t/10) * D2R;
        delta_r_c = -40 * sin(t/10) * D2R;

    elseif n == 1   % roll attitude and sideslip hold

        if t < 30
            phi_ref = 0;
        elseif t >= 30 && t < 50
            phi_ref = 15 * D2R;
            phi_c = exp(-h/5) * phi_c + (1 - exp(-h/5)) * phi_ref;
        elseif t >= 50 && t < 150
            phi_ref = 5 * D2R;
            phi_c = exp(-h/5) * phi_c + (1 - exp(-h/5)) * phi_ref;
       else
           phi_c = exp(-h/5) * phi_c;
       end
       % ************************* CALL TO YOUR FUNCTIONS ***************************
       [delta_a_c,phi_int] = rollAttitudeHold(a_phi1,a_phi2,p,phi,phi_c,phi_int,h);
       [delta_r_c,beta_int] = sideslipHold(a_beta1,a_beta2,beta,beta_int,h);
       % ****************************************************************************

    else  % course and sideslip hold
```

```
        if t <= 30
            chi_c = 0;
        elseif t > 30 && t < 60
            chi_ref = 10 * D2R;
            chi_c = exp(-h/5) * chi_c + (1 - exp(-h/5)) * chi_ref;
        else
            chi_c = exp(-h/5) * chi_c;
        end

       % ************************** CALL TO YOUR FUNCTIONS **************************
        [phi_c,chi_int] = courseHold(chi,chi_c,Vg,chi_int,h);
        [delta_a_c,phi_int] = rollAttitudeHold(a_phi1,a_phi2,p,phi,phi_c,phi_int,h);
        [delta_r_c,beta_int] = sideslipHold(a_beta1,a_beta2,beta,beta_int,h);
       % ****************************************************************************

    end

    % Control signals
    delta_a = delta_a_c;
    if abs(delta_a_c) > delta_a_max                % aileron saturation
        delta_a = sign(delta_a_c) * delta_a_max;
    end

    delta_r = delta_r_c;
    if abs(delta_r_c) > delta_r_max                % rudder saturation
        delta_r = sign(delta_r_c) * delta_r_max;
    end
    u = [ delta_a delta_r ]';

    % Store simulation data in a data table
    simdata(i,:) = [t psi chi beta phi p r...
        delta_a delta_a_c delta_r delta_r_c phi_c chi_c];

    % Propagate states to x[k+1] by Euler's method
    d  = -0.1176 * [ A(1,1) A(2,1) A(3,1) 0 0]' * D2R;    % disturbance
    x = x + h * ( A * x + B * u + d );                     % aircraft model

end
```

The disturbance `d = -0.1176 * [A(1,1)A(2,1)A(3,1)0 0]' * D2R` will shift the equilibrium point of β with -0.1176 degrees. This is equivalent to the β value obtained for the wind load `v_w = [14.3 0.5 -0.3]''` in the open-loop stability analysis. When testing without integral action, you can multiply the disturbance with zero to avoid drift and tune your PD controller.

There are three test-cases in the `closedLoop.m` file that can be run by setting the parameter $n = 0, 1, 2$.

```
% n = 0: open-loop test (just for tesing)
% n = 1: roll attitude and sideslip hold (Problem 2a)
% n = 2: course and sideslip hold (Problem 2b)
n = 1
```

2a (20 points)

For this problem, you should set $n = 1$ in the header of `closedLoop.m`.

Follow the successive-loop closure design method in Beard & McLain (2012, Section 6.3) where the closed-loop system is described by (assuming $k_{i_\phi} = 0$)

$$H_\phi(s) = \frac{k_{p_\phi} a_{\phi_2}}{s^2 + (a_{\phi_1} + a_{\phi_2} k_{d_\phi})s + k_{p_\phi} a_{\phi_2}} = \frac{\omega_{n_\phi}^2}{s^2 + 2\zeta_\phi \omega_{n_\phi} s + \omega_{n_\phi}^2} \tag{11}$$

Compute the PID gains using the method of Beard & McLain (2012, Section 6.3). Check that the natural frequency of the inner loop is close to the cross-over frequency obtained from Problem 1. Furthermore, it is important that the inner loop has a much higher frequency than the outer loop. If this is not fulfilled, the approximation of the inner loop as a unity gain will be violated. Finally, implement the PID controller in the Matlab function.

```
function [delta_a_c,phi_int] = rollAttitudeHold(a_phi1,a_phi2,p,phi,phi_c,phi_int,h)

% Add your controller gains and PID roll attitude controller here
delta_a_c = 0;

% Euler's method: phi_int[k+1]
phi_int = phi_int + h * (phi_c - phi);

end
```

where `delta_a` is the aileron (rad), `a_phi1` and `a_phi2` are the transfer function parameters, `phi` is the measured roll angle (rad), `phi_c` is the roll angle command (rad) and `h` is the sampling time in seconds used to propagate the integral state `phi_int`. Save the function as a m-file function rollAttitudeHold.m and upload the function to Blackboard.

Hint: *When designing the inner loop, choose* $\delta_a^{max} = 21°$, $e_\phi^{max} = 15°$ *and* $\zeta_\phi = 1.0$. The PD controller should work before you add integral action, which is needed to remove steady-state errors due to wind.

Note that the root-locus method in Beard & McLain (2012, Section 6.3) can be used to tune the integral gain. You can also use the SISO pole-placement method in Fossen (2021), that is $k_{i_\phi} = k_{p_\phi} w_{n_\phi}/10$. However, you will also get a maximum score by trial and failure.

2b (20 points)

The next step is to minimize the sideslip angle β such that the bank-to-turn equation is valid. For this purpose, design a PI controller (you must find the parameters yourself, e.g., by using the method of Beard & McLain, 2012),

```
function [delta_r_c, beta_int] = sideslipHold(a_beta1,a_beta2,beta,beta_int,h)

% Add your controller gains and sideslip PI controller here
delta_r_c = 0;

% Euler's method: beta_int[k+1]
beta_int = beta_int + h * beta;

end
```

Finally, run both controllers and check that $\beta \to 0$ for a nonzero disturbance. This requires a nonzero value for δ_r. At the same time, the roll angle ϕ should converge to the command ϕ_c. You are supposed to upload the m-file functions rollAttitudeHold.m (Problem 2a) and sideslipHold.m (Problem 2b) and the Matlab plot rollAndSideslip.png (Problem 2b) on Blackboard.

Problem 3: Lateral Autopilot Design for Course Hold

For this problem, you should set $n = 2$ in the header of `closedLoop.m`.

It is assumed that you have solved Problems 2a and 2b such that your autopilots `rollAttitudeHold` and `sideslipHold` work satisfactorily. Then the next step is to add a third autopilot for generation of the command `phi_c`, which is an input to `rollAttitudeHold` according to:

```
[phi_c,chi_int] = courseHold(chi,chi_c,Vg,chi_int,h);
[delta_a_c,phi_int] = rollAttitudeHold(a_phi1,a_phi2,p,phi,phi_c,phi_int,h);
[delta_r_c,beta_int] = sideslipHold(a_beta1,a_beta2,beta,beta_int,h);
```

In some cases, it may be beneficial to remove the integral gain in the inner loop because it can lead to delay and instability when designing an outer loop with integral action. Choose the solution which is best for your setup and tuning parameters. In fact, both approaches should work. The control objective is to regulate χ to χ_c using successive loop closure.

3a (20 points)

The outer-loop course hold controller should be implemented in the Matlab function

```
function [phi_c,chi_int] = courseHold(chi,chi_c,Vg,chi_int,h)

% Add your controller gains and course hold PI controller here
phi_c = 0;

% Euler's method: chi_int[k+1]
chi_int = chi_int + h * ssa(chi_c - chi);

end
```

where `chi` is the measured course angle (rad) and `chi_c` is the course angle command (rad). You are supposed to upload the m-file courseHold.m and the corresponding Matlab plot course.png on Blackboard.

References

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

Fossen, T. (2021). *Handbook of Marine Craft Hydrodynamics and Motion Control*, 2nd edn, John Wiley & Sons.

Fossen, T. I. (2022). Aircraft Notes, Lecture notes in TTK4190 Guidance, Navigation and Control of Vehicles.