# AE4-311
# Nonlinear and Adaptive Flight Control

## *Take Home Assignment 1*

## Fault Tolerant Flight Control
## Using Model and Sensor based Nonlinear Dynamic Inversion

(June 1, 2024)

Assignment submission deadline: 8 September 2024

**Problem statements:**

This assignment is about Fault Tolerant Flight Control (FTFC) using two nonlinear adaptive control approaches: Adaptive Nonlinear Dynamic Inversion (ANDI) and Incremental Nonlinear Dynamic Inversion (INDI). ANDI is an indirect model based control approach which involves aircraft online model identification, using the two-step approach learnt in the course on System Identification of Aerospace Vehicles, AE4-320, and Nonlinear Dynamic Inversion (NDI), learnt in the course of Advanced Flight Control, AE4-311. INDI is a sensor based nonlinear adaptive control approach learnt in the course of Advanced Flight Control. This assignment must be done with the Cessna Citation simulation model supplied in Simulink, and this model can be steered by means of a joystick. In this assignment, a manual version of an FTFC algorithm will be developed and will be evaluated for one failure scenario.

As learnt in AE4-320, the two step method consists of two major steps, namely aircraft state estimation and aerodynamic model identification. Estimation of aircraft states is done by making use of the kinematic and observation model, based upon redundant but contaminated information from all sensors (air data, inertial, magnetic and GPS measurements). For this purpose, a Kalman Filter is used. After the aircraft flight trajectory has been estimated through the aircraft states, the aerodynamic model identification is done in the subsequent step. The aerodynamic model identification can be performed with the Recursive Least Squares Approach.

Since the implementation of the two step method has been performed already in the assignment of AE4-320 in a batch setup, this assignment will ignore the Kalman Filter for the state estimation step. This means that the available sensor information in the to be used Cessna Citation simulation model can be considered as <u>not contaminated</u>. Mind that this is an idealization of the reality, but it is better to focus on new developments rather than repeating the design of algorithms which has already been done before.

The INDI approach uses the angular velocity measurements to reconstruct the angular acceleration body components which are used in the inner-loop rate NDI controllers.

This assignment consists of 8 major steps:

1. Initialization of the Cessna Citation 550 simulation model, setup of the FlightGear visualization software.
2. Implement the specified failure dynamics and the given joystick control input in the simulation model.
3. Implement the Aerodynamic Model Identification algorithm in the simulation model
4. Design and implement a monitoring algorithm that triggers the reset switch in the Aerodynamic Model Identification step after a failure has occurred
5. Design and implement a manual adaptive nonlinear dynamic inversion algorithm for the angular rates in roll, pitch and yaw.
6. Analyse and compare the handling qualities of the aircraft with classical and ANDI based fault tolerant control, before and after activation of the failure.
7. Design the INDI inner-loop rate controller without using the system identification in the control loop.
8. Analyse and compare the handling qualities of the aircraft with INDI based fault tolerant control, before and after activation of the failure.

Aircraft data are listed as follows

Ixx = 11187.8 kg.m$^2$
Iyy = 22854.8 kg.m$^2$
Izz = 31974.8 kg.m$^2$
Ixz = 1930.1 [kg.m$^2$]
wing span: b = 13.3250 m
wing area: S = 24.9900 m$^2$
chord: c = 1.9910 m

The air density is dependent of the altitude and can be calculated on board as follows:

$R = 287.05\ m^2/s^2K$ $\quad T_0 = 288.15\ K$
$\lambda_{trop} = -0.0065\ K/m$ $\quad \rho_0 = 1.225\ kg/m^3$
$h_0 = 0\ m$ $\quad h_{tropopauze} = 11\ km$

$$T_{tropopauze} = T_0 + \left(h_{tropopauze} - h_0\right)\lambda_{trop} \qquad [K]$$

$$\rho_{tropopauze} = \rho_0 \left(1 + \lambda_{trop}\frac{h_{tropopauze}}{T_0}\right)^{\left(-\frac{g}{R\lambda_{trop}}-1\right)} \qquad \left[kg/m^3\right]$$

$h_{stratopauze} = 20\ km$

$$\rho_{stratopauze} = \rho_{tropopauze} \cdot e^{\left(-\frac{g}{RT_{tropopauze}}\left(h_{stratopauze}-h_{tropopauze}\right)\right)} \qquad \left[kg/m^3\right]$$

In the troposphere $\left(h \le h_{tropopauze}\right)$:

$$\rho = \rho_0 \left(1 + \lambda_{trop} \frac{h}{T_0}\right)^{\left(-\frac{g}{R\lambda_{trop}} - 1\right)} \qquad \left[kg/m^3\right] \tag{1.1}$$

In the stratosphere $\left(h > h_{tropopauze}\right)$:

$$\rho = \rho_{tropopauze} + \rho_{tropopauze} \cdot e^{\left(-\frac{g}{RT_{tropopauze}}\left(h - h_{tropopauze}\right)\right)} \qquad \left[kg/m^3\right] \tag{1.2}$$

# 1. Initialization of the Cessna Citation 550 model, setup the FlightGear visualization software.

To simulate an aircraft model, we make use of the Cessna Citation 550 simulink model. For visualization we make use of the open-source simulator software called FlightGear.

**Intialize the Cessna Citation 550 simulink model**

This assignment requires Matlab/Simulink. We will use a model of the Cessna Citation 550 bravo contained in the "Citation_FlightGear_v1.mdl" Simulink model. Before being able to run the simulation, you should run the script "initcit.m" which initializes all required aircraft constants, and loads a trim file.

**Setup the FlightGear visualization software**

To help you assess the validity of your simulation and later your controller, you can make use of the free to download (open-source) flight simulation package FlightGear. You can download the latest version of the simulator here: https://www.flightgear.org/download/ . The current assignment was tested on FlightGear version 2020.3 and earlier. Because the Simulink model communicates with FlightGear using network packages, you can also run the simulation without FlightGear being started (or even installed).

Before being able to visualize your simulation, you need to modify the file "runfg_c172.bat" such that the directories in the .bat file match the install directories of your FlightGear installation. See "EXAMPLE_runfg_c172.bat" for an example where FlightGear was installed in the C:\Program Files\FlightGear2020.3\ directory. In this case we make use of the Cessna C172 aircraft which is included in the stock install, but of course you can also download and use your own models. In that case you need to change the "--aircraft=c172p" command in the .bat file to mirror your particular aircraft model. When you run the .bat file, the FlightGear should startup and should be ready to receive network packages from your Simulink model.

## 2. Implementation of the specified failure dynamics and adaptation of the given joystick control input in the simulation model,

The simulation model as provided contains a specific block in order to modulate the joystick input for this aircraft. This block involves a.o. a correction for biases and scaling factors in the joystick sensors. Sometimes, even a dead-zone is required if your joystick gives occasionally undesired small inputs in its neutral position. It is very important to modify these modulations for your specific joystick. The provided Simulink file `Joystickcalibtest.mdl` can help you for this purpose.

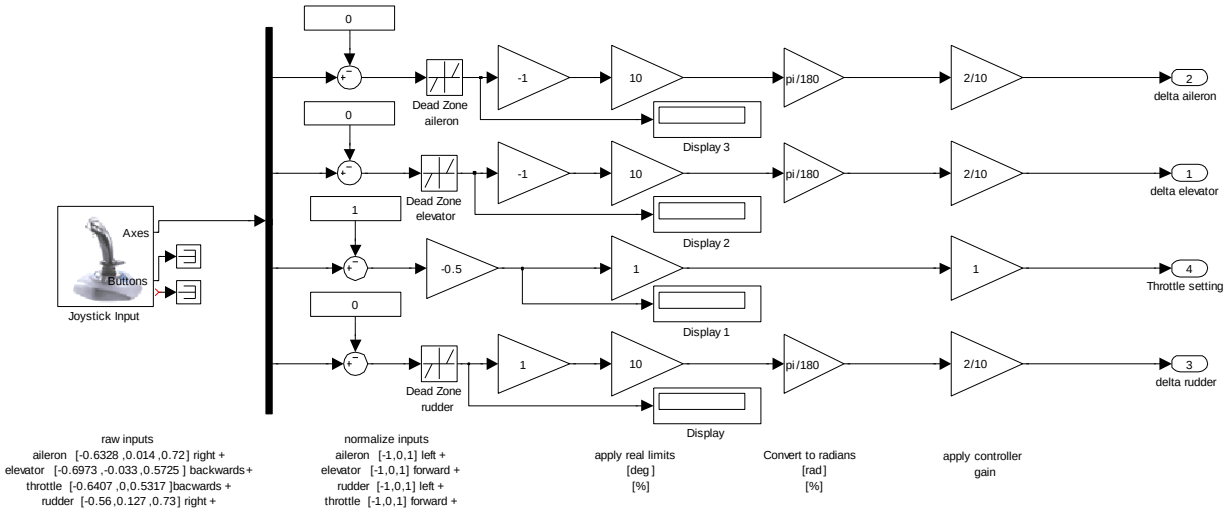


**Figure 1: Lay out of the Joystickcalibtest.mdl Simulink model.**

The failure under consideration for this scenario is the so-called aileron hardover. One of the ailerons deflects to an extreme position and gets stuck, for example due to a failure in the hydraulics. This failure dynamics need to be implemented in the simulation model, but first they have to be modelled realistically.

If one aileron gets stuck while one remains operative, the pilot will ”feel” an aileron channel that is half as effective as before. If in an even worse case, the aileron gets stuck in a non-neutral position, the aircraft will start rolling to one side, which needs to be compensated for by the aileron that is still functional. Since the individual ailerons are not present in the aircraft model, the scenario of one aileron stuck has to be implemented in a single aileron control channel.

For unfailed situation the total aileron deflection $\delta_a$ is calculated as follows:

$$\delta_{a_{\text{actual}}} = \delta_{a_{\text{left}}} + \delta_{a_{\text{right}}} = \delta_{a_{\text{desired}}} \tag{1.3}$$

If we assume that the right aileron gets stuck in a non-neutral position, the command for aileron deflection $\delta_a$ is now disturbed in the following way:

$$\delta_{a_{\text{actual}}} = \delta_{a_{\text{left}}} + \delta_{a_{\text{offset}}} = \frac{1}{2}\delta_{a_{\text{desired}}} + \delta_{a_{\text{offset}}} \tag{1.4}$$

Figure 2 schematically depicts this aileron failure. $\delta_{a_{\text{offset}}}$ represents the angle at which the aileron gets stuck.

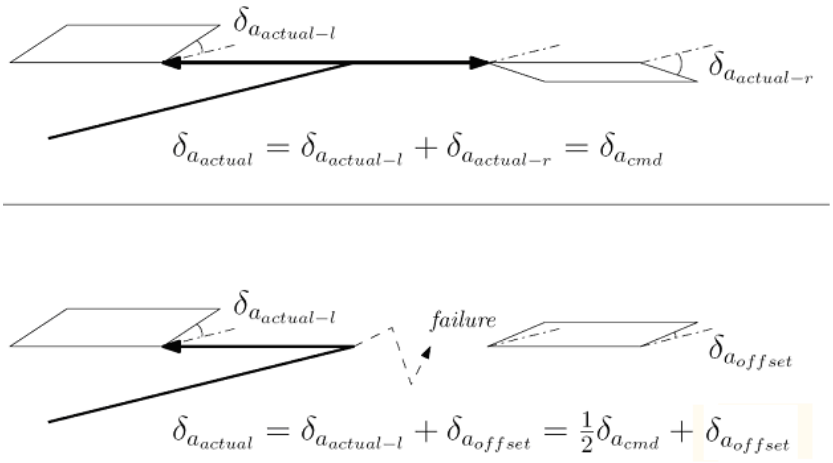


**Figure 2: One aileron stuck**

This failure can be represented in the actuators block based upon the dynamics described in eq.(1.4).

## Assignment

Modify the joystick input biases and scaling factors for your specific joystick
Implement the failure dynamics as specified and illustrated above with $\delta_{a_{\text{offset}}} = -0.30 rad \left(= -17.19°\right)$ in the actuators block "under the mask" and explain the expected influence on the to-be-identified parameters.

Note that if you do not have a joystick available, then comment out the "JoyStick" block in the "Pilot" block using Ctrl+Shift+u.

*Hints:*

- *If you run the Simulink model without any modifications and there is an error regarding the block "sfun_rttime", just remove the block "sfun_rttime".*
- *If you get an error stating that the demux block just in front of the Cessna Citation 500 model has an incorrect dimension, you have not connected a joystick to the computer. If you do not have a joystick, then in the "**Pilot**" block switch the manual "**Input Selector Switch**" to the "**Virtual joystick input**" signal, and comment out the "**Joystick**" block using Ctrl+Shift+u. Note that in that case, you have to design your own input sequences using the signal generators in the "**Virtual Joystick**" block.*
- *To get into the model of the Citation in the Simulink file: right click and choose "look under mask". The failure dynamics have to be implemented in the actuators block under the mask, since the aileron failure takes place in the servo actuator.*

- *Some joysticks may have more than 4 axes, depending on all features included in the joystick. Make sure that you adjust the Joystick input accordingly, by adjusting the demux block or the number of terminators.*
- *Bear in mind that the joystick axes may not be in the good order, depending on your type of joystick.*
- *An activation switch, e.g. to activate the failure dynamics, can be made using a so-called switch button on your joystick. This can be done by means of a so-called "Triggered Subsystem" block in the "Ports and subsystems" folder of the Simulink library browser. You can also use the Simulink file "*`switchonoff.mdl`*".*

## 3. Implementation of the Aerodynamic Model Identification algorithm

The aerodynamic force components can be calculated with

$$C_X = \frac{X}{\frac{1}{2}\rho V^2 S} = \frac{mA_x}{\frac{1}{2}\rho V^2 S};\ C_Y = \frac{Y}{\frac{1}{2}\rho V^2 S} = \frac{mA_y}{\frac{1}{2}\rho V^2 S};\ C_Z = \frac{Z}{\frac{1}{2}\rho V^2 S} = \frac{mA_z}{\frac{1}{2}\rho V^2 S} \tag{1.5}$$

The aerodynamic moment components can be calculated with

$$\begin{aligned} C_l &= \frac{L}{\frac{1}{2}\rho V^2 Sb} = \frac{\dot{p}I_{xx} + qr\left(I_{zz} - I_{yy}\right) - \left(pq + \dot{r}\right)I_{xz}}{\frac{1}{2}\rho V^2 Sb} \\ C_m &= \frac{M}{\frac{1}{2}\rho V^2 S\bar{c}} = \frac{\dot{q}I_{yy} + rp\left(I_{xx} - I_{zz}\right) + \left(p^2 - r^2\right)I_{xz}}{\frac{1}{2}\rho V^2 S\bar{c}} \\ C_n &= \frac{N}{\frac{1}{2}\rho V^2 Sb} = \frac{\dot{r}I_{zz} + pq\left(I_{yy} - I_{xx}\right) + \left(qr - \dot{p}\right)I_{xz}}{\frac{1}{2}\rho V^2 Sb} \end{aligned} \tag{1.6}$$

The symmetric aerodynamic forces and moment can be modeled as:

$$\begin{aligned} C_X &= C_{X_0} + C_{X_\alpha}\alpha + C_{X_{\alpha^2}}\alpha^2 + C_{X_q}\frac{q\bar{c}}{V} + C_{X_{\delta_e}}\delta_e \\ C_Z &= C_{Z_0} + C_{Z_\alpha}\alpha + C_{Z_q}\frac{q\bar{c}}{V} + C_{Z_{\delta_e}}\delta_e \\ C_m &= C_{m_0} + C_{m_\alpha}\alpha + C_{m_q}\frac{q\bar{c}}{V} + C_{m_{\delta_e}}\delta_e \end{aligned} \tag{1.7}$$

The asymmetric aerodynamic force and moments can be modeled as:

$$\begin{aligned} C_Y &= C_{Y_0} + C_{Y_\beta}\beta + C_{Y_p}\frac{pb}{2V} + C_{Y_r}\frac{rb}{2V} + C_{Y_{\delta_a}}\delta_a + C_{Y_{\delta_r}}\delta_r \\ C_l &= C_{l_0} + C_{l_\beta}\beta + C_{l_p}\frac{pb}{2V} + C_{l_r}\frac{rb}{2V} + C_{l_{\delta_a}}\delta_a + C_{l_{\delta_r}}\delta_r \\ C_n &= C_{n_0} + C_{n_\beta}\beta + C_{n_p}\frac{pb}{2V} + C_{n_r}\frac{rb}{2V} + C_{n_{\delta_a}}\delta_a + C_{n_{\delta_r}}\delta_r \end{aligned} \tag{1.8}$$

It should be noted that the thrust is not included in the aerodynamic models. This is due to the fact that in this assignment, only the attitude angles or rates need to be controlled. If airspeed also needs to be controlled, then the thrust needs to be included in the aerodynamic models. However, this is beyond the scope of this assignment.

The measured control surface deflections are available in the simulation model, they can be found “under the mask”. Mind that you have to measure the control deflections before the failure block. Mind also that the specific forces in the output in the simulation model are given in g’s instead of the unit m/s$^2$.

**Assignments:**

A Recursive Least Squares approach needs to be set up in Simulink in order to identify the aerodynamic parameters.

This identification step can be incorporated in Simulink using an Embedded Matlab Function Block. A short manual to Embedded Matlab is attached.

In the RLS algorithm algebraic loops can occur, in order to prevent this, memory blocks can be used.

*Hints:*

- *Matlab will give error messages of algebraic loops. This can be corrected by adding "memory blocks" (folder "Discrete" in the Simulink Library Browser) in the paths concerned.*
- *The initial value of the covariance matrix should be a diagonal matrix with very large values on the diagonal, e.g. 10000. You have the possibility to give an m-file as initial value in the memory block, if desired.*
- *Make sure that you select the proper control surface deflections for the identification routine. You cannot use directly the joystick input, since the identification routine applies only for the aerodynamics, e.g. yaw damper[1] behaviour cannot be included in the identified model, so make sure you make the proper choice. You can do this by:*
    1. *selecting the control surface deflections under the mask, after the actuator dynamics (integrators and limits), and bringing them out of the masked block.*
    2. *selecting the joystick input directly, but add the simulated actuator dynamics thereafter before using the signal in the ID routine.*

## 4. Design and implementation of a monitoring algorithm that triggers the reset switch in the Aerodynamic Model Identification step after a failure has occurred

One option to guarantee flexibility of the identification algorithm to changing model dynamics is the use of the forgetting factor $\lambda$.

Another option is the use of other statistical monitoring metrics which trigger a resetting of the covariance matrix P in the recursive least squares procedure. This involves redefining the covariance matrix P in its original diagonal structure $P_0$. As a consequence, the estimated parameters, which were gradually "frozen" during the identification process, are "melted" again and are sensitive for new changes in the independent variables.

Valid statistical metrics are:

- The autocorrelation function:

[1] Be careful with the use of the yaw damper: this function is necessary for manual control, but is taken over by sideslip feedback for NDI and classic rate control.

$$\pi_{k_{gap}} = \frac{1}{N}\sum_{t=0}^{N-k_{gap}} \Delta(t)\Delta\left(t+k_{gap}\right) \tag{1.9}$$

where $k_{gap}$ is a positive non-zero integer which needs to be selected. More information about the autocorrelation function can be found in Stoica's paper which is supplied with the assignment material.

- The moving average of the square innovation:

$$\bar{\Delta}(k) = \frac{1}{n_{\text{av}}}\sum_{i=0}^{n_{\text{av}}} \Delta(k-i)^2 \tag{1.10}$$

where $n_{\text{av}}$ is the number of samples over which this average is taken (a proper range is 25-100 data samples)

- A spectral analysis:

$$\sigma_{\Delta}^2 = \frac{1}{N-1}\sum_{k=1}^{N}\left(\Delta(k)-\bar{\Delta}\right)^2 \qquad \text{with:} \qquad \bar{\Delta} = \frac{1}{N}\sum_{k=1}^{N}\Delta(k) \tag{1.11}$$

In all these equations, the innovation of the estimated aerodynamic model Δ(k) is defined as follows:

$$\Delta(k) = z(k) - \mathbf{A}(k)\underline{\hat{\theta}}_{RLS}(k) \tag{1.12}$$

in which *z(k)* is the state measurement from the actual aircraft, **A***(k)* is the data matrix and $\underline{\hat{\theta}}_{RLS}(k)$ is the vector of estimated parameters.

**Assignments:**

Implement the forgetting factor λ as well as the different statistical metrics, and analyse their performance in the presence of the failure. Also, analyse their sensitivity for so-called false alarms. In order to make a fair comparison, also include a portion of flight resembling a cruise condition, i.e. straight flight with no inputs.

After comparing the different metrics, select the most appropriate one and set up the resetting trigger for the covariance matrix P as explained earlier. These resetting triggers must be defined individually for the six separate channels X, Y, Z, L, M and N. E.g. for an asymmetric failure only Y, L and N need to be re-identified, the others can be kept invariant.

## 5. Design and implementation of a manual adaptive nonlinear dynamic inversion algorithm for the angular rates in roll, pitch and yaw.

A manual control loop needs to be designed for this part. This loop controls the angular rates in pitch roll and yaw via the aerodynamic moments.

**Assignments:**

Rewrite the rotational equation of motion given:

$$\mathbf{M} = \mathbf{I}\dot{\omega} + \omega \times \mathbf{I}\omega$$

$$\text{with: } \mathbf{M} = \begin{bmatrix} L \\ M \\ N \end{bmatrix} = \frac{1}{2}\rho V^2 S \begin{bmatrix} bC_L \\ \bar{c}C_M \\ bC_N \end{bmatrix}, \mathbf{I} = \begin{bmatrix} I_{xx} & 0 & -I_{xz} \\ 0 & I_{yy} & 0 \\ -I_{xz} & 0 & I_{zz} \end{bmatrix}, \omega = \begin{bmatrix} p \\ q \\ r \end{bmatrix} \tag{1.13}$$

by expanding the dimensionless moment coefficient in their first order Taylor expansion as given in eq. (1.7) and (1.8). Defining the angular accelerations $\dot{\omega}$ as virtual input vector $v$ for the NDI loop allows to rewrite eq. (1.13) in the form:

$$u = \mathbf{b}^{-1}(x)\left[\nu - \mathbf{a}(x)\right], \qquad \text{where: } u = \left[\delta_a \quad \delta_e \quad \delta_r\right]^T \tag{1.14}$$

Give the definition of **a**($x$) and **b**($x$) in eq. (1.14) in terms of the variables defined in eq. (1.13). Explain the limitations imposed by the structure of eq. (1.14). As shown during the lecture, this control law renders the closed loop behaviour between the virtual input vector $v$ and the aircraft states $x$ de facto into a pure single integrator. As a result, a linear controller can be defined fairly straightforward in order to control this integrator, as illustrated in Figure 3.

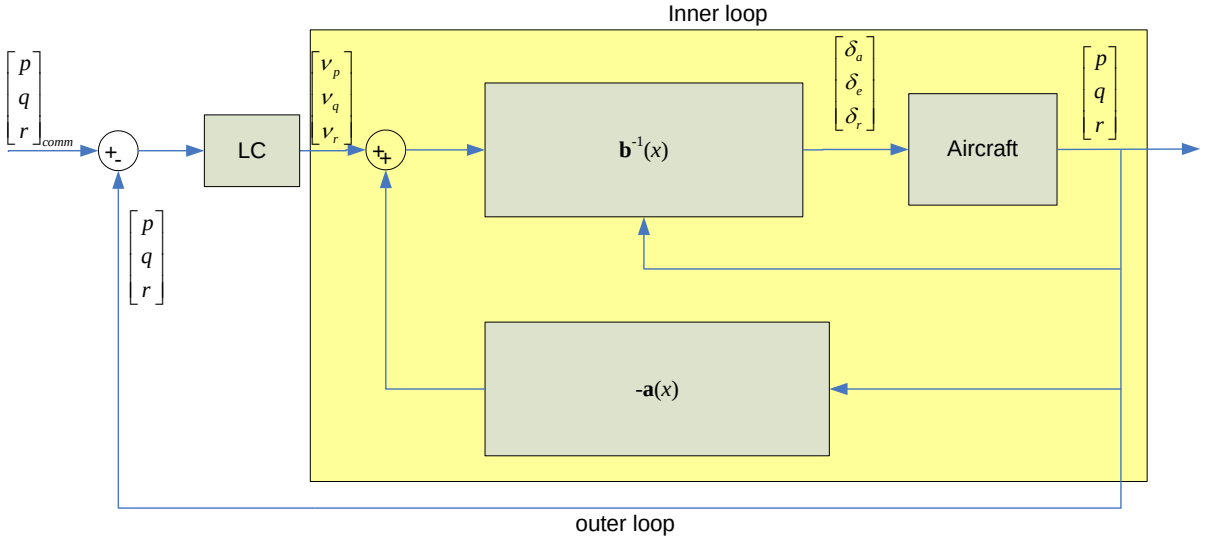


**Figure 3: NDI-loop structure for the Fault Tolerant Controller**

Implement the controller structure shown by Figure 3 in the simulation model via Embedded Matlab, where the linear controller is a PI-controller, tuned in order to achieve comfortable handling characteristics. All P- and I-gains can be positive non-zero integers which lie in the range between 1 and 10. The input as shown in Figure 3 can be connected to the joystick, but mind that this leads to very brusque robot-like reactions of the NDI-controlled vehicle with respect to the pilot inputs. Therefore, some first order filters should be placed between the joystick and the rest of the model. Another point of sensitivity is the control of the rudder. Pure yaw rate as pilot input to the rudder is not practical. A more realistic quantity is the sideslip $\beta$. Therefore, an outer sideslip control loop should be placed on top of the inner yaw rate loop. The reference sideslip angle $\beta_{ref}$ is then supplied by the twist handle of the joystick.

Mind that the aerodynamic derivatives, included in eq. (1.14) need to be used from the aerodynamic model identification block, implemented in part 2.

*Hints:*

- *Mind the order of the following quantities in the Simulink model: rates p, q and r and control deflections $\delta_e$, $\delta_a$, $\delta_r$, are not consistent. Make sure to re-order them accordingly.*
- *When one switches between classic rate control and NDI control, or between failure and non-failure, it is important to reset the integral action of the PI controllers in order to avoid exaggerated transient behaviour.*

## 6. Analysis and comparison of the handling qualities of the aircraft with classical and fault tolerant control, before and after activation of the failure.

This is the last step of the assignment, focusing on the validation of the algorithms implemented until now.

The student is asked to fly an aircraft trajectory similar to the example shown in Figure 4. The aileron hardover failure can be activated after the first right hand turn.

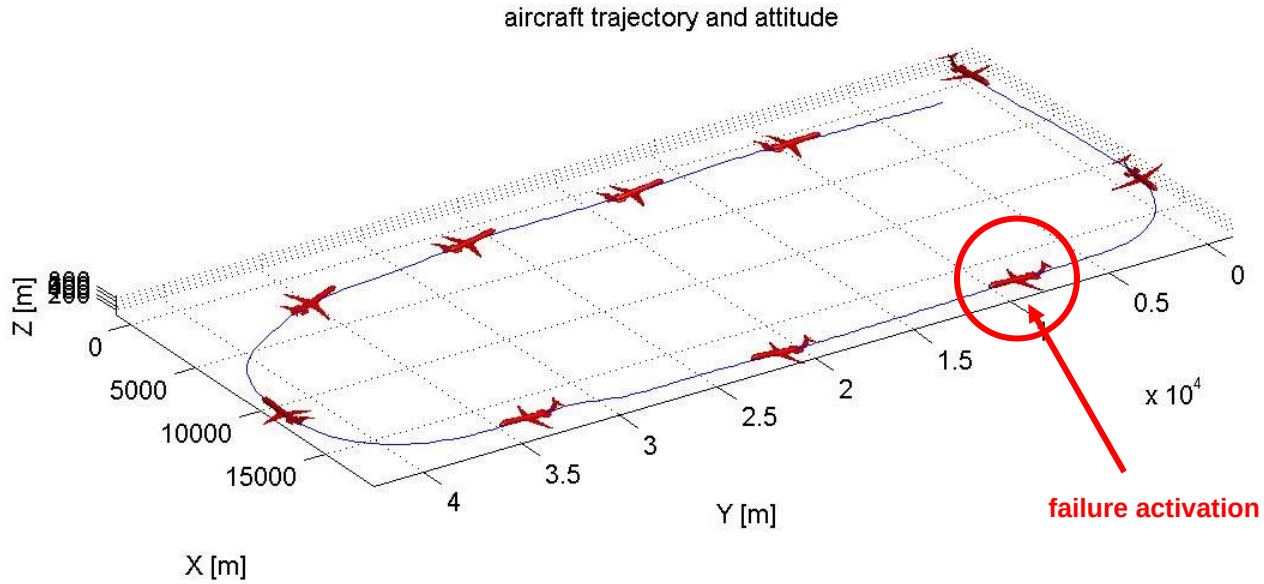


**Figure 4: Validation trajectory to be flown in Simulink by hand with joystick**

In order to evaluate the NDI-controller properly, its performance should be compared with a classical rate controller. This controller can be PI with the following gains:

- Aileron: proportional: -1; integral: -0.5
- Elevator: proportional: -2; integral: -0.1
- Rudder: proportional: -2; integral: -0.1

If these gains do not work well (the aircraft is still unstable), you can tune these gains yourself.

Make sure to switch off the yaw damper and to include beta feedback on the yaw channel, as for the NDI-control loop. Also, make sure to reset the integral action of the PI controllers in order to avoid exaggerated transient behaviour, as for the NDI-control loop.

**Assignments:**

In order to validate the performance of the fault tolerant controller, the following actions need to be performed.

Perform four simulations:

- Classical aircraft control without failure
- Classical aircraft control with failure
- Fault Tolerant Flight Control without failure
- Fault Tolerant Flight Control with failure

Show and analyse the following graphs:

1. Time histories of the statistical metric for monitoring as selected in part 4.
2. Time histories showing the convergence of the relevant aerodynamic and control derivatives before and after the failure.
3. The trajectories for the four simulation runs.
4. Time histories of the control surface deflections, and compare them with the reference signals as provided by the joystick command inputs.

**Above mentioned steps are repeated for the INDI control approach for steps 7 and 8. However, the control derivatives used by the INDI approach should be fixed (not obtained through online model identification). In reality, a prior knowledge about these derivatives is always available. In this assignment, you can use the model identification results from the adaptive NDI approach as the prior information about the control derivatives.**