Technical Note describing the joint Airbus-Stellenbosch University Industrial Benchmark on Fault Detection and Fault Tolerant Control.

IFAC World Congress 2023
--

Japie Engelbrecht, Stellenbosch University, South Africa – {jengelbr@sun.ac.za}
Philippe Goupil, Airbus, Toulouse, France – {philippe.goupil@airbus.com}
Simon Oudin, Airbus, Toulouse, France – {simon.oudin@airbus.com}

This benchmark and its associated Matlab/Simulink © model may only be used for the purpose of competing in the IFAC WC 2023 "Aerospace Industrial Benchmark on Fault Detection and Fault Tolerant Control" competition. For permission to use the benchmark outside the IFAC competition, please contact Philippe Goupil and Japie Engelbrecht.

Abstract: this technical note accompanies the Matlab/Simulink © model (any version from R2015a should work) as part of the Industrial Benchmark jointly proposed by Airbus and Stellenbosch University for the 22nd IFAC World Congress, to be held in Yokohama, Japan, July 9-14 2023. It aims at describing the industrial context of the benchmark as well as to provide the reader with the technical problem to solve. This is the second edition of this competition following the first edition organized during the 21st IFAC World Congress in Berlin, in 2020.

Additional Note: This benchmark is a competition based on the evaluation of two separate contributions: (i) **the design** (Simulink subsystem blocks to be added in the global Simulink model (see explanations below) and that shall be able to detect all the fault cases and to perform control law reconfiguration according to the requirements detailed here); (ii) a regular paper (up to 6 pages) or a discussion paper (up to 4 pages) detailing the principles of the proposed design and submitted through the classical IFAC submission system. Then the organizers will review all submitted designs and papers and proceed to the **selection of the 5 best teams**. These teams will be allowed to present their work during a special invited session during the IFAC World Congress. The best team will receive an Airbus award at the end of the invited session.

1. CONTEXT

This benchmark is dedicated to Fault Detection (FD) and Fault Tolerant Control (FTC) in the Flight Control System (FCS) of a civil commercial aircraft. For this competition, FTC means proposing a strategy for Flight Control Law Reconfiguration after FD, including control reallocation, and does not mean designing a new control law.
The FCS is part of the aircraft avionic systems and consists of all the elements located between the pilot inputs (in the cockpit) and the movable parts (the control surfaces), including these two elements. It comprises as well sensors, probes, wiring, actuators, numerical buses, power sources, etc… It is used to control the aircraft attitude, trajectory and speed. It is one of the most critical systems on-board the aircraft.
The Electrical Flight Control System (EFCS, a.k.a. Fly-By-Wire), first developed by Aerospatiale for civil aircraft and installed on Concorde (as an analogue system) and then designed with digital technology on Airbus aircraft from 1980s (A310), provides more

sophisticated control of the aircraft and flight envelope protection functions [1]. The main characteristics are that high-level control laws in normal operation allow all control surfaces to be controlled electrically and that the system is designed to be available under all circumstances. The EFCS is a critical system designed to meet very stringent requirements in terms of availability. The detection of all related failures is therefore a very important point to be considered in the aircraft design. In particular, in the context of aircraft overall optimization and their increasing size, system design objectives originating from structural loads design constraints are more and more stringent. The main issue is weight saving to improve aircraft performances (e.g. consumption, noise, range). Consequently, for system failures impacting the aircraft structure, performance of detection methods must be improved, while retaining a perfect robustness. This benchmark deals with a particular EFCS failure which has an influence on aircraft structural loads. This failure is called in literature oscillatory failure case (OFC) [2] [3]. From a control perspective, the failure case affects the quality of the airplane's closed-loop responses. In the nominal (fault-free) case, the aircraft is controlled using the so-called "Normal Law", which means the highest level of automation and protections. For some failure cases, the Normal Law minimum requirements cannot be met and reconfiguration to a less sophisticated control scheme is preferable [4]. For example, after an OFC detection, the loss of one control surface like the elevator causes a reconfiguration to "Alternate Law" (i.e. a degraded flight control law). The Alternate longitudinal controller is still a load factor scheme, but it uses simplified feedbacks and it is less demanding on actuation power (i.e. its gains are lower).

As explained in more details below (Section 7), the provided model is a simple longitudinal model which includes two control surfaces (typically the elevators installed on the horizontal tail plane of a civil aircraft) and its associated control loops (the actuator servoloop and a simplified longitudinal aircraft flight control law). The goal of this competition is two sides: (i) to detect an OFC simulated on one control surface (the left elevator), and that will lead to the deactivation of this control surface; (ii) following this FD step, then to proceed to a control law reconfiguration, i.e. to perform a transition from the Normal Law to the Alternate Law. For both steps (FD and FTC), the requirements detailed below must be satisfied (cf. Sections 5 and 6).

2. OSCILLATORY FAILURE CASES (OFC)

OFCs are mainly due to electronic components in fault mode or to a mechanical breakage. These malfunctions create an erroneous sinusoidal signal which propagate through the control loop and which may lead, under some circumstances, to an unwanted oscillation of a control surface. OFCs may generate loads on the structure when located within the actuator bandwidth. These failures, coupled with the aero elastic behavior of the aircraft, may lead to high loads or vibrations. The worst case corresponds to resonance phenomena with aircraft natural modes. It is very improbable as the OFC frequencies are uniformly distributed. But one cannot prove that it is impossible, so this case has to be covered. OFC amplitude must be contained by system design within an envelope function of the frequency. Usual monitoring techniques cannot always guarantee staying within an envelope with acceptable robustness; a specific OFC detection algorithm may be used. The capability to detect these failures is very important because it has an impact on the structural design of the aircraft. The load envelope constraints must be respected. More precisely, if OFC of given amplitude cannot be detected and "passivated" (i.e. the propagation of the fault is stopped), this amplitude must be considered for load computations. The result of this computation can lead to reinforce the structure. In order to avoid reinforcing the structure and consequently to save weight, low amplitude OFC must be detectable sufficiently early and in a robust way.

OFC signals are considered as sinusoidal signals with frequency and amplitude uniformly distributed over the frequency range [0–f_{max}] Hz, with f_{max}<20Hz. Beyond f_{max}, OFC have no significant effects because of the low-pass behavior of the actuator. For structure-related system objectives, it is necessary to detect OFC beyond the given amplitude in a given number of periods, whatever the OFC frequency. Time detection is expressed in period numbers, which

means that, depending on the failure frequency, the time really allowed for detection will not be the same. OFC is said to be 'liquid' when it results in a sinusoidal signal added to the nominal servo-loop signal or 'solid' when the sinusoidal signal replaces the nominal signal. The OFC detection methodology is supposed to take into account the specificities of these two different cases.
In this benchmark, only OFCs located in the servo-loop control of the moving surfaces are considered, that is, between the Flight Control Computer (FCC) and the control surface, including these two elements. See Figure 1 for a typical control loop architecture. OFCs can originate from input or output of the FCC, from rod sensor or from the FCC itself. Depending of the fault source the effect could be different.

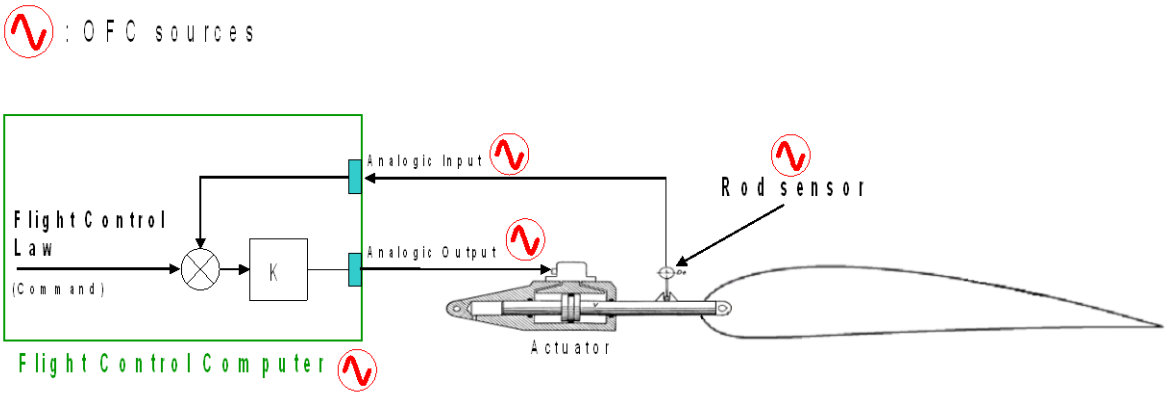

Figure 1: Control surface control loop and OFC sources

3. FLIGHT CONTROL LAW RECONFIGURATION

In normal conditions, with the EFCS the aircraft is protected against critical events [5]. The corresponding flight control law is called the ''normal law''. It requires a high level of integrity and redundancy of the computers, the peripherals (i.e. sensors, actuators and servoloop), and the hydraulics. Operation under normal laws provides flight envelope protection against excessive load factors, overspeed, stall, extreme pitch attitude and extreme bank angle. However some protection can be lost following failures, for example the loss of a control surface, IRS (Inertial Reference System), ADR (Air Data Reference) or a Flight Control Computer type. As a result of the loss of protection, there is a reversion to lower-level laws, called ''alternate laws''. Continuation of flight is still possible, but full protection of the flight envelope is no longer guaranteed. The lowest level law is the ''direct law'' where there is no protection. Manual trim of the aircraft is required. The probability of reverting to a low-level law is very small. This reconfiguration is a way to be fault tolerant and is due to a loss of hardware redundancy.

4. PRINCIPLE OF THE COMPETITION

Independent requirements for FD and FTC are given in the following sections (5 and 6) but the complete challenge to tackle shall be understood as follows. The typical maneuver that will be tested by the organizers consists of a first load factor step in Normal law, followed by the injection of an OFC once the step is completed. Then, after the OFC detection the flight control law reconfiguration must be performed and then a second load factor step allows to verify the alternate control law behavior.
All other requirements described below shall be satisfied (e.g. requirement number 3 in section 6.1 about the transition from Normal to Alternate mode in the middle of a run with a step-input on the commanded NZ). See all the comments provided in Section 7.3 as well.

5. REQUIREMENTS: OFC DETECTION SCHEME

5.1 MINIMUM REQUIREMENTS

The OFC detection scheme proposed by the reader shall satisfy the following minimum requirements:

1. OFC signals with amplitudes as small as possible shall be detected.
2. OFC signals with a fixed but unknown frequency between 1 and 10 Hz shall be detected.
3. OFC signals shall be detected within three periods of oscillation, whatever the OFC frequency.
4. Both liquid and solid failures shall be detected.
5. OFC signals at the servo current input or at the rod position sensor shall be detected.
6. The OFC detection scheme shall produce no false alarms for
    a. normal flight in the presence of no turbulence, light turbulence, moderate turbulence, and severe turbulence
    b. a load factor step input, sine input, or chirp signal input.

5.2 BONUS REQUIREMENTS

The OFC detection scheme proposed by the reader will receive an advantage if it satisfies one or more of the following bonus requirements:

7. The OFC detection scheme must be implemented, if possible
    a. as a Simulink model using only graphical components, and without using embedded Matlab code
    b. using no matrix-vector computations
    c. using no state space representations
    d. using no looping code structures, such as *for* loops, *while* loops, or *do while* loops.
8. If the OFC detection scheme contains high-level tuning parameters, a description of the tuning parameters and the tuning procedure must preferably be provided.

5.3 SUBMISSION REQUIREMENTS

The Simulink model with the proposed OFC detection scheme must be submitted with a one page description summarizing the functional blocks of the system, and a description of how the detection scheme operates.

6. REQUIREMENTS: CONTROL LAW RECONFIGURATION

6.1 MINIMUM REQUIREMENTS

The reconfiguration algorithm proposed by the competing teams shall satisfy the following minimum requirements:

1. The ‘Normal Law’ and ‘Alternate Law’ feedbacks cannot be modified by the user: the Normal Law must subtract the value of equilibrium load factor to the load factor target/feedback while the Alternate Law cannot use it for simplicity's sake. The Simulink color code is gray for “cannot be changed”.
   The rest of the blocks for control law computation (add, integrator, etc) can be modified and new blocks can be added for both laws. The Simulink color code is yellow.

2. The transition from ‘Normal Law’ to ‘Alternate Law’ shall be quick (t<3sec). We assume the airplane remains in Alternate law until the end of the scenario.
3. The airplane responses shall be bumpless during transition: no pitch departure (i.e. sudden rate of change of longitudinal state parameters), regardless of the time of transition. In other words, transition from Normal to Alternate mode shall be satisfactory even during a maneuver without elevator failure (it shall be demonstrated in the middle of a run with a step-input on the commanded NZ).
4. After OFC detection on the left elevator servocommand, the corresponding elevator order shall be quickly set to zero.
5. After OFC detection on the left elevator servocommand, the airplane response shall quickly be as close as possible to the nominal Alternate Law performance (as if 2 elevators were working). A run with step-input on the commanded NZ after reconfiguration shall demonstrate it.

6.2 BONUS REQUIREMENTS

The reconfiguration algorithm proposed by the competing teams will receive an advantage if it satisfies one or more of the following bonus requirements:

6. The law order ‘DQC’ is continuous during the transition.
7. The law order ‘DQC’ is continuous and its derivative is continuous as well during the transition.
8. The reconfiguration for control law selection & left/right elevator management are considered ‘perfect’ if performed instantaneously (1 sample/cycle). At least two solutions exist to meet this requirement and extra points will be awarded if two schemes are proposed.
9. The reconfiguration scheme must be implemented, if possible
    - as a Simulink model using only graphical components,
    - without using embedded Matlab code
    - without using ‘if’ or ’else’ statements

6.3 SUBMISSION REQUIREMENTS

The Simulink model with the proposed reconfiguration scheme must be submitted with a one page description summarizing the functional blocks of the system, and a description of how the detection scheme operates.

7. BENCHMARK

The goal of this competition is two sides: (i) to detect an OFC simulated on one control surface (the left elevator), and that will lead to the deactivation of this control surface; (ii) following this FD step, then to proceed to a control law reconfiguration, i.e. to perform a transition from the provided Normal Law to the Alternate Law. For both steps (FD and FTC), the requirements detailed above must be satisfied.

The Simulink model associated to this benchmark is a good fidelity model of a control loop, however limited to a specific flight phase (cruise), to the longitudinal axis and to a given flight point. It allows generating the position (corresponding to a given command) of two control surfaces, typically the elevators installed on the horizontal tail plane of a civil aircraft. It allows to inject OFCs originating from the actuator or from the FCC, at different frequencies and for a given amplitude. It also allows to reconfigure from a nominal control law to an alternate law.

The main goals are to develop a design (software solution) for detecting several OFC scenarios and then to perform a subsequent control law reconfiguration. The criteria for selection of the most promising designs include:

- Robustness: no false alarm during fault-free scenarios
- Performance: ability to detect predefined OFCs and detection time.
- Smooth and quick transition from the Normal law to the Alternate law.
- Complexity: computational design load and high-level parameter tuning.

7.1 THE SIMULINK MODEL

Two Simulink models are provided, namely `ofc_benchmark.slx` and `ofc_benchmark_example.slx`. The top-level diagram of both Simulink models is shown in Figure 2. The difference between the two Simulink models is that `ofc_benchmark.slx` contains an empty subsystem block for the OFC Detection Scheme, while `ofc_benchmark_example.slx` contains an example architecture for the OFC detection scheme in the subsystem block.

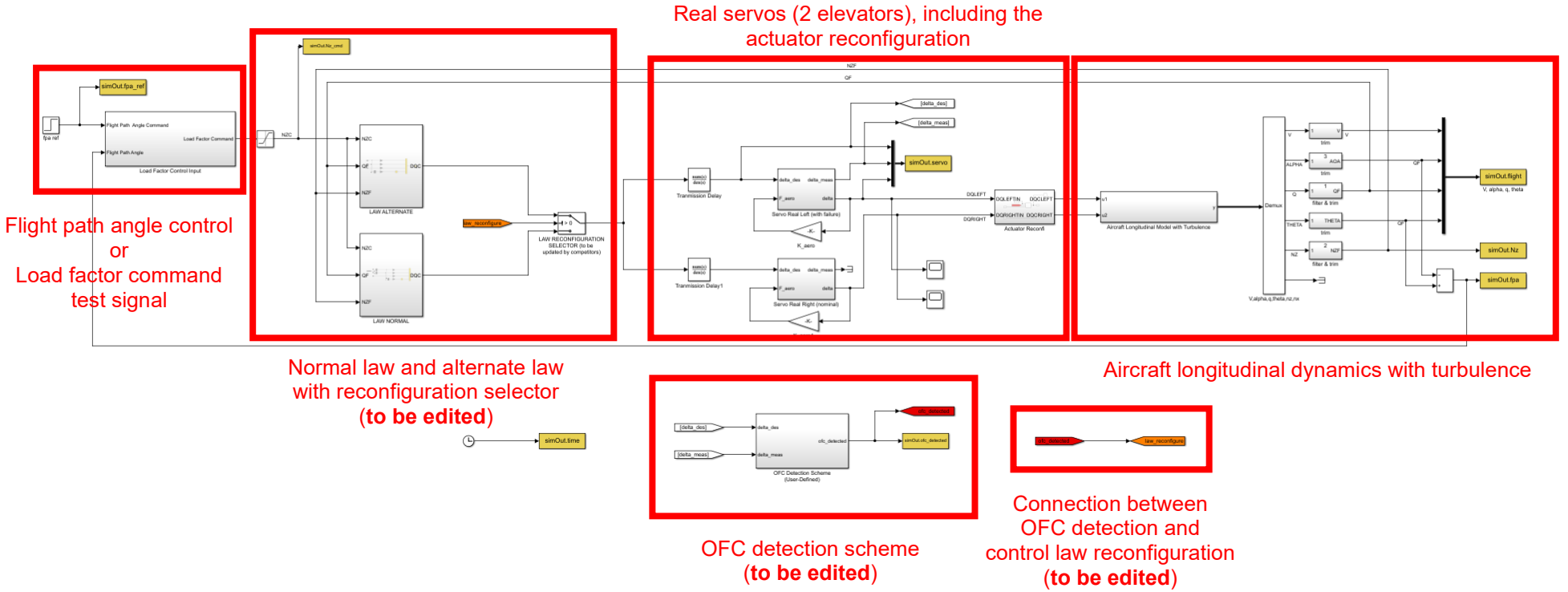

Figure 2: Simulink model for the benchmark.

The Simulink model consists of the following functional blocks:

- Real servo actuators (e.g. 2 elevators), with possibility to inject oscillatory failure on the left actuator, and to perform the actuator reconfiguration (solution to be proposed by the competing teams).
- Aircraft longitudinal dynamics, with turbulence
- Normal and alternate laws, including a law reconfiguration selector (to be updated by the competing teams).
- Flight Path Angle controller.
- OFC detection scheme

The **real servo actuator** block models the motion of the servo actuator in response to a control surface deflection command from the flight control system. It duplicates two times the same model, one for the left actuator, one for the right actuator. It also allows an OFC to be injected on the left actuator only. The oscillatory failure may be injected at either the servo current input, or at the servo rod position sensor output. The actuator model includes models of the servo hydraulics, the servo current input, the rod position sensor and control surface deflection sensor outputs, and the feedback control system that controls the rod position. The linear motion of both the servo rod and the associated deflection of the control surface are modelled. The real servo actuator receives a control surface deflection command delta_des from the load factor controller, and which is common to both actuator. It outputs the true deflection delta and the measured deflection delta_meas of the control surfaces. Internally, the servo input current (both before and after the OFC is injected, for the left actuator), the true rod position, and the rod position sensor output are recorded in the simulation.

The **aircraft longitudinal dynamics** block models the longitudinal motion of the aircraft in response to the elevator control surface deflection, and with the effect of external wind turbulence disturbances included. The aircraft longitudinal dynamics receive the elevator control surface deflections from the servo model and outputs the airspeed, angle of attack, pitch rate, pitch angle, normal load factor, and flight path angle. The wind turbulence is modelled using a Von Karman Wind Turbulence Model from the Simulink Aerospace Blockset.

The **Normal Law and Alternate Law** block models the nominal “Normal Law” and the lower-level “Alternate Law” (see explanations in Section 3 and reference herein). It receives in input the commanded load factor, from the Flight Path Angle Control block, and outputs the desired control surface deflection to the real servo block, after a “law reconfiguration selector”, to be proposed by the competing teams. Modifications are allowed in the subsystems “Law Normal” and “Law Alternate”, according to the rules defined in Section 5.1.

The **Flight Path Angle Control** block provides the load factor command to the **Normal Law and Alternate Law** block. The **Flight Path Angle Control** input block may in fact be operated either in **flight path angle control mode** or in **test signal mode**. In **flight path angle control mode**, the flight path angle is controlled for level flight by providing the load factor command using the flight path angle of the aircraft as feedback. In **test signal mode**, a load factor command test signal (step signal, sine wave, or chirp) is provided directly to the load factor controller.

The **OFC detection scheme** block is an empty block for the user to implement their OFC detection scheme. The block receives the commanded control surface deflection and the measured (left) control surface deflection as inputs, and must output an OFC detected flag. If no OFC is present, then the block must output an OFC detected value of **zero**; if an OFC is present, then the block must output an OFC detected value of **one**.

An **example OFC detection architecture** is shown in Figure 3. The scheme consists of an analytically redundant servo model, a residual generation block, and a residual evaluation block. The analytically redundant servo model receives the commanded control surface deflection and determines the modelled control surface deflection. The residual generation block subtracts the modelled deflection from the measured deflection to determine the residual signal. The residual evaluation block then evaluates the residual signal to determine whether an OFC is present. Internally, the modelled control surface deflection and the residual signal are stored.

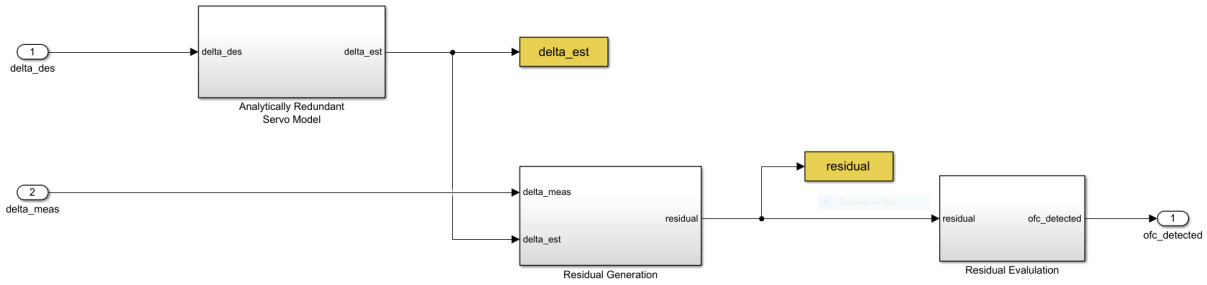

Figure 3: Example OFC detection architecture.

7.2 GETTING STARTED

1. Open the Simulink model `ofc_benchmark.slx`.
2. Initialise the simulation parameters from the Matlab command line

```
>> ofc_benchmark_init
```

3. Randomise the parameters of the real servo actuator to represent parameter variation and uncertainty from the Matlab command line

```
>> servoReal.randomiseServoParameters()
```

4. Set the location, type, amplitude, bias, frequency, phase and start time of the oscillatory failure:

   The following instructions sets a liquid oscillatory failure at the position sensor of the actuator rod, with an amplitude of 10 mm and a frequency of 1 Hz. (The bias and phase are both set to zero.) The oscillatory failure is set to start at simulation time t = 30 seconds.

```
>> ofc.setLocation('sensor')
>> ofc.setType('liquid')
>> ofc.setAmplitude(10)
>> ofc.setBias(0)
>> ofc.setFrequency(2*pi*1)
>> ofc.setPhase(0)
>> ofc.setStartTime(30)
```

5. Set the turbulence level to moderate turbulence

```
>> aircraft.setLightTurbulence()
```

6. Run the simulation

```
>> ofc_benchmark_run
```

7. Plot the simulation results

```
>> ofc_benchmark_plot
```

7.3 SIMULATION SCRIPTS, PARAMETERS, AND FUNCTIONS

7.3.1 Simulation scripts

The Matlab scripts shown in Table 1 are provided to initialise, run, and plot simulations from the Matlab command line, or from a Matlab script. For example, the `ofc_benchmark_run` script can be called from a *for loop* to execute Monte Carlo simulations.

Table 1: Simulation scripts

Script	Description
`ofc_benchmark_init`	Create the simulation objects and initialise their parameters
`ofc_benchmark_run`	Run a simulation
`ofc_benchmark_plot`	Plot the results of a simulation

The simulation parameters are stored in the `simulation` object in the Matlab workspace, and are shown in Table 2.

Table 2: Simulation parameters

Variable name	Description	Units
`simulation.stopTime`	simulation end time	seconds
`simulation.Fs`	sampling frequency	Hz

The method `setSimulinkModel` is provided to allow different Simulink models to be used with the simulation scripts. For example, the simulation scripts can be used with either `ofc_benchmark.slx` or `ofc_benchmark_example.slx`.

Table 3: Functions for setting Simulink model.

Function
`simulation.setSimulinkModel(simulink_model)`
Usage
`simulation.setSimulinkModel('ofc_benchmark')` `simulation.setSimulinkModel('ofc_benchmark_example')`

`simulink_model`	Name of the Simulink model to be used for the OFC benchmark simulations [string]

7.3.2 Simulation outputs

The simulation results are stored as variables in the `simOut` object in the Matlab workspace, as shown in Table 4 and Table 5 (for `ofc_benchmark_example.slx`).

Table 4: Simulation output: base variables

Variable name	Description	Units
`simOut.Nz`	normal load factor	g

`simOut.servo`
`1st column: delta_des`
`2nd column: delta_meas`
`3rd column: delta`
	
	control surface deflection (command)
	control surface deflection (measured)
	control surface deflection (true)
		
		deg
		deg
		deg

`simOut.Nz_cmd`	normal load factor command	g
`simOut.current`	servo current command	mA

`simOut.flight`
`1st column: Vbar`
`2nd column: alpha`
`3rd column: q`
`4th column: theta`
	
	airspeed
	angle of attack
	pitch rate
	pitch angle
		
		m/s
		deg
		deg/s
		deg

`simOut.fpa`	flight path angle	deg
`simOut.fpa_ref`	flight path angle command	deg
`simOut.precurrent`	servo current, before adding current OFC	mA
`simOut.rod_pos`	rod position (true)	mm
`simOut.rod_sensor`	rod position (measured), after adding sensor OFC	mm
`simOut.time`	Time	s
`simOut.ofc_detected`	OFC detected flag? `ofc_detected =` `0:` OFC not detected `1:` OFC detected	[0/1]

Table 5: Simulation output: additional variables (for ofc_benchmark_example.slx)

Variable name	Description	Units
`simOut.delta_est`	control surface deflection (modelled)	deg
`simOut.residual`	residual; `delta_meas - delta_est` difference between measured deflection of real servo (`delta_meas`) and modelled deflection of analytically redundant servo model (`delta_est`)	deg

7.3.3 Parameter settings: Simulation

The parameters for the simulation can be set using the functions of the `simulation` object in the Matlab workspace. The `simulation` functions are shown in Table 6, and can be used to set the time at which the OFC detected flag is triggered to force the left actuator to be deactivated and the control law to be reconfigured.

Table 6: Functions for setting simulation parameters.

Function	Description
`simulation.setOFCDetectedTime (ofcDetectedTime)`	Set the time [seconds] at which the `ofc_detected` flag is triggered to force the left actuator to be deactivated and the control law to be reconfigured.

7.3.4 Parameter settings: Oscillatory Failure Case (OFC)

The parameters for the oscillatory failure case can be set using the functions of the `ofc` object in the Matlab workspace. The `ofc` functions are shown in Table 7, and can be used to set the location, type, amplitude, bias, frequency, phase, and start time of the OFC. The function `enableRandomOFC` is provided to generate a random OFC for Monte Carlo simulations.

Table 7: Functions for setting oscillatory case parameters.

Function	Description
`ofc.disableOFC()`	Disable all OFCs
`ofc.setLocation(location)`	Set the OFC location `location =` `'current'`: OFC at servo current input `'sensor'` : OFC at rod position sensor
`ofc.setType(type)`	Set the OFC type `type =` `'none'` : no failure `'liquid'` : liquid failure `'solid'` : solid failure
`ofc.setAmplitude(amplitude)`	Set the OFC amplitude [mA / mm] units is [mA] for `location = 'current'` units is [mm] for `location = 'sensor'` The amplitude units depend on the location of the oscillatory failure (current or sensor).

`ofc.setBias(bias)`	Set the OFC bias [mA / mm] units is [mA] for `location = 'current'` units is [mm] for `location = 'sensor'` The bias units depend on the location of the oscillatory failure (current or sensor).
`ofc.setFrequency(frequency)`	Set the OFC frequency [rad/s]
`ofc.setPhase(phase)`	Set OFC phase [rad]
`ofc.setStartTime(start_time)`	Set OFC start time [seconds]
`ofc.enableRandomOFC(sim_time)`	Enable a random OFC (random location, type, amplitude, bias, frequency, phase, and start time) `sim_time`: simulation end time [seconds]

7.3.5 Parameter settings: Physical servo actuator

The model parameters for the physical servo actuator can be set using the functions of the `servoReal` object in the Matlab workspace. The mathematical model of the physical servo actuator is shown below

$$\dot{p}(t) = v(t)$$

$$v(t) = v_c(t)\sqrt{\frac{\Delta P - \frac{F_{\text{aero}}(t) + F_{\text{damping}}(t)}{S}}{\Delta P_{\text{REF}}}}$$

with external forces

$$F_{\text{damping}} = K_d v^2$$

$$F_{\text{aero}} = -\text{sgn}(v_c) K_{\text{aero}} \delta$$

and the rod position control laws

$$i_c(t) = K[\, p_{REF}(t) - p(t)\,]$$

$$v_c(t) = K_c\, i(t)$$

Model variables:

$p(t)$ and $v(t)$ are the position and velocity of the actuator rod
$v_c(t)$ is the actuator rod speed commanded by the flight computer
$F_{\text{damping}}(t)$ is the servo-control load of the adjacent actuator in damping mode, in case of a Duplex active/passive scheme.
$F_{\text{aero}}(t)$ is the aerodynamic force acting on the control surface

$i(t)$ is the servo actuator input current
$p_{REF}(t)$ is the actuator rod position command generated by the flight control computer

Model parameters:

ΔP is the hydraulic differential pressure delivered to actuator
ΔP_{REF} is the differential pressure reference corresponding to maximum rod speed
S is the actuator piston surface area
K_{d} is the actuator damping coefficient
K_{aero} is the aerodynamic load coefficient
K is the servo rod position controller gain
K_{c} is the conversion factor from servo actuator input current to actuator rod speed

The model parameters ΔP (hydraulic differential pressure) and K_{d} (actuator damping coefficient) of the physical (real) servo actuator are assumed to contain parameter uncertainty, and may vary from the nominal parameter values assumed by the analytically redundant servo actuator model. The aerodynamic force $F_{\mathrm{aero}}(t)$ acting on the control surface is also not known to the analytically redundant servo actuator model. The other model parameters (ΔP_{REF}, S, K, K_{c}) are assumed to contain very little uncertainty, and are close to the nominal parameters assumed by the analytically redundant servo actuator model.
For this reason, the `servoReal` **object only provides functions to set the** ΔP **and** K_{d} **parameters.**

The `servoReal` object stores the model parameters for the physical (real) servo actuator in the variables shown in Table 8. The model parameter are initialised to their default values when the `servoReal` object is created.

Table 8: Physical servo actuator parameters

Property	Symbol	Description	Default	Units
`dP`	ΔP	Hydraulic differential pressure delivered to actuator	29	N/mm^2
`dP_ref`	ΔP_{REF}	Hydraulic differential pressure reference corresponding to maximum rod speed	33.5	N/mm^2
`K`	K	Servo rod position control gain	0.6	mA/mm
`K_aero`	K_{aero}	Aerodynamic load coefficient	647.7	N/deg
`K_d`	K_d	Actuator damping coefficient	8.45	N/(mm/s)^2
`S`	S	Actuator piston surface area	5800	mm^2

The model parameters ΔP and K_{d} can be changed using the functions shown in Table 9. The functions `setdP` and `setK_d` are provided to set ΔP and K_{d} to specified values, while the function `randomiseServoParameters` is provided to assign random values to ΔP and K_{d} for Monte Carlo simulations.

Table 9: Functions for setting physical servo actuator parameters

Function	Description
`servoReal.setdP(dP)`	Set the hydraulic differential pressure parameter ΔP (must be between 16 and 30).
`servoReal.setK_d(K_d)`	Set the actuator damping coefficient K_d (must be between 6.8 and 10).

`servoReal.randomiseServoParameters()`	Randomise the servo constants ΔP and K_d to represent parameter variation in the real servo actuator. Assign a random value to ΔP uniformly distributed between 16 and 30. Assign a random value to K_d uniformly distributed between 6.8 and 10.

7.3.6 Parameter settings: Aircraft model

The model parameters for the aircraft longitudinal dynamics, including the flight path angle controller gains, and the load factor controller gains, are stored in the `aircraft` object in the Matlab workspace. The model parameters are initialised to their default values when the `aircraft` object is created.

The `aircraft` object provides functions to set load factor command and the turbulence level, but does not provide any functions to change the longitudinal dynamics or the flight control gains.

7.3.6.1 Set load factor control input

The load factor command can be set to the following input types:

- flight path angle control
- load factor step input
- load factor sine input
- load factor chirp input

The functions shown in Table 10 are provided to set the load factor command input

Table 10: Function for setting load factor command input signal

Function
`aircraft.setControlInput(input_type, input_amplitude, input_start time, input_stop_time, input_sine_frequency_hz)`
Usage

`aircraft.setControlInput('FPA_control')`
 	
`aircraft.setControlInput('NZ_step', input_amplitude, input_start_time, input_stop_time)`
 	
`aircraft.setControlInput('NZ_sine', input_amplitude, input_start_time, input_stop_time, input_sine_frequency_hz)`
 	
`aircraft.setControlInput('NZ_chirp', input_amplitude)`
 	
`aircraft.setControlInput(input_type, input_amplitude, input_start time, input_stop_time, input_sine_frequency_hz)`

Arguments	
`input_type`	The load factor command type `input_type =` `'FPA_control'`: flight path angle control `'Nz_step'` : load factor step input `'Nz_sine'` : load factor sine wave input `'Nz_chirp'` : load factor chirp input
`input_amplitude`	Load factor amplitude [g] The interpretion of `input_amplitude` depends on the `input_type`: `input_type =` `'Nz_step'` : load factor step size `'Nz_sine'` : sine wave amplitude `'Nz_chirp'` : chirp amplitude Not applicable when `input_type = 'FPA_control'`.
`input_start_time` `input_stop_time`	The start time and stop time of the load factor step input or load factor sine wave input [in seconds]. The control input starts in flight path angle control by default, switches to the load factor step input or load factor sine wave input between `input_start_time` and `input_stop_time`. After `input_stop_time` the control input returns to flight path angle control. Not applicable when `input_type = 'FPA_control'` or `input_type = 'NZ_chirp'`.
`input_sine_frequency`	The frequency of the load factor command sine input [Hz]. units is [mA] for `location = 'current'` units is [mm] for `location = 'sensor'` Only applicable when `input_type = 'NZ_sine'`.

When the load factor command is set to the step input type (`'NZ_step'`), then the following function allows a second load factor step to be applied after the first one. This is useful for testing the aircraft response both before and after the reconfiguration.

Function
`aircraft.setControlInput2(input_amplitude2, input_start_time2, input_stop_time2)` Only applicable when `input_type = 'NZ_step'`.

Arguments	
`input_amplitude2`	Load factor amplitude [g] of the second load factor step.
`input_start_time2` `input_stop_time2`	The start time and stop time of the second load factor step input.

7.3.6.2 Set turbulence levels

The turbulence level can be set to light, moderate, or severe turbulence, or can be disabled, using the functions shown in Table 11.

Table 11: Functions for setting turbulence level.

Function	Description
`aircraft.setNoTurbulence()`	Disable turbulence
`aircraft.setLightTurbulence()`	Set light turbulence
`aircraft.setModerateTurbulence()`	Set moderate turbulence
`aircraft.setSevereTurbulence()`	Set severe turbulence

7.3.7 Parameter settings: Analytically redundant servo model

The model parameters for the analytically redundant servo actuator are stored in the `servoModel` object in the Matlab workspace. The model parameters are initialised to their default values when the `servoModel` object is created.

The analytically redundant servo actuator model uses the same model as the physical (real) servo actuator, except that it uses the nominal values for the model parameters, and does not model the external aerodynamic force $F_{\text{aero}}(t)$ acting on the control surface. ($F_{\text{aero}} = 0$ is assumed.)

The `servoModel` object therefore does not provided any functions to change the parameters for the analytically redundant servo model.

8. REFERENCES

[1] Favre, C. (1994). Fly-by-wire for commercial aircraft: The Airbus experience. International Journal of Control, 59(1), 139–157.
[2] Besch, H. M., Giesseler, H. G., Schuller, J. (1996). Impact of electronic flight control system (EFCS) failure cases on structural design loads. AGARD Report 815, Loads and Requirements for Military Aircraft.
[3] Goupil, P. (2010). Oscillatory failure case detection in the A380 electrical flight control system by analytical redundancy. Control Engineering Practice, 18(9), 1110-1119.
[4] Jeanneau, M. (2006). Unsteady character of fly-by-wire flight controls: round-up and research orientations at Airbus. Proceedings of the Asian Control Conference.
[5] Oudin, S. (2017). Low Speed Protections for a Commercial Airliner: a Practical Approach, AIAA Scitech.