SIMULATION-BASED SYSTEM RELIABILITY ANALYSIS OF ELECTROHYDRAULIC ACTUATOR WITH DUAL MODULAR REDUNDANCY

Maxim Andreev*, Artem Kolesnikov, Uwe Grätz, Julia Gundermann

ESI ITI GmbH, Schweriner Str. 1, 01067 Dresden, Germany
* Corresponding author: Tel.: +49 351 260 50 249; E-mail address: maxim.andreev@esi-group.com

ABSTRACT

This paper describes the failure detection system of an electro-hydraulic actuator with dual modular redundancy based on a *hybrid twin*™ concept. *Hybrid twin*™ is a combination of *virtual twin* that operates in parallel with the actuator and represents its ideal behaviour, and a *digital twin* that identifies possible failures using the sensor readings residuals. *Simulation-based system reliability analysis* helps to generate a dataset for training the digital twin using *machine learning* algorithms. A *systematic failure detection approach* based on decision trees and the process of analysing the quality of the result is described.

Keywords: failure detection system, reliability, digital twin, machine learning

1. INTRODUCTION

The objective of *reliability engineering* is to provide the ability of the device to function within the specified period of time. The problem is that at the design phase it is very difficult to predict all the possible factors that could lead to failure, so unplanned failures are almost impossible to be completely eliminated from the device life cycle.

The consequences of failures can be significantly reduced by, for example, developing *fault-tolerant systems*, providing *predictive maintenance* and *system diagnostics* design. This makes possible to correct the inaccuracy of design-stage reliability estimations by adjusting to the real operating conditions. But reliability engineers often face at this point a conflict of increasing device complexity to improve reliability (e.g. by *redundancy*), which in turn makes maintenance and diagnostics more difficult.

However, *digital transformation* brings new opportunities in these challenges. As an example, model-based development is becoming more and more important. Device simulation models can be used both in designing and predicting the reliability of devices, as well as during operation of a real device [1].

The aim of this paper is to demonstrate the workflow of a simulation-based system reliability improvement by providing an accurate *failure detection system*. We use *machine learning* methods that allow us to move from virtual prototype simulation models, already widely used in product development by many manufacturers, to a system that accurately identifies failure at the earliest stages of its occurrence with minimal use of physical experiment.

2. RELIABILITY OF AN ELECTROHYDRAULIC ACTUATOR

The object of this study is an electrohydraulic aileron actuator with *dual modular redundancy*. The typical structure of such actuators used in civil aviation, as described in [2], is taken as an example.

We assume that the optimal actuator structure was obtained at the previous development stage (for example, with the help of *fault tree analysis*) and thus it consists of two electro-hydraulic actuators with independent energy sources connected in parallel to the load (aileron). In normal mode, both actuators work in parallel, so

that failure of one of them does not lead to a loss of control and implements the *fault-tolerant system* concept.

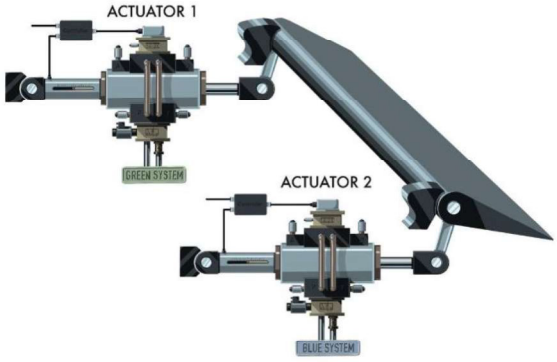


Figure 1: Electrohydraulic aileron actuator with dual modular redundancy

As with any system with *modular redundancy*, it must be equipped with an automatic diagnostic and control system that automatically switches off the failed actuator. A specific problem for *dual modular redundancy* systems is the difficulty of detecting which of the two actuators failed. One failed actuator causes disturbances in the system, both at the mechanical system level and via the control system, which leads to abnormal behaviour of the other actuator (which can be monitored by sensors).

The next challenge is to accurately localize the failure. Not all failures require the same response from the control system. For example, a leakage in an actuator requires its immediate shutdown to provide the availability of the remaining hydraulic system; while a pressure sensor failure allows the actuator to remain functional by switching the control mode. Thus, the failure detection system must be able to differentiate between such cases and prevent false actuator shutdowns.

On the other hand, failure must be detected and localized at the earliest stage with minimum intensity. This would avoid the emergency consequences of gradual failures and eliminate them through *predictive maintenance*.

These problems can be solved by a failure detection system based on a system simulation model in combination with machine learning.

3. HYBRID TWIN™ BASED FAILURE DETECTION SYSTEM

The failure detection system is shown in **Figure 2.** The *electro-hydraulic actuator 2* (for simplicity, only one actuator is shown) receives from the *flight control computer 1* an electrical signal proportional to which the aileron *4* must be turned. Various sensors (position, pressure, temperature) are also available in the actuator, which can be used for control and diagnostics, and whose data from both actuators is collected in the receiver *5*.

The same input signal is received by the actuator's *virtual twin 3*. The *virtual twin* is a real-

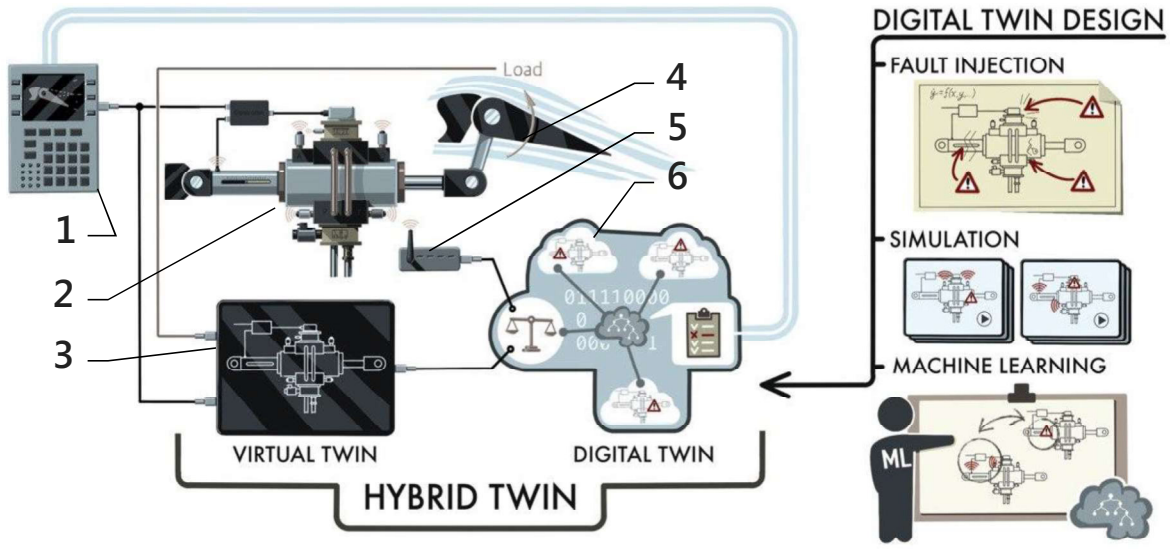


Figure 2: Hybrid Twin™ based failure detection system

time (or semi-real-time) simulation model based on an ODE describing the *nominal behaviour* of the actuator. To perform the twin function, the model must receive data from the sensors on the current values of the boundary conditions (supply and drain pressure, fluid temperature at the inlet, etc.) as well as load values. The output variables of the *virtual twin* are the calculated sensor data, which essentially represent their required values. Deviation of the real sensor readings from the virtual sensors readings of the *virtual twin* (assuming that the load sensors and boundary conditions are correct) will indicate that there are faults in the system. To extract more detailed information on failure, a *digital twin 6* is required.

The *digital twin* is a real-time capable decision algorithm whose input receives all sensor values and whose output provides information about a possible failure. To train a *digital twin*, we use thousands of variations of the *virtual twin*, containing models of different failures. The result of the digital twin prediction can be used to update the virtual twin by fitting parameters and to trigger a flight control computer for correction (e.g. to generate a control signal to shut down one of the actuators).

This combination of physically-based models and data-science is called a "*hybrid twin*™" and represents the next generation of twins by *ESI Group* [3].

3.1. Simulation model as a virtual twin

The first step to the actuator's *hybrid twin* is to create its *virtual twin*. The *virtual twin* design starts at the product prototype stage, when simulation models are created to check the requirements. In the process of product design, models are corrected, modified, validated in laboratory conditions and by the certification stage, as a rule, there is a set of models with a given accuracy simulating the device behaviour.

In the next step, based on this model set, a compromise simulation model should be created, that can be used for real-time (or semi-real-time) applications. For our research we developed a detailed simulation model of the electro-hydraulic actuator in *SimulationX*, the general structure of which is shown in **Figure 3**.

The virtual twin of each actuator is placed in a *virtual environment*, that is a model of the environment of the real device with the same behaviour. This means that the *virtual twin* must receive not only control signals, but also environment information, such as the supply and tank pressure and temperature at the input of the output hydraulic line of each actuator, inertial and aerodynamic load on aileron.

These data can be obtained either simply from sensors (as in the case of fluid pressure and temperature that can be measured by the sensors) or a combination of sensor data and a simulation model (as in the case of aerodynamic aileron load

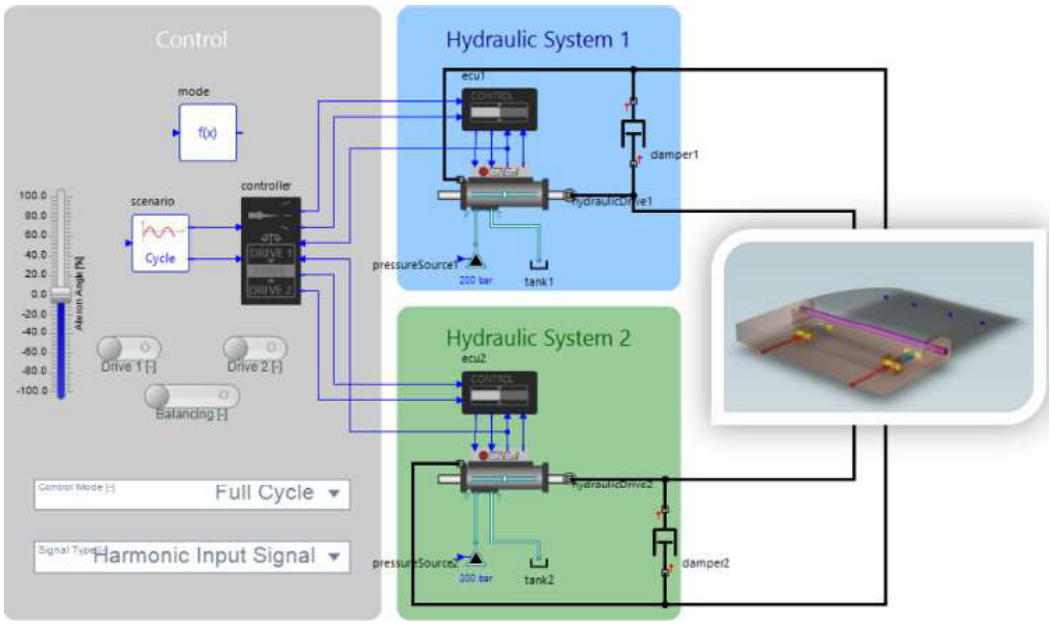


Figure 3: Diagram view of the aileron control system model in *SimulationX*

that can be simulated based on altimeter data, the spatial orientation of the aircraft and its air speed). For further steps, it is important to note that for reliable use of the *virtual environment* data, all sensors and simulation models must be equipped with their own failure diagnostic system. Data on the state of the object can be obtained only based on the assumption that the *virtual environment* is working properly.

Another important requirement for the *virtual twin* model is the degree of detail of its physical content. The model assumptions should not be limited to the calculation of system state variables in which sensors are installed but should also be able to simulate the behaviour in case of possible failures. **Figure 4** shows the detailed structure of the model of the electrohydraulic actuator developed in *SimulationX*.

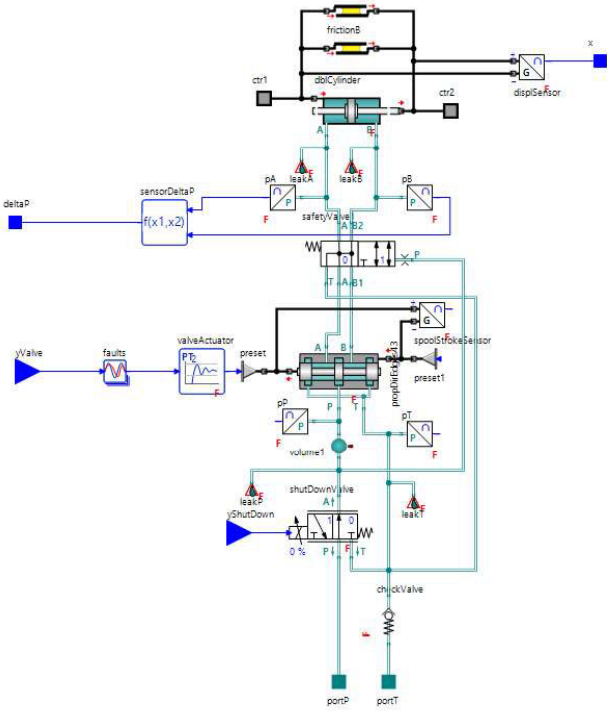


Figure 4: Diagram view of the electro-hydraulic actuator model in *SimulationX*

The following requirements should be considered in the hydraulic drive simulation model used as *virtual twin*:

- The model of the *hydraulic control servo valve* contains a spool model based on edge geometry. This is important for further modelling of the clogging of one of the control edges.
- All sensor models not only transmit state variables, but also consider the process of converting a physical value into an electrical or a dimensionless value. At conversion points, fault models will be built in later.

The final stage of virtual twin design is the validation of the model with the real actuator. The result of the validation is a field of *confidence intervals* of the model depending on the input variables. These confidence intervals are a basis of the failure diagnostics system, as we will show later.

3.2. Fault-augmented virtual twin

A *digital twin* training requires a large amount of data where the sensor data are clearly related to the detected failures. Collecting enough data in real-world conditions seems to be a big challenge that is difficult to realize in practice because of the need to create crashable prototypes and a large number of experiments. However, the same simulation model that was used to create the *virtual twin* can be used to generate this data. For this purpose, the model has been augmented with fault models that can occur in real operation. Manual *fault augmentation* requires a deep understanding of the system and analysis of device operation history data. At the same time, a precise physical description of all possible failures is hardly possible due to their difficult predictability. Paper [4] describes a way to simplify and partially automate this process. The main idea is that faults are semi-automatically injected in the simulation model. The faults are similar for different physical domains (leakage, increased friction, signal noise, etc.), simulated as simple as possible, but can be parametrically changed with different intensity. This allows to generate a large dataset containing many theoretically possible failures without requiring manual modelling of each failure individually.

The following types of failures were simulated in our study:

- Fluid internal and external leakage
- Valve edge clogging
- Servo valve actuator fault
  - Natural frequency reduction
  - Damping loss
- Signal faults
  - Bias
  - Calibration error

A detailed description of the simulation of these failures is given in [5].

We presume that failures should be detected at an early stage. As a result, we can assume that the occurrence of failure combinations is unlikely and thus significantly reduce the required number of models. Nevertheless, even in this case we are talking about hundreds and thousands of possible failures with different intensities. All failures should be classified into a limited number of categories to be used in a failure detection system. **Table 1** shows the classification we propose.

Table 1: Failure Categories

Failed Actuator	Command	Failure Information	Components	Faults
Actuator 1 (2) Failed	Shut Down!	External Leakage	4	40
		Control Valve Failure	1	100
		Supply Valve Failure	1	10
		Tank Valve Failure	1	10
	Change Control Mode!	Cylinder Pressure Sensor Failure	2	80
		Displacement Sensor Failure	1	40
	Warning!	Cylinder Internal Leakage	1	10
		Valve Displacement Sensor Failure	1	40
		Supply/Tank Pressure Sensor Failure	2	80
		Sum (x2 Actuators):	28	820

First of all, failures differ by the actuator in which they occur. After that, failures should be classified according to the required control system actions. Three categories have been defined:

- "*Shut Down!*" includes failures that require the immediate disconnection of the failed actuator from the hydraulic system and its switch to passive damping mode.
- *"Change Control Mode!"* refers to failures that require a change in the operating mode of the *ECU*.
- *"Warning!"* faults do not require an immediate response but must be fixed by technical staff during maintenance.

To fix a failure during maintenance, it is necessary to accurately localize it by the component in which it occurred. However, within the same component, failures may have different nature. For example, category *"Control Valve Failure"* includes both control edge clogging and actuator fault or accuracy loss.

3.3. Failure detectability and sensor readings

Any failures that occur in the actuator during the operation lead to deviations of variables, that defining the system state $\boldsymbol{X}(t)$ at time t from the *nominal* behaviour $\boldsymbol{X}^{Nom}(t)$, called *errors* $\boldsymbol{E}(t)$.

$$\boldsymbol{E}(t) = \boldsymbol{X}(t) - \boldsymbol{X}^{Nom}(t) \quad (1)$$

A failure can be *detected* if one of the components of $\boldsymbol{E}(t)$ at the time point t_F exceeds a certain threshold ε:

$$|e_i| > \varepsilon \quad (2)$$

In the failure detection system, the *unobservable* nominal behavior is replaced by the behavior of a *virtual twin* $\boldsymbol{X}^{VT}(t)$ with output discrete virtual sensor variables $\boldsymbol{S^{VT}}(s_1^{VT}, s_2^{VT}, \dots, s_n^{VT})$. The state variables of the real actuator are estimated using the sensor readings $\boldsymbol{S}(s_1, s_2, \dots, s_n)$. Thus, the theoretical *error* value $\boldsymbol{E}(e_1, e_2, \dots, e_n)$ is replaced by the observable *residual* value $\boldsymbol{R}(r_1, r_2, \dots, r_n)$:

$$r_i = s_i - s_i^{VT} \quad (3)$$

It is necessary to note that in this case only those failures can be detected and classified which lead to *errors* of state variables $x_i = x(t_i)$ measured by available sensors. The first problem is that the *errors* in sensors themselves lead to *deviations* of their readings s_i.

The next problem is that the results of the *virtual twin* are *calculated* virtual sensor readings s_i^{VT}, which we assume to be distributed around the real $\boldsymbol{X}^{Nom}(t)$.

This leads to a distribution of the residual value r_i. Thus, not any deviation of r_i from zero indicates an error but could be only a result of sensor error and/or virtual twin model error. A condition must be defined for when the value(s) of r_i significantly deviate(s) from zero. This

condition and with it the accuracy of failure detection both depend on the accuracy of the sensors used, the accuracy of the virtual twin, as well as the size of the dataset sample.

In this demonstrator is assumed, that failure is detectable if one of the residuals to be out of ±3% band for each scaled residual value $r_i' = \frac{r_i}{x_{Scale}}$:

$$r'_i \notin (-0.03, +0.03) \quad (4)$$

The random nature of the model inaccuracy requires statistical processing of the *residuals*. **Figure 5** and **Figure 6** show histograms of scaled sensor residual probability distribution for two types of failures (external leakage and pressure sensor bias). The histograms were recorded with sampling rate 100 Hz during 90 s of simulation of the actuator's operation.

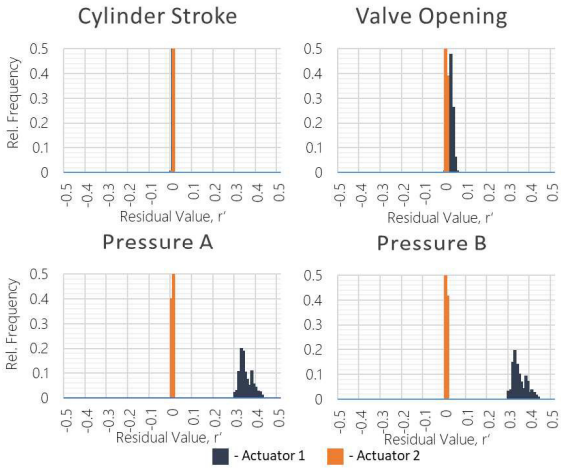


Figure 5: Residuals relative frequency distribution. External leakage in the hydraulic cylinder.

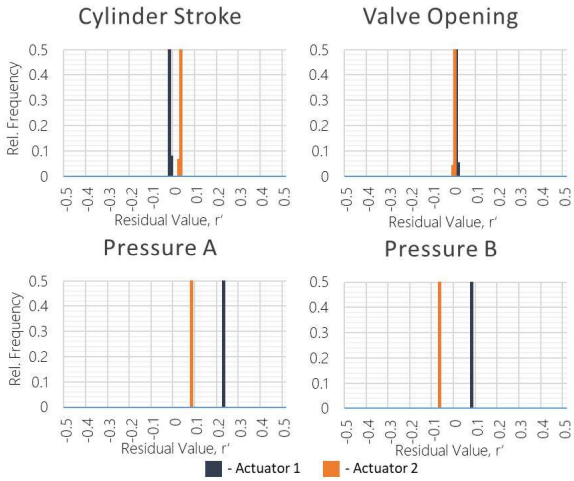


Figure 6: Residuals relative frequency distribution. Cylinder pressure sensor bias.

A residual in one part of the system leads to the spread of residuals into other parts of the system, including the faultless actuator. This is also observed in the residual distribution: leakage in one of the cylinder chambers leads to an increase in the pressure residual of the other chamber, and a variation in the control valve behaviour. If the pressure sensor fails, this affects the pressure behaviour in the chambers of all actuators and the cylinder stroke sensor residual.

To detect and localize failures, it is important to *extract features* from the residuals sample. Regardless of the type of residual distribution, the main features that can be used to identify faults are:

- Residual sample mean value:

$$\bar{r}' = \frac{\sum(s_i' - s_i^{VT'})}{N} \quad (5)$$

- Residual variance of sample mean:

$$Var(r') = E[(r_i' - \bar{r}')^2] \quad (6)$$

In addition, a discrete indicator is required to show whether the residual is inside the confidence interval or not.

The indicator is defined by the following equation:

$$r_{noDev} = \begin{cases} 1 & |\bar{r}'| < 0.03 \wedge Var(r') < 0.01^2 \\ 0 & \text{otherwise} \end{cases} \quad (7)$$

Thus, each *residual* has three *features* that will be used to identify faults.

3.4. Dataset generating

Machine learning algorithms require a training dataset as input data. The dataset is based on simulation results and should be as close as possible to the conditions of the real device operation. Within the framework of this demonstrator it was generated three white noise-based control signals with different amplitudes and spectrum and simulated each of them for two load levels (50% and 100%).

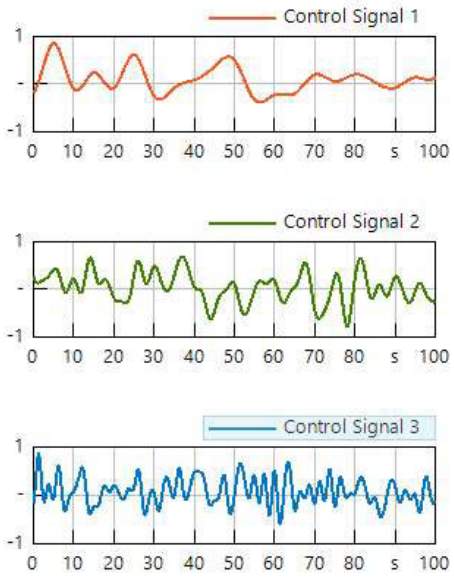


Figure 7: White noise-based control signals for dataset generating

Thus, the dataset will consist of six sets of possible failures (each of 820 variants), in total 4920 variants. It would take a lot of work to create so many model modifications and collect results manually. To solve this problem more effectively, the *System Reliability Analysis (SRA) AddIn* of *SimulationX* was used.The *SRA-AddIn* has the following functions: automatically create a specified number of models with different failures and fault intensities, simulate all variants using all allocated processor cores and collect the results on the dataset [4].

After generation, the dataset must be prepared for machine learning algorithms. Namely, simulated failures must be classified and the variants whose residuals do not exceed the confidence interval have to be filtered out. The structure of the dataset is shown in **Figure 8**.

Variant No	Faults Intensities				Fault Classifier			Residuals Features			
	fault1.intensity	fault2.intensity	...	faultF.intensity	Failed Actuator	Command	Detailed Info	Residual 1, Mean	Residual 1, Var	...	Residual R, ...
1	0.1	0	...	0	Actuator 1	Shut Down!	Valve Failure	0.112	0.0012	...	0.453
2	0.2	0	...	0	Actuator 1	Warning!	Sensor Failed	0.453	0.0354	...	0.0012
...	...	...	...	...	...	...	...	...	...	...	...
n	0	0	...	1	Actuator 2	Warning!	Sensor Failed	0.112	0.0012	...	0.453

Figure 8: Dataset structure after processing

3.5. Machine learning based failure analysis

Recently methods of machine learning in tasks for error detection and forecasting of failures in technical systems have become widespread [6, 7]. Currently, a large number of machine learning algorithms are available for fault diagnosis and predictive maintenance [8, 9]. We analysed various available classifiers such as:

- Linear Support Vector
- Decision tree
- Multi-layer Perceptron
- Gaussian Naïve Bayes
- Random forest
- Linear Discriminant Analysis
- AdaBoost classifier
- Logistic Regression

All classification methods have strengths and weaknesses depending on the available data size, needed accuracy and interpretability as well as selected features [10]. In the applied tasks for multiclass classification, it is recommended to compare the main metrics of the algorithms to each other and select the most exact and reliable one. According to the sufficient accuracy and interpretability the decision tree classifier using the *CART* algorithm with the *Gini* impurity metrics were chosen for the data set with faults [11]. The data set was investigated by the

decision three in the data analytics tool *ESI Mineset*.

The objective of the classification algorithm in the context of *failure detection* is to find a rule based on the sensor residuals (column *Residuals Features*), which will uniformly group the failures of one type (column *Fault Classifier* in **Figure 8**). Using the resulting rules for the real sensor residuals analysis, it is theoretically possible to obtain in real time a decision about the occurrence of a certain failure class in the system.

In reality, if all available sensor residual features are used to identify detailed failure information, the following problems may occur:

- the algorithm will be very sensitive to changes in training data
- the resulting rules will be very complex and contain many conditions caused by the randomness contained in the dataset

An illustration of the last point can be the result we have obtained with a direct approach to learning: some key decisions depend on a value lying within a range of tank pressure sensor residual which is less than 0,001% of the whole residual range. This effect is called *overfitting* and such algorithms cannot be used in practice.

To avoid these problems, we used a *systematic failure detection approach*. First, the detection of the failed actuator is trained on the complete dataset. Then, on a dataset containing only the failures of one actuator, the classifier of the control command is defined. Third, on the corresponding samples detailed failure detection is trained. This allows us to reduce the complexity of each classifier by limiting the height of the tree to 5-7 levels and avoid possible *overfitting* due to *early stopping*.

As mentioned above, to identify the failed actuator, a *decision tree algorithm* is used. All residual features of all sensors are included in the training set, but the maximum height of the tree is limited to 7. The result is shown in **Figure 9**.

The resulting tree uses mostly the discrete features $r_{NoDev,}$ and final splitting criteria do not have extremely small ranges. 1/3 of the dataset was used to estimate the accuracy of the algorithm. As result the model quality was 91,49% (error rate $-0{,}085 \pm 0{,}005$). The analysis of the most problematic terminal nodes (with the lowest *purity value*) shows that the most difficult case for the algorithm is to recognize and differ the failure of the cylinder piston stroke sensor of one actuator from another. This shows that possible actuator design changes (that would, for

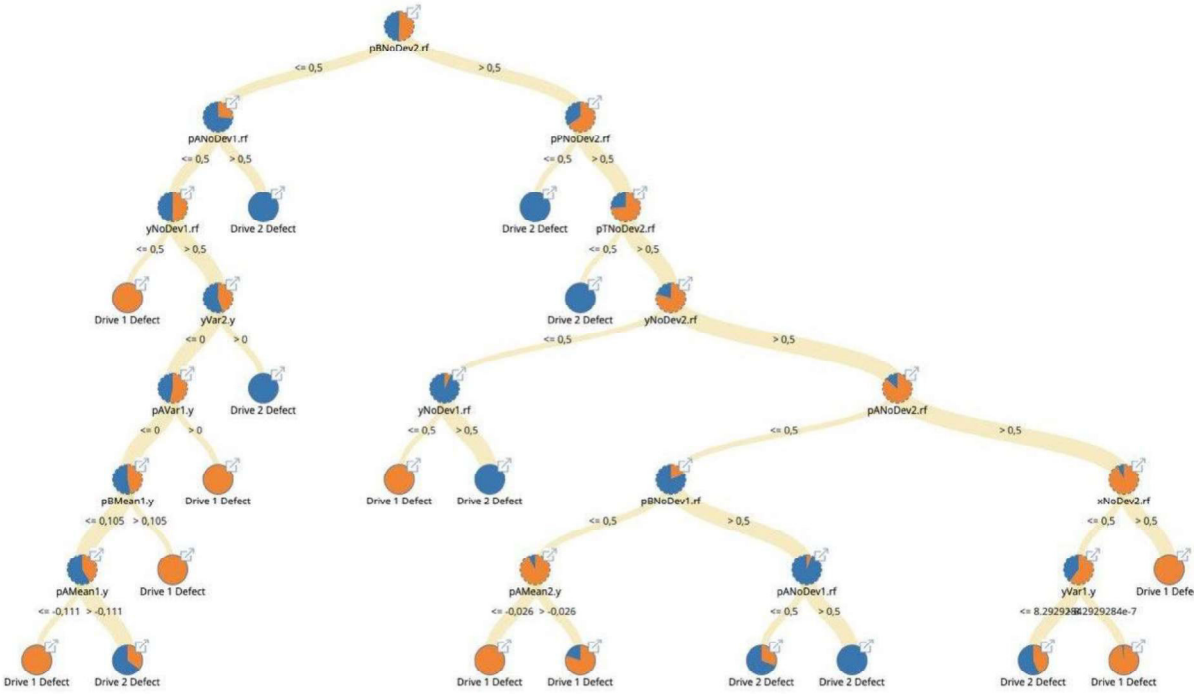


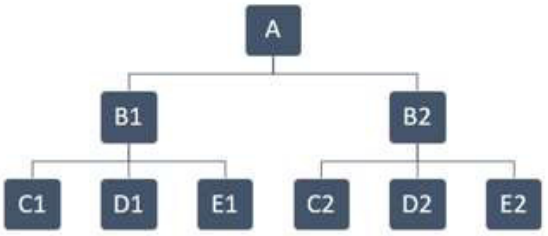
Figure 9: Structure of the decision tree that detects failure of one of the actuators.

example, identify a piston stroke sensor failure on a majority vote basis) would increase the accuracy of the failure detection system.

Further, based on a data sample containing only *Actuator 1* failures, the algorithm for recognition of failures classified by the command of the control system is similarly trained. The result obtained using the features of all the sensors (incl. *Actuator 2*) by very good model quality showed clear *overfitting* symptoms: many secondary features were involved in the decision process (such as drain pressure sensor variance), as was the focus on small changes in the cylinder piston stroke sensor readings. Using only *Actuator 1* sensors for training improved the quality of the algorithm but did not solve the drain sensor problem. The best result was obtained after excluding the drain sensor readings: model quality – 98,44% (error rate – 0,016 ± 0,006).

The training approach for *detailed failure detection* algorithms is similar to that described above, so this paper does not cover it.

As a result of a systematic approach, a complex symmetrical structure of 9 decision trees is developed (see **Figure 10**), on the basis of which an algorithm for device diagnostics can be implemented into the flight control computer.



A – Failed actuator classification,

B – Control system trigger classification,

C, D, E – Classification of detailed failure information

Figure 10: Macrostructure of failure classification algorithm

4. CONCLUSION

1. The implementation of the *hybrid twin*™ approach shown in this paper allows to significantly increase the reliability of the electro-hydraulic actuator with dual modular redundancy through a failure detection system.
2. Successful detection of the failure categories (with very small deviations from the nominal behaviour) depends on the accuracy of the *virtual twin* representing the real actuator.
3. Generation of the failure detection algorithm for the *digital twin* requires the calculation of many fault augmented variations of the simulation model of the device, that can be significantly accelerated through parallel computing.
4. The developed *systematic failure detection approach* using the *decision tree* algorithm is sensitive to the overfitting problem which can be resolved by training gradually from a more general category of failures to more detailed.

NOMENCLATURE

ODE	Ordinary Differential Equations
ECU	Electronic Control Unit
SRA	System Reliability Analysis
CART	Classification and Regression Trees

REFERENCES

[1] Negretti D (2019) Modeling and simulation for predictive maintenance: a survey, Master's thesis, Polytechnic University of Milan, Italy, http://hdl.handle.net/10589/148668

[2] Shumilov I.S (2009) Sistemy Upravleniya Rulyami Samolyotov (Aircraft rudder control systems), Moscow.

[3] Chinesta F, Abisset-Chavanne E, Cueto E (2018) Virtual, Digital and Hybrid Twins: A New Paradigm in Data-Based Engineering and Engineered Data – Archives of Computational Methods in Engineering

[4] Kolesnikov A, Tretsiak D, and Cameron M (2019) Systematic Simulation of Fault Behavior by Analysis of Vehicle Dynamics, Proceedings of the 13th International Modelica Conference, March 4-6, 2019, Regensburg, Germany, doi: 10.3384/ecp19157451

[5] Kolesnikov A, Andreev M, and Abel A (2018) The Fault-Augmented Approach for the Systematic Simulation of Fault Behavior in Multi-Domain Systems in Aerospace, SAE Technical Paper 2018-01-1917, doi:10.4271/2018-01-1917.

[6] Liu W.Y, Tang B.P, Han J.G, at all (2015) The Structure Healthy Condition Monitoring and

Fault Diagnosis Methods in Wind Turbines: A Review - Renewable and Sustainable Energy Reviews. Volume 44, 466–472

[7] Ali, E (2016) A Machine Learning Approach for Tracking the Torque Losses in Internal Gear Pump - AC Motor Units, 10th International Fluid Power Conference. Dresden, March 8-10, 2016, Dresden, Germany

[8] Choi S, Haque M. Sh, Tarek M. T. B (2018) Fault Diagnosis Techniques for Permanent Magnet AC Machine and Drives—A Review of Current State of the Art, IEEE Transactions on Transportation Electrification. Volume 4, Issue 2, doi: 10.1109/TTE.2018.2819627

[9] Miller S, (2019) Predictive Maintenance Using a Digital Twin, MathWorks Technical Articles, https://www.mathworks.com/company/newsletters/articles/predictive-maintenance-using-a-digital-twin.html Accessed 23 Jan 2020

[10] Jegadeeshwaran R, Sugumaran V (2013) Comparative Study of Decision Tree Classifier and Best First Tree classifier for fault diagnosis of automobile hydraulic brake system using statistical features - Measurement. Volume 46, Issue 9, 3247–3260

[11] Loh Wei-Yin (2011) Classification and Regression Trees - WIREs Data Mining and Knowledge Discovery. Volume 1, 14–23