# 1.5

# Real-Time Predictive Maintenance – Model-Based, Simulation-Based and Machine Learning Based Diagnosis

**Franz Wotawa[1], David Kaufmann[1], Adil Amukhtar[1], Iulia Nica[1], Florian Klück[2], Hermann Felbinger[2], Petr Blaha[3], Matus Kozovsky[3], Zdenek Havranek[3] and Martin Dosedel[3]**

[1]Graz University of Technology, Austria
[2]AVL List GmbH, Austria
[3]Brno University of Technology CEITEC, Czech Republic

## Abstract

Predictive maintenance focuses on forecasting faulty or unwanted behaviour and defines appropriate countermeasures to be taken. Diagnosis, i.e., the detection of failures, the identification of faults, and repair provides useful foundations for predictive maintenance. In this article, we show how diagnosis, and in particular model-based, simulation-based and machine learning based diagnosis, can be used in practice. For this purpose, we introduce a simplified DC e-motor simulation model with the capability of fault injection to be used to show the efficiency of the introduced diagnosis methods based on the model's behaviour. A simulation run of the system under test with pre-defined injected faults during runtime is used to validate the results obtained by the diagnosis methods. The results outline a promising application of these diagnosis methods for industrial applications, since each algorithm shows a time efficient and reliable diagnosis in relation to find the root cause of an observed faulty behaviour within the model. Further, the root cause analysis, performed with the introduced diagnosis methods, offers an excellent starting point for future development of self-adapting systems.

**Keywords:** abstract model, AI based diagnosis, AI based predictive maintenance, digital twin, model-based diagnosis, machine learning, simulation-based diagnosis, reliability, validation, fault model, fault injection, root cause analysis.

## 1.5.1 Introduction and Background

In this article, we focus on the application of model-based and machine learning-based diagnosis outlined in the article "Foundations of Real-Time Predictive Maintenance with Root Cause Analysis" making use of an e-motor use case. In the foundations, we already discussed the underlying background, and an architecture of a real-time diagnosis tool for detecting root causes of faults based on different diagnosis methods, which distinguish in the applied methodologies.

Besides providing more information regarding the application of the different diagnosis methods, we want to solve the question of whether model-based reasoning can be used for obtaining explanations for the given models, a simplified DC motor model with the capability of fault injection was developed to capture the individual ideas of diagnosis tools.

The first approach, model-based diagnosis, considers an abstract model that can be represented as logical rules for diagnosis. This model captures the abstract values for quantities/signals. Using an abstraction function, it is possible to map given values to their abstract representation. The second approach, simulation-based diagnosis, utilizes simulation models directly. A pre-requisite is that the models not only capture the correct behaviour but also faulty behaviour like the influence of different parameters on the behaviour. The last approach, an AI-based diagnosis model also uses simulation models to gather information about the behaviour based on different parameters of the system. The produced knowledge base is further used to train a model. After the training, it is plausible to evaluate the feasibility of the diagnosis model in terms of decision process optimization in real time.

In summary, we deal with the following diagnosis methods in this article:

- Model-based diagnosis
    - Abstract model represented with logic rules.
    - Diagnosis based on state change observations.
    - Classify the model in components and identify a normal or abnormal behaviour of each.
- Simulation-based diagnosis

  - Detailed simulated system model ("digital twin") with the capability of fault injection.
  - Use of simulation models directly.
  - Generate labelled reference data by simulating the model with health and fault condition in different scenarios.
  - Real-time diagnosis of observed model values based on pre-simulated reference data.
- AI-based diagnosis
  - Machine learning to diagnose unexpected behaviour.
  - Artificial neural networks to diagnose and predict fault behaviour.
  - Physical model (digital twin) with the capability of fault injection to produce training data.
  - Train and evaluate an AI based model on collected labelled reference data to detect fault behaviour in real-time within cyber-physical systems.

For all diagnosis methods, the assumption is to find faults occurring at runtime. To show the architecture and applicability of these approaches, the focus is on describing the mechanism and show the results based on examples to highlight the idea, the problems, and solutions. In the following section a simplified DC e-motor model with the capability of fault injection is described and the obtained results based on different diagnosis method implementations are demonstrated.

## 1.5.2 Application of Diagnosis Systems Based on Simplified DC e-Motor Model

In this section we introduce a developed simplified DC e-motor model with fault injection capability in all used components which comprises the battery, switch, resistor, load on the motor and the e-motor parts. The ability of fault injection is used to discuss three different diagnosis algorithms to detect faults in a system based on the simplified DC e-motor. The first promising approach is the model-based diagnosis algorithm. The model-based diagnosis system uses logic to represent the e-motor to perform model-based reasoning to search for diagnosis candidates given an unexpected behaviour caused by faults in the system. The second introduced diagnosis system is based on simulation which uses digital twin models of the e-motor directly to simulate faults in advance to use the generated data to find correlations with

the original system. The last approach deals with machine learning using gathered fault data of the e-motor model to train the system to detect faulty behaviour.

#### 1.5.2.1 Simplified DC e-Motor Model With Fault Injection Capabilities

The proposed use case of a DC e-motor comprises a battery, a switch for turning the motor on and off, a resistor, which we may use to adapt the voltage provided to the motor, the e-motor, and a load attached to the motor. In Figure 1.5.1, we find the schematics of the motor that also comprises the internals of the battery and the motor. We assume that the battery comprises an internal resistance, the motor resistance, inductance, as well as a part coupling the electric components to mechanic ones. For the model we consider a brushed e-motor comprising a wound rotor and a permanent-magnetic stator. The rotational speed of the motor is proportional to the voltage applied and its torque is proportional to the applied current. Table 1.5.1 shows a list of all components with the applicable health states including faults that can be set during runtime to simulate different behaviour of the e-motor. The DC e-motor simulation model is built with the equation-based language Modelica to simulate the complex physical system. For the diagnosis approaches based on this model, we use the simulated outputs

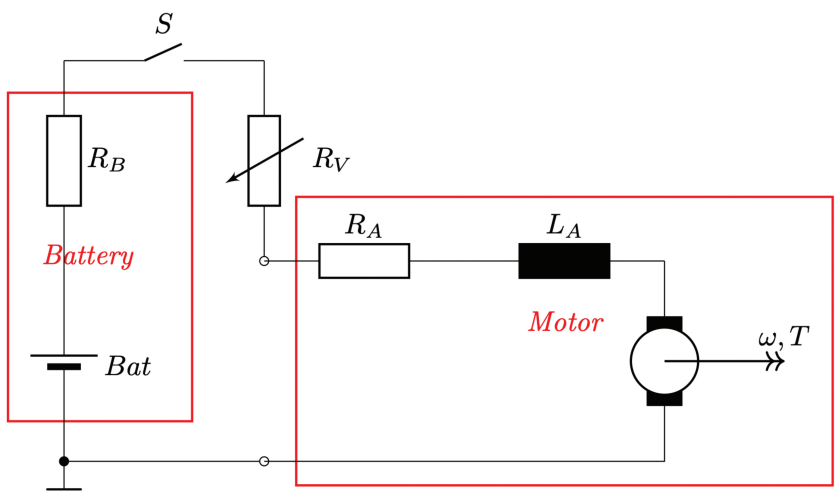


**Figure 1.5.1** Simplified DC e-motor circuit.

**Table 1.5.1** Simplified DC motor component state description.

| Component | Health state | Description |
|---|---|---|
| Motor | *ok* | Ordinary behaviour of a motor given by its internal components and the equations provided for DC motors allowing to map electrical quantities to mechanical ones. |
| | $f_1$ | In this fault mode we assume that 1/3 of the resistor and inductivity values is lost. |
| | $f_2$ | In this fault mode we assume that 2/3 of the resistor and inductivity values is lost. |
| Load | *ok* | The load applied to the motor is set to its normal value. |
| | *empty* | There is no load anymore applied to the motor. |
| | $f_1$ | The load is 50% higher than its normal value. |
| | $f_2$ | The load is 50% lower than its normal value. |

directly or a generated FMU (Functional Mockup Unit) from the model to be able to run simulations of the DC e-motor in other programming environments.

#### 1.5.2.2 Model-based Diagnosis for Simplified DC e-Motor

In the following, we outline the use of model-based diagnosis for the identification of root causes. For this purpose, we discuss the necessary steps required to diagnose the simplified DC motor use case depicted in Figure 1.5.1. Specifically, we consider the following faulty case where a certain load is higher than expected, indicated as load fault $f_1$ in Table 1.5.1. In Figure 1.5.2 we depict the behaviour of the e-motor when switching it on without load (*empty)*, with the expected load (*ok*), and the higher load ($f_1$). We see that when switching on the motor during time 0.5 and 1.5 seconds, there is a deviation between the observed rotational velocity and the current drawn to drive the e-motor in all three cases. Although this deviation is not that high between the ordinary "normal load" scenario and the "high load" scenario, as it can be observed.

We require observations and a logic model for computing diagnoses. The observations in the case of model-based diagnosis are assumed to be available at certain points in time where we probe the system. In Figure 1.5.2, we consider 3 probing time points ❶, ❷, and ❸ at time steps 0.25, 1.00, and 1.75 seconds respectively. In ❶ there is no difference between the three observed signals. In ❷ we see that both the rotational velocity as well as the current are different when comparing "normal load" with the "high load" scenario. In the "high load" scenario, the velocity is lower and the absolute

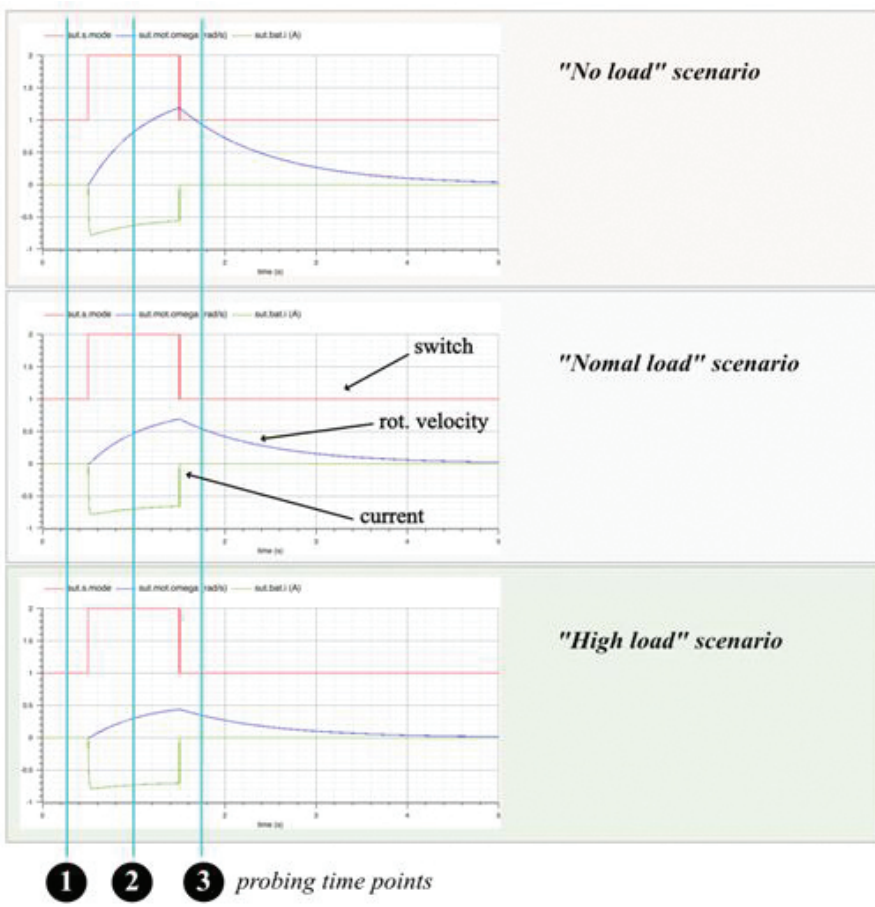


**Figure 1.5.2** Simplified DC e-motor diagnosis observations used for model-based diagnosis.

value of the current is slightly higher. In time step ❸, only the velocity is still lower for "high load".

Such deviations can be obtained automatically comparing a simulation run considering the e-motor to work as expected with observations obtained from monitoring the real e-motor implementation. Deviations trigger diagnosis and the question is how a model of the e-motor example can be utilized for obtaining the root cause responsible for the behavioural differences observed. For diagnosis, we will map the deviations or values to their corresponding logic representation. But before discussing this issue, we have a look at modelling for diagnosis.

We have to formalize the behaviour of components and their interconnections. For the behaviour, we use rules of the form $\neg ab(C) \rightarrow behav(C)$. A battery component, for example, can be easily formalized stating that in case of correct behaviour, it is delivering power using the following logic rule:

$$\neg ab\,(C) \wedge type\,(C, battery) \rightarrow val(pow\,(C)\,, nominal) \tag{1.5.1}$$

The predicate $type$ is used to say that component $C$ is of a type, e.g., $battery$. The predicate $val$ is for stating a value, e.g., $nominal$, for a component port, e.g., $pow$. In addition, we may also formalize that a malfunctioning battery is not delivering any electricity, i.e.:

$$ab\,(C)\ \wedge type\,(C, battery) \rightarrow val(pow\,(C)\,, zero) \tag{1.5.2}$$

We can do the same for switches, resistors, and the motor. A switch if being switched-on provides electricity (but only if there is electricity at one port). If switched-off no electricity is provided. A resistor is for passing electricity, and a motor makes use of provided electrical power to speed-up its rotor. Depending on the load the velocity reached can be higher or lower, requiring less or more power.

$$\begin{gathered}
\neg ab\,(C) \wedge type\,(C, switch) \wedge on\,(S) \wedge val\,(inpow\,(C)\,, V) \\
\rightarrow val(outpow\,(C)\,, V) \\
\neg ab\,(C) \wedge type\,(C, switch) \wedge on\,(S) \wedge val\,(outpow\,(C)\,, V) \\
\rightarrow val(inpow\,(C)\,, V) \\
\neg ab\,(C) \wedge type\,(C, switch) \wedge off\,(S) \\
\rightarrow val(outpow\,(C)\,, zero) \\
ab\,(C) \wedge type\,(C, switch) \rightarrow val(outpow\,(C)\,, zero)
\end{gathered} \tag{1.5.3}$$

$$\begin{gathered}
\neg ab\,(C) \wedge type\,(C, resistor) \wedge val\,(inpow\,(C)\,, V) \\
\rightarrow val(outpow\,(C)\,, V) \\
\neg ab\,(C) \wedge type\,(C, resistor) \wedge val\,(outpow\,(C)\,, V) \\
\rightarrow val(inpow\,(C)\,, V) \\
ab\,(C) \wedge type\,(C, resistor) \rightarrow val(outpow\,(C)\,, zero)
\end{gathered} \tag{1.5.4}$$

$$\begin{gathered}
\neg ab\,(C) \wedge type\,(C, motor) \wedge val\,(inpow\,(C)\,, V) \rightarrow val(outpow\,(C)\,, V) \\
\neg ab\,(C) \wedge type\,(C, motor) \wedge val\,(inpow\,(C)\,, V) \rightarrow val(speed(C), V) \\
\neg ab\,(C) \wedge type\,(C, motor) \wedge val\,(speed\,(C)\,, V) \rightarrow val(inpow\,(C)\,, V) \\
ab\,(C) \wedge type\,(C, motor) \rightarrow \neg val(speed\,(C)\,, nominal) \\
ab\,(C) \wedge type\,(C, motor) \rightarrow \neg val(outpow\,(C)\,, nominal)
\end{gathered} \tag{1.5.5}$$

Note that in the above model we do not distinguish values to be higher or lower than expected. Instead, we state that the speed (power requested) is not allowed to be nominal in case of a fault in the e-motor. This formalization captures the faulty behaviour required for diagnosis in the mentioned use case. However, we are also able to come up with a model considering different faulty states (and not only $ab$).

The described model formalizes the behaviour of the components. What is missing, is the description of the structure of the system. In our case, we have 4 components, i.e., a battery b, a switch s, a resistor r, and a motor m, that are directly connected. We first, declare the components via stating logical facts:

$$type\,(b, battery) \wedge type(s, switch) \wedge type(r, resistor) \wedge type(m, motor) \tag{1.5.6}$$

Afterward, we define the connections between the components using the predicate $conn$:

$$\begin{array}{c} conn\,(inpow\,(s)\,, pow\,(b)) \wedge conn(outpow\,(s)\,, inpow\,(r)) \\ conn\,(outpow\,(r)\,, inpow\,(m)) \end{array} \tag{1.5.7}$$

To complete the formalization, we state that values are transferred via a connection (in both directions), and that it is not allowed to have different values on any connection:

$$\begin{array}{l} val\,(X, V) \wedge conn\,(X, Y) \rightarrow val(Y, V) \\ val\,(Y, V) \wedge conn\,(X, Y) \rightarrow val(X, V) \\ \neg(val\,(X, V) \wedge val\,(X, W) \wedge V \neq W) \end{array} \tag{1.5.8}$$

This logic model can be now used for diagnosis. Note that in this context a diagnosis is a setting of health states to components. Hence, we are interested in assigning either $ab$ or $\neg ab$ to any component, e.g., in our case $b$, $s$, $r$, and $m$ considering the given observations. In Table 1.5.2, we summarised the diagnosis results obtained when using the diagnosis engine described in [2] and the model introduced in this section. Based on the foundations elaborated in [2] we introduced a more complex physical system also taking the factor time into consideration for observation to show the efficiency of the developed diagnosis method for a broader field of application.

We see that in case of the second and third observations, we only obtain the motor being responsible for the deviation between the expected and the observed values. Note that this – because of the formalization – states that the load is higher than expected. The required diagnosis time was less

**Table 1.5.2** Diagnosis results obtained using model-based diagnosis.

| Section | Observation | Diagnosis |
|---|---|---|
| ❶ | $off\,(s)$<br>$\wedge\, val\,(pow\,(b)\,, nominal)$<br>$\wedge\, val\,(speed\,(m)\,, zero)$<br>$\wedge\, val(outpow\,(m)\,, zero)$ | $\neg ab(b) \wedge \neg ab(s) \wedge \neg ab\,(r) \wedge \neg ab(m)$ |
| ❷ | $on\,(s)$<br>$\wedge\, val\,(pow\,(b)\,, nominal)$<br>$\wedge\, \neg val\,(speed\,(m)\,, zero)$<br>$\wedge\, \neg val\,(speed\,(m)\,, nominal)$<br>$\wedge\, \neg val(outpow\,(m)\,, zero)$<br>$\wedge\, \neg val\,(outpow\,(m)\,, nominal)$ | $\neg ab(b) \wedge \neg ab(s) \wedge \neg ab\,(r) \wedge ab(m)$ |
| ❸ | $on\,(s)$<br>$\wedge\, val\,(pow\,(b)\,, nominal)$<br>$\wedge\, \neg val\,(speed\,(m)\,, zero)$<br>$\wedge\, \neg val\,(speed\,(m)\,, nominal)$<br>$\wedge\, val(outpow\,(m)\,, zero)$ | $\neg ab(b) \wedge \neg ab(s) \wedge \neg ab\,(r) \wedge ab(m)$ |

than 0.0021 seconds for all observations. It is also worth noting that the observations represent our knowledge. For ❷ we know that the speed and the power consumption (the latter represented using the port $outpow$) are both not nominal and also not zero. Similarly, we represent the observations for ❸.

In summary, the provided model was able together with the given observations to come up with the expected solution. No other single fault diagnoses were obtained in any case. Modelling relied on the assumption of the particular fault case, and the transfer of power through the circuit. This simplified model may not be appropriate in all cases. Diagnosis time was very short making the approach feasible for this kind of application having a limited smaller number of components and taking care of simple models. Modelling, however, has always been an issue and more sophisticated models are maybe required for other application scenarios. The presented approach assumes that a simulation model (a-kind-of a digital twin) is running concurrently for allowing to generate observations.

#### 1.5.2.3 Simulation-Based Diagnosis for Simplified DC e-Motor

The idea behind the simulation-based diagnosis is to make use of digital twin models to simulate pre-configured faulty behaviour and thus find correlations with the original cyber-physical system's measured values, which allows to

diagnose faults as well as fault combinations and the correlated root causes of a physical system.

To diagnose a physical system with a simulation-based approach we use a system to induce fault modes to measure relevant signals to gather information about the behaviour under these configurations. This can be performed on a real system or at least on a digital twin (simulation model) with the ability to perform fault injection and output all relevant measured signals. To use the knowledge about the behaviour under fault conditions is the main idea of the simulation-based diagnosis approach. This leads us to the question of how to use the measured information to detect a fault system and additional to diagnose the root cause of such a faulty behaviour?

The main part of the simulation-based diagnosis approach is a precise cyber physical model, or a simulation model of the physical system (digital twin) with the capability of fault injection. Besides that, the algorithm itself is categorized into three subsections. First the reference data, a pre-simulated fault data generation for the diagnosis, second the model signals processing, a preparation phase of the measured model signals on which the diagnosis is performed and last the diagnosis phase where the measured data is brought into comparison with the pre-simulated fault reference data to find the best correlation and compute diagnoses to explain the actual system behaviour.

To evaluate the simulation-based diagnosis approach on the DC e-motor model we use the FMU version of the simulation model running in a python environment instead of a real system to produce fault reference data for the diagnosis method. In addition, we used another instance of the simulated DC e-motor model as a system to be diagnosed. We set the focus on the diagnosis of the faults in regards to the motor and torque parameter. As stated in Table 1.5.1 the motor and load state can be set with different modes. However, for the validation we concentrate on the specific faults as *empty*, $f_1$ and $f_2$ for the motor as well as $f_1$ and $f_2$ for the torque. In addition, we use the *ok* state to diagnose a health system as a reference to a faulty system.

$$\begin{aligned} load_{state} &\in \{ok,\ empty,\ f_1,\ f_2\} \\ motor_{state} &\in \{ok,\ f_1, f_2\} \end{aligned} \tag{1.5.9}$$

To generate the reference data for diagnosis, the simulation is configured with the option to inject faults at runtime. To generate a reference dataset with a broad range of different scenarios and signal behaviour characteristics, the fault states $f_x$ ($load_{state}$, $motor_{state}$) are injected at various time points and initial parameters of the DC motor model simulation. Besides the single fault injection, also all possible combinations of faults are considered to cover

most of the feasible fault diagnosis. While performing the simulations with fault injection, we measure the most significant signals $\overrightarrow{x_s}$ (1.5.10) of the e-motor model as the battery current $i$ and voltage $u$, the motor rotation speed $\omega$ and angular acceleration $\alpha$ at a sampling rate of 0.001 seconds.

$$x_s\,(t) \in \{i(t),\ u(t),\ \omega(t),\ \alpha(t)\} \tag{1.5.10}$$

Next the observations $\overrightarrow{x_s}$ are processed with a moving average method (see equation (1.5.11)) on a time window $\Delta w$ of 0.05 seconds. With the moving average we obtain an averaged signal value for each time step we simulate.

$$x_r\,(t) = \frac{1}{n}\sum_{i=0}^{n} x_s(t-i)\ ,\ n := \Delta w \tag{1.5.11}$$

The average is built since we perform the diagnosis based on an averaged time window $\Delta w$ to avoid losing information during state changes and quick responses. The averaged reference data $\overrightarrow{x_r}$ (1.5.12) is stored in a table for later usage in the diagnosis algorithm. Since all fault states $f_x$ are known for every measurement, we obtain a labelled dataset as reference data. The corresponding state information is appended to the reference data in the table.

$$\overrightarrow{x_r} = \left[\ x_{r0}\ \ \ldots\ \ x_{rn}\ \ ,\ \ \boldsymbol{f_{load}}\ \ \boldsymbol{f_{motor}}\ \right] \tag{1.5.12}$$

After generating the labelled reference data $\overrightarrow{x_r}$, we can run the DC motor simulation with the option to measure the signals $\overrightarrow{x_s}$, as mentioned before with a sampling rate of 0.001 seconds. For the diagnosis we constantly store the latest signal values within the same time window $\Delta w$ length as selected for the reference data (0.05 seconds). By selection of an equal-sized time window, it is possible to make a direct comparison on the reference and measured data. The diagnosis is requested continuously within a time interval of 0.4 seconds. With every request, the latest measured signals are averaged at the request time point equal to equation (1.5.11) and result in $\overrightarrow{x_m}$. The generated averaged measured signal vector $\overrightarrow{x_m}$ is further used in the diagnosis process.

Within the diagnosis process the highest correlation between the averaged measured signals and the averaged reference data is searched. After the global minimum of the deviation is found, the related reference signal $\overrightarrow{x_r}$ is read out to get access to the parameter states $f_x$ used as a label for the reference data. Finally, the identified states $f_x$ ($f_{load}$ and $f_{motor}$) are returned as the actual diagnosis. Figure 1.5.3 shows a detailed description of the complete diagnosis

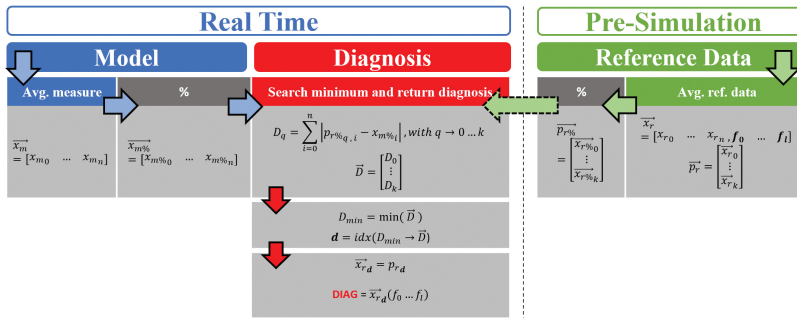


**Figure 1.5.3** Simulation-based diagnosis algorithm description.

algorithm equations for the search process starting with a triggered diagnosis based on the averaged measured signals $\overrightarrow{x_m}$ and reference signals $\overrightarrow{x_r}$.

Figure 1.5.4 illustrates the e-motor simulation, where the first three graphs show the battery current flow, the motor angular velocity and the angular acceleration over a time of 5 seconds. In addition, the markers indicate a diagnosis request of the system, whereby a green dot depicts a health system and a red cross means that a fault is detected at this point by the diagnosis algorithm. The last graph describes the actual set of states $f_x$ in the e-motor simulation (blue rectangle) and the system diagnosis (red arrow), whereby the diagnosis holds until a change in diagnosing is recognized.

We see that the system starts at health conditions ($ok$). After the first second the load fault $f_1$ (high load) is injected into the DC e-motor simulation. The algorithm recognizes the fault and returns the correct diagnosis. At the time of 2 seconds, the system is brought back to health conditions for 1 second when the motor state is set to fault $f_2$ (66% inductivity and resistor value lost). Since this fault is injected within a transient zone where no diagnosis request is triggered, the fault is detected with the next diagnosis request what explains the time delay of the diagnosis. With a higher diagnosis request rate, we obtain faster results, but this is limited in terms of real time diagnosis and the necessary computation time. The last fault injection into the system is a combined fault, it consists of a motor fault $f_2$ and a load fault $f_1$. The fault is again recognized and diagnosed correctly by the algorithm. We see again a delay between the injection and the computed diagnosis, due to the selected diagnosis interval of 0.4 seconds.

From this we conclude that the simulation-based diagnosis system is worth to be considered for further research since the overall algorithm is easy to implement and the system is robust in detecting different kinds of

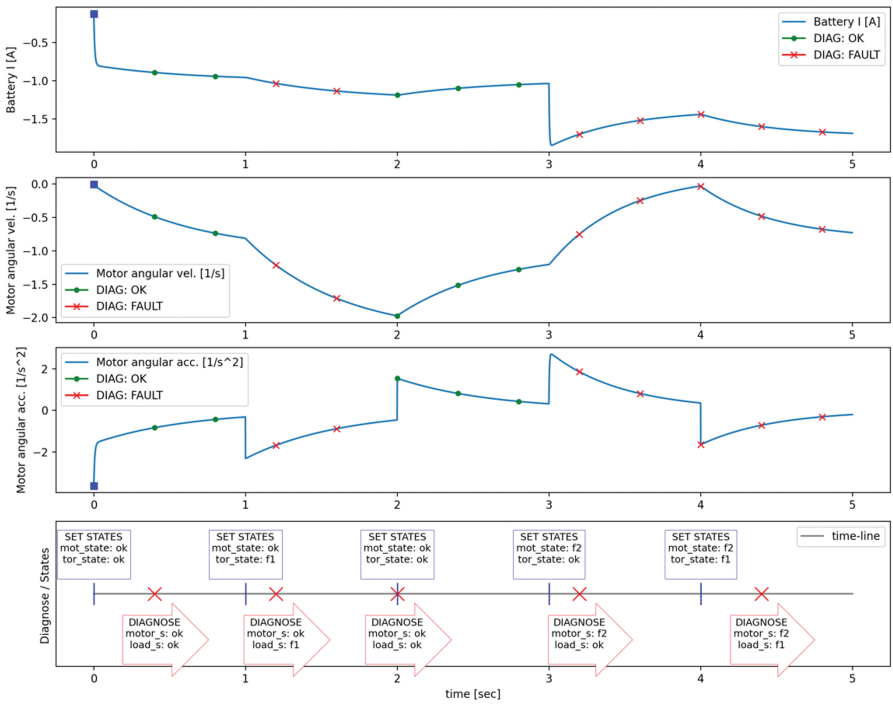


**Figure 1.5.4** Simplified DC e-motor diagnosis observation with simulation-based diagnosis.

faults and fault combinations if the faults to be diagnosed are known and the digital twin is able to simulate the behaviour precisely enough. Weaknesses are zones where different faults can raise similar characteristics during the initial phase that may result in wrong diagnoses. The reference table can also cause problems in terms of storage space, if too many different faults and fault combinations need to be diagnosed, which also has a direct negative impact on the computation time. Since we are only interested in short sections starting with the fault injection and ending when signals reach a certain equilibrium level, the storage of reference data is minimized.

### 1.5.2.4 Machine Learning for Diagnosis of Simplified DC e-Motor

As already discussed, that machine learning algorithms are not unfamiliar with the domain of fault diagnosis. There exist classes of algorithms e.g., Support vector machines, Decision Tree, K-Means, etc., which can be utilized to solve a complex problem related to fault diagnosis. For this case study, fault

diagnosis of DC motor, we modelled the fault diagnosis problem with one of the ensembles-technique-based machine learning algorithms called *Bootstrap Aggregation (Bagging)*. We have multiple fault states of the simplified DC motor for the diagnosis. Hence, we will use a classifier model for the classification task. We already have faulty behaviour simulated for each faulty state, that's why we adopted a supervised methodology to train the model. Now we will discuss the methodology for the fault diagnosis using machine learning in more detail.

In a classification problem, machine learning models take input of predictor variables corresponding to the dependent variable. In this case study, we have nine predictor variables and two dependent variables namely $load_{state}$ and $motor_{state}$ of the e-motor system. Dependent variable $load_{state}$ and $motor_{state}$ have a set of categorical labels defined as shown in (1.5.9). In order to transform the problem into the multi-classification problem, we combined the $load_{state}$ and $motor_{state}$ and introduced a new variable called *target* defined as follows:

$$target{=}load_{state}\ .motor_{state} \tag{1.5.13}$$

Finally, the distribution of the dependent variable is almost equally divided into faulty and non-faulty categories i.e., 42% and 58% respectively. As the dependent variable has the sequence of values across predictor variables, therefore, a machine learning model can be trained to learn the underlying pattern associated with each state of the system. As there are more than two states, a multi-classification model is trained as opposed to binary classification. Furthermore, as machine learning models require data to be numeric, we encoded the dependent variable with a label encoder in order to train and evaluate the model performance. Label encoder simply assigns a unique numeric integer value to a categorical label.

As discussed earlier that we used *Bagging* algorithm for the modelling, Random Forest is one of the machine learning models from *Bagging* classifiers. Random Forest is a bagging technique that simply combines (average) the outcome of multiple models and makes more accurate prediction than one model.

Model selection is one of the important and crucial parts of the training. The main reason to select *Random Forest* is that it performs well on both large and small datasets, and it can select the best subset of features that perform better and adds more information into the modelling. There is a number of hyper parameters associated with most of the machine learning models which can be fine-tuned to achieve the best performance. In this case

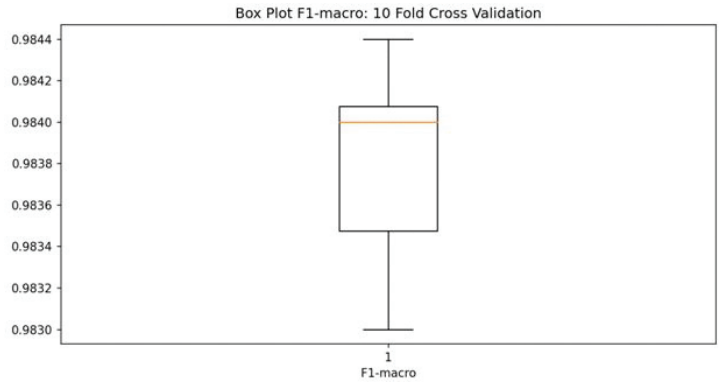


**Figure 1.5.5** Box plot 10-fold cross validation.

study we used the important parameters for random forest i.e., *n_estmiators = 100, criterion = gini, min_samples_split = 2, min_samples_leaf = 1* etc. to train and evaluate the model. For the evaluation of the model, K-fold cross-validation is used where *k=10*. K-fold cross-validation is used to ensure that the model is not overfitting [1] and it generalizes well. In this setting, the dataset is randomly divided into *K* chunks, and *K* models are trained on each chunk. Each model is trained using *K-1* chunks and validated on the remaining dataset. Finally, as an evaluation metric, we used *F1-macro* (macro-averaged), used to assess the quality of the model for multiple classes. *F1-macro* is an average of label-wise F1 scores, whereas the F1 score is basically a harmonic mean of precision and recall. For each fold, the *F1-macro* is calculated, and then averaged score for 10-folds is used to evaluate the performance of the model. Once the model is passed through the validation process to estimate the overall performance, the final model is trained and tested on the test data. Please note that the test data was not part of the training and validation process.

Next, we will discuss the results obtained from the diagnosis using the machine-learning model. Figure 1.5.5. shows the distribution of *F1-macro* over the 10 folds. For each fold, our model performs well as there are no outliers. The average score for 10-fold cross-validation is **0.9838**, which shows that model was able to classify and detect the faults correctly, for most of the states.

Figure 1.5.6. shows the confusion matrix for the test data, where each cell along the diagonal represents the correct classification of diagnosis and the rest of the cells show the misclassifications predicted by the model. Results

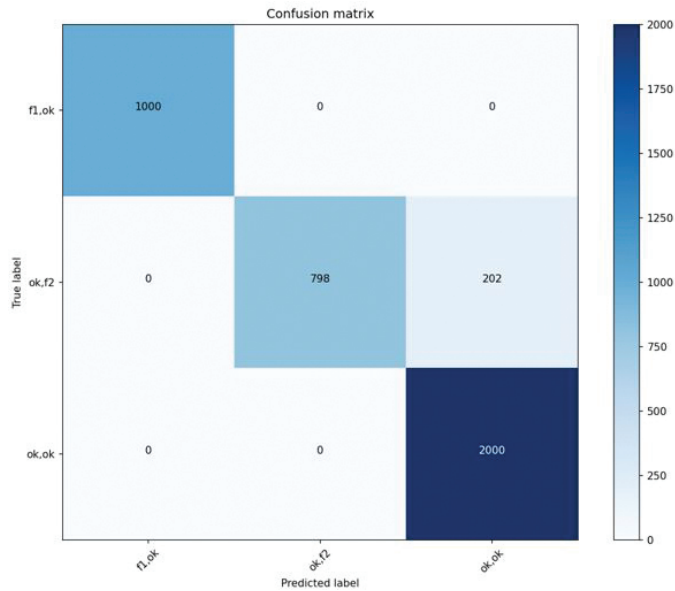


**Figure 1.5.6** Normalized confusion matrix - model testing/verification.

suggest that the model was able to correctly diagnose two faults i.e. $\{f_1, ok\}$ and $\{ok, ok\}$, whereas we have very few misclassifications for faulty state $\{ok, f_2\}$. The *F1-macro* score for the test data is **0.9465**. Results indeed show that machine learning can be useful for the fault diagnosis of the e-motor.

From this, we conclude that we developed a machine learning algorithm to classify the faulty states of the DC e-motor, based on the sequence of variables. Results indeed suggest that machine learning algorithms have the potential to learn the fault behaviour of the DC e-motor system. Although, there are few misclassifications of the faulty states, but it is indeed part of the learning as learning cannot be perfect. Bagging algorithms take the decision from multiple models. Hence, these algorithms have the ability to perform well. It is important to note that learning highly depends on the quality of data, as the model learns from the underlying distribution of the data and correlations (if exist). Results also suggest that each faulty state has the learning curve associated with it, as a result, advanced machine-learning algorithms e.g., boosting, neural networks, and deep learning can be tested and evaluated for future work. As machine learning algorithms are not pre-programmed, it gives them the advantage over other traditional faulty diagnosis techniques.

#### 1.5.2.5 Comparisons and Limitations

The used underlying methodology has some limitations due to the assumptions required. In the following section, the limitations and problems of each applied diagnosis algorithm on the DC e-motor model are summarized for comparison reasons.

The model-based diagnosis uses a detailed logic representation describing the model's components separately from the available simulation model. The logic models may be made for a particular purpose, e.g., hardware diagnose, and must be adapted to serve other goals, e.g., design diagnose of a system. Since the logic representation of complex cyber-physical systems is applied, the diagnosis is limited to observe state changes. Therefore, abstraction might not be that simple to map to real behaviour without ambiguity. Further, there is a lack of tools supporting the development and easily going through obtained results.

The simulation-based approach uses digital twin models directly to simulate healthy and faulty behaviour. The used parameter and obtained outputs from the simulation are analysed, processed, labelled and stored in a lookup table for further usage as a reference basis for the diagnosis search algorithm. The main requirement for this approach is to generate an accurate representation of the real system with the capability of fault injection. In addition, the faults and fault combinations must be previously defined to be able to diagnose them. With an increasing number of possible faults, limitation factors as computation time and storage space come to the fore. Another limitation is the adaptability to hardware or parameter changes in the real system, since a precise and realistic behaviour representation to obtain a correct diagnosis is needed.

Machine learning for diagnosis uses labelled reference data to train the system. Since the algorithm depends on the quality of the observed labelled data it is essential to have access to a precise simulated representation of the real system. Variability in the system hardware or parameter requires the machine learning model to be retrained which takes up an enormous amount of time. Models usually do not generalize well, and when deployed in real-time, results are affected by the data points which were not part of the training dataset. In addition, model selection is a crucial part of learning. Results may vary based on the model selected for the type of data, e.g., sequential and non-sequential and underlying distribution of the data. Further, if the labels of the fault type are not simulated properly the model will be biased towards the noise and the misclassification rate increases.

### 1.5.3 Conclusion

For the simplified DC e-motor, we introduced two types of components (motor, load) with the ability to inject faults as resistor and inductivity loss and a varying load factor. Based on this model, three methods, model-based diagnosis, simulation-based diagnosis and machine-learning diagnosis are introduced to be able to detect unexpected behaviour and outline its root cause. The model-based diagnosis method uses a logical representation of the simplified DC motor model to identify abnormal state changes. With this approach, we were able to come up with the expected solution in the particular case applying a high load fault during normal operation, still, the modelling complexity increases for more sophisticated models.

The simulation-based approach makes use of digital twin models directly to simulate normal and faulty behaviour to cover possible scenarios which are of interest for the diagnose part. The measurements and the corresponding state parameter are stored and used as reference data for the diagnosis search process during real-time observation of the simplified DC motor model. The simulation-based diagnosis approach delivers accurate diagnoses in real time with the limitation that only pre-simulated faults are considered to be diagnosed.

The last approach is the machine-learning diagnosis, which is capable to classify the faulty states based on the real-time measured signals of the model. As training data for the bagging algorithm, we use the simulated labelled reference data from a simulation-based approach which already covers different behaviours caused by fault injection. The machine-learning diagnosis model is validated with a 10-fold cross-validation method and the verification is done on unseen data which was not part of the validation set. We generated new instances of the system under test using the simulation-based approach architecture to run the DC e-motor model simulation to test the machine-learning diagnosis model.

## Acknowledgements

This work is conducted under the framework of the ECSEL AI4DI “Artificial Intelligence for Digitising Industry” project. The project has received funding from the ECSEL Joint Undertaking (JU) under grant agreement No 826060. The JU receives support from the European UnionâĂŹs Horizon 2020 research and innovation programme and Germany, Austria, Czech Republic, Italy, Latvia, Belgium, Lithuania, France, Greece, Finland, Norway. The

work was co-funded by grants of Ministry of Education, Youth and Sports of the Czech Republic, by the Austrian Federal Ministry of Transport, Innovation and Technology (BMVIT) under the program "ICT of the Future" between May 2019 and April 2022 (more information can be retrieved from https://iktderzukunft.at/en/). The work was also supported by the infrastructure of RICAIP that has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 857306 and from Ministry of Education, Youth and Sports under OP RDE grant agreement No CZ.02.1.01/0.0/0.0/17_043/0010085.

## References

[1] G. a. W. D. a. H. T. a. T. R. James, An Introduction to Statistical Learning: with Applications in R, Springer, 2013.

[2] Kaufmann, D., Nica, I., Wotawa, F.: Intelligent agents diagnostics - enhancing cyber-physical systems with self-diagnostic capabilities. Advanced Intelligent Systems, 2021. DOI https://doi.org/10.1002/aisy.202000218.