Isolation and Localization of Unknown Faults Using Neural Network-Based Residuals

Daniel Jung

Department of Electrical Engineering, Linköping University, Linköping, 581 83, Sweden
daniel.jung@liu.se

ABSTRACT

Localization of unknown faults in industrial systems is a difficult task for data-driven diagnosis methods. The classification performance of many machine learning methods relies on the quality of training data. Unknown faults, for example faults not represented in training data, can be detected using, for example, anomaly classifiers. However, mapping these unknown faults to an actual location in the real system is a non-trivial problem. In model-based diagnosis, physical-based models are used to create residuals that isolate faults by mapping model equations to faulty system components. Developing sufficiently accurate physical-based models can be a time-consuming process. Hybrid modeling methods combining physical-based methods and machine learning is one solution to design data-driven residuals for fault isolation. In this work, a set of neural network-based residuals are designed by incorporating physical insights about the system behavior in the residual model structure. The residuals are trained using only fault-free data and a simulation case study shows that they can be used to perform fault isolation and localization of unknown faults in the system.

1. INTRODUCTION

An important task of fault diagnosis of industrial systems is fault localization, i.e., identifying where faults are located in the system. Increasing system complexity and autonomous operation require that the system is reliable and able to detect faults early before any accidents or damages occur. Being able to identify a faulty component gives important information when deciding for suitable counter-measures to minimize costs and the risk of potential dangers.

Machine Learning has been very successful in many applications, including image classification and text analysis. One example of such methods are neural networks and deep learning. However, some of the recent successes have been made possible thanks to the access to large amounts of training data (Jia, Lei, Lin, Zhou, & Lu, 2016). In many fault diagnosis applications, collecting representative data is complicated and expensive, especially during the system development phase and early system life (Sankavaram, Kodali, Pattipati, & Singh, 2015). Even though incremental classification algorithms are able to improve performance over time, as more training data become available, it is still relevant to be able to identify likely fault locations of fault scenarios not covered in training data. This is important in, for example, troubleshooting (Pernestål, Nyberg, & Warnquist, 2012). One solution to limited training data is to use physical-based models when implementing machine learning algorithms.

The idea of using physical-based model structures in data-driven neural network design for fault diagnosis has been proposed in (Pulido, Zamarreño, Merino, & Bregon, 2019). With respect to the mentioned work, this paper presents how to use neural network-based residuals and physical-based models to localize unknown faults in the system. In (Garcia-Alvarez, Bregon, Fuente, & Pulido, 2011) a model parameter estimation approach based on a partitioned system model is proposed.

The benefits of combining model-based and data-driven fault diagnosis methods have also been discussed in (Tidriri, Chatti, Verron, & Tiplica, 2016). Hybrid diagnosis system designs, combining model-based residuals and machine learning classifiers, have been proposed in, for example, (Jung & Sundström, 2017; Tidriri, Tiplica, Chatti, & Verron, 2018; Jung, Ng, Frisk, & Krysander, 2018). The methods in these mentioned papers, relies on residuals to perform fault isolation and classification. Therefore, residual generation is an important task during the diagnosis system design to achieve satisfactory fault isolation performance. With respect to these previous works, not only fault isolation is considered here but also localization of unknown faults.

1.1. Problem Statement

Even though a data-driven classifier is able to identify when an unknown fault has occurred, it is non-trivial to localize the

fault in the actual system without training data from that fault. One solution is to utilize physical insights about the system when implementing a machine learning algorithm.

In this work, a simulation study is performed to investigate if it is possible to point out the fault location in a system using a set of unconventional neural network-based residuals where the network design represents the structure of the system. The neural network design is implemented using Python 3 and PyTorch (Paszke et al., 2017). It is assumed that training data to train the neural network models are available from nominal system operation only, i.e. no data from any fault scenario are used during the training phase. The study shows that including some physical insights in the neural network design makes it possible to detect and localize system faults even though the networks are trained using data from nominal system operation only.

For the neural network design, structural model decomposition methods are used on a structural representation of the system. A structural model describes the relationship between system variables without considering the analytical relation, i.e., it only describes which variables are included in each model equation (Blanke, Kinnaert, Lunze, Staroswiecki, & Schröder, 2006). Different residual generation algorithms use the structural model to create computational graphs to evaluate the model equations from sensor data to compute a residual, see for example (Frisk, Krysander, & Jung, 2017; Pulido & González, 2004). If a detailed analytical model of the system is not available, a structural model representing the physical-based relations, describing the system behavior, can still be used to design neural-networks for residual generation.

2. A NON-LINEAR TWO TANK SIMULATION CASE STUDY

To illustrate the proposed method, a non-linear two-tank simulation model is used to simulate sensor data. An illustration of the system is shown in Figure 1 and the model dynamics are derived from the Bernoulli equation as

$$\dot{x}_1 = -d_1\sqrt{x_1} + d_2 u, \quad \dot{x}_2 = -d_3\sqrt{x_1} + d_4\sqrt{x_2},$$
$$y_1 = x_1, \quad y_2 = x_2, \quad y_3 = d_5\sqrt{x_1}, \quad y_4 = d_6\sqrt{x_2} \quad (1)$$

where x_i is water level in tank i, u is a known input flow in tank one, y_1 and y_2 measure the water level in each tank, respectively, y_3 and y_4 measure the out-flow from each tank, respectively, and $d_1, \ldots, d_6$ are model parameters.

In this case study, it is assumed that an accurate model of the system is not available for the diagnosis system design. Instead, a qualitative model is available that describes the gen-
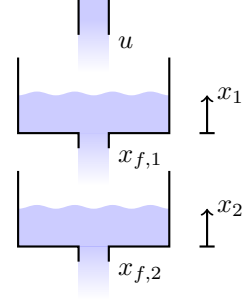

Figure 1. An illustration of the two tank system.

eral system behavior as follows:

$$
\begin{aligned}
&e_1 : \dot{x}_1 = h_1(x_{f,1}, u) && e_5 : y_1 = x_1 \\
&e_2 : \dot{x}_2 = h_2(x_{f,1}, x_{f,2}) && e_6 : y_2 = x_2 \\
&e_3 : x_{f,1} = g_1(x_1) && e_7 : y_3 = x_{f,1} \\
&e_4 : x_{f,2} = g_2(x_2) && e_8 : y_4 = x_{f,2}
\end{aligned}
\quad (2)
$$

where $x_{f,i}$ is out-flow from tank i, u is a known input flow into tank one, and y_1, y_2, y_3, y_4 are sensor data. The functions $h_1(\cdot)$ and $h_2(\cdot)$ state that the change in water level in each tank depend on the inflow and outflow. The functions $g_1(\cdot)$ and $g_2(\cdot)$ say that the outflow depends on the water level in the tank.

An example of simulated data from the system is shown in Figure 2. For the evaluation, the simulation model can be used to simulate different faults, for example leakages in the tank, clogging in the outflow pipes, and sensor faults.
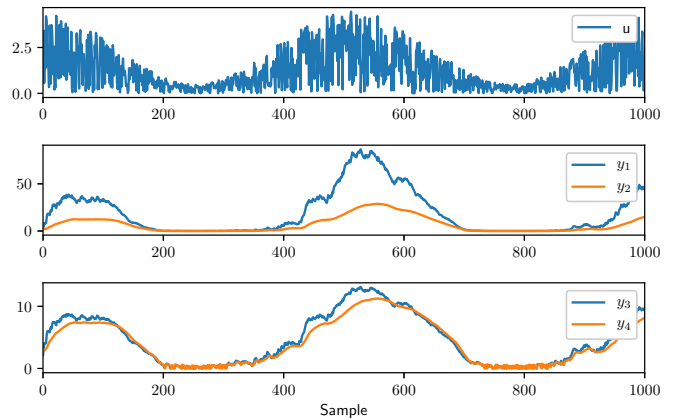

Figure 2. An example of simulated data from the two tank system.

3. BACKGROUND

First, a brief summary of artificial neural networks is presented. Then, the principles of model-based diagnosis and structural analysis methods are summarized.

3.1. Artificial Neural Networks

Artificial neural networks and deep learning are a set of machine learning methods that can be used to approximate non-linear functions (Schmidhuber, 2015). Neural networks consist of a set of neurons where the output from some neurons are inputs to other neurons and can be represented as a computational graph. Each neuron is a non-linear function of the inputs to the neuron, for example

$$x_{i,\text{out}} = h_i(w_i^T x_{i,\text{in}} + \beta_i) \tag{3}$$

where $x_{i,\text{in}}$ denote the inputs to neuron i, $x_{i,\text{out}}$ the output, w_i is a vector of weights, β_i is a bias, and h_i is a non-linear activation function, e.g. rectified linear unit (ReLU), sigmoid or hard tan (Aggarwal, 2018). A common method for training neural networks is to use back-propagation.

Conventional neural network designs arrange the neurons in different layers. The first layer of the neural network is denoted the input layer, the final layer is called the output layer, and all layers in between are called hidden layers. One type of neural networks, called recurrent neural networks, can be used to model temporal dynamic systems (Pearlmutter, 1995). In recurrent neural networks, the output from some neurons are used as input to other neurons at concurring time steps. This is shown in Figure 3 where the state variable $\hat{x}_1$ is used as input in the next time instance of the recurrent neural network. The variable u_t is an input signal and $\hat{y}_t$ is an output signal at time instance t.
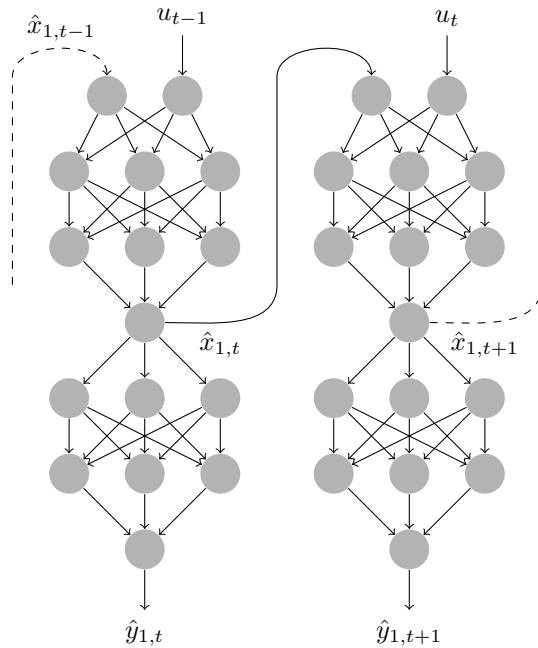

Figure 3. An example of a recurrent neural network. The figure illustrates the neural network structure of residual r_5 in Eq. (11) that is used to compute $\hat{y}_t$.

3.2. Model-Based Fault Diagnosis

In model-based fault diagnosis, faults are identified by detecting inconsistencies between sensor data y and predictions $\hat{y}$ from a physical-based model of the system, using residuals $r = y - \hat{y}$. To generate residuals require analytical redundancy in the model (Travé-Massuyès, 2014). A residual is a function of known variables and is, ideally, zero in the nominal case. Because of model uncertainties and sensor noise, different types of statistical tests are used to determine when a significant change in the residual output has occurred. Each residual models nominal system behavior, and can thus be interpreted as an anomaly classifier (Gupta, Gao, Aggarwal, & Han, 2014).

3.2.1. Structural Methods

A useful analysis tool for model-based diagnosis is structural methods (Blanke et al., 2006). A structural model is a bipartite graph describing the relationship between variables and equations and can be represented as an incidence matrix. Figure 4 shows an example of an structural representation of the two tank model Eq. (2). Each row represents a model equation and each column a model variable. Equations e_9 and e_{10} are used in the structural model to state the relationship that $\dot{x} = \frac{dx}{dt}$ where I in Figure 4 is used in these equations to highlight the state variable and D its derivative (Frisk et al., 2012).

The structural model is not dependent on the actual analytical expression which makes it a useful tool for analysis during early system design since no parameter values are needed. By using a method called Dulmage-Mendelsohn decomposition on the structural model, it is possible to, for example, perform fault detectability and isolability analysis but also find redundant equation sets for residual generation (Krysander, Åslund, & Nyberg, 2008).

An example of a redundant equation set given Eq. (2) is $\{e_1, e_3, e_5\}$. The three equations contain two unknown variables x_1 and $x_{f,1}$ and can be used to generate a residual. Matching algorithms can be used with redundant equation sets to generate residuals, see for example (Frisk et al., 2017). In principle, a matching algorithm finds a computational sequence describing how the known signals should be used to, sequentially, compute the unknown variables in the model using the equation set when one of the equations is used as a residual equation.

Example 3.1 *Asssume that the functions h_1 and g_1 in Eq. (2) are known. The equation set $\{e_1, e_3, e_5\}$ can be used to design different residuals, for example,*

$$\dot{\hat{x}}_1 = h_1\left(g_1(\hat{x}_1), u\right)$$
$$r = y_1 - \hat{x}_1 \tag{4}$$
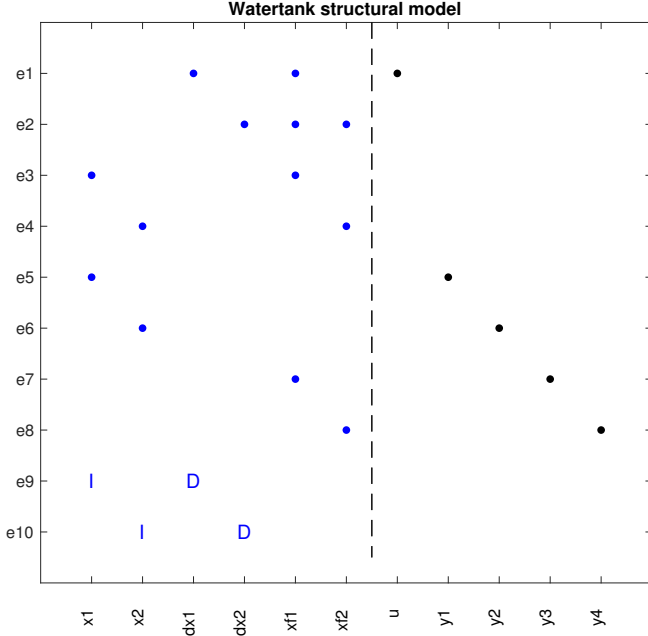
Watertank structural model

Figure 4. A structural representation of the two tank model.

or

$$r = \dot{y}_1 - h_1\left(g_1(\hat{x}_1), u\right). \tag{5}$$

Equation (4) is an example of a residual with integral causality and Eq. (5) with derivative causality (Frisk et al., 2012). If the redundant equation set does not contain any dynamic equations the residual is said to be an algebraic relation. In this work, only integral causality will be considered for residuals with dynamic equations.

3.2.2. Model-Based Residual Design

Depending on which redundant set of model equations is used to generate a residual, the residual will be sensitive to faults in a certain part of the system. If different residuals are designed using different sets of redundant equations, referred to as the *model support* of the residual, the set of residuals will give a specific fault pattern depending on where a fault occurs in the system (Travé-Massuyès, 2014). Residuals that are designed to monitor the part of the system where a fault occurs should, ideally, deviate from their nominal behavior, while the other residuals should not be affected. By comparing the model support of the residuals that have deviated from their nominal behavior, it is possible to identify possible locations of the fault in the system based on the model equations (Pucel, Mayer, & Stumptner, 2009).

4. NEURAL NETWORK-BASED RESIDUAL GENERATION

Based on the structural model of the system, a set of different redundant equation sets are identified using the fault diag-

nosis toolbox (Frisk et al., 2017). Based on each redundant equation set, a residual is modeled using a recurrent neural network where the location of the state variables are given by the structural model. For example, the residual in Eq. (4) is reformulated as

$$\dot{\hat{x}}_{1,t} = \xi\left(\hat{x}_{1,t-1}, u_{t-1}\right)$$
$$r_t = y_{1,t} - \hat{x}_{1,t} \tag{6}$$

where subscript t denotes time index, the unknown function $\xi : \mathbb{R}^2 \to \mathbb{R}$ is modeled using a neural network and the state variable is approximated using the Euler forward method to formulate a time-discrete model. The resulting residual function is a recurrent neural network with only one state variable, as illustrated in Figure 3.

Similarly, six additional residuals are implemented from different redundant equation sets, using the principles described in for example (Frisk et al., 2012), and the final residual set is summarized as follows:

$$r_{1,t} = y_{4,t} - \xi_1\left(y_{2,t}\right) \tag{7}$$

$$r_{2,t} = y_{3,t} - \xi_2\left(y_{1,t}\right) \tag{8}$$

$$\hat{x}_{2,t} = \xi_{3a}\left(y_{3,t-1}, \hat{x}_{2,t-1}\right)$$
$$r_{3,t} = y_{4,t} - \xi_{3b}\left(\hat{x}_{2,t}\right) \tag{9}$$

$$\hat{x}_{2,t} = \xi_4\left(y_{1,t-1}, \hat{x}_{2,t-1}\right)$$
$$r_{4,t} = y_{2,t} - \hat{x}_{2,t} \tag{10}$$

$$\hat{x}_{1,t} = \xi_{5a}\left(\hat{x}_{1,t-1}, u_{t-1}\right)$$
$$r_{5,t} = y_{3,t} - \xi_{5b}\left(\hat{x}_{1,t}\right) \tag{11}$$

$$\hat{x}_{1,t} = \hat{x}_{1,t-1} + \xi_6\left(y_{3,t-1}, u_{t-1}\right) + 0.01\left(y_{1,t} - \hat{x}_{1,t}\right)$$
$$r_{6,t} = y_{1,t} - \hat{x}_{1,t} \tag{12}$$

$$\hat{x}_{1,t} = \xi_{7a}\left(\hat{x}_{1,t-1}, u_{t-1}\right)$$
$$\hat{x}_{2,t} = \xi_{7b}\left(\hat{x}_{1,t-1}, \hat{x}_{2,t-1}\right)$$
$$r_{7,t} = y_{2,t} - \hat{x}_{2,t} \tag{13}$$

where each function $\xi_i(\cdot)$ is modeled as a neural network. Residuals $r_3, ..., r_7$ are modeled as recurrent neural networks with similar structures as in Figure 3. The model support of all residuals are summarized in Table 1.

Residuals r_1 and r_2 are static algebraic relations modeling the relation between measured water levels in each tank and the measured outflow in each tank. Residuals $r_3, ..., r_7$ have internal dynamics, where a small feedback term was introduced in the dynamic equation of residual r_6 in Eq. (12) because of difficulties to achieve satisfactory prediction error when training the model. Also, note that in r_6, with respect to the other dynamic residuals, the term $\hat{x}_{1,t-1}$ is not included as an in-

Table 1. A summary of the equation sets used to design each neural network-based residual.

	e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8
r_1				X		X		X
r_2			X		X		X	
r_3		X		X			X	X
r_4		X	X	X	X	X		
r_5	X		X				X	
r_6	X					X	X	
r_7	X	X	X	X		X		

Table 2. A summary of the equation sets used to design each neural network-based residual.

	e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8
e_1	X							
e_2		X		X				
e_3			X					
e_4				X				
e_5					X			
e_6				X		X		
e_7							X	
e_8				X				X

put in the neural network model $\xi_6(\cdot)$ but kept outside. This is necessary to maintain the redundant model structure and to make sure that faults not directly affecting the equations in the model support of r_6 are decoupled. Note that r_6 resembles a ResNet structure (He, Zhang, Ren, & Sun, 2016).

Ideal fault localization performance, i.e. when model uncertainties are not considered, given the selected residual set is summarized in Table 2. A fault in e_i is isolable from a fault in e_j if there is a residual where e_i is part of its model support but not e_j. An X at position (i, j) means that a fault affecting equation e_i cannot be isolated from a fault affecting equation e_j. Based on the selected residuals it is possible to, ideally, localize a fault to a part of the system modeled by one equation. The exceptions are faults in e_2, e_6, and e_8 that cannot be isolated from a fault in e_4.

4.1. Implementation and Training of Residuals

Each residual is implemented in Python and PyTorch and trained using simulated fault-free data, see Figure 2. Each non-linear function $\xi(\cdot)$ is here modeled using a neural network with three hidden layers, 32 neurons in each layer, and ReLU as activation function. The training is performed by simulating the system and minimizing the mean square error $\sum_t (y_t - \hat{y}_t)^2$ using the ADAM solver (Kingma & Ba, 2014) and truncated back-propagation through time (Werbos, 1990). It is important that training data are representative of nominal system operation since model validity is not expected for operating points not covered by training data.

An example of the evaluated residual outputs in the nomi-

nal case are shown in Figure 5. The left column shows the residual time-series data, while the second column shows the histogram of each residual. The two histograms in each plot show the residual distribution during the first and second half of the time-series. The red dashed lines represent the 1% and 99% quantiles of the blue histograms representing the first half of the data set. These will be used analyze the residual outputs when the distributions are affected by different faults. Note that more sophisticated change detection algorithms can be used, instead of thresholding the residual, to automatically detect changes in the residual output, for example CUmulative SUM (CUSUM) (Page, 1954).
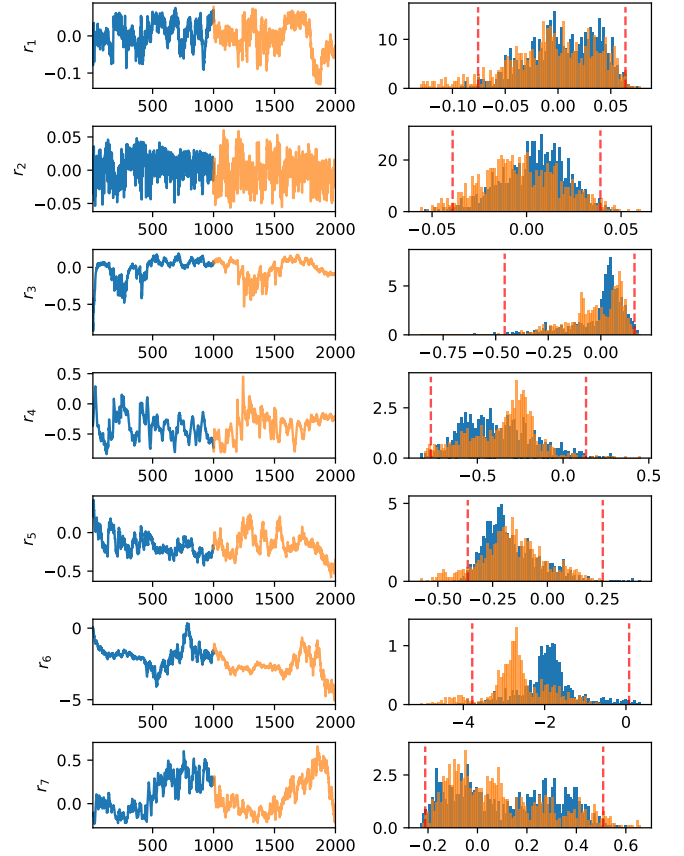

Figure 5. The residual outputs using simulated fault-free data.

5. EVALUATION

To evaluate the fault localization performance of the neural network-based residuals, different fault scenarios are simulated. For single-fault scenarios, fault localization can be performed by analyzing the intersection of the model support of all residuals that significantly deviate from their nominal behavior. To handle multiple-fault scenarios, minimal hitting set algorithms, such as the one proposed in (De Kleer & Williams, 1987), can be applied to identify likely fault localizations.

The first simulated fault scenario is a leakage fault in tank one occurring after sample 1000. The residual outputs are shown in Figure 6 where the time-series data are plotted in the left column and the histogram of the residuals, before and after the fault occurs, in the right column. When comparing the distributions in the right column it is visible that residuals r_5 and r_6 deviate significantly from their nominal behavior while the other ones does not.

When comparing the model support for r_5 and r_6 in Table 1, the intersection of the corresponding equation sets is $\{e_1, e_7\}$, indicating that the fault should affect the part of the system described by one of the two equations. Equation e_1 describes the water level dynamics in tank one and e_7 the sensor measuring the outflow from tank one. A leakage will affect the dynamics of the water level in the tank since there is an additional outflow from the tank not captured by the nominal model e_1 thus correctly narrowing down the location of the fault. Ideally, r_7 should also react to the leakage but is not deviating significantly in this case. An explanation could be that the accuracy of the residual model is not good enough to distinguish the fault.
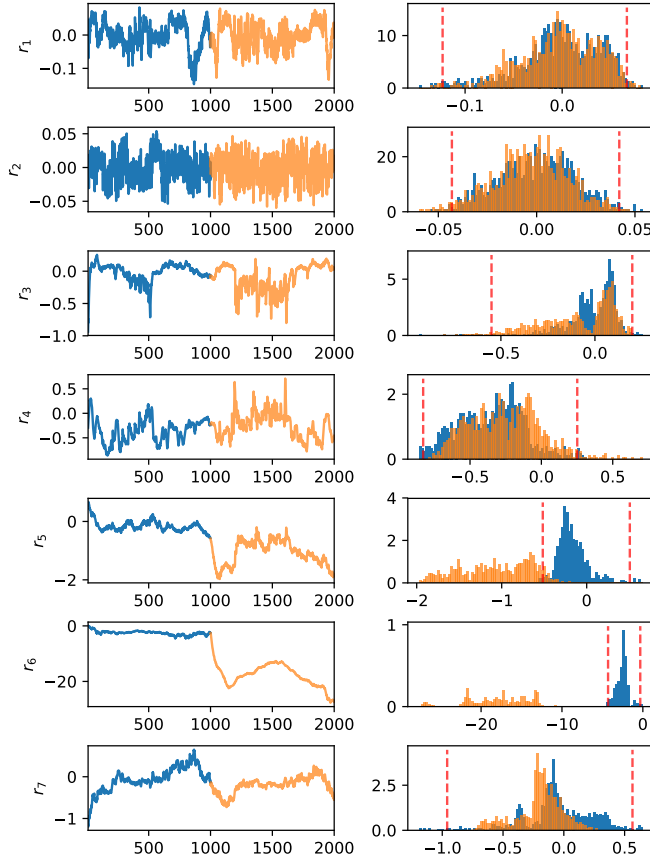

Figure 6. The residual outputs when simulating a leakage in tank one.

In the second fault scenario, a clogging affecting the outflow

from tank two is simulated and the residual outputs are shown in Figure 7. In this case, there is a significant change in the distributions of residuals r_1 and r_4, while a small change can be noticed in r_7. Based on the model support in Table 1 the intersection is $\{e_4, e_6\}$. Equation e_4 describes the relation between water level in tank one and the resulting outflow and e_6 the sensor measuring the level in tank two. The clogging fault is identified since the fault results in a decreased outflow described by e_4. Note that residual r_3, which is sensitive to a fault in e_4 makes a sudden change when the fault occurs but then goes back to nominal behavior. If a change detection algorithm applied to r_3 also triggers an alarm, e_4 would be isolated uniquely.
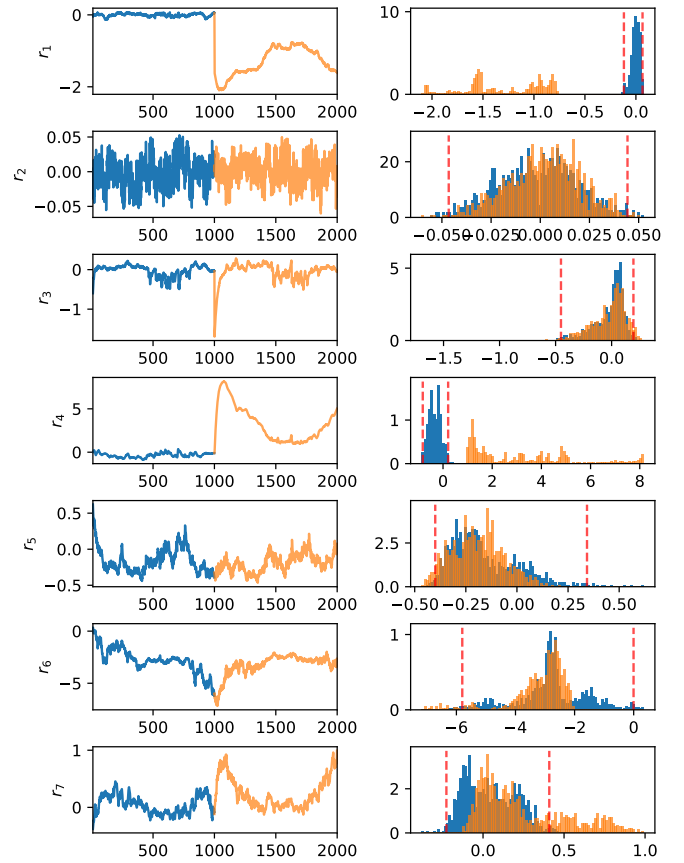

Figure 7. The residual outputs when simulating clogging in the outflow pipe of tank two.

The results from the two fault scenarios show that the trained set of neural network-based residuals can be used to identify the fault location in the actual system. Even if there were more than one equation where the fault could be located in the two scenarios, it gives useful information to a technician where to start troubleshooting.

6. CONCLUSIONS

The case study shows that it is possible to perform fault localization of unknown faults using neural network-based residuals without the need of training data from faults. Developing accurate physical-based models can be time consuming. Hybrid methods combining qualitative models and machine learning can be one solution to reduce development time while still be make to make use of the structural properties of the physical-based model.

REFERENCES

Aggarwal, C. C. (2018). *Neural networks and deep learning*. Springer.

Blanke, M., Kinnaert, M., Lunze, J., Staroswiecki, M., & Schröder, J. (2006). *Diagnosis and fault-tolerant control* (Vol. 2). Springer.

De Kleer, J., & Williams, B. (1987). Diagnosing multiple faults. *Artificial intelligence*, *32*(1), 97–130.

Frisk, E., Bregon, A., Åslund, J., Krysander, M., Pulido, B., & Biswas, G. (2012). Diagnosability analysis considering causal interpretations for differential constraints. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, *42*(5), 1216–1229.

Frisk, E., Krysander, M., & Jung, D. (2017). A toolbox for analysis and design of model based diagnosis systems for large scale models. *IFAC-PapersOnLine*, *50*(1), 3287–3293.

Garcia-Alvarez, D., Bregon, A., Fuente, M., & Pulido, B. (2011). Improving parameter estimation using minimal analytically redundant subsystems. In *2011 50th ieee conference on decision and control and european control conference* (pp. 7788–7793).

Gupta, M., Gao, J., Aggarwal, C., & Han, J. (2014). Outlier detection for temporal data: A survey. *IEEE Transactions on Knowledge and Data Engineering*, *26*(9), 2250–2267.

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 770–778).

Jia, F., Lei, Y., Lin, J., Zhou, X., & Lu, N. (2016). Deep neural networks: A promising tool for fault characteristic mining and intelligent diagnosis of rotating machinery with massive data. *Mechanical Systems and Signal Processing*, *72*, 303–315.

Jung, D., Ng, K., Frisk, E., & Krysander, M. (2018). Combining model-based diagnosis and data-driven anomaly classifiers for fault isolation. *Control Engineering Practice*, *80*, 146–156.

Jung, D., & Sundström, C. (2017). A combined data-driven and model-based residual selection algorithm for fault detection and isolation. *IEEE Transactions on Control Systems Technology*, *PP*(99), 1-15.

Kingma, D., & Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Krysander, M., Åslund, J., & Nyberg, M. (2008). An efficient algorithm for finding minimal overconstrained subsystems for model-based diagnosis. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, *38*(1), 197–206.

Page, E. (1954). Continuous inspection schemes. *Biometrika*, *41*(1/2), 100–115.

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., . . . Lerer, A. (2017). Automatic differentiation in pytorch. In *Nips-w.*

Pearlmutter, B. (1995). Gradient calculations for dynamic recurrent neural networks: A survey. *IEEE Transactions on Neural networks*, *6*(5), 1212–1228.

Pernestål, A., Nyberg, M., & Warnquist, H. (2012). Modeling and inference for troubleshooting with interventions applied to a heavy truck auxiliary braking system. *Engineering applications of artificial intelligence*, *25*(4), 705–719.

Pucel, X., Mayer, W., & Stumptner, M. (2009). Diagnosability analysis without fault models. In *20th international workshop on principles of diagnosis (dx-09)* (pp. 67–74).

Pulido, B., & González, C. (2004). Possible conflicts: a compilation technique for consistency-based diagnosis. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, *34*(5), 2192–2206.

Pulido, B., Zamarreño, J., Merino, A., & Bregon, A. (2019). State space neural networks and model-decomposition methods for fault diagnosis of complex industrial systems. *Engineering Applications of Artificial Intelligence*, *79*, 67–86.

Sankavaram, C., Kodali, A., Pattipati, K., & Singh, S. (2015). Incremental classifiers for data-driven fault diagnosis applied to automotive systems. *IEEE Access*, *3*, 407–419.

Schmidhuber, J. (2015). Deep learning in neural networks: An overview. *Neural networks*, *61*, 85–117.

Tidriri, K., Chatti, N., Verron, S., & Tiplica, T. (2016). Bridging data-driven and model-based approaches for process fault diagnosis and health monitoring: A review of researches and future challenges. *Annual Reviews in Control*, *42*, 63–81.

Tidriri, K., Tiplica, T., Chatti, N., & Verron, S. (2018). A generic framework for decision fusion in fault detection and diagnosis. *Engineering Applications of Artificial Intelligence*, *71*, 73 - 86.

Travé-Massuyès, L. (2014). Bridging control and artificial intelligence theories for diagnosis: A survey. *Engineering Applications of Artificial Intelligence*, *27*, 1–16.

Werbos, P. (1990). Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, *78*(10),

1550–1560.

BIOGRAPHIES

Daniel Jung was born in Linköping, Sweden in 1984. He received a Ph.D. degree in 2015 from Linköping University, Sweden. During 2017, he was a Research Associate at the Center for Automotive Research at The Ohio State University, Columbus, OH, USA. Since 2018, he is an Assistant Professor at Linköping University. His current research interests include theory and applications of model-based and data-driven fault diagnosis, smart grids, and optimal control of hybrid electric vehicles.