Real time model-based diagnosis enabled by hybrid modeling

Ion Matei[1], Alexander Feldman[2], Johan de Kleer[3] and Alexander Perez[4]

[1,2,3,4] *Palo Alto Research Center, Palo Alto, CA, 94304, USA*
imatei@parc.com
afeldman@parc.com
dekleer@parc.com
aperez@parc.com

ABSTRACT

In this paper we propose a hybrid modeling approach for generating reduced models of a high fidelity model of a physical system. We propose machine learning inspired representations for complex model components. These representations preserve in part the physical interpretation of the original components. Training platforms featuring automatic differentiation are used to learn the parameters of the new representations using data generated by the high-fidelity model. We showcase our approach in the context of fault diagnosis for a rail switch system. We generate three new model abstractions whose complexities are two order of magnitude smaller than the complexity of the high fidelity model, both in the number of equations and simulation time. Faster simulations ensure faster diagnosis solutions and enable the use of diagnosis algorithms relying heavily on large numbers of model simulations.

1. INTRODUCTION

In model-based approaches, the diagnosis engine is provided with a model of the system, nominal values of the parameters of the model and values of some of its inputs and outputs. The main goal of a diagnosis engine is to determine from only this information the presence of a fault and to isolate it. There is a rich literature on model-based diagnosis results proposed independently by the artificial intelligence (de Kleer, Mackworth, & Reiter, 1992) and control (Gertler, 1998),(Isermann, 2005),(Patton, Frank, & Clark, 2000) communities. Model-based diagnosis requires accurate models to detect and isolate faults in physical systems. For real-time diagnosis, such models need to simulate withing an allotted time interval. Typically, the more accurate models are, the more complex become and hence it takes more time to simulate them. Traditional model-based diagnosis include filters (e.g., Kalman filter (Kalman, 1960), particle filter (Arulampalam, Maskell, & Gordon, 2002)), or optimization based-techniques that estimate a number of parameters whose deviation from their nominal values indicate the presence of a fault. These methods rely on model simulations either for one sample period (Kalman and particle filters) or for some time horizon (optimization based). The simulation time becomes even more stringent in the case of the particle filter where a possibly large number of particles require their own model simulations. One may argue the many system admit ordinary differential equations (ODEs) models as representations and a one time step forward propagation is not necessarily a complex operation. Although this is true for some physical systems, many others require differential algebraic equations (DAEs) as mathematical representations. DAE simulations require the use of the Newton-Rhapson algorithm that in turn requires the inversion of a Hessian matrix. The Hessian matrix size depends on the complexity of the model (e.g., number of equations and variables).

In this paper we propose a hybrid modeling approach to reduce the complexity of a high fidelity model of a physical system. The reduced complexity model is used by a diagnosis-engine to detect and isolate system faults. Since we use the model for model-based diagnosis, here we care on the effects of the model complexity on the simulation time. The hybrid modeling approach is based on: (i) *identifying* the system components responsible for making the simulation time taking a long time, (ii) finding new parameterized *representation*s for such components, and (iii) *learn* the parameters of the new components We make sure that the chosen representations preserve, at least in part, the physical meaning of the original physical components. Such a meaning is particularly useful in diagnosis since it points to a physical explanation of a faulty behavior.

2. PROBLEM DESCRIPTION

We consider physical systems whose behavior can be described by a set of DAEs of the form

$$0 = F(\dot{x}, x, u), \quad (1)$$
$$y = h(x, u), \quad (2)$$

where x represents the state vector, u is a vector of inputs, and y is a vector of outputs. We consider *parametric faults*: faults that can be described through changes in system parameter values. Parametric faults do not impose significant constraints on the type of faults we can detect and isolate. Indeed, as shown in our previous work (Honda & et al., 2014; Minhas et al., 2014; Saha & et al., 2014), we can augment the physical model with fault modes inspired by the physics of failure. The physics-based fault augmentation process adds additional equations to the model. These new equations are dependent on parameters whose activation induces the simulated faulty behavior. The type of faults introduced are domain dependent. We can cover electrical (short, open connections, parameter drifts), mechanical (broken flanges, stuck flanges, torque losses due to added friction, efficiency losses), or fluid (blocked pipes, leaking pipes) domains.

Let $\mathcal{F} = \{F_0, F_1, \ldots, F_L\}$ denote a set of faults we would like to detect and isolate, where F_0 denoted the normal behavior. The diagnosis objective is to determine a classifier $f : Y \rightarrow \{F_0, F_1, \ldots, F_L\}$, where Y is a set of observations of the system behavior, typically given as time series that are processed sample by sample (online) or as a batch (offline). We associate a set of fault parameters $\{\theta_1, \ldots, \theta_L\}$ to each of the fault modes with nominal values θ_i^* for $i \in \{1, \ldots, L\}$. The classifier fault detection scheme is defined as a variation of the observations from their expected values. The fault isolation is based on the deviation of the fault parameters from their nominal values, i.e., $\|\theta_i - \theta_i^*\| \geq \epsilon_i$, where ϵ_i is a fault specific threshold that can depend on the noise statistics for example. Several fault parameter deviations are simultaneously possible, hence there may be some ambiguity in the fault diagnosis. This case happens when the sensor measurements do not contain enough information to differentiate between distinct faults. The fault are tracked either online or offline using filters or optimization based parameter estimation techniques.

3. RAIL SWITCH MODEL

As a case study, we consider a rail switch system used for guiding trains from one track to another. The rail switch is composed of a servo-motor and a gear-mechanism for scaling the rotational motion and for amplifying the torque generated by the electrical motor. The rail load is composed by a mechanical adjuster, and tongue-rails. The rail switch model has 5522 equations, and the rail component on its own has 4768 equations. We note that the majority of the model complexity is concentrated on the rail model which was obtained by using a finite element analysis approach (FEA). In particular, each beam was approximated as a sequence of connected 2D mass-spring-dampers, where the springs and dampers oppose the rotational motion. Hence producing a reduced representation of this model improves its usability, especially in real time applications. The input for the rail switch signal is a reference signal for the servo-motor controller for each of the two direction of motions. The time horizon for each input reference signal is 7 sec. Using the high-fidelity model, it takes more than 7 sec to simulate the model for 14 sec, simulation time that includes the rail displacement in both directions. Our objective is to replace the rail component with a simpler representation, to significantly reduce the simulation time.

4. FAULT AUGMENTATION

In this section we describe the modeling artifacts that were used to include in the behavior of the system four fault operating modes: misaligned adjuster bolts (left and right), obstacle and missing bearings. These fault modes were reported to be of interest by a rail system operator we collaborated with. Obviously there are many other fault modes of interest at the level of the point machine for example. Such faults are more readily detected due to the rich instrumentation present at the servo-motor.

Misaligned adjuster bolts: In this fault mode the bolts of the adjuster deviate from their nominal positions. As a result, the instant at which the drive rod meets the adjuster (and therefore the instant at which the switch rail starts moving) happens either earlier or later. For example, in a left-to-right motion, if the left bolt moves to the right, the contact happens earlier. The reason is that since the distance between the two bolts decreases, the left bolt reaches the adjuster faster. As a result, when the drive rod reaches its final position, there may be a gap between the right switch blade and the right stock rail. In contrast, if the left bolt moves to the left the contact happens later. The model of the adjuster includes parameters that can set the positions of the bolts, and therefore the effects of this fault mode can be modeled without difficulty. Figures 1 and 2 show a comparison between the nominal behavior and the misaligned left and right bolts, respectively on the motor current and angular velocity.

Missing bearings: To minimize friction, the rails are supported by a set of rolling bearings. When they become stuck or lost, the energy losses due to friction increase. A component connected to the rail was included to account for friction. This component has a parameter that sets the value for the friction coefficient. By increasing the value of this parameter, the effect of the missing bearings fault can be simulated. Figure 3 shows a comparison between the nominal behavior and the missing bearing behavior on the motor current and angular velocity.

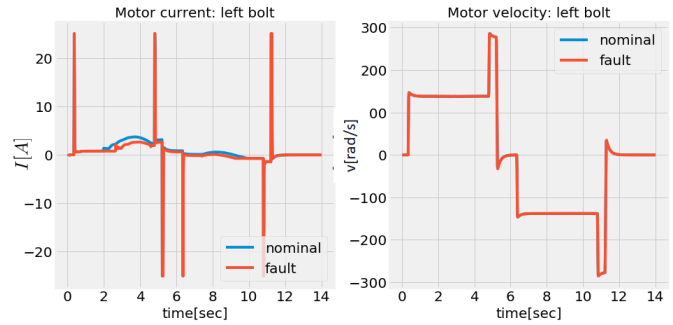

Figure 1. Effects of a misaligned left adjuster bolt on the motor current and angular velocity

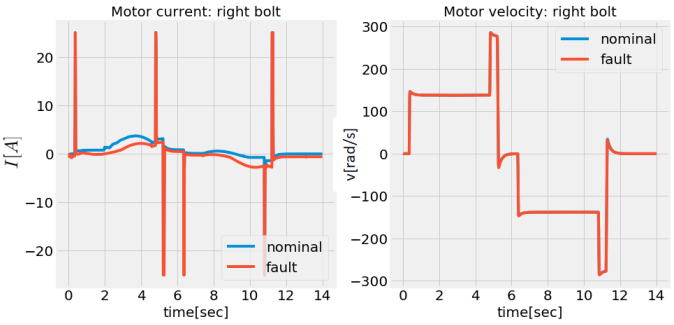

Figure 2. Effects of a misaligned right adjuster bolt on the motor current and angular velocity

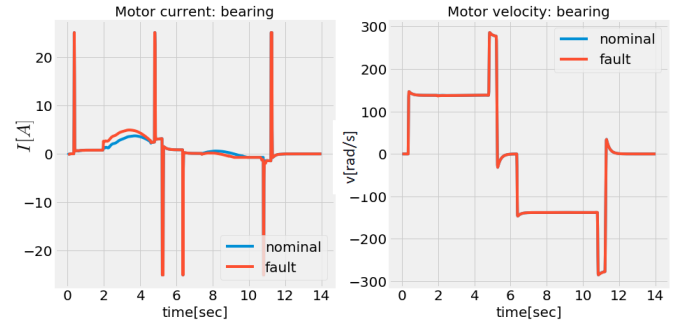

Figure 3. Effects of missing bearings on the motor current and angular velocity

Obstacle: In this fault mode, an obstacle obstructs the motion of the switch blades. In case the obstacle is insurmountable, a gap between the switch blades and the stock rail appears. The effect on the motor torque is a sudden increase in value, as the motor tries to overcome the obstacle. To model this fault we included a component that induces a localized, additional friction phenomenon for the switch blades. This component has two parameters: the severity of the fault and the position. For very high severity the switch blades cannot move beyond a certain position. Figure 4 shows a comparison between the nominal behavior and the obstacle present behavior on the motor current and angular velocity.

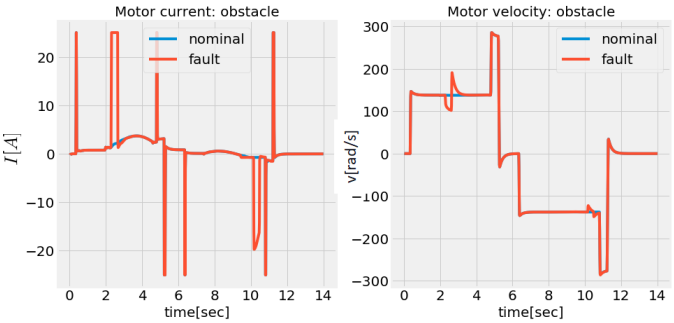

Figure 4. Effects of obstacle presence on the motor current and angular velocity

Remark 4.1 *The behavior of the angular velocity seems to be little affected by the presence of rail switch faults. This observation can be explain by the fact that the angular velocity is the controlled variable. Hence the motor will vary the torque (current) in an attempt to track the angular velocity reference.*

5. ACAUSAL MODELING

Acausal models are physics based models typically constructed from first principles. Unlike the causal models used in signal processing and control, components of acausal models do not have inputs and outputs but ports (connectors) through which energy is exchanged with other components or the environment. This is the modeling formalism used in the Modelica (Fritzson, 2015) language or in Simscape. Ports are characterized by variables whose type determines how they are manipulated when two ports are connected. For example at a connection point, all flow variables sum up to zero (flow conservation), while all non-flow variables are equal. Examples of flow variables include current, force, torque while examples of non-flow variables include potential, velocity, angular velocity. Typically, the product between a flow and a non-flow variable has the physical interpretation of instantaneous power. The acausal modeling formalism is an instance of the more general port-Hamiltonian formalism (van der Schaft & Jeltsema, 2014). The behavior of acausal components is determined by a set of constitutive equations of the form $f(x; w) = 0$, rather than by a causal map (with or without memory). The vector of variables x can include port variables (flow, non-flow) and internal variables (states, algebraic variables), while w is a vector of component parameters.

We use acausal models to give simplified representations of the rail component initially constructed using a finite element approach which typically induces a higher complexity. To learn the parameters of the constitutive equations there are

two main scenarios that can be considered. In the first scenario, we assume that we can directly measure the component variables. This has the advantage that we can in theory perform the model learning in isolation, without considering the entire model. For this approach to work we need to careful choose the model representation to avoid learning trivial models. The second scenario assumes we have only indirect information about the behavior of the model through measurements that do not include the rail component variables. In this case, the learning must include the entire rail-switch model and it is more computationally intense. Since we have access to the high fidelity model and hence we can directly measure every model variable, we consider the first scenario. We use two type of representations for the (acausal) rail component: causal[1] and acausal.

It the causal case we assume that some variables are inputs while other variables are outputs. This assumption is not ad-hoc. It comes from a causal analysis of the entire system model that produces causal relationships between the system variables. This causal analysis is typically performed before simulating a dynamical system represented as a DAE (Casella, 2011). Once the input/output variable assignment is done, we select a representation for the constitutive equations e.g., a neural network (NN), and move to the parameter learning step. Note that instead of assigning the component variables to an input/output category, we can try to learn the component parameters by assuming that all variables are inputs and the output is zero for all inputs. This approach can only work when considering the entire system model, case which introduces a regularization effect that prevents learning a trivial equation such as the constant zero map. Indeed, a zero map playing the role of a constitutive equation can make the system model unsimulatable due to a singular Jacobian of the system DAE.

In the acausal case the constitutive equations emulate physical laws. In what follows, we discuss different options for the constitutive equations that guarantee that the overall system model can be simulated. Since the behavior of the component can be fairly complex, we may need a large set of constitutive equations. To avoid arbitrary choices of constitutive equation maps, we propose using networks of generalized mass spring dampers (gMSD). In such a network, each node is a composition of one generalized mass, spring and damper in a parallel connection, and each link is a composition of one spring and damper. To ensure that the component modeled as a network of gMSDs does not destabilize the overall system model, we impose conditions on the gMSDs that ensure that the model can be simulated. Such a condition is *dissipativity*. A dissipative component cannot generate energy internally. A formal definition of a dissipative component is given in what follows.

Definition 5.1 *Let $E(t) = E(t_0) - \int_{t_0}^{t} p(\tau)d\tau$ be the energy of a physical component, where $p(t)$ is its power. The component is* dissipative *if $E(t) \leq E(t_0)$ for all $t \geq t_0$.*

We propose two types of maps for the gMSD. The first type if based on a polynomial representation as described in the following proposition.

Proposition 5.1 *Consider a component represented as a network of gMSD where the behavior of the masses, springs and dampers are given by:*

$$\begin{aligned} F_m &= \sum_{i=0}^{n} m_{2i+1}\ddot{x}^{2i+1} + \sum_{i=1}^{n} m_{2i}sign(\dot{x})\ddot{x}^{2i}, \\ F_c &= \sum_{i=0}^{n} c_{2i+1}(\dot{x})x^{2i+1} + \sum_{i=1}^{n} c_{2i}sign(\dot{x})x^{2i}, \\ F_d &= \sum_{i=0}^{n} d_{2i+1}\dot{x}^{2i+1} + \sum_{i=1}^{n} d_{2i}sign(\dot{x})\dot{x}^{2i}, \end{aligned}$$

respectively, where the scalars m_i, d_i and c_i are non-negative, and n is the polynomial order. Then the component is dissipative.

An alternative definition for the gMSD is given in the following proposition.

Proposition 5.2 *Consider a component represented as a network of gMSD where the behavior of the masses, springs and dampers are given by:*

$$F_m = m(x, \dot{x}, \ddot{x})\ddot{x},\ F_c = k(x)x,\ F_d = d(x, \dot{x})\dot{x}$$

respectively, where $m(\cdot,\cdot,\cdot)$, $k(\cdot)$ and $d(\cdot,\cdot)$ are non-negative scalar functions. Then the component is dissipative.

Note that we have a lot of freedom with respect to how we can model the functions $m(\cdot,\cdot,\cdot)$, $k(\cdot)$ and $d(\cdot,\cdot)$. We can modeled them for example as NNs, where we make sure that the last layer imposes a non-negative output through a "ReLu" layer or by taking the square of the output of the last linear layer. Since the constitutive equations may contain differential equations, we will need to use learning platforms with ODE solving capabilities (e.g., Pytorch (Paszke & et al., 2017), TensorFlow (Abadi & et al., 2015), DAETools (Nikolić, 2016)), if the state derivatives are not measured.

6. HYBRID RAIL SWITCH MODEL

In this section we introduce several approaches for simplifying the rail switch component model. In addition to model simplification, we will also focus on preserving the physical interpretation of the reduced model, through appropriate choices of constitutive equation maps. We assume that we have access to the variables at the connection point between the adjuster and the rails. In particular, we assume we can directly measure the force F, position x, velocity v and acceleration a. We use the two modeling approaches:
Causal approach: we determine a causal relation between the force, position, velocity an acceleration and use a causal map

[1] We force a causal behavior for the acausal component.

such as a NN to model the relation between them. The resulting component model is still acausal though, with an imposed variable dependence.

Acausal approach: we model the rail component as a combination of generalized mass, spring, dampers as defined in Propositions 5.1 and 5.2. We will show that one mass-spring-damper component is sufficient. The training data is generated by simulating the high fidelity rail switch model. The input signal is the current applied to the servo-motor, which correlated with a desired velocity profile. Typically, predetermined current trajectories are fed to the servo-motor to generate the rail motion. In our case, we will use random inputs to push the rail. We will record the force, position, velocity and acceleration trajectories and use them as training data. Each time series corresponds to a time interval of 100 sec, sampled at 0.1 sec. When appropriate, we use one time series for training or several of them.

6.1. Causal modeling

In this approach, we assign causality relations to the variables at the connection point between the adjuster and the rails. Since the serve-motor tracks a pre-specified speed pattern, our intuition should tell us that the position and velocity of the rails are set by the motor. This intuition is confirmed by a causal analysis performed by looking at the block lower triangular (BLT) transformation (Casella, 2011) that depicts the causal relations between the system variables [2]. Hence, we model the rail behavior by using a causal map $F = g(u; w)$, where $g : \mathbb{R}^3 \rightarrow \mathbb{R}$ is a map described by a NN with one hidden layer $g(u) = W^{[1]}\left(\tanh\left(W^{[0]}u + b^{[0]}\right)\right) + b^{[1]}$, where, the input $u = [x, \dot{x}, \ddot{x}]$ is a vector containing the position, speed and acceleration, the output F is the force, and $w = \{W^{[0]}, b^{[0]}, W^{[1]}, b^{[1]}\}$ is the set of parameters of the map g. We have employed a two steps training process. In the first step we train the parameters of the map in isolation, considering the map g only. We used 15 time series containing trajectories of the force, position, speed and acceleration. We used the Keras (Chollet & et al., 2015) deep-learning training platform, proceeded by splitting the data into training (70%) and test (30%) data sets. We chose the hidden layer dimension to be 50, and trained the NN parameters using a decaying learning rate. The validation results are shown in Figure 5, where we depict the true vs. predicted output samples using as input the test data set. The MSE for the validation data is $MSE_{test} = 415.46$. Although it may appear a large value, it must be interpreted relative to the values of the force used in training and validation, since the training data was not normalized to maintain the physical interpretation. We used the weights of the Keras model to implement a Modelica component with one port and the constitutive equation given by $F = W^{[1]}\left(\tanh\left(W^{[0]}u + b^{[0]}\right)\right) + b^{[1]}$, where $u = [x, \dot{x}, \ddot{x}]$. Next, we executed a fine tuning of the component parameters by performing a parameter learning step using the entire rail switch model. This way, the rest of the model equations are considered, adding an additional regularization effect. We chose a gradient-free optimization algorithm, namely the Powell algorithm, to avoid using gradient approximations. The Modelica rail switch model was converted into a functional mockup unit (FMU) (Blochwitz & et al., 2011), and integrated with the Powell algorithm in Python. Although gradient free algorithms are typically slow for a large number of variables, we did not have to run the algorithm for a large number of iterations since we used the Keras solution as initial parameter values. The result of this additional step was a 20% improvement of the loss function applied to the test data. The newly learned Modelica component has 8 equations.

[2] The BLT transformation is too large to be included in a plot.

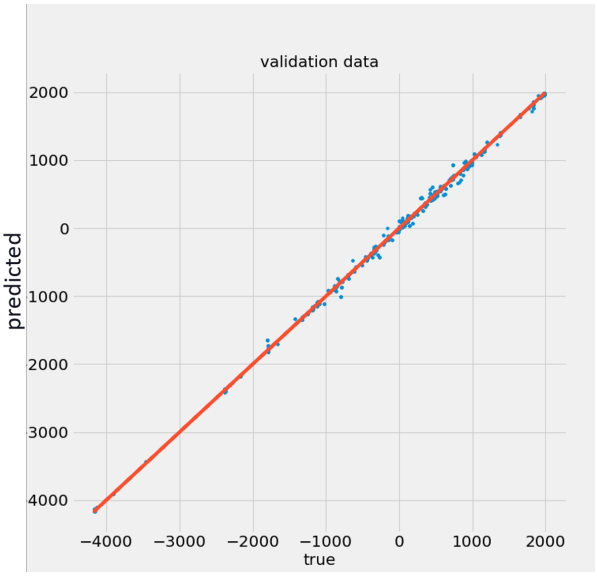

Figure 5. Validation of the learned model

6.2. Acausal modeling

We showed in the previous section how we can use causal maps inside acausal components. The advantage of the causal representation is that we can use main stream deep learning platforms to learn the parameters of the causal map. There is a significant disadvantage though: it is not clear if the obtained component is reusable. By reusability we understand the ability to use the component in different configurations and still behaving as expected. From a numerical perspective view, this means that we should be able to compute the acceleration when the force becomes the input (position and speed are state variables and considered known from the previous simulation step). The acausal modeling approach guarantees this. Using the observation that the rail opposes motion, we modeled the rail as a combination of a generalized mass-spring damper in a parallel connection. We use two types of gMSD models: polynomial and NN. We considered a linear mass model: $F_m = m\ddot{x}$

for both cases. In the polynomial case, we considered the following models for the spring and damper, respectively: $F_c = c_0(x - x_{fix}) + c_1(x - x_{fix})^3 + c_2(x - x_{fix})^5$ and $F_d = d_0\dot{x} + d_1\dot{x}^3 + d_2\dot{x}^5$. The set of parameters we have to learn is $w = \{m, c_0, c_1, c_2, d_0, d_1, d_2, x_{fix}\}$. Unlike to previous section, we considered as input the force, and as outputs the position and velocity. The model parameters are the solution of the following constrained optimization problem:

$$\min_{w \geq 0} \frac{1}{2N} \sum_{i=1}^{N} \|x(t_i) - \hat{x}(t_i)\|^2 + \|\dot{x}(t_i) - \hat{\dot{x}}(t_i)\|^2$$
$$\text{subject to:}$$
$$m\hat{\ddot{x}}(t_i) + F_c(t_i) + F_d(t_i) = F(t_i),$$
$$F_c(t_i) = c_0(x(t_i) - x_{fix}) + c_1(x(t_i) - x_{fix})^3 + c_2(x - x_{fix})^5,$$
$$F_d(t_i) = d_0\dot{x}(t_i) + d_1\dot{x}(t_i))^3 + d_2\dot{x}(t_i)^5,$$
$$w = \{m, c_0, c_1, c_2, d_0, d_1, d_2, x_{fix}\}.$$

where t_i are time samples of the time series. We chose to include only the position and velocity since as states they are automatically generated during the simulation process. The optimization problem used one time series only and used a nonlinear least square algorithm. We relied on the DAETools (Nikolić, 2016) Python package to implement the optimization algorithm since it provides access to the gradients of the cost function, hence gradient approximations are not needed. In particular we used the IDAS solver with default parameters that is endowed with sensitivity analysis. The resulting optimal parameters are as follows: $c_0^* = 6.5 \times 10^3$, $c_1^* = 0.45$, $c_2^* = 4.15 \times 10^4$, $d_0^* = 5.96 \times 10^2$, $d_1^* = 0$, $d_2^* = 0$, $m^* = 1.5 \times 10^2$, $s_0^* = 1.077$. We repeated the learning process when the acausal rail model is represented using NN representations. In particular we chose as models for the spring and damper $F_c = c(x, \dot{x})^2(x - x_{fix})$ and $F_d = d(x, \dot{x})^2\dot{x}$, respectively, where $c(x, \dot{x})$ and $d(x, \dot{x})$ are modeled as a NN with one hidden layer of size 15 and `tanh` as activation function. Using the DAETool, we solved the following optimization problem:

$$\min_{w} \frac{1}{2N} \sum_{i=1}^{N} \|x(t_i) - \hat{x}(t_i)\|^2 + \|\dot{x}(t_i) - \hat{\dot{x}}(t_i)\|^2$$
$$\text{subject to:}$$
$$m\hat{\ddot{x}}(t_i) + F_c(t_i) + F_d(t_i) = F(t_i),$$
$$F_c(t_i) = c(x(t_i), \dot{x}(t_i))^2(x(t_i) - x_{fix}),$$
$$F_d(t_i) = d(x(t_i), \dot{x}(t_i))^2\dot{x}(t_i),$$
$$c(x(t_i), \dot{x}(t_i)) = W_c^{[1]} \tanh(\left(W_c^{[0]}[x(t_i), \dot{x}(t_i)]^T + b_c^{[0]}\right)) + b_c^{[1]},$$
$$d(x(t_i), \dot{x}(t_i)) = W_d^{[1]} \tanh(\left(W_d^{[0]}[x(t_i), \dot{x}(t_i)]^T + b_d^{[0]}\right)) + b_d^{[1]},$$
$$w = \{m, W_c^{[1]}, b_c^{[1]}, W_c^{[0]}, b_c^{[0]}, W_d^{[1]}, b_d^{[1]}, W_d^{[0]}, b_d^{[0]}, x_{fix}\},$$

With the neural network representation we are able to recover a more detailed behavior for the speed. The number of equations of the rail component under the acausal polynomial and NN representations are 7 and 11, respectively.We validated the learned models by integrating them within the overall rail switch model. We generated 25 time series with random inputs for the servo-motor used for the four rail switch models: the high fidelity one, and three low fidelity corresponding to the causal NN, acausal polynomial and acausal NN representations, respectively. An example of such time series that corresponds to the force applied to the rail is shown in Figure 6.

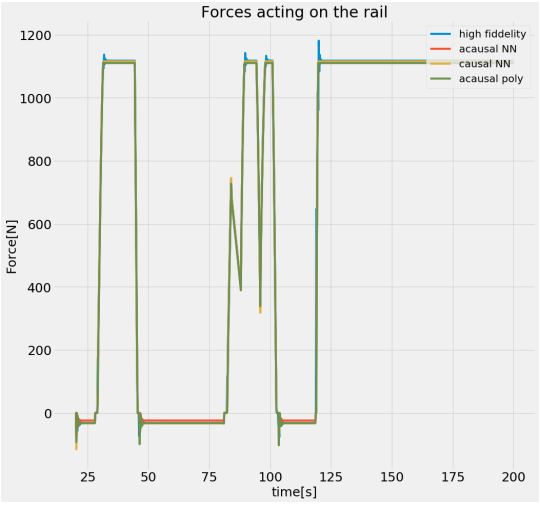

Figure 6. Rail force for the high and low fidelity models

We used the 25 time series to compute MSE statistics for the position, velocity and force. The results are shown in Figures 7, 8 and 9, shown as box plots. A first observation is that the MSEs corresponding to the force have large values as compared to the position and velocity. This should not be a surprise since the absolute values of the force are in the thousands. The position and velocity MSEs are similar for all three cases. In the case of the force, the acausal representations have roughly the same statistics, while in the causal case, the MSE has both the variances and mean comparable, but slightly smaller. This again should not be a surprise since the rail causal model is tailored for our scenario. In other words the model may be overfitted. In a different usage scenario, the casual representation may not even simulate. Hence we have a trade-off between accuracy and generalizability.

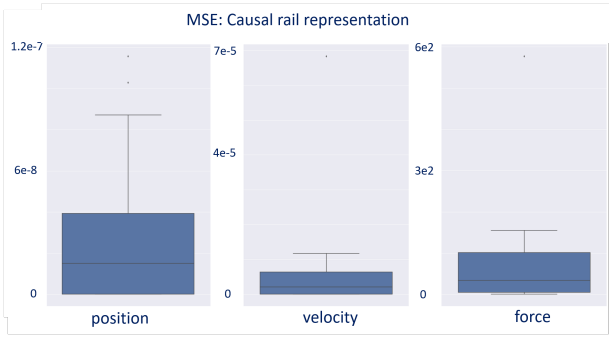

Figure 7. Validation statistical results: causal rail representation

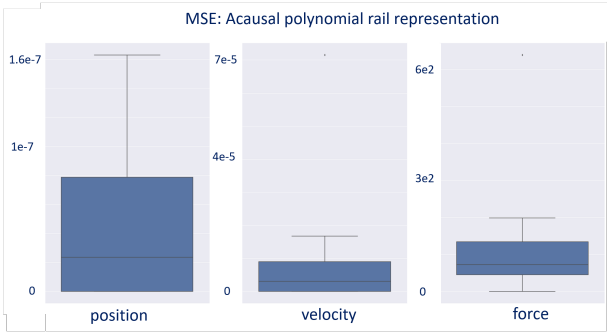

Figure 8. Validation statistical results: acausal polynomial rail representation

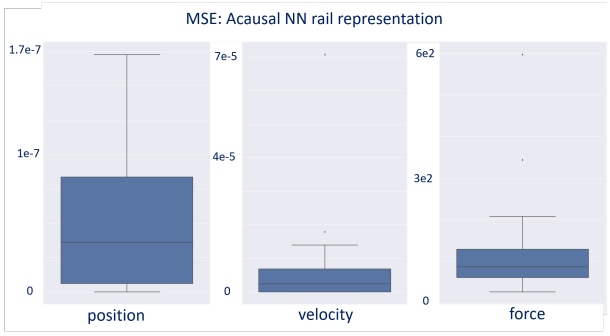

Figure 9. Validation statistical results: acausal NN rail representation

7. FAULT DIAGNOSIS

We use the high-fidelity model as the ground truth, and use the fault component and parameters to generate faulty behavior. *Note that since the faults are not at the level of the rail component, there is no need to train surrogate models for the rail in each fault mode.* For the adjuster bolt misalignment, we consider one at a time, 50 mm and 200 mm to the left and to the right bolt misalignment. Two parameters have been introduced in the adjuster model that allow for bolt misalignment modeling, whose nominal values are zero. In the case of the missing bearing fault, the missing bearing component introduces a viscous type of friction corresponding to a viscous coefficient of $d = 5000\ Ns/m$. If necessary we can model other type of friction models, e.g., Coulomb friction. The component responsible for the simulation of an obstacle has two parameters: the fault intensity and the obstacle location. The fault intensity dictates how much opposing force the obstacle generates against the rail motion induced by the motor. We model the opposing force as a localized viscous force. The localization of the force was achieved by allowing the viscous coefficient to be non-zero only in a neighborhood of the obstacle location. For example d can be modeled as $d = 10^5 \times (\mathbb{1}(x - x_o + \delta) - \mathbb{1}(x - x_o - \delta))$ Ns/m, where x_o is the obstacle location, $\mathbb{1}(x)$ is the step function and δ is a small positive scalar. This means that d is non-zero only inside the interval $[x_o - \delta, x_o + \delta]$. The obstacle position is chosen at 10 cm from the left side initial position of the rail. The effects of the fault modes on the motor current and angular velocity for the chosen parameter were shown in Figures 1-4. The objective of the fault diagnosis is to detect which of the four fault modes is present by tracking the parameters of the fault modes. We consider the single fault scenario, that is only one of the four fault modes is active at some time instant. We can use simultaneous parameter tracking (all parameters of the fault model are tracked) or we can run in parallel tracking algorithms that estimate the parameters of one of the four fault modes only. In our case we would have four parallel algorithm. Based of the parameter deviation from their nominal values we declare the presence of a fault mode.

7.1. Optimization-based parameter estimation

Filtering based techniques either have constraints on the class of models they can work with (e.g., Kalman like filters), or they require significant computational resources (e.g., particle filter). In this work we estimated the fault parameters for each of the four fault modes using an optimization-based parameter estimation algorithm. The loss function was defined as the mean square error (MSE) between the simulated variables and the "observed" variables (motor current, motor angle and angular velocity). The observed variables were generated using the high-fidelity models and contain simulations over a time horizon of 14 sec containing both switch motions: left to right and right to left. The variables are sampled at 0.1 sec. The optimization algorithm requires loss function evaluations that in turn requires model simulations. The model simulations were done using Functional Mockup Units (FMU) (Blochwitz & et al., 2011) representations of the Modelica models. We tested the optimization algorithm for the three versions of the reduced complexity models: causal NN, acausal NN and acausal polynomial representations of the rail. We tested the parameter estimation using several optimization algorithm including gradient-based and gradient-free algorithm. The best results were produced by the the differential evolution algorithm and they are presented in what follows. Since such an algorithm requires many loss function evaluations it is imperative for the model simulations to be fast. In average, the acausal polynomial, acausal NN and causal NN representations take 0.3 sec, 0.5 sec and 0.9 sec, respectively over the 14 sec time horizon. For the same time interval, the high fidelity model takes 7 sec. The FMUs were used in Python scripts implementing the parameter estimation algorithms. The model simulations were executed on PC with a 12 cores Intel Xeon 3.5 GHz CPU, with 64 GB of RAM. We recall that the starting position of the rail is 1 m, value dictated by the initial conditions of the motor and the positions of the different reference points in the rail model.

Left bolt fault: the left bolt fault mode is active with a devia-

tion from its nominal value of 50 mm. Tables 1-3 present the results of the parameter estimation algorithms corresponding to the three representations, when tracking separately the fault parameters. Using as metric the MSE it is clear that we are correctly identifying the left bolt fault as the current fault mode. In addition, the fault parameter values are within 3% of the value used to generate the faulty behavior.

Tracked fault parameters	Paramater value	MSE
Left bolt[mm]	49.53	0.006
Right bolt[mm]	6.03	0.324
Missing bearing[Ns/m]	39.73	0.326
Obstacle{[Ns/m],[m]}	$\{1.5\times10^5, 1.49\}$	0.334

Table 1. Left bolt fault mode (acausal polynomial representation)

Tracked fault parameters	Paramater value	MSE
Left bolt[mm]	48.69	0.011
Right bolt[mm]	11.26	0.352
Missing bearing[Ns/m]	28.45	0.353
Obstacle{[Ns/m],[m]}	$\{1.67\times10^5, 10.04\}$	0.386

Table 2. Left bolt fault mode (acausal NN representation)

Tracked fault parameters	Paramater value	MSE
Left bolt[mm]	50.42	0.005
Right bolt[mm]	8.91	0.344
Missing bearing[Ns/m]	33.67	0.304
Obstacle{[Ns/m],[m]}	$\{6.31\times10^4, 0.0772\}$	0.341

Table 3. Left bolt fault mode (causal NN representation)

We estimated simultaneously all the fault parameters as well. The results for the three representations of the rail model are shown in Table 4. We obtained reasonable small MSE values, but it is more challenging to distinguish between the faults modes. Recalling that the obstacle was introduced at 1.1 m we can exclude the obstacle fault mode (the fault intensity is irrelevant outside the obstacle position). The parameter corresponding to the missing bearing fault mode has a value in the hundreds for two of the rail representations. Although they may appear not to have a significant impact on the behavior of the rail switch, without some prior information about what is a significant value it is difficult to draw a conclusion about this fault mode. The good news is that the left bolt fault parameter was reasonably well estimated. Although not zero, the right bolt fault parameter values are small enough to eliminate this fault mode as a possible source of faulty behavior.

Right bolt fault: the bolt fault mode is active with 200 mm deviation from its nominal value. Tables 5-7 present the results of the parameter estimation algorithms corresponding to the three representations, when tracking separately the fault parameters. The MSE values show that we can indeed identify the correct fault mode. Moreover, the fault parameter values are within 6% of the value used to generate the faulty behavior.

Tracked fault parameters	Acausal poly	Acausal NN	Causal NN
Left bolt[mm]	49.69	50.097	48.22
Right bolt[mm]	0.151	1.624	0.394
Missing bearing[Ns/m]	244.37	1.624×10^2	6.186×10^2
Obstacle{[Ns/m], [m]}	$\{7.58\times10^4$, 1.769$\}$	$\{1.297\times10^5$, 1.7314$\}$	$\{1.381\times10^5$, 1.223$\}$
MSE	0.004	0.005	0.012

Table 4. Left bolt fault mode: simultaneous parameter estimation

Tracked fault parameters	Paramater value	MSE
Left bolt[mm]	72.75	1.025
Right bolt[mm]	197.35	0.029
Missing bearing[Ns/m]	35.71	1.767
Obstacle{[Ns/m],[m]}	$\{1.11\times10^5, 0.191\}$	1.786

Table 5. Right bolt fault mode (acausal polynomial representation)

Tracked fault parameters	Paramater value	MSE
Left bolt[mm]	71.84	1.03
Right bolt[mm]	187.66	0.091
Missing bearing[Ns/m]	49.12	1.818
Obstacle{[Ns/m],[m]}	$\{6.12\times10^4, 1.04\}$	1.855

Table 6. Right bolt fault mode (acausal NN representation)

Tracked fault parameters	Paramater value	MSE
Left bolt[mm]	73.97	1.022
Right bolt[mm]	198.42	0.011
Missing bearing[Ns/m]	20.20	1.792
Obstacle{[Ns/m],[m]}	$\{1.05\times10^4, 0.977\}$	1.792

Table 7. Right bolt fault mode (causal NN representation)

Bearing fault: the bearing fault mode is active with the viscous coefficient taking the value 5000 Ns/m. Tables 8-10 present the results of the parameter estimation algorithms, when tracking separately the fault parameters. The smaller MSE values correspond to the bearing fault mode. We note the parameter estimation error variance is roughly 3%.

Tracked fault parameters	Paramater value	MSE
Left bolt[mm]	0.04	0.412
Right bolt[mm]	4.40	0.3869
Missing bearing[Ns/m]	5060.706	0.03
Obstacle{[Ns/m],[m]}	$\{3.5\times10^3, 1.37\}$	0.367

Table 8. Bearing fault mode (acausal polynomial representation)

Tracked fault parameters	Paramater value	MSE
Left bolt[mm]	0.06	0.377
Right bolt[mm]	16.42	0.365
Missing bearing[Ns/m]	4919.18	0.00744
Obstacle{[Ns/m],[m]}	$\{1.83\times10^5, 1.04\}$	0.404

Table 9. Bearing fault mode (acausal NN representation)

Tracked fault parameters	Paramater value	MSE
Left bolt[mm]	0.126	0.377
Right bolt[mm]	5.25	0.361
Missing bearing[Ns/m]	4845.50	0.0032
Obstacle{[Ns/m],[m]}	$\{1.84\times10^5, 1.04\}$	0.378

Table 10. Bearing fault mode (causal NN representation)

Obstacle fault: we simulated the high fidelity model with an obstacle at 1.1 m and a viscous coefficient with the value 10^5 Ns/m. The parameter estimation results when tracking the fault parameter separately are shown in Tables 11-13. The smallest MSE values were obtained for the correct fault parameters. In addition, the maximum estimation error for the fault intensity and fault location parameters is 0.2% and 0.09% of the nominal values, respectively.

Tracked fault parameters	Paramater value	MSE
Left bolt[mm]	3.17	69.618
Right bolt[mm]	49.57	69.20
Missing bearing[Ns/m]	5915.78	67.67
Obstacle{[Ns/m],[m]}	$\{1.01\times10^5, 1.099\}$	0.020

Table 11. Obstacle fault mode (acausal polynomial representation)

Tracked fault parameters	Paramater value	MSE
Left bolt[mm]	0.178	69.40
Right bolt[mm]	44.477	69.054
Missing bearing[Ns/m]	5870.32	67.62
Obstacle{[Ns/m],[m]}	$\{9.98\times10^4, 1.099\}$	0.047

Table 12. Obstacle fault mode (acausal NN representation)

Tracked fault parameters	Paramater value	MSE
Left bolt[mm]	0.679	69.384
Right bolt[mm]	54.353	69.071
Missing bearing[Ns/m]	5867.27	67.560
Obstacle{[Ns/m],[m]}	$\{9.98\times10^4, 1.099\}$	0.0122

Table 13. Obstacle fault mode (causal NN representation)

8. CONCLUSIONS

We proposed a hybrid modeling approach to simplify a high fidelity model of a rail-switch system. In particular, we used simplified representations for the rail component using machine learning inspired models. The representations preserved the physical interpretation of the rail component. The model complexity of the model abstractions (i.e., number of equations) is reduced by two orders of magnitude. A similar reduction in the order of magnitude is obtained with respect to the simulation time of the rail switch model over a full motion cycle of the rail. The new model abstractions were used for the rail fault diagnosis. The rail switch model was augmented with additional behavior to include parameterized fault modes. An optimization based approach was used to estimate the fault parameters. We demonstrated that using algorithms that track separately the fault parameters of each of the four fault modes produce accurate diagnosis results. The MSEs and the parameter values are used by the diagnosis engine to produce a diagnosis solution.

ACKNOWLEDGMENT

This material is based upon work supported in part by the Defense Advanced Research Projects Agency (DARPA) Award HR00111890037 Physics of AI (PAI) Program.

REFERENCES

Abadi, M., & et al. (2015). *TensorFlow: Large-scale machine learning on heterogeneous systems.* Retrieved from `https://www.tensorflow.org/` (Software available from tensorflow.org)

Arulampalam, M. S., Maskell, S., & Gordon, N. (2002). A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking. *IEEE TRANSACTIONS ON SIGNAL PROCESSING*, *50*, 174–188.

Blochwitz, T., & et al. (2011). The functional mockup interface for tool independent exchange of simulation models. In *In proceedings of the 8th international modelica conference.*

Casella, F. (2011, March). *Introduction to the Dynamic Modelling of Thermo-Fluid Systems using Modelica - mini course.* Dipartimento di Elettronica e Informazione, Politecnico di Milano. Retrieved from "`http://staff.polito.it/roberto.zanino/sub1/teach_files/modelica_minicourse/index_modelica.html`"

Chollet, F., & et al. (2015). *Keras.* `https://keras.io`.

de Kleer, J., Mackworth, A., & Reiter, R. (1992). Characterizing diagnoses and systems. *"Journal of Artificial Inteligence"*, *56*(2–3), 197–222.

Fritzson, P. (2015). *Principles of object-oriented modeling and simulation with modelica 3.3: A cyber-physical approach* (2nd ed.). Hoboken, NJ: Wiley.

Gertler, J. (1998). *Fault-detection and diagnosis in engineering systems*. New York: Marcel Dekker.

Honda, T., & et al. (2014, August). A simulation and modeling based reliability requirements assessment methodolog. In *Proceedings of international design engineering technical conferences and computers and information in engineering conference (asme 2014)* (Vol. 7).

Isermann, R. (2005). Model-based fault-detection and diagnosis - status and applications. *Annual Reviews in Control*, *29*(1), 71 - 85.

Kalman, R. (1960). A new approach to linear filtering and prediction problems. *Transactions of the ASME–Journal of Basic Engineering*, *82*(Series D), 35–45.

Minhas, R., de Kleer, J., Matei, I., Saha, B., Janssen, B., Bo-

brow, D., & Kurtoglu, T. (2014). Using fault augmented modelica models for diagnostics. In *Proceedings of the 10th international modelicaconference* (pp. 437–445).

Nikolić, D. D. (2016, April). Dae tools: equation-based object-oriented modelling, simulation and optimisation software. *PeerJ Computer Science*, *2*, e54. Retrieved from https://doi.org/10.7717/peerj-cs.54 doi: 10.7717/peerj-cs.54

Paszke, A., & et al. (2017). Automatic differentiation in PyTorch.

Patton, R. J., Frank, P. M., & Clark, R. N. (2000). *Issues of fault diagnosis for dynamic systems*. Springer-Verlag London.

Saha, B., & et al. (2014, August,). Model-based approach for optimal maintenance strategy. In *Proceedings of second european conference of the prognostics and health management society.*

van der Schaft, A., & Jeltsema, D. (2014). Port-hamiltonian systems theory: An introductory overview. *Foundations and Trends in Systems and Control*, *1*(2-3), 173-378. Retrieved from http://dx.doi.org/10.1561/2600000002 doi: 10.1561/2600000002