A Hybrid Qualitative and Quantitative Diagnosis Approach

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

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

ABSTRACT

Parametric faults are detected and isolated using parameter tracking algorithms based on optimization algorithms or filtering techniques (e.g., Kalman filter, particle filter). Online, simultaneous tracking of all parametric faults can fails since there may be too many combinations of parameter values that explain the observed behavior. Hence, a correct diagnosis solution is not obtained. An alternative in the single fault case is to track separately each parametric fault in parallel and choose the one that best explains the observed behavior according to some chosen metric (e.g., mean square error). This approach is feasible but computationally expensive, since there may be too many tracking algorithms running in parallel. Analytic redundancy relations (ARRs) are used to reduce the number of parametric faults that are tracked simultaneously. ARRs qualitatively point to a set of possible explanations but usually require a large number of sensors to achieve good isolability of fault causes. They induce a fault signature matrix (FSM) that can be derived offline. The parameter tracking algorithms will be instantiated for the faults in the set of possible explanations produced by the ARRs. By combining ARRs with online parameter tracking algorithms we can obtain a good tradeoff between computational effort and fault isolability. The approach is validated by diagnosing faults in a rectifier circuit.

1. INTRODUCTION

Our goal is to design a model-based diagnosis engine for detecting and isolating parametric faults in a dynamical system. The diagnosis engine is provided with a model of the system, nominal values of the parameters and values of some of its inputs and outputs over time. 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. Data-driven approaches are widely used as well. They use statistical models (classifiers or regressors) trained with labeled data that describe the behavior of the system under various fault modes. They usually require a large amount of data to build statistically meaningful models. This requirement is not always easy to satisfy, since system are designed not to fail. In addition, we may lose explainability of the root causes.

In this paper we focus on parametric faults. A significant change in a parameter value as compared to its nominal value indicates a fault in the component associated with the respective parameter. There are several approaches for parameter tracking. Filter based approaches such as the Kalman filter (Kalman, 1960) and its extensions to non-linear systems (extended and unscented Kalman filters) are an option. They do not always perform well on nonlinear systems and non-Gaussian noise. Alternatively, we can use a particle filter algorithm (Arulampalam, Maskell, & Gordon, 2002). The computational effort required to implement such a filter may be prohibitive though.

Filters track the state of a dynamical system and hence the state needs to be augmented by including the system parameters. Tracking the state and all parameters simultaneously does not guarantee accurate results since a multitude of parameter value combinations may explain the system behavior. Another option for tracking the system parameters is to use an optimization based approach. Time series reflecting the system behavior are compared with the simulated behavior, and the model parameters are adjusted until the real and simulated behaviors match. Typically, the parameters are adjusted as part of solving a nonlinear least square problem.

Tracking multiple parameters simultaneously faces the same challenges as in the case of the filtering methods. Since the least square cost function typically depends nonlinearly on the system parameters, the more parameters we have the more local minima exist. Hence, unless we use global optimization

algorithms, which are slower, we cannot guaranteed convergence to the global minimum.

When dealing with single fault cases, each parameter is tracked using an instantiation of an optimization algorithm. For large number of parameters this may prove computationally costly. Therefore we need a procedure to select a subset of these parameters. We propose using analytic redundancy relations (ARRs) to implement this procedure.

ARRs are constraints that when not satisfied indicate the presence of a fault. They induce a fault signature matrix (FSM) that shows which parameters are sensitive to the variables in the ARRs. It encodes signatures in the observations associated with the faults. We cannot guarantee that the signatures are all different. That is, several faults may have the same signatures.

The FSM structure depends on the number of sensors and what variables are actually measured. We use the fault signatures to derive a set of possible faults, and use optimization (or filtering) methods to estimate the parameters associated to a particular fault signature in parallel. This way we limit the computational effort by tracking only a subset of parameters. In this paper we track the parameter by applying optimization algorithms. The approach can be easily adapted to work with filter-based methods.

Paper structure: We start with a description of the problem in Section 2. In Section 3 we introduce background material on ARRs and optimization-based parameter tracking. In the same section we discuss how the two approaches can be combined to improve the computational efficiency of the parameter tracking procedure. In Section 4 we demonstrate our approach when diagnosing faults in a rectifier.

2. PROBLEM SETUP

The objective is to detect and isolate faults in dynamical systems expressed as a set differential algebraic equations (DAEs) of the form:

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

where x is the state vector, u is the input for the system, y is the vector of output measurements and θ is a vector of system parameters. Often, we can separate the DAE into an ODE and a set of algebraic equations of the form

$$\dot{x} = f(x, z, u; \theta), \quad (3)$$
$$0 = g(x, z, u; \theta), \quad (4)$$
$$y = h(x, z, u; \theta), \quad (5)$$

where we have a new vector z called vector of algebraic variables. This is the type of DAE we will use in the example section. In addition, the sensing model is based on measuring a subset of the state and algebraic variables x and z. Time series of the input and output vectors are periodically produced and used by a diagnosis engine to detect and isolate a fault, if present. Such time series are of the form $u_{T_1:T_2} = \{u_{t_k}\}_{k=N_1}^{N_2}$, $y_{T_1:T_2} = \{y_{t_k}\}_{k=N_1}^{N_2}$, where $N_1 = \lfloor k = T_1/h \rfloor$, $N_2 = \lfloor k = T_2/h \rfloor$, $t_k = kh$, with h the sampling period.

The diagnostic engine processes the time series and produces a diagnosis result reflecting what parameter has drifted and by how much. The engine processes the time series all at once unlike the filter based implementations that can process data one sample at a time. The length of the time series should sufficiently large to ensure convergence of the optimization algorithm. This is equivalent to ensuring the invertability of the Hessian of the cost function at a local minimizer.

For linear minimum-square problem, convergence can be tested by computing the rank of a particular matrix. For nonlinear cases, it is rather challenging to specify when we have enough information since we would actually need to know the local minima. We are left with choosing "long enough" time series and making them longer in case convergence is not achieved.

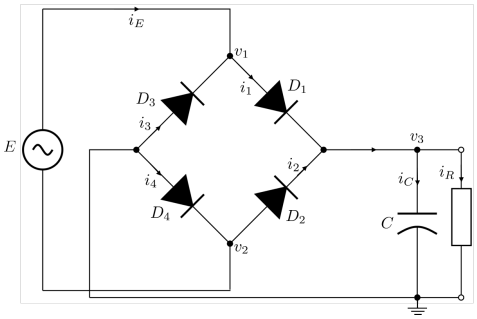


Figure 1. Rectifier circuit

Focusing on parametric faults only is not as constraining as it may seem. In our previous work we showed how we can introduce a variety of parameterized fault models based on the physics of failure. Under this approach the initial nominal model is augmented with fault modes, each mode depending on one or more parameters. The type of faults introduced are domain dependent. We 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.

Fault augmented models enabled us to execute a number of system analytics tasks, ranging from fault diagnosis (Minhas et al., 2014), reliability analysis (Honda et al., 2014) to maintenance scheduling (Saha et al., 2014). Hence parametric

faults cover a significant number of possible faults that can affect a system.

The example on which we test our approach is shown in Figure 1. It is a rectifier circuit powered by an AC source, with a smoothing capacitor and a resistive load. The diodes' behavior follow the ideal nonlinear characteristics

$$i = i_S \left(e^{\frac{v}{n v_T}} - 1 \right), \tag{6}$$

where i_s is the reverse bias saturation current, v_T is the thermal voltage and n is the ideality factor. As we will see in the example section, the behavior of the rectifier is described by a DAE rather than an ordinary differential equation.

We are interested in detecting faults in the diodes represented as open connections, and drifts in the parameters C and R.

3. ARRS AND OPTIMIZATION BASED APPROACH FOR PARAMETER TRACKING

In this section we briefly describe how ARRs are used for fault diagnosis and how optimization algorithms are used to track parameters of dynamical systems. In the last part of this section we combine the two approaches. More details on the use of ARRs for fault diagnosis can be found in (Staroswiecki & Comtet-Varga, 2001), (Staroswiecki, 2000), (Samantaray & Bouamama, 2008).

3.1. Analytic redundancy relations

The mathematical representation of the system model is a set of differential equations that are used to produce ARRs. The ARRs represent various constraints among a set of known process variables (Staroswiecki & Comtet-Varga, 2001). To account for the modeling uncertainties and noise the thresholds δ_i need to be adaptive. Typically, function Φ is of the form $\Phi(r) = |r|$. This defines a basic change detection scheme. A smoothing of the change detection scheme be applied by using a moving average scheme to compute $\Phi(r)$, that is $\Phi(r(t)) = \frac{1}{N}\sum_{i=0}^{N-1} |r(t-i)|$, were the time is assumed to be discrete. Other change detection schemes can be employed, such as the cumulative sum control chart (CUSUM) test. We can define the cumulative sum quantities $S_i^+(t) = \max\{0, S_i^+(t-1) + r_i(t-1) - d\}$ and $S_i^-(t) = \max\{0, S_i^- + r_i(t-1) + d\}$, where d is the size of the shift to be detected. They can be used in a CUSUM based coherence vector definition:

$$c_i(t) = \begin{cases} 1, \ S_i^+(t) > \delta_i \text{or } S_i^-(t) < -\delta_i \\ 0, \ \text{otherwise} \end{cases} \tag{7}$$

A fault is detected if $c \neq [0, 0, \ldots, 0]$, that is, at least one element in the coherence vector is non-zero. A fault is isolated by matching the coherence vector to a binary fault signature matrix (FSM). The FSM denoted by S embeds the structural sensitivity of each residual to the faults in the system components (Staroswiecki, 2000). The entries of the FSM are defined as

$$S_{ji} = \begin{cases} 1, \ i^{th} \text{ residual is sensitive to } j^{th} \text{ component} \\ 0, \ \text{otherwise.} \end{cases} \tag{8}$$

A fault in a component is detectable if at least one residual is sensitive to it. Fault isolation is performed using the FSM. Note that if there are too few sensors, the resulting ARRs may not be able to differentiate among some faults. The ideal case is when each residual is sensitive to only one fault. This type of residuals are called *structured* residuals. If we try to monitor more faults than the number of sensors, we cannot construct structured residuals. A significant of ARRs is the fact that we do not need to explicitly include the fault models. A fault appears when a significant change in the variables and parameters of a component takes place. Therefore, a residual is affected by a component fault if the variables contributing to the computation of the residual are sensitive to the respective fault,

ARRs can be determined using a structured approach, where a bipartite graph describes the dependence between the variables and the constraints that define the system behavior (Staroswiecki & Comtet-Varga, 2001). If a variable appears in a constraint then an edge in the graph exists between them. The structural analysis is done through matching on the bipartite graph: system variables are associated with the constraints from which they can be computed. ARRs are generated from over-constrained subsystems and are completely expressed when only known variables appear (inputs, outputs and parameters) (Staroswiecki, 2000).

The elimination of unknown variables is done iteratively. At the first iteration, we find the variables that can be computed directly from the current set of initial set of known variables. These variables are added to the known variables set which are used to express additional unknown variables at the next iteration. The algorithm stops when all variables are expressed in terms of inputs, outputs and parameters. The constraints that were unused in this process become the ARRs. We emphasize that this determines a structural dependency between known and unknown variables. This dependency may be broken for particular choices of system parameters. More details on the ARR-based diagnosis and other model-based diagnosis techniques can be found in (Samantaray & Bouamama, 2008). In Section 4 we show how we can construct ARRs for the rectifier circuit.

3.2. Optimization-based parameter tracking

The goal is to estimate the parameters of the system using field data. The field data is represented by input and out-

put time series $u_{T_1:T_2}$ and $y_{T_1:T_2}$. The output time series are used to construct the least square cost function $J(\theta) = \frac{1}{N_2-N_1}\sum_{k=N_1}^{N_2}\|y_k - \hat{y}_k(\theta)\|^2$, where $\hat{y}_k(\theta)$ is the simulated sampled output having as input the time series $u_{T_1:T_2}$. The parameter estimates are computed as solution of the nonlinear least-square problem

$$\begin{aligned} \min_{\theta} \quad & \boldsymbol{J}(\theta) \\ \text{subject to:} \quad & \theta \in \Theta \end{aligned} \tag{9}$$

where Θ is a constraint set. The constraint set ensures that the system parameters take values that make physical sense. For example, the resistance of a electrical resistor must be positive. For box type constraints, that is $\theta_i^{min} \leq \theta_i \leq \theta_i^{max}$, where θ_i is an entry of θ, we can transform the constrained optimization problem (9) into an unconstrained one by applying the transformation $\theta_i = \theta_i^{min} + \left[\sin(\tilde{\theta}_i) + 1\right]\frac{\theta_i^{max}-\theta_i^{min}}{2}$ (James & Roos, 1975), and minimizing $\boldsymbol{J}(\tilde{\theta})$. Similar transformations can be used when the parameters are only partially bounded. Namely for $-\infty < \theta_i \leq \theta_i^{max}$ we have $\theta_i = \theta_i^{max} + 1 - \sqrt{\tilde{\theta}_i^2 + 1}$ and for $\theta_i^{min} < \theta_i \leq \infty$ we use $\theta_i = \theta_i^{min} - 1 + \sqrt{\tilde{\theta}_i^2 + 1}$. This is useful since typically, unconstrained optimization problems are simpler to solve, and there are more optimization algorithms available for such problems.

We can use both gradient-free and gradient based optimization methods to solve the unconstrained optimization problem. Both methods rely on model simulations. In the case of gradient-based methods, in the best case scenario we can symbolically obtain the gradient vector. When this is not the case possible options include automatic differentiation or numerical approximations.

We usually prefer to avoid approximating the gradient numerically, since the process introduces numerical errors that tend to accumulate. One avenue for automatic differentiation is using the deep-learning platform such as Tensorflow (Abadi et al., 2015) or Pytorch (Subramanian, 2018). This will require building customized objects that described the behavior of the system. If the system is dynamic, solving the optimization problem requires solving an ODE (or a DAE). Tensorflow already includes a numerical solver for ODEs. Alternatively, we can implement explicit numerical solvers such as the Runge-Kutta algorithms, the simplest of such solvers being the Euler first order approximation. Implementing such solvers is equivalent with designing a "custom" recurrent neural network (RNN), where the number of identical cells is given by the length of the input and output time series. Reasonable choices for gradient-based optimization methods are part of the class of nonlinear least square algorithms such as trusted-region-reflective or Levenberg-Marquardt (Kanzow, Yamashita, & Fukushima, 2004), which take advantage of the cost function's structure.

For large scale optimization algorithms, first order methods such the gradient decent algorithms and its extensions are the state of the art. Gradient-free methods include Nelder-Mead (Powell, 1973), pattern search (Hooke & Jeeves, 1961), or Powell (Powell, 1964) algorithm with its different versions (Powell, 1964). The optimization problem (9) is usually nonlinear and non-convex. Hence there are no guarantees that the optimization algorithm converges to the global minimum. There is the option of using global optimization algorithms (e.g., particle swarm optimization, (Kennedy & Eberhart, 1995), genetic algorithms, (Mitchell, 1998), evolution strategy) (Beyer & Schwefel, 2002)) but they are usually too slow. The optimization algorithm uses the model to generate the simulated output. If the model is dynamic, it requires the initial state for simulation. If the initial state is not available, we can treat its entries as optimization variables and estimate both parameter and initial conditions.

Note that estimating the parameters using an optimization based procedure is one option among many. Alternatively, we can use filtering based techniques based on the extended Kalman filter or particle filter (Arulampalam et al., 2002).

3.3. ARRs enabled parameter tracking

In the previous subsections we discussed how ARRs are used for diagnosis and how optimization algorithms can be used for parameter tracking. The ARRs approach gives an qualitative diagnosis, that is, it provides a list a parameters that are the cause of the abnormal behavior. Parameter tracking provides quantitative values for the parameters, which gives an idea about the severity of the fault. The isolability property of the ARR approach depends on what variables can be measured and how many. We have at most 2^m signatures with m being the number of sensors. Hence with two sensors we can encode in the best case scenario at most four modes, nominal mode included. Even if a signature corresponds to a list of parameters, rather than a single one, this list is typically a subset of the set of all parameters. Hence we need to track fewer parameters. In other words, the ARRs based diagnosis is used to filter out parameters that are unlikely to be responsible for the abnormal behavior.

Figure 2 depicts the architecture behind the combination of ARRs and optimization based parameter tracking. First the input and output time series are used to evaluate the ARRs and generate a coherence vector c. This vector is checked against the FSM which produces a list of possible parameters $\{\theta_{i_1}, \ldots, \theta_{i_M}\}$ responsible for the abnormal behavior. Next, for each parameter θ_{i_j}, an optimization procedure is instantiated which estimates the parameter value $\hat{\theta}_{i_j}$ that best explains the observed behavior. Finally, the likely parame-

ter responsible for the faulty behavior is selected by choosing the one for which the cost function is the smallest, that is, $\hat{\theta}_{i^*}$, where $i^* = \arg\min_{\theta \in \{\theta_{i_1},\ldots,\theta_{i_M}\}} J(\theta)$.

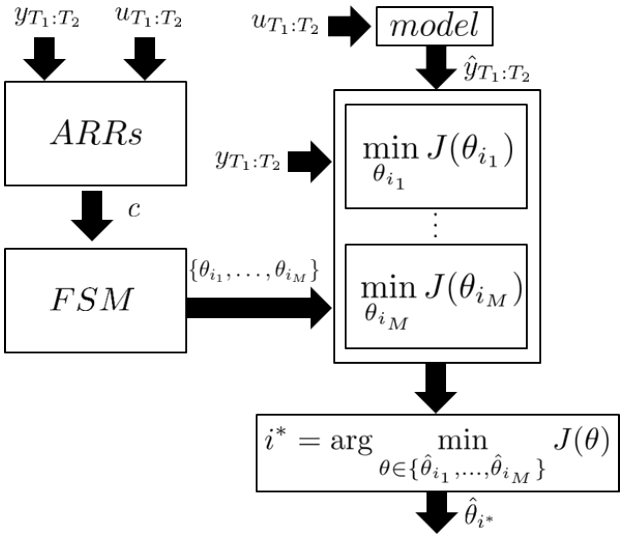


Figure 2. ARRs enabled parameter tracking architecture

The optimization blocks in Figure 2 can be replaced with filters that track each parameter θ_{i_j}, if filtering based techniques for parameter tracking are used. Since filters require the statistics of the initial state, we would need to track the state of the system until an abnormality is detected, and the statistics of state estimate at the respective point will be used as initial state for the bank of filters.

4. ILLUSTRATIVE EXAMPLE

In this section we demonstrate the ARRs enabled parameter tracking approach for parametric fault diagnosis. The schematic of the rectifier was introduced in Figure 1. The input for the rectifier is an sinusoid signal $E = E_a \sin(2\pi f t)$. The nominal parameter values of the rectifier's components are shown in Table 1.

Table 1. Rectifier parameters

Notation	Description	Value
E_a	amplitude	8.91 [V]
f	frequency	60 [Hz]
R	resistor	100 [Ohm]
C	capacitor	2.2e-6[F]
i_S	saturation current	1e-12[A]
v_T	thermal voltage	26e-3[V]
n	ideality factor	1.5

We consider two types of faults: open connections in the diodes and parameter drifts for the capacitor and resistor. The open connections are simulated by making the saturation current i_S zero.

We first construct a set of ARRs and the FSM. We assume we measure two variables: potentials v_2 and v_3. The behavior of the rectifier is described by the following DAE system:

$$v_1 = E + v_2 \tag{10}$$
$$i_3 = i_{S_3}(e^{-\frac{v_1}{nv_T}} - 1) \tag{11}$$
$$i_1 = i_{S_1}(e^{\frac{v_1 - v_3}{nv_T}} - 1) \tag{12}$$
$$i_E = -i_1 + i_3 \tag{13}$$
$$i_4 = i_{S_4}(e^{-\frac{v_2}{nv_T}} - 1) \tag{14}$$
$$i_2 = i_{S_2}(e^{\frac{v_2 - v_3}{nv_T}} - 1) \tag{15}$$
$$i_2 = i_4 + i_E \tag{16}$$
$$Ri_R = v_3 \tag{17}$$
$$i_C = i_1 + i_2 - i_R \tag{18}$$
$$C\dot{v}_3 = i_C \tag{19}$$

To solve the above DAE, it is needed to solve, at each time-step, the optimization problem

$$\min |i_2 - i_4 - i_E| \tag{20}$$

where (20) is subject to the following constraints:

$$v_1 - E - v_2 = 0 \tag{21}$$
$$i_3 - i_{S_3}(e^{-\frac{v_1}{nv_T}} - 1) = 0 \tag{22}$$
$$i_1 - i_{S_1}(e^{\frac{v_1 - v_3}{nv_T}} - 1) = 0 \tag{23}$$
$$i_E + i_1 - i_3 = 0 \tag{24}$$
$$i_4 - i_{S_4}(e^{-\frac{v_2}{nv_T}} - 1) = 0 \tag{25}$$
$$i_2 - i_{S_2}(e^{\frac{v_2 - v_3}{nv_T}} - 1) = 0 \tag{26}$$

This problem can in fact be simplified to an iterative Newton-Raphson algorithm where the iteration variable is the potential v_2. Once v_2 is solved, the rest of the unknown variables can be computed.

To find the ARRs we apply the structured approach. Since we have two sensors we can generate two ARRs. Note that there is no unique set of two ARRs. In fact, any two equations from the constraints (10) – (19) can serve as ARRs. However, they will induce the same fault signature matrix. The structured approach uses a greedy procedure to compute the unknown variables. In the case of derivatives, explicit equations are added. For example for $\dot{v}_3$ we add the equation

$$\dot{v}_3 = \frac{dv_3}{dt} \tag{27}$$

This is a causal equation though, meaning that we can compute $\dot{v}_3$ from v_3 but not the other way around. The "computation" is in fact an approximation since we have samples of v_3 in time. Hence we have 11 equations and 9 unknowns. From the measurement variables v_2 and v_3, in the initial step we can find solutions for the unknowns v_1, i_2, i_R, i_4 and $\dot{v}_3$ from constraints 10, 15, 17, 14 and 27, respectively. In the second step, we can derive the unknowns i_C, i_1 and i_3 from the constraints 19, 12 and 11, respectively. Finally, in the third step we can compute i_E from constraint 13.

We have solved for all the unknowns and we are left with the constraints 16 and 18. They are the ARRs for our example, resulting in the residuals $r_1 = i_2 - i_4 - i_E$ and $r_2 = i_C - i_1 - i_2 + i_R$. The next step is to construct the FSM. This involves tracking the dependence of each variable in the constraints 16 and 18 on the parameters or the components of the system. In particular, we will track the dependance of the variables on the four diodes and on the parameters C and R. For example, i_2 depends only on the state of diode D_2 while i_E depends on the state of diodes D_1 and D_3. We repeat the dependency analysis for all variables involved in the ARRs, resulting in the FSM depicted in Table 2.

Table 2. Fault signature matrix

	r_1	r_2
D_1	1	1
D_2	1	1
D_3	1	0
D_4	1	0
R	0	1
C	0	1

We discover that we can differentiate between three pairs of components $\{D_1, D_2\}$, $\{D_3, D_4\}$ and $\{R, C\}$.

Figures 3a-3b depict the residuals r_1 and r_2 over a time interval of 0.1s, and their average values, in the nominal model. As we expect, their values are close to zero.

Figures 3c and 3d depict the residuals r_1 and r_2 over a time interval of 0.1s, and their average values in the case diode D_1 suffers from an open connection fault. The simulated residual values are unrealistically large, but it has the correct signature.

The detection thresholds can be chosen based on the desired sensitivity of the ARRs. For example if we want to detect a $\pm 25\%$ change in the parameters R and C, we would need to choose a threshold of maximum 0.01 for the average residual values. This value was derived by ignoring any noise in the measurements. It would need to be revised to accommodate for the noise when used on a real system.

Once a fault is detected using the ARRs, a list of possible causes is generated. For each possible cause, an optimization algorithm is instantiated to find the best explanation. Using the magnitude of the ARR residuals to draw a conclusion may be misleading. For example a short in resistor R generates residual values comparable in magnitude with the residual values generated in the case of an open connection in D_1. For the resistor and capacitor we track the parameters R and C. In the case of the diode faults, we track the saturation currents i_S.

We used Powell's gradient free algorithm for estimating the fault parameters. With this approach we do have to worry about the gradient approximation. Since the number of optimization variables is small, the algorithm runs fast enough.

Tables 3 and 4 show the results when the ARRs indicate that either R or C may be faulty. The first column of the table depicts the real mode. This means that the second row corresponds to the resistor having drifted from 100 Ohm to 150 Ohm. We estimate each parameter at a time (columns 2 and 3) and the results when R and C are estimated simultaneously.

Table 3 presents the parameter estimates while Table 4 shows the cost values computes at the estimate values. First we note that the cost estimates can indeed be use to differentiate between the faults. We also note, that when estimating the two parameters simultaneously, the results are not correct although the resulting cost value may be small.

Table 3. Optimization results for faults in R and C - parameter estimates

mode	$\hat{R}[\Omega]$	$\hat{C}[\mu F]$	$\hat{R}[\Omega], \hat{C}[\mu F]$
$R = 150[\Omega]$	**149.99**	3.33	105.01, 3.18
$C = 5[\mu F]$	149.99	**4.99**	149.99,3.28

Table 4. Optimization results for faults in R and C - cost function estimates

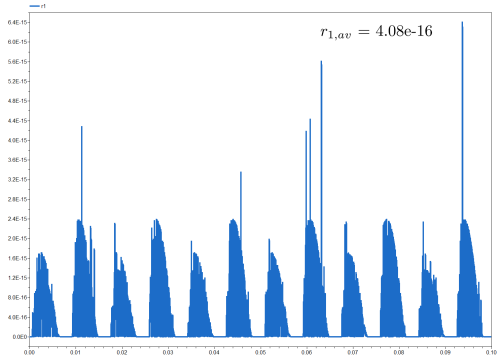
mode	$J(\hat{R})$	$J(\hat{C})$	$J(\hat{R}, \hat{C})$
$R = 150[\Omega]$	**9.83e-12**	1e-4	8.4e-5
$C = 5[\mu F]$	5.6e-2	**2.1e-5**	1.9e-4

We repeat the same steps for differentiating between faults in diodes D_1 and D_2.

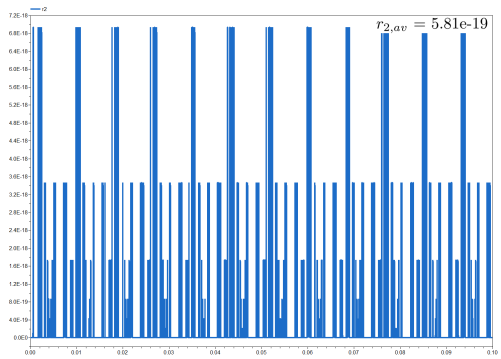
Table 5. Optimization results for faults in D_1 and D_1 - saturation currents estimates

mode	$\hat{i}_{S_1}[A]$	$\hat{i}_{S_2}[A]$	$\hat{i}_{S_1}[A], \hat{i}_{S_2}[A]$
$\hat{i}_{S_1} = 0[A]$	**1e-22**	7.57e-13	1.05e-22, 1e-12
$\hat{i}_{S_2} = 0[A]$	7.19e-13	**9.99e-23**	9.99e-13,1e-22

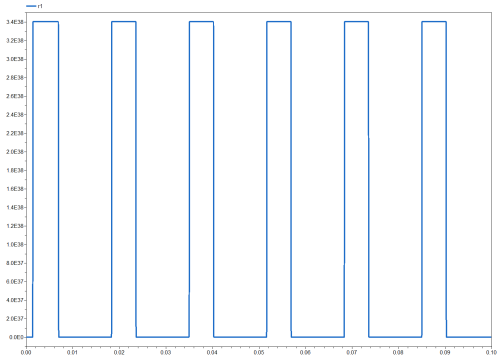
We note that in this case even when estimating simultaneously the saturation currents in the two diodes we get correct results (recall that the nominal value for the saturation current is i_S=1e-12 A). Since the optimization problem in nonlinear and usually non-convex, we cannot guarantee convergence to



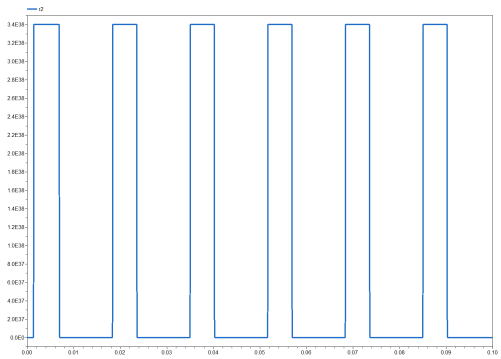
(a) Residual r_1 in nominal mode, $r_{1,0:0.1} = 4.08e-16$



(b) Residual r_2 in nominal mode,$r_{2,0:0.1} = 5.81e-19$



(c) Residual r_1 for open connection in D_1, $r_{1,0:0.1} = 3.4e38$



(d) Residual r_2 for open connection in D_1,$r_{2,0:0.1} = 3.4e38$

Figure 3. Residuals r_1 and r_2 for various nominal and fault modes

Table 6. Optimization results for faults in D_1 and D_1 - cost function estimates

mode	$J(\hat{i}_{S_1})$	$J(\hat{i}_{S_2})$	$J(\hat{i}_{S_1}, \hat{i}_{S_2})$
$\hat{i}_{S_1} = 0[A]$	**0**	2.4976	1.55e-3
$\hat{i}_{S_2} = 0[A]$	0.038	**3.83e-13**	4.03e-13

the global minima. We can imagine an architecture where different types of optimization algorithms initialized at different initial conditions are instantiated. We can then select the parameter estimates based on their cost functions evaluated at the estimates. The approach discussed above can be applied online as well, by using a moving window idea to generate time series. This provides the advantage of a good guess for the initial conditions of the optimization variables under the assumption that there are no sudden changes in the parameter values. Some of the simulated residual values take values unrealistically large. In a real system we would expect these value to be between some physically realizable values. Still, we expect them to be significantly large to induce correct values in the coherence vectors.

5. CONCLUSIONS

We discussed an approach under which qualitative based diagnosis is used to improve the numerical efficiency of an optimization based approach for parameter estimation. The qualitative diagnosis uses ARRs to generate a list of candidate components/parameters that would explain an abnormal behavior. It is based on a FSM derived offline. This list is used to instantiate a set of optimization algorithms that learn separately the values of the parameters in the list. We demonstrate our approach for diagnosing faults in a rectifier circuit.

As a future work we plan to extend our approach to handling non-parametric faults as well. Parasitics, for example, is a growing problem in modern integrated circuits and plan to revise our approaches for diagnosing these kinds of problems.

REFERENCES

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., ... Zheng, X. (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.

Beyer, H.-G., & Schwefel, H.-P. (2002, May). Evolution strategies - a comprehensive introduc-

tion. *Natural Computing*, *1*(1), 3–52. doi: 10.1023/A:1015059928466

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

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

Honda, T., Saund, E., Matei, I., Janssen, B., Saha, B., Bobrow, D. G., ... Lattmann, Z. (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).

Hooke, R., & Jeeves, T. A. (1961, April). "Direct Search" Solution of Numerical and Statistical Problems. *J. ACM*, *8*(2), 212–229. doi: 10.1145/321062.321069

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

James, F., & Roos, M. (1975). Minuit: A System for Function Minimization and Analysis of the Parameter Errors and Correlations. *Comput. Phys. Commun.*, *10*, 343-367. doi: 10.1016/0010-4655(75)90039-9

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.

Kanzow, C., Yamashita, N., & Fukushima, M. (2004). Levenberg-Marquardt methods with strong local convergence properties for solving nonlinear equations with convex constraints. *Journal of Computational and Applied Mathematics*, *172*(2), 375 - 397. doi: http://dx.doi.org/10.1016/j.cam.2004.02.013

Kennedy, J., & Eberhart, R. C. (1995). Particle swarm optimization. In *Proceedings of the ieee international conference on neural networks* (pp. 1942–1948).

Minhas, R., de Kleer, J., Matei, I., Saha, B., Janssen, B., Bobrow, D., & Kurtoglu, T. (2014). Using fault augmented modelica models for diagnostics. In *Proceedings of the 10th international modelicaconference* (pp. 437–445).

Mitchell, M. (1998). *An introduction to genetic algorithms*. Cambridge, MA, USA: MIT Press.

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

Powell, M. J. D. (1964). An efficient method for finding the minimum of a function of several variables without calculating derivatives. *The Computer Journal*, *7*(2), 155-162. doi: 10.1093/comjnl/7.2.155

Powell, M. J. D. (1973, April). On Search Directions for Minimization Algorithms. *Mathematical Programming*, *4*(2), 193–201.

Saha, B., Honda, T., Matei, I., Saund, E., de Kleer, J., Janssen, W. C., ... Bobrow, D. G. (2014, August,). Model-based approach for optimal maintenance strategy. In *Proceedings of second european conference of the prognostics and health management society.*

Samantaray, A. K., & Bouamama, B. O. (2008). *Model-based process supervision: A bond graph approach* (1st ed.). Springer Publishing Company, Incorporated.

Staroswiecki, M. (2000). Quantitative and qualitative models for fault detection and isolation. *Mechanical Systems and Signal Processing*, *14*(3), 301 - 325. doi: https://doi.org/10.1006/mssp.2000.1293

Staroswiecki, M., & Comtet-Varga, G. (2001). Analytical redundancy relations for fault detection and isolation in algebraic dynamic systems. *Automatica*, *37*(5), 687 - 699. doi: https://doi.org/10.1016/S0005-1098(01)00005-X

Subramanian, V. (2018). *Deep learning with pytorch: A practical approach to building neural network models using pytorch* (1st ed.). Packt Publishing.