Efficient Online Analysis of Accidental Fault Localization for Dynamic Systems using Hidden Markov Model

Ning Ge, Shin Nakajima, Marc Pantel

Efficient Online Analysis of Accidental Fault Localization for Dynamic Systems using Hidden Markov Model

Ning Ge[1], Shin Nakajima[2] and Marc Pantel[1]
[1]University of Toulouse, IRIT/INPT, Toulouse, France
[2]National Institute of Informatics, Tokyo, Japan
{ning.ge | marc.pantel}@enseeiht.fr | nkjm@nii.ac.jp

Keywords: Simulation; Online Analysis; Accidental Fault Localization; Hidden Markov Model;

Abstract

This paper proposes a novel approach to do online analysis of accidental fault localization for dynamic systems by using Hidden Markov Model (HMM). By introducing reasonable and appropriate abstraction of complex system, HMM is used to represent the fault and no-fault states of system's components and system's behaviour. The HMM is parametrized to be statistically equivalent to real system's behaviour. Inspired by the principles of Fault Tree Analysis and maximum entropy in Bayesian probability theory, we propose the algorithms to estimate HMM's parameters, instead of learning, because in real systems the learning data for accidental fault is difficult to obtain. We design a specific test bed to generate large quantity of test cases, and give out the experimental results to assess the accuracy and efficiency. Meanwhile, we apply the approach to a simple helicopter control system case study, and give out convincing results.

1. INTRODUCTION

Fault Detection and Isolation (FDI) is dedicated to monitoring a system, identifying when a fault has occurred, and pinpointing the type of fault and its location. One main FDI approach derives the faults from some model, classified into the category Model-Based FDI, while another main category is Signal Processing based FDI. In model-based FDI, the system model may be mathematical, or knowledge based, including observer-based approach, parity-space approach, parameter identification based methods, etc [1]. As the dynamic systems' complexity increases, the resource-consuming simulation technology is insufficient for detecting faults in systems. Thus, it becomes urgent to use abstract model to represent complex system by keeping necessary and sufficient information. The efficiency and accuracy of model-based approach depends on the appropriate abstraction and reasonable assumptions.

Many theories and techniques exist for the analysis and simulation of large dynamic systems. Our previous work [2] proposes a co-analysis framework for the automated analysis of cyber-physical systems (CPS) [3], in which the behaviour of Simulink is taken in an as-is manner determined by the simulation algorithms. The approach combines logic-based formal analysis methods with numerical simulations to enable the analysis of an under-constrained controller design, which cannot be handled by co-simulation. An open question proposed by the perspective in [2] concerns the fault detection and localization in systems. CPS introduce a new paradigm to software-intensive systems, in which the controller (system) is strongly affected by feedback from the plant (environment). As shown in Fig. 1, a controller may have several control modes and changes itself with transitions between the modes. Since some of the Simulink model descriptions represent hardware components, the controller is expected to be able to do online analysis of the accidental fault localization $\mathcal{F}(t)$ occurring in the hardware.

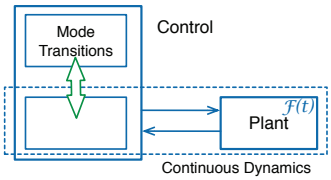

Figure 1. Cyber-Physical System with Control Modes

In this paper, we aim to solve the above problem by using a statistical model, Hidden Markov Model (HMM) [4]. In HMM, the system being modeled is assumed to be a Markov process with unobserved (hidden) states. By analysing the observed outputs of the hidden states, system's behaviour can be deduced. HMM has thus been widely used in temporal pattern recognition such as speech, handwriting, gesture recognition, etc. HMM has been used to diagnose and prognose the whole system's health condition in some existing works [5] [6]. The learning tasks are used to train the parameters in HMM. However, the learning task is based on rich observed sequences with enough fault experience, which is not easy to obtain for accidental faults. In this work, we propose a novel approach to use HMM to do online analysis of accidental fault localization in dynamic systems. We introduce appropriate abstraction of complex dynamic systems and reasonable assumptions of HMM parameters, to represent the fault and no-fault states of system's components and system's behaviour. Inspired by the principles of Fault Tree Analysis (FTA) [7] and of maximum entropy [8] in Bayesian probabil-

ity theory, we propose an algorithm to estimate the parameters in HMM without learning task. We evaluate the accuracy and efficiency of proposed algorithm by using a specific test bed to generate 1000 test cases, and then apply the approach to a simple helicopter control system case study.

In paper [9], the author aims to prove the validation of CPS's behavioural properties by statistical model. Our main ideas are identical: using statistical model to deduce the conclusion, except that [9] introduces a formal description in BLTL (Bounded Linear Temporal Logic), instead of HMM, to interpret the probable result sequence, which is the same concept as our observation sequence. By using importance-sampling and cross-entropy (two classic statistical methods), the main obstacle [9] encounters is that it needs much more samples to give out a convincing conclusion. The reason behind is that BLTL treats each element in a result sequence in a pure statistical thus relatively independent way, while HMM assumes that there must be some explicit dependency between the elements.

This paper is organized as follows: Section 2. details the problem we aim to solve; Section 3. gives an overview of our approach by introducing HMM modeling and analysis methods; Section 4. presents the approach of online analysing accidental fault localization for complex dynamic systems by using HMM; Section 5. experiments the approach in a specific test bed; applies the approach to a simple helicopter control system case study; and discusses the generalization of the approach; Section 6. gives some concluding and perspective remarks.

2. PROBLEM DESCRIPTION

The accidental fault may occur on the hardware during system's execution. Since the accidental fault does not frequently occurs, a reasonable assumption is made in this work.

Assumption 1. *A system encounters accidental fault means at this moment, at most one device is the fault source. Two devices cannot encounter accidental fault at the same time.*

The prior knowledge in this work consists of devices' default parameters and system's specification. The device's default parameters indicate a device's fault occurrence probability. This parameter is usually provided by the device manufacture according to the testing result before selling. Alternatively, MTBF (Mean Time Between Failures) can be used to deduce this parameter if information is lacking. System's specification describes the expected functional constraints.

Example 1 (Problem Description Example). *During a finite simulation period T, we choose N observation time points for the system with components A, B, C, D, etc (Fig. 2). Assume M functional constraints (FC) are described in the specification. $v_i (i = 1...11)$ are output/input variables between the components.*

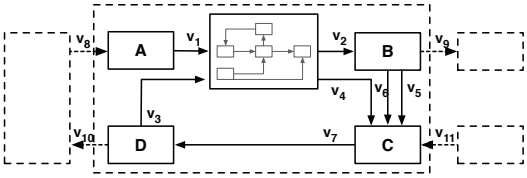

Figure 2. Problem Description Example

After time T, we obtain the violation status (R for respecting specification, V for violating specification) for the M functional constraints at time points t ($t = 1...N$), i.e.

$$FC_1^t(v_1, v_3, v_5) = V$$
$$\dots$$
$$FC_M^t(v_2) = R$$

When a functional constraint is violated, one of the devices must have encountered an accidental fault. However once we detect a wrong output from a device, usually the others' outputs will be bad due to failure propagation. Thus the problem is distinguishing and locating accidental fault's source device. This can be considered as a pattern recognition question. Moreover, the accidental fault localization approach should support online analysis, which implies that the method needs to be computation-economic and easy to implemented by most onboard dynamic systems.

The simulation technology is able to provide analysable evidence to help fault identification. Nevertheless, it lacks the mechanism to automatically locate the accidental fault. We introduce HMM in this work, because HMM, a good compromise between Bayesian network and time-based sequence, disposes of a natural capacity to deal with time-related pattern recognition problems. As shown in Fig. 3, HMM can be used to diagnostics, prognostics and online analysis.

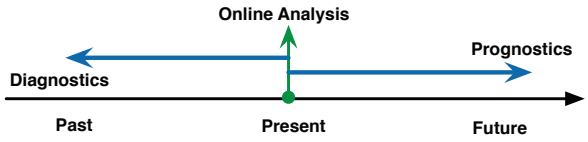

Figure 3. HMM Diagnostics and Prognostics

On the other hand, HMM, as the abstraction of real system, simulates statistically system's inner states and their behaviours. Once HMM is trained or built, it can be used to diagnose system's past and current health condition and prognoses future condition. Some works are aimed to detect the whole system's health condition, while in this work we aim to go further, to locate the fault source in the system.

3. HMM MODELING AND ANALYSIS

An HMM is defined as a statistical model used to represent stochastic processes, where the states are not directly observed. A basic HMM can be described as follows:

- **N**: number of states
- **M**: number of observations
- $\mathbf{M_I}$: initial probability distribution; $\sum_{i=1}^{N} \mathbf{M_I}(i) = 1$
- $\mathbf{M_T}$: probability distribution of transitions from states to states; $\sum_{j=1}^{N} \mathbf{M_T}(i,j) = 1, i = 1...N$
- $\mathbf{M_E}$: emission distribution for the observations associated with states; $\sum_{j=1}^{M} \mathbf{M_E}(i,j) = 1, i = 1...N$

Example 2 (HMM Example). *A two states HMM example abstracting a system's health condition is given by Fig. 4, where the system owns two states Healthy and Faulty, and two observations which represent whether the ouputs respect the functional constraints (R) or violate the functional constraints (V). The three distributions* $\mathbf{M_I}\mathbf{M_T}\mathbf{M_E}$ *are:* $\begin{pmatrix}0.6 & 0.4\end{pmatrix}\begin{pmatrix}0.7 & 0.3\\ 0.4 & 0.6\end{pmatrix}\begin{pmatrix}0.9 & 0.1\\ 0.2 & 0.8\end{pmatrix}$.

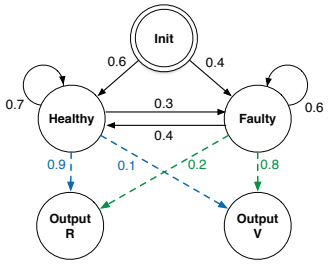

Figure 4. A Two States Hidden Markov Model

HMM, as abstract model of real system, is statistically identical to system's real behaviour. When modeling a system, HMM separates the concept into two conceptually independent paradigms: behaviour and observation. Behaviour refers to what the system really is; while observation to what the system exhibits that is used for its recognition. $\mathbf{M_I}$ gives indication about the probability that a behaviour becomes the first behaviour when system runs. $\mathbf{M_T}$ decides how probably will the system behave from one state to the other states. This is statistically equivalent to the real system's behaviour. $\mathbf{M_E}$ provides a distribution that connects the behaviour and the observation: if at a given time the behaviour is known, how probably an observed sequence will occur.

$\mathbf{M_I}, \mathbf{M_T}$ and $\mathbf{M_E}$ can be obtained by modelling or through a learning process. Once all these matrix parameters are estimated, HMM is capable to deduce, given an observed output sequence or a set of such sequences, the maximum likelihood estimation of inner-state transition sequences. The work [5] shows a traditional way of applying HMM to system diagnosis and prognosis, as shown in Fig. 5. The trained HMM will

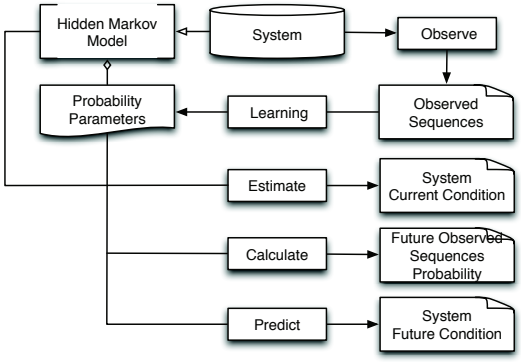

Figure 5. Traditional HMM Approach

be used to estimate system's past condition, predict system's future condition and compute a future observed sequence's occurrence probability.

In Ex. 2, we can observe an output sequence $Seq(t)$ from the observers Output R and Output V within a certain observation time T, e.g. $\begin{pmatrix}V & V & V & V & R & V & V & V\end{pmatrix}$.

As the parameters are pre-defined, we can derive system's behaviour $State(t)$ during T by using Viterbi algorithm [10], in which H means healthy, and F means faulty, i.e. $\begin{pmatrix}F & F & F & F & H & F & F & F\end{pmatrix}$.

In this work, our objective is to find out how system performs in terms of behaviour, however, the only available information source for the external world is the observation. Therefore a backward analysis is necessary to recover the behaviour from the observation. As we aim to tell the exact location of the fault, not only detecting unhealthy condition of a system, the core issue is how to construct HMM representing a system with probable fault.

The learning task cannot be used in this work. Because the accidental fault does not frequently happen, thus the learning task cannot be trained with rich fault experience. The modeling method in our research allows to pre-define some HMM parameters by using prior knowledge of devices' fault occurrence probability.

4. ONLINE ANALYSIS OF ACCIDENTAL FAULT LOCALIZATION

The approach takes all the components in a system as a whole model. When modeling HMM, a hidden state represents, at this moment, on which components one accidental fault occurs, or not. Instead of using learning task to train the parameters in HMM, we define the specific states and observations dedicated to accidental fault localization, and propose our own algorithms to compute the HMM's parameters by analysing system's architecture and using devices' fault occurrence probability. The proposed algorithms have been evaluated in the experiments by using our test bed and real case study, which will be presented in the next part.

Definition 1 (**Physical Variable**). *A component C in a system has N_I inputs and N_O outputs variables, which connect C with other components. These N_I together with N_O variables are the physical variables of C.*

By running the system, for some physical variables, whether it violates or not the functional constraints will be known. If it is possible to observe M physical variables' value sequence, the whole system can be modeled by M HMMs. Each represents whether current system's behaviour will lead to one physical variable violating the functional constraints, and which kind of violation.

4.1. System States

System states represent the health condition of all its components. A system with N components is seen as a coupling entity with $N+1$ states, each of them represents at the given moment, which component encounters a certain context that leads it to give out unwanted output. HMM states are either all healthy (N_F) or having one faulty component (F_i), modeled as $\begin{pmatrix} N_F & F_1 & F_2 & \dots & F_i & \dots & F_N \end{pmatrix}$.

4.2. Observations & Observed Sequence

The observations represent that physical variables violate the functional constraints or not.

Definition 2 (**Non-Violation** (V_N)). *When a physical variable does not violate functional constraint, this is defined as non-violation.*

Definition 3 (**Independent Violation** (V_I)). *When a physical variable v violates functional constraint with only this variable, i.e. $FC(v)$, this is defined as independent violation.*

Definition 4 (**Coupling Violation** (V_C)). *When a physical variable v violates functional constraint with other variables, i.e. $FC'(v, v_1, v_2, \dots)$, the source of violation cannot be confirmed among the variables. The probability of each is thus given out. It is defined as coupling violation.*

Each HMM has these 3 observations: V_N, $\mathcal{V}_I$ and $\mathcal{V}_C$. A system with M physical variables has M HMMs (Fig. 6).

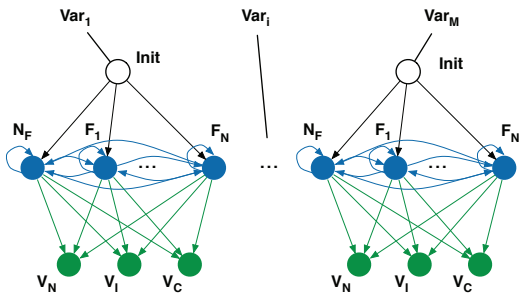

Figure 6. System's HMM Structure

4.3. Initial Probability Distribution $\mathbf{M_I}$

The devices' default occurrence parameters are provided by the manufacture. A component's fault parameter can be estimated when it is well designed, e.g. a functional block provided by the library of Simulink can be fully trusted, with fault occurrence parameter 0. For a system with N components, let $\omega_i (i = 1 \dots N)$ be the fault occurrence parameter of the i^{th} component. $\mathbf{M_I}$ is thus computed as follows:

$$\mathbf{M_I}(0) = \frac{\prod_{i=1}^{N}(1-\omega_i)}{\prod_{i=1}^{N}(1-\omega_i) + \sum_{i=1}^{N}\omega_i} \tag{1}$$

$$\mathbf{M_I}(i) = \frac{\omega_i}{\prod_{i=1}^{N}(1-\omega_i) + \sum_{i=1}^{N}\omega_i}, i = 1 \dots N \tag{2}$$

Eqs. 1 and 2 describe the likelihood that no component has failed at time 0 (i.e., no failure at the start) and the likelihood that some component i has failed at the start.

If prior knowledge about ω_i is unknown, MTBF is used to deduce the initial parameters. Therefore, the most reasonable hypothesis is $\sum_{i,j=1}^{N} |\omega_i - \omega_j| = 0$. If some indications exist, like the i^{th} component is twice more probable to have fault than the j^{th}, then $\omega_i = 2\omega_j$ will replace $\omega_i = \omega_j$. $\sum_{i,j=1}^{N} |\omega_i - \omega_j| = 0$ will be generalized to $min(\sum_{i,j=1}^{N} |\omega_i - \omega_j|)$. This can be solved because $\sum_{i=0}^{N} \mathbf{M_I}(i) = 1$.

4.4. Transition Probability Matrix $\mathbf{M_T}$

$\mathbf{M_T}(i, j)$ represents the probability of the transition from state i to state j. We propose to compute $\mathbf{M_T}$ by using the devices' fault parameters ω_i. When the system transits from current state to all components healthy state, the transition probability is $\prod_{k=1}^{N}(1-\omega_k)$. When the system transits to one component faulty state j, the probability is $\omega_j \prod_{k=1,k\neq j}^{N}(1-\omega_k)$. After normalization, $\mathbf{M_T}$ is as follows:

$$\mathbf{M_T}(i,0) = \frac{\prod_{k=1}^{N}(1-\omega_k)}{\prod_{k=1}^{N}(1-\omega_k) + \sum_{n=1}^{N}(\omega_n \prod_{k=1,k\neq n}^{N}(1-\omega_k))}, \quad i = 0 \dots N \tag{3}$$

$$\mathbf{M_T}(i,j) = \frac{\omega_j \prod_{k=1,k\neq j}^{N}(1-\omega_k)}{\prod_{k=1}^{N}(1-\omega_k) + \sum_{n=1}^{N}(\omega_n \prod_{k=1,k\neq n}^{N}(1-\omega_k))}, \quad i = 0...N, j = 1...N \tag{4}$$

Eqs. 3 and 4 are the probability from component i failing to no component failing at a time t > 0 and from component i failing to component j failing at time t > 0.

4.5. Emission Probability Matrix $\mathbf{M_E}$

$\mathbf{M_E}$ represents, if a component occurs fault, how probably it will influence the functional constraints. More precisely, how probably the physical variables will violate the functional constraints. Inspired by the principles of FTA and the principle of maximum entropy in Bayesian probability theory, we propose the algorithm for computing $\mathbf{M_E}$ in Fig. 7. We will explain the definitions and the proposed algorithms in the following parts.

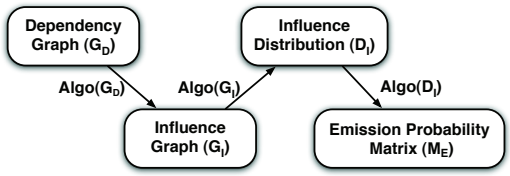

Figure 7. $\mathbf{M_E}$ Algorithm

4.5.1. Computing Influence Graph $\mathbf{G_I}$

Definition 5 (Dependency Graph $\mathbf{G_D}$). *Dependency graph represents the dependency between system's components.*

Definition 6 (Influence Graph $\mathbf{G_I}$). *Influence graph is a graph representing how components' accidental fault influences physical variables. Influence graph is topologically identical to dependency graph, with supplementary influence weight indicating the probability that the component C influences the variables v_i. The influence weights $\mathbf{G_I}(C, v_i)$ are computed by using Influence Layout Algorithm.*

Example 3 (Influence Graph Example). *Fig. 8 is an influence graph with 9 components and 15 physical variables. $\mathbf{G_I}(A, v_3)$ is the influence weight between A and v_3.*

Algorithm 1 (Influence Layout Algorithm). *Inspired by the principles of FTA, we propose this influence layout algorithm. An influence graph with N components and M variables can be solved by applying linear programming to Eq. 5-8 .*

$$\sum_{i=1}^{M} v_i = 1 \tag{5}$$

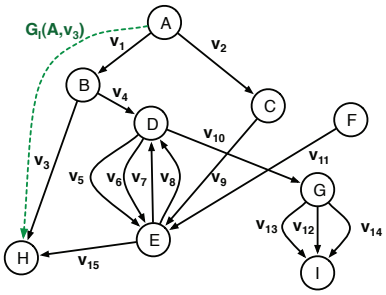

Figure 8. Influence Graph Example

$$v_i > \frac{1}{\delta M}, i = 1...M, \delta > 1 \tag{6}$$

For each component C, the sum of input value equals to the sum of its output value, to ensure C does not introduce supplementary side effect to system's failure analysis (Eq. 7).

$$\sum_{in} v_i - \sum_{out} v_j = 0 \tag{7}$$

According to the principle of maximum entropy in Bayesian probability theory, without any prior knowledge, the difference between variables should be minimised (Eq. 8).

$$\min_{v_1...v_M} \sum_{i=1}^{M} |v_i - \frac{1}{M}\sum_{j=1}^{M} v_i| \tag{8}$$

4.5.2. Computing Influence Distribution $\mathbf{D_I}$

Definition 7 (Influence Distribution $\mathbf{D_I}$). *When component C_i occurs fault, the influence it has for physical variable v_j is defined as $\mathbf{D_I}(i, j)$.*

The direct way to get $\mathbf{D_I}$ is the use of simulation. If enough simulation scenarios covering the range of all the variables are provided, $\mathbf{D_I}$ can be exactly constructed with simulation results. However, the use of simulation is not practical for the non-frequently happening accidental fault. To make a compromise between the computation time and the precision, instead of simulation, we compute $\mathbf{D_I}$ by using the *Influence Graph*. This approach may sacrifice some precision, however, the experimental results in Sec. 5. will prove its accuracy and efficiency for the accidental fault problem.

When a component's output physical variable violates functional constraints, either because it encounters an accidental fault, or because its dependent components has spreadable fault. The propagation of fault is related to system's architecture that:

- If C_i has encountered an accidental fault, it will probably generate an output violation. The reason why it is not a definitive violation is that in one simulation, the required-context for violation may not happen. This probability is defined as $\mathbf{D_A}$.

- If C_i depends on C_j (C_i directly or indirectly takes C_j's output as input), and if an accidental fault occurs on C_j, C_i will probably have an output violation. It is not a definitive violation, because either in one simulation, the required-context for violation may not happen; or the intermediate component's design has fault-tolerance consideration. This probability is defined as $\mathbf{D_B}$.

Algorithm 2 (Influence Distribution Algorithm). *As* $\mathbf{D_A}$ *and* $\mathbf{D_B}$ *are independent, we have* $\mathbf{D_I} = \mathbf{D_A} + \mathbf{D_B}$. $\mathbf{D_A}$ *and* $\mathbf{D_B}$ *can be computed using* $\mathbf{G_I}$.

- *If variable v_j does not depend on component C_i (C_i has no path to v_j),* $\mathbf{D_I}(i,j) = 0$
- *If v_j directly depends on C_i,* $\mathbf{D_A}(i,j) = \mathbf{G_I}(C_i, v_j)$
- *If v_j indirectly depends on C_i, assume there are P paths from C_i to v_j. Let O_{ik} be the k^{th} output of C_i, and $\boldsymbol{S}(C_i)$ is the index of C_i's successor component on current path. The recursive computation is as follow:*

$$\mathbf{D_B}(i,j) = \sum_{k=1}^{P} \mathbf{G_I}(C_i, O_{ik}) \cdot \mathbf{D_I}(\boldsymbol{S}(C_i), j) \tag{9}$$

Eq. 9 describes system's fault propagation and fault exposition. If some faults occur, the most probable faulty component should be the one that exchanges the most often the data/information. That's why there is always an accumulation.

4.5.3. Computing Emission Probability Matrix ($\mathbf{M_E}$)

Algorithm 3 (Emission Matrix Algorithm). *For the HMM of variable v, if the state representing C_i is failing, we use the following algorithm to compute the probability of generating the observations V_I, V_C and V_N.*

For V_I, $\mathbf{M_E}$ should be a function of $\mathbf{D_I}$. According to our test, we found that function y=x is good enough to give out a relatively accurate result, as shown in Eq. 10.

$$\mathbf{M_E}(C_i, V_I) = \mathbf{D_I}(C_i, v) \tag{10}$$

For V_C, if v violates several functional constrains with other variables (v_i), the occurrence time of v_i in these constraints is n_i, and the occurrence time of v is n. The coupling violation is computed as follows:

$$\mathbf{M_E}(C_i, V_C) = \frac{n\mathbf{D_I}(C_i, v)}{\sum_i n_i \mathbf{D_I}(C_i, v_i)} \tag{11}$$

The emission probability from state to V_N is:

$$\mathbf{M_E}(C_i, V_N) = 1 - \mathbf{M_E}(C_i, V_I) - \mathbf{M_E}(C_i, V_C) \tag{12}$$

Since the only valuable information/entropy in fault localization method will become maximized only when a fault happens, therefore it is difficult (even impossible) to find directly a statistical distribution for those fault-free behaviour. In this case, in order to not introduce some new assumption, we choose to use the HMM's mathematical property (emission matrix must be semantically exclusive and the sum should be 1) to present the normal state-observation pair by Eq. 12.

Obviously, the healthy state (N_F) will lead to $\mathbf{M_E}(N_F, V_N) = 1$, $\mathbf{M_E}(N_F, V_I) = 0$ *and* $\mathbf{M_E}(N_F, V_C) = 0$.

4.6. Online Locating Accidental Fault

If we choose T_S simulation time points for all the M HMMs, we observe $T_S \cdot M$ sequences, with which the most probable system behaviour can be derived. By computing the statistic of each state, the state with maximum probability is located as the fault source. To online locate accidental fault for time point t, we use the observed sequences in the past time of t according to the feature of markov process. However, when t is too long, we might have huge quantity of observed sequences, which makes the computation time increases. To solve this problem, a sampling window $\mathcal{W}$ is introduced to limit the length of t.

Example 4. *At time point 5, The observed sequences for 3 variables are:*

$$\begin{array}{l} Seq_1 : \begin{pmatrix} N_F & F_1 & F_1 & F_2 & F_1 \end{pmatrix} \\ Seq_2 : \begin{pmatrix} F_1 & F_1 & F_1 & F_3 & F_1 \end{pmatrix} \\ Seq_3 : \begin{pmatrix} N_F & F_1 & F_1 & F_2 & F_1 \end{pmatrix} \end{array}$$

The state with maximum probability is F_1, which means component C_1 is most probably the fault source.

5. EXPERIMENTAL RESULTS

5.1. Test Bed

Since it is costly to generate test data for real system, and it is less convincing by replaying a sample, we design a specific test bed to evaluate the method's accuracy and efficiency by generating a large quantity of simulated use cases. Each use case includes: the **system architecture** which defines the components and physical variables, and their interconnections; the **failure probability** of each component; the **functional specification** corresponding to each physical variable or some group of them; the **running time counter** after which the system will stop producing output.

The method assumes that: Each component in the system has a chance to fail. This probability can be 0; All functional constraints are based on variable's value itself, and for simplification, they are all range constraint, which means they delimit only the min/max value of the variables.

If an accidental fault occurs, the test bed will give out an out-of-range value for this component's output. This emulates how a system falls into fault, whatever the model is.

The test bed will extract its architecture and give each component a random low probability of falling into accidental fault as the prior knowledge of device's fault parameter.

Each component's input and output will be allocated to a variable by the test bed. It guarantees that the interconnected ones share the same variable. The variable will be associated with a random range, which is the functional specification. If a device is more probable to fail, the test-bed-generated value for its entire output variables will be more probable to go out of the defined range.

The approach will use the generated data and the functional specification to locate the most probable component as the accidental fault source if there is some violation observed. The test bed will then compare this computed conclusion with the initial context to deduce whether the method is efficient.

5.2. Experiments

Our test bed has generated 1000 test cases to evaluate the method's performance in terms of accuracy. The criterion that impacts the accuracy is the complexity of system's architecture. This can be measured by component's average input & output number (Fig. 9), and component number. (Fig. 10). We generate different scale test groups from 1 to 50 inputs/outputs or components. Each group has 20 test cases. The test results show the max/min/average accuracy ratio in each group. We find out this method is more sensible to the average input & output number, while more scalable to component number. This method deals with the accidental fault localization for middle-range systems with accuracy superior to 95%.

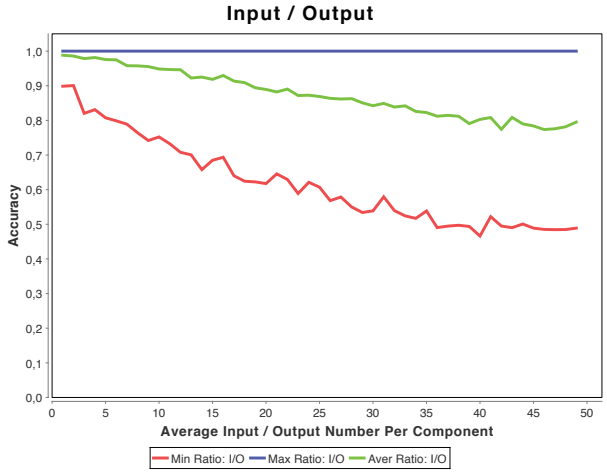

Figure 9. Accuracy by Input Number

The computation of the HMM parameters by resolving linear system (the influence layout algorithm) and eventually optimising non-linear system (the initial matrix without complete prior knowledge) consumes time, ranging from seconds to several minutes according to the system's complexity. However, it computes only once before the fault localization process begins, because the HMM's initialization is architecture-dependent-only. Once the HMM is precomputed, the fault localization algorithm is very fast and depends only on the observation sequence length. Since the sampling window is bounded, this method gives a good response to the online accidental fault localization problem.

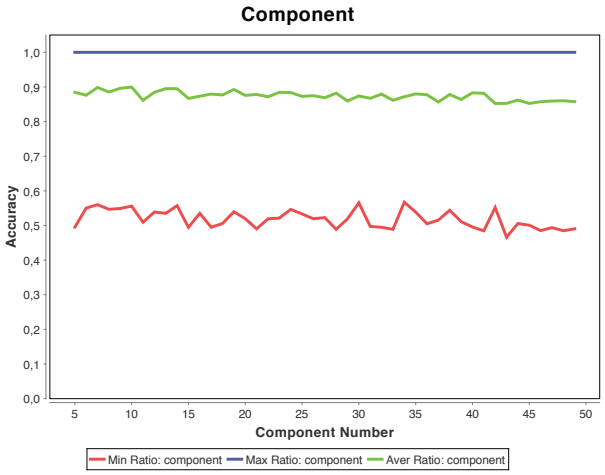

Figure 10. Accuracy by Component Number

5.3. Helicopter Control System Case Study

After certifying statistically the method's performance using the test bed, we rely on a classical Simulink teaching model of simple helicopter control system [11] to evaluate, whether the proposed approach works also well for real system. A complete mathematical model, including propeller dynamics, forces generated by the propellers, static and dynamic friction of the bearings, etc. is an overkill for method proofing objectives, therefore a simplified version is chosen. This helicopter is a linear 4^{th} order system, where the 6 control matrices are pre-defined.

In order to introduce the accidental fault into the test, we modify the Simulink model to have some components generating faulty outputs at some given time point according to its original functionality. These components will have a computed failure probability which is based on the ratio between failure count and total simulation time tick. The total simulation time is 10000, while the sampling window size is 1000. The fault localization method gives a good accuracy throughout the test, as shown in Fig. 11: the average accuracy degrades with the widening of sampling windows, but once the sampling window is selected, the accuracy stabilizes and the total accuracy is superior to 90%.

5.4. Discussion

This method needs less complete specification to derive fault analysis conclusion, because it uses architecture concept. This is an advantage for analysing old-fashioned system, when people focus mainly on conception but not formal specification.

At the very beginning of this paper, we reasonably assume there is at most one accidental fault in the whole system at

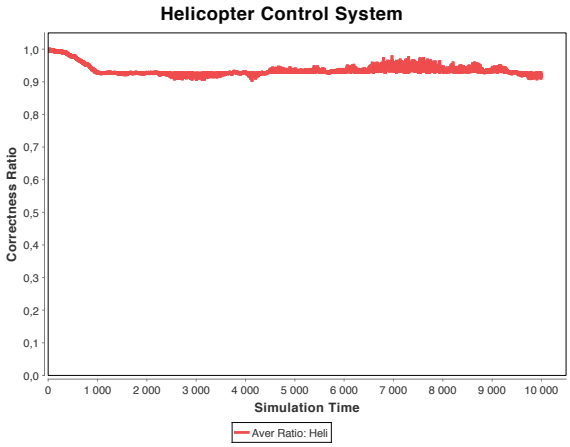

Figure 11. Helicopter Control System Case Study Result

time point t. In this situation, the computational complexity is only O(n), which is scalable. If we consider there might be n accidental fault at time t, it is $O(2^n)$ complicated in space and time, which makes it difficult to scale for large system with complex architecture. A compromise can be applied in practice if we pre-know that at most only K component will encounter accidental fault at the same time. Therefore the complexity reduce to $O(C_{(n,1)} + C_{(n,2)} + ... + C_{(n,k)})$.

A weak point of this method is the quantification of some hypothesis based on the influence distribution $\mathbf{D_I}$. This is also why we find in the experiment, when the complexity of system architecture increases, the method's accuracy degrades. To solve this problem, we should introduce an approach which makes good compromise between architecture-based HMM parametrization and observation sampling learning-based HMM initialization. This is our ongoing work.

6. CONCLUSION

This paper proposes a novel approach to do online analysis of accidental fault localization in dynamic systems by using HMM. HMM can be used as an abstraction of real system, and emulates real system's behaviour. Inspired by the principles of FTA and of maximum entropy in Bayesian probability theory, we propose the algorithms to estimate HMM's parameters, avoiding to pass through learning task, because the learning data for accidental fault in real system is difficult to obtain. Once HMM is completed, by observing output sequence within the sampling window, we can online estimate the source of accidental fault. We design a specific test bed to generate large quantity of test cases. The experimental results have assessed the accuracy and efficiency of our approach. The accuracy of accidental fault localization is superior to 95% for normal scaled systems. Meanwhile, we apply the approach to a simple helicopter control system case study, which shows the accuracy is superior to 90%.

We will improve the concreteness of the influence distribution by introducing heuristic algorithms and experiment the approach on more complicated use cases in future work. Meanwhile, we will introduce an approach which makes good compromise between architecture-based HMM parametrization and observation sampling learning-based HMM initialization. The later is capable be generalized to more applications, e.g. for detecting and locating design faults in systems.

REFERENCES

[1] S. X. Ding, *Basic ideas, major issues and tools in the observer-based FDI framework*. Springer Berlin Heidelberg, 2008, pp. 13–19.

[2] S. Nakajima, S. Furukawa, and Y. Ueda, "Co-analysis of sysml and simulink models for cyber-physical systems design," in *2nd Workshop on Cyber-Physical Systems, Networks, and Applications*, 2012, pp. 473 – 478.

[3] J. M. Wing, *Cyber-Physical Systems*, 2009, vol. 21, no. 1.

[4] L. Rabiner, "A tutorial on hidden markov models and selected applications in speech recognition," *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257 –286, 1989.

[5] M. Bjerkeseth, "Using hidden markov models for fault diagnostics and prognostics in condition based maintenance systems," Tech. Rep., May 2010.

[6] D. Tobon-Mejia, K. Medjaher, N. Zerhouni, and G. Tripot, "A data-driven failure prognostics method based on mixture of gaussians hidden markov models," *Reliability, IEEE Transactions on*, vol. 61, no. 2, pp. 491 –503, June 2012.

[7] W. E. Vesely, F. F. Goldberg, N. H. Roberts, and D. F. Haasl, *Fault Tree Handbook*. U.S. Nuclear Regulatory Commission, 1981.

[8] E. T. Jaynes, "Information Theory and Statistical Mechanics - Jaynes," *The Physical Review*, vol. 106, no. 4, pp. 620–630, 1957.

[9] E. M. Clarke and P. Zuliani, "Statistical model checking for cyber-physical systems," in *Proceedings of the 9th international conference on Automated technology for verification and analysis*, ser. ATVA'11. Berlin, Heidelberg: Springer-Verlag, 2011, pp. 1–12.

[10] A. Viterbi, "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm," *Information Theory, IEEE Transactions on*, vol. 13, no. 2, pp. 260 –269, april 1967.

[11] B. Cazzolato, *Automatic Control II - 2DOF Helicopter Tutorial & Lab*.