A novel implementation of supervisory based Fault Tolerant Control

Tushar Jain, Joseph Julien Yamé, Dominique Sauter

A novel implementation of supervisory based Fault Tolerant Control

Tushar Jain, Joseph J. Yamé and Dominique Sauter

***Abstract*— In this note, we discuss a performance-based supervisor for the fault tolerant control (FTC). We conduct the notion of stability, which is often misinterpreted in our approach with the classical arbitrary switching control. Moreover, we only deal with the trajectories generated by the system in real time and we do not have any access to plant parameters, states of the system etc. Thus, we clearly distinguished this notion employed in both the approaches. A novel switching logic is also proposed that root out the problems seen in while utilizing fixed dwell-time switching, and hysteresis switching logic in the supervisor. The proposed structure serves a significant purpose for real-time model free fault tolerant control. The use of virtual reference tool to analyze the performance of "off-the-shelf" controllers is also highlighted. Thus the explicit use of fault diagnosis module can easily be dropped out in a supervisor based FTC.**

I. INTRODUCTION

Fault tolerance is the property of a system that aims at guaranteeing the control objectives are achievable in spite of faults [1]. . However, in some cases the fault is not accommodated completely and the control objective has to be lowered down so that the system is still controllable in some sense. Various researchers in FTC community have studied this particular case and it is termed as *disgraceful degraded performance*. Here we assumed, the system is controllable at any time (pre-fault/post-fault scenario) satisfying the original control objectives.

Patton [10] introduced a scheme of fault tolerant control system with a supervision subsystem as shown in Fig. 1. This figure shows the general function scheme of fault tolerant control with four main components: the plant itself (including sensors and actuators), the fault detection and diagnosis (FDD) unit, the feedback (or feed-forward) controller, and the supervision sub-system. The main controller activities are represented by solid line. The dashed line represents the operation of FDD unit, and the dotted line represents adaptation (tuning, scheduling, accommodation, and reconfiguration). The plant is considered to have potential faults in sensors, actuators (or other components). In the faultless case, FDD unit remains inactive and the nominal feedback controller attenuates the disturbances and ensures set-point following and other requirements on the closed loop system. The FDD unit is responsible for providing the supervision system with information about the onset, location and severity of any faults. On the supervision level the diagnostic block simply recognizes that the closed-loop system is faultless and no change of the control law is necessary. If a fault occurs, the supervision sub-system makes the control loop fault-tolerant. The diagnostic block identifies the fault. Based on the system inputs and outputs together with fault decision information from the diagnostic block, the supervision sub-system reconfigures the sensor set and/or actuators to isolate the faults, and tune or adapt the controller to accommodate the fault effects so that the closed loop satisfies the performance specifications. The architecture has various superior characteristics from the classical scheme of fault tolerant control system [1, pg.11]. In classical scheme, fault accommodation (FA) unit generally takes necessary actions depending on the information fed by fault diagnosis (FD) module. The uncertainty and robustness issues in FD module are quite known. On the other hand, there is a two-way flow of information between the FD unit and supervisor. Therefore, the supervisor serves a main executing block to FTC while in [1, pg.11], the integrated FD-FA forms the executing unit on occurrence of fault. Taking the benefits from this scheme, we proceed further in this direction of supervisory based FTC.

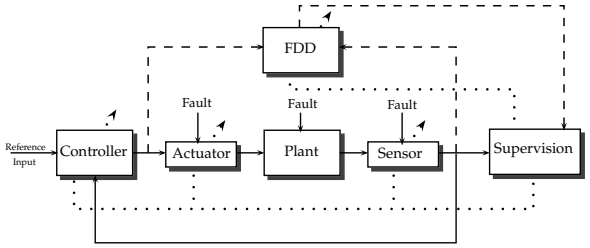

Fig. 1. Fault Tolerant Control system architecture with supervision subsystem [10].

The supervisory scheme to FTC has been studied in [5], [13], [7] that utilizes the *complete architecture* as shown in Fig. 1. The robustness proof was examined in [2]. A novel approach to FTC was studied in [12] using the virtual reference tool [11], though this tool is quite in-practice in control community since a decade. Therefore, taking the benefits of this approach in FTC community, we further explored it in this paper. Here, the main emphasis is given on constructing the supervisor, which is quite significant to build model-free scheme to FTC. Later we propose a new switching logic that assists in optimal selection of candidate controller and its superiority over other two switching mechanisms.

II. SUPERVISION BASED FTC

Logic based (or supervision) based approach to FTC has emerged as a significant technique that effectively answers

The authors are with Centre de Recherche en Automatique de Nancy (CRAN), UMR 7039, Faculté des Sciences et Techniques, Université Henri Poincare Nancy 1, 54506 Vandoeuvre-les-Nancy, France {tushar.jain, joseph.yame, dominique.sauter}@cran.uhp-nancy.fr

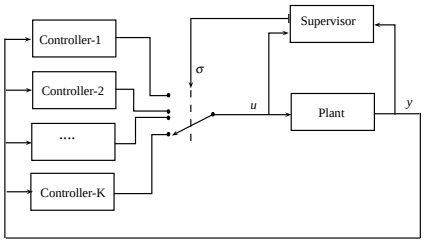

Fig. 2. Structure of logic (or supervision)-based switching controller

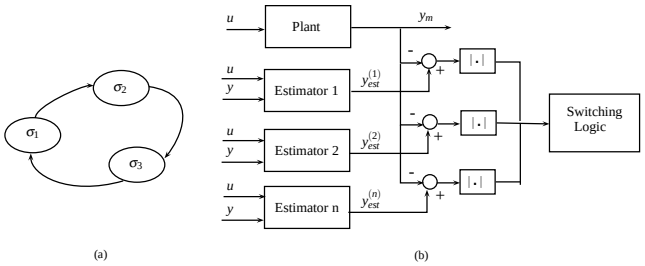

Fig. 3. (a) Pre-routed supervision (PRS); (b) Estimator-based supervision

some of the issues dealt with integrated design of fault tolerant control in real-time [14]. A *supervisor* is defined as a high-level decision maker that orchestrates switching among candidate controllers. The key idea is to construct a bank of controllers such that each controller in bank takes care of a particular fault. Therefore, this approach has its limitation in the sense that it can only handle a particular class of fault. However, the novelty of the approach lies in the efficient treatment of that particular class of fault *in real-time*. The architectural overview is shown in Fig. 2. In the analysis and development (AD) phase, it has been assumed that a finite set of controllers

$$\mathbf{C} = \{C_1, C_2, ..., C_K\} \tag{1}$$

is constructed in such a way so that in every situation either healthy or any faulty mode of the plant, there is at least one controller, or certain combination of controllers in that set which has the appropriate control action and is able to satisfy control objectives. It is the job of supervisor that selects a suitable configuration of controllers (at least one or more than one) depending on the type of supervision that seems more promising based on the available information. Therefore, it can be treated as a finite dimensional convex (or quasiconvex) optimization problem, i.e., the search of a corrective controller would be done in a finite set.

The motivation of logic-based switching control is two-fold:

- In the control of systems where traditional methodologies based on a single controller do not provide satisfactory performance. This scenario is often handled by periodic or continuous switching between two or more controllers. Moreover, the number of switches might not be restricted in this case.
- In the control of systems where a single controller can achieve the desired performance subjected to the presence of right corrective controller in the bank. Therefore, restricted number of switches is ensured.

The former motivation is very sensitive to instability. Thus, it demands rigorous stability analysis at the switching time as well as at the overall stability of the system. On the other side, the advantage is that if there is no stabilizing controller present in the bank then continuous switching between destabilizing controllers can also achieve the performance objectives [6]. However, this case should also be pre-considered in AD phase. While in later approach, depending on the type of switching logic one can guarantee the closed-loop stability of system. Therefore, we based our study on the second motivation in this paper. A supervision-based system is mainly categorized into three types:

A. Pre-routed supervision

In this type of supervision, a dedicated order of switches is designed in such a way the system satisfy certain performance specifications. The switching may stop in finite time [6] or it could be continuous i.e. periodic [13] depending on the control objectives. A switching order is illustrated in Fig. 3(a). In continuous case, the stability due to finite switching is not possible, thus it is guaranteed by carefully selecting a common or multiple Lyapunov function for each mode such that the Lyapunov function should be monotone non-increasing in time. However, this function can only be constructed if the plant knowledge is available.

One of the drawbacks of utilizing pre-routed switching is that each controller has to be tested in loop after another in a pre-defined sequence. Thus, the effectiveness of this approach is generally argued in case the set **C** is large.

B. Estimator-based Supervision

Estimator-based supervision (EBS) is also known as indirect supervision as the switching decision depends on the reconstructed output of the current plant-working mode. It is true that to build a controller bank, prior plant information is necessary but in this approach, that information is required in real-time as well. The supervisor block shown in Fig.3(b) is made-up of a set of estimators, followed by performance evaluation, and a switching logic scheme. Each estimator reconstructs the actual plant output in either one of the healthy or faulty working modes. Its performance is evaluated by computing a norm of the output estimation error, and the estimator that yields the smallest performance index corresponds to the present working mode. Consequently, corresponding controller to the estimator that yields the smallest performance index is applied to the process.

The drawback of this approach lies in the use of nominal set of plant-models in real-time that can have a significant impact on switching logic. This mainly occurs due to the estimation delay, model uncertainties etc. The approach works on the same principle as *Certainty Equivalence* since the controller is selected based on the current estimates. We will discuss another type of supervision, *viz.* Performance-based supervision, in next section.

III. Performance-based Supervision

The key idea of performance-based supervision (PBS) is to keep the controller while the observed performance satisfies the control objectives. When the performance of current controller becomes unacceptable, then switch to controller that leads to best expected performance. This performance is solely based on the available data in real-time. PBS system includes three main ingredients that does not incorporate any kind (estimation/nominal) of plant knowledge in real-time.

Assumption :

Given a set of candidate controllers: $\mathbf{C} = \{C_k : k \in K\}$

Performance monitor :

$J_k \equiv$ measure of the expected performance of "off-the-shelf" controllers C_k inferred from past data

Decision logic :

- If J_k is acceptable then keeps the current controller,
- If J_k is unacceptable then switch to controller $C_{\hat{k}}$ that corresponds to deliver best $J_{\hat{k}}$

Note k and $\hat{k}$ are indices of current and off-the-shelf controller respectively.

A. Evaluation of performance

To analyze the performance of any controller in bank, that controller should be tested in loop. Unfortunately, in real-time environment, it is not possible to ensure the stability employing this methodology. Safonov et.al. [11] suggests an approach to evaluate the performance of candidate controllers by generating a *fictitious (or virtual) reference signal*. Since last decade, this tool becomes significant in inferring the performance of "off-the-shelf'" controllers in closed loop [3], [4], [8].

Definition 1: $\mathfrak{L}_e^\infty$ denotes the space of trajectories which are bounded for finite time, i.e.

$$\mathfrak{L}_e^\infty = \{ f : \mathbb{R}^+ \to \mathbb{R} / \sup_{t>\tau} |f(\tau)| < \infty, \forall t \in \mathbb{R}^+ \}$$

Mathematically, fictitious reference signal is generated by

$$\tilde{r}_k(t) = C_k^{-1} u(t) + y(t) \tag{2}$$

This helps in analyzing the behavior of all controllers in the bank using a cost function given by:

$$J_k = \frac{\|u\|_\tau^2 + \|\tilde{e}_k\|_\tau^2}{\|\tilde{r}_k\|_\tau^2 + \kappa} \tag{3}$$

where $\tilde{e}_k(t) = y(t) - \tilde{r}_k(t)$, κ is an arbitrary positive constant used to prevent division by zero, and $\| \bullet \|$ represents the norm of time trajectory.

Remark 1: All trajectories belong to $\mathfrak{L}_e^\infty$ can grow exponentially, even the cost function as well therefore belong to this space.

So from (3), we see that no plant knowledge is required to access the performance of candidate controllers. The only thing known about the plant is that it generated the experimental data $\mathscr{D}$ on some time interval τ, i.e.

$$\mathscr{D} = \{(u,y) \in \mathfrak{L}_e^\infty : y(t) = p(t) \otimes u(t) \text{ on } [0,\tau]\} \tag{4}$$

where $\otimes$ is the convolution operator and $p(t)$ is the impulse response of plant. The cost function is the main component in these ingredients that judge the undesired behavior in closed loop.

Definition 2: Let r denote the input and $\mathscr{D}$ denotes the resulting plant data collected with C_k as the current controller. The pair $(J_k, \mathbf{C}), \forall k \in K$ is said to be *cost-detectable* if, without any assumptions on the plant and for every $C_k \in \mathbf{C}$, stability of the closed-loop system is falsified/unfalsified by the pair $(r, \mathscr{D})$.

Therefore, $J_k, \forall k \in K$ must satisfy cost-detectability such that it can reliably detect any stability/instability in the closed-loop. Cost-detectability is completely determined from the knowledge of cost function and candidate controllers, without reference to plant.

Remark 2: In formulation of cost function, the only necessary condition imposed on it is of cost-detectability that can only influence the decision about the current controller in loop. Therefore, a destabilizing controller can also be selected following this phenomenon.

Remark 3: The insertions of destabilizing controller definitely lead to increase in the value of cost function. However, this phenomenon does not conflict with the convexity of problem . It is the final stabilizing controller that determines the convexity. Thus, it still maintains even if the value of cost function increases after the insertion of destabilizing controller.

1) Norm evaluation: Equation (3) is determined by evaluating the norm of trajectories that defines the behavior of system. Here, we will briefly discuss the possible ways to evaluate those norms in real time. Since we mentioned that the cost function is determined based on the previous data collected during the time interval τ, there are three ways to tackle this:

a) Static window (fixed τ): This type of evaluation is generally used in fixed dwell time switching logic. At every τ duration, cost function is reset so that it does not have any memory for $t < \tau$. This window is chosen in such a way so that the dynamics of system settles down and satisfies the control objective.

b) Cumulative norm: In this method, we do not define a certain window, instead the norm of all signals are evaluated on the whole time axis starting from zero up to the current time. Therefore, the collected data until the current time has its impact on cost function. This results in non-decreasing behavior of cost function. Thus, for a stable system, the cost function might reach to infinity but it will be bounded. Cumulative evaluation of norm is generally used in Hysteresis switching logic.

c) Dynamic window (sliding fixed τ): In this method, we define a window of length τ in such a way so that the cost function has a monotone non-increasing behavior while the stabilizing controller is in the loop. This assumption is less restrictive in comparison to first method. The major advantage is its effectiveness to monitor the rate of variation of cost function.

Remark 4: The rate of variation aims to analyze the

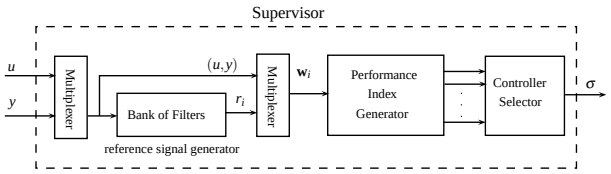

Fig. 4. Structure of reconfiguration mechanism

monotonic behavior of cost functional. Since the insertion of destabilizing controller cannot be prevented, our objective is to prevent this insertion while the stabilizing controller is working in the loop.

2) Impact of faults: Here we do not deal with the modeling of faults explicitly. Since we are considering a model-free approach to FTC, therefore modeling of faults is trivial. On the contrary, we consider fault as an unknown behavior that commute the system behavior in such a way it no longer satisfies the control objective. Many researchers studied the problem of FTC by considering an additive faults as an additive unknown signal to state or input. See a bibliography note [14] and references therein. Interestingly, in closed loop environment, certain types of fault (i.e. step etc.) do not inject any instability unless the saturation limits are violated.

Therefore, any impact of fault is considered as an unpermitted behavior that violates the desired behavior and a corrective controller for that faulty behavior is present in the bank. Thanks to cost-detectability with which it is possible to detect any instability occurring in the loop. Thus, the main aim is to construct a supervisor or reconfiguration mechanism that selects the right controller based only on the trajectories generated by the system in real-time.

IV. Switching logic

The main job of supervisor is to perform *when-which* task, i.e. when to substitute the acting controller and which controller to switch on. This significant task is executed by a switching logic that selects a suitable controller for the current plant-working mode (Fig. 4). It can also be termed as scheduling-routing task that is carried out in real-time by monitoring cost functional, each related to a different controller. Thanks to the virtual reference tool, this task is achieved without switching on all candidate controllers in the feedback loop in order to assess their performance with the plant.

The switching logic plays a significant role to maintain the convexity of problem such that it is able to select the right controller and does not switch to destabilizing controller if it is already connected into the loop. Here we will discuss two popularly known performance based switching logic and their limitations. Later covering those limitations, we propose a novel switching logic.

A. Fixed dwell time

Dwell time switching is generally known as slow switching that introduce a number $\tau_D > 0$ such that the switching times $t_1, t_2, \ldots$ satisfy the inequality $t_{i+1} - t_i \geq \tau_D$ for all i. This

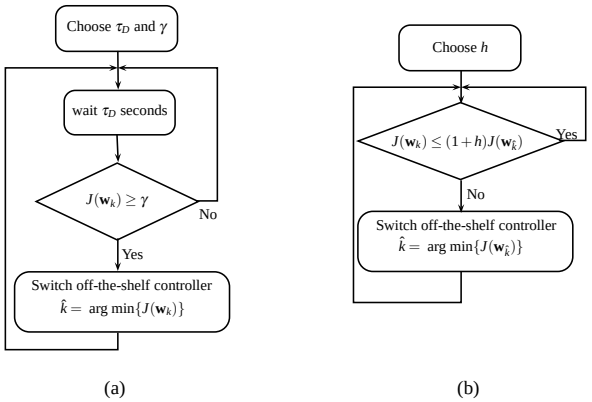

Fig. 5. (a) Fixed dwell time; (b) Adaptive dwell time

number τ_D is usually called the *dwell time*. A performance margin γ is chosen such that the controller is falsified if it satisfies the inequality $J_k > \gamma$. On falsification, a new controller is switched into the loop according to the rule $\hat{k} = \arg\min\{J(\mathbf{w}_k)\}$, where $\hat{k} \in K$ and $\mathbf{w}_k = (r_k, u, y)$. The computer program is illustrated in Fig. 5(a).

1) Issues in fixed dwell time: The dwell time is chosen in such a way so that the dynamics of closed loop settles down and the cost functional for the switched system satisfies the inequality $J_k \leq \gamma$. It is a well known fact that choosing a large τ_D assures the asymptotic stability assuming the switched controller is stabilizing. Since the switching does not guarantee the selection of right controller in one shot, a wrong controller can also be selected, thus, a necessary lower bound is required. Liberzon et. al. [6] developed a lower bound on dwell time considering exponential decay of signals. The formulation of bound requires the knowledge of switched system, and most important, the initial state. Information about the switched system can be assumed in AD phase as well however, the knowledge of initial state is problematic that cannot be pre-determined.

Consider a scenario analyzed by a time-map as shown in Fig. 6. We installed three controllers in the bank: one initial working (in healthy plant-working mode), one destabilizing, and one stabilizing controller. Since the selection of right controller is not guaranteed after the occurrence of fault, the stabilizing controller is selected after two switchings. In addition, after a first switch the destabilizing controller is falsifiable in $\theta < \tau_D$ duration. Adopting the flow in Fig. 5(a) the interval between two nocks symbolizes the dwell time where the switching can occurs. Here t_f and t_d represents the fault occurrence time and its detection time respectively. Starting from the origin, the current controller is falsified at t_d but it remains in the loop until time $2\tau_D$. On switching to destabilizing controller, suppose it is falsifiable after θ duration. Finally, the right controller is switched into the loop on next switch. Therefore, the duration for which the loop is being operated by destabilizing controllers after the detection of fault is

$$t_{des} = 3\tau_D - t_d$$

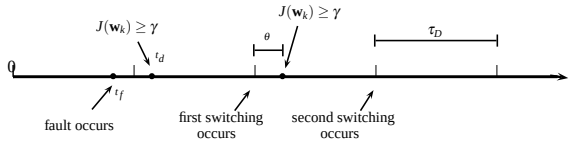

Fig. 6. Time-map

where $t_d = \tau_D + \delta$. Suppose δ is very less than dwell time, thus

$$\delta \lll \tau_D \Rightarrow t_{des} \cong 2\tau_D$$

The above observation actually reveals a significant disadvantage of utilizing dwell-time switching. With a prespecified dwell time, the performance of system due to presence of destabilizing controllers in loop for a long time might deteriorate to an unacceptable level before the next switch occurs.

B. Adaptive dwell time

A different way to slow the switching down is by means of hysteresis. We generate the switching signal σ by means of a so called *hysteresis switching logic*, illustrated via the computer-like diagram in Fig. 5(b). Fix a positive number h called the hysteresis constant. Switch off-the-shelf controller when it satisfy the inequality $J(\mathbf{w}_k) \leq (1+h)J(\mathbf{w}_{\hat{k}})$. This hysteresis constant plays the role of non-zero dwell time. In this switching logic, we do not fix any performance threshold. However, the switching occurs if the value of cost function corresponding to off-the-shelf controller becomes *significantly* smaller than the cost function for the current controller. Thus, the hysteresis constant makes an adaptive lower bound on dwell time.

Unlike to fixed dwell time switching logic, here we employed cumulative norm based method to evaluate the cost functional because of the unavailability of fixed time window. Thus, as mentioned before the cost function $J(\mathbf{w}_k), \forall k \in K$ has always monotonic non-decreasing behavior. Moreover, it is bounded for stabilizing controller.

1) Switching from stabilizing controller: It is true when a fault occurs the current controller becomes destabilizing and one cannot guarantee to switch to right controller directly. Here we will show that in the initial learning phase, if at once the stabilizing controller has the probability to switch in the loop it can switch to other destabilizing controller as well. Let us analyze this scenario. Given a cost function (3), k and $\hat{k}$ are the indices of stabilizing current controller in loop and destabilizing "off-the-shelf" controller with one right-hand-pole (RHP). When C_k is working in the loop, $\|u\|$ and $\|y\|$ are bounded that results in bounded $\|\tilde{r}_k\|$ by (2). Thus collectively $\mathbf{w}_k$ makes a bound on J_k. For off-the-shelf controller $C_{\hat{k}}$, if we prove a bound on $\|\tilde{r}_{\hat{k}}\|$ then $J_{\hat{k}}$ is also bounded. Since $C_{\hat{k}}$ is unstable with one RHP, its inverse $C_{\hat{k}}^{-1}$ is a stable transfer function. Moreover, C_k is working in the loop, thus $(\|u\|, \|y\|)$ is bounded. Following (2), $\|\tilde{r}_{\hat{k}}\|$ becomes bounded as well. Therefore, $J_{\hat{k}}$ is bounded too. This

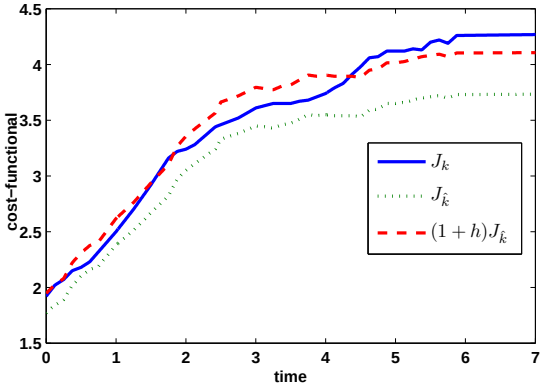

Fig. 7. Cost-functional trajectories

case is illustrated by a simulation for an unknown system in Fig. 7. In figure, we see that the switching to destabilizing controller can occur following the hysteresis switching logic.

Since the norm of all trajectories is evaluated considering the past data starting from origin, it is a fact that after a learning process, the final controller will be stabilizing controller only. However, switching to destabilizing controller when stabilizing controller is already present can devolve the overall performance of system.

C. Novel switching logic

Reacting to the issues discussed above, here we propose a switching logic scheme for real-time FTC. The computer program is shown in Fig. 8. Choose a dynamics settling time (DST) $\tau_{ds} < \tau_D$ and like fixed dwell time approach, we set a performance threshold γ. Since we employ sliding window (of length τ) norm evaluation of time-governed trajectories, the role of τ is to ensure the quantized monotonic behavior of cost functional. The length of τ_{ds} must be chosen in such a way that for a stabilizing current controller it satisfies the inequality

$$\frac{J(\mathbf{w}_k(t)) - J(\mathbf{w}_k(t-\tau_{ds}))}{\tau_{ds}} \leq 0 \qquad (5)$$

where $\tau_{ds} \neq \tau$. Since our one of the main objectives is to guarantee that once the stabilizing controller is switched in the loop, it should not make a transit to destabilizing one even in the learning phase as well. If after switching, stabilizing controller is selected then at t^+ there might be two cases: either $J_k > \gamma$ or $J_k \leq \gamma$, where t^+ denotes the time just after switching. In case it is less than γ, it remains bounded to γ. On contrary, if it satisfies (5) then it becomes bounded as $t \to \infty$. It is justified as in AD phase we already determined that if any particular controller is the correct controller for the current plant-working mode then it make a bound $J_k \leq \gamma$ irrespective of the initial conditions. Here we are considering only one occurrence of fault at a time, so if a controller is falsified then that controller will not be test for unfalsification.

1) Controller falsification: The switching logic falsifies any current controller if one of the conditions is true:

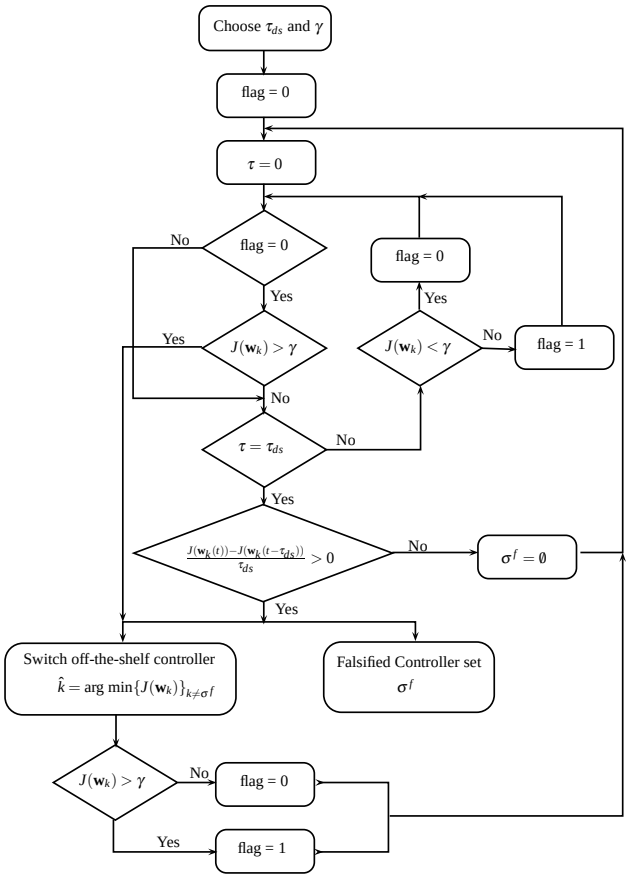

Fig. 8. Proposed switching logic

- In case $J_k \leq \gamma$ before the occurrence of fault, the controller is falsified if $J_k > \gamma$.
- In case $J_k \leq \gamma$ just after the switching (i.e. t^+), the controller is falsified if $J_k > \gamma$.
- In case $J_k > \gamma$ just after the switching (i.e. t^+), the controller is falsified if J_k has monotonic increasing behavior and does not satisfy (5).

The first two conditions are justified. Suppose at t^+, corrective controller is selected. Prior to its selection, it might be possible that $J_{\hat{k}} > \gamma$ as its behavior is being inferred by another destabilizing controller in the loop. Therefore, if we do not include third condition then the supervisor can reject the right controller as well, and thus leads to instability.

2) Stability aspects: Narendra et.al. [9] prove that the overall system will be globally stable for any switching sequence, provided that the intervals between successive switches have a nonzero lower bound, which can be chosen to be arbitrarily small. This result is intuitively reasonable since it is known that each one of the K controllers C_k results in global stability if used alone. The validation does not follow directly, however, since switching between stabilizing controllers need not result in a stable system. One can guarantee this type of stability if we make a bound on states at switching instants. This can also be achieved by employing a bumpless transfer technique.

In fixed dwell time approach, τ_D make a bound while hysteresis constant ensure the non-zero dwell time in hysteresis switching. For the proposed switching mechanism, since all trajectories can grow exponentially (Remark 1), it develops a non-zero dwell time when $J_k < \gamma$ while τ_{ds} make a lower bound when $J_k > \gamma$. The prevention of switching wrong controller also affects the stability of overall system. Corresponding to fixed dwell time approach, the destabilizing controller following the proposed logic can either stay for θ-duration or τ_{ds}-duration where, θ or $\tau_{ds} \ll (t_{des} \approxeq 2\tau_D)$. Moreover, it is guaranteed that once the correct controller is selected then it will not make a transition.

V. Conclusion

In this paper, we have discussed a supervisory-based approach to achieve model-free FTC. This approach is very useful to deal with integrated design issues in FTC. Here we construct a bank of controllers assuming that one of them is the correct stabilizing for the current plant-working mode. Supervisory-based approach is regulated by a switching logic, *viz.* fixed dwell time and adaptive dwell time (hysteresis switching). First, we have shown few shortcomings of these switching mechanisms and then proposed a new switching logic that effectively handles those limitations. Further investigation is required to determine the restrictions of this novel logic.

References

[1] M. Blanke, M. Kinnaert, M. Staroswiecki, and J. Lunze. *Diagnosis and Fault tolerant control*. Springer-Verlag, 2003.

[2] J. Bošković and R. Mehra. A multiple model-based reconfigurable flight control system design. In *37th IEEEConference onDecision and Control (CDC98)*, 1998.

[3] M.C. Campi, A. Lecchini, and S.M. Savaresi. Virtual reference feedback tuning: a direct method for the design of feedback controllers. *Automatica*, 38:1337–1346, 2002.

[4] E. G. Collins, Y. Zhao, and R. Millett. A genetic search approach to unfalsified pi control design for a weigh belt feeder. *Int. J. Adapt. Control Signal Process*, 15:519–534, 2001.

[5] S. Kanev and M. Verhaegen. A bank of reconfigurable lqg controlles for linear systems subjected to failures. In *Proc. 39th CDC*, pages 3684–3689, 2000.

[6] D. Liberzon and A. Morse. Basic problems in stability and design of switched systems. *IEEE Control Systems Magazine*, 19(5):59–70, 1999.

[7] S. Médar, P. Charbonnaud, and F. Noureddine. Active fault accommodation of a three tank system via switching control. In *Proceedings of the 15th TriennialWorld Congress of IFAC*, 2002.

[8] E. Mosca and T. Agnoloni. Inference of candidate loop performance and data filtering for switching supervisory control. *Automatica*, 37:527–534, 2001.

[9] K. S. Narendra and J. Balakrishnan. Improving transient response of adaptive control systems using multiple models and switching. *IEEE Transactions On Automatic Control*, 39:1861–1866, 1994.

[10] R. J. Patton. Fault-tolerant control systems: The 1997 situation. In *Proc. 3rd IFAC symposium on fault detection, supervision and safety for technical processes*, pages 1033–1055, 1997.

[11] M.G. Safonov and T-C. Tsao. The unfalsified control concept and learning. *IEEE Transactions on Automatic Control*, 42(6):843–847, 1997.

[12] J.J. Yamé and D. Sauter. A real-time model-free reconfiguration mechanism for fault-tolerance: Application to a hydraulic process. In *Proc. 10th International Conference on Control, Automation, Robotics and Vision(ICARCV)*, pages 91–96, 2008.

[13] H. Yang, V. Cocquempot, and B. Jiang. Supervisory fault tolerant control design via switched system approach. In *IEEE Conference on Control and Fault-Tolerant Systems, Systol'10*, 2010.

[14] Y. Zhang and J. Jiang. Bibliographical review on reconfigurable fault-tolerant control systems. *Annual Reviews in Control*, 32:229–252, 2008.