Kalpa Publications in Computing

Volume 4, 2018, Pages 110–123

28th International Workshop on
Principles of Diagnosis (DX'17)

Comparing Switching vs Mixing Model-Predictive Controllers for Robust Fault-Tolerant Control

Yves Sohége[1] and Gregory Provan[2]

[1] Department of Computer Science,
University College Cork,
Ireland
yves.sohege@insight-centre.org
[2] Department of Computer Science,
University College Cork,
Ireland
g.provan@cs.ucc.ie

Abstract

We conduct a comparative study between two approaches for combining signals from several Model-Predictive Controllers (MPCs) designed for different fault scenarios. The first is MPC switching where a switch dictates which of the MPC controllers is currently active. The second is MPC mixing where all MPCs are running concurrently and their outputs are blended in proportion to the current estimate of fault state. We demonstrate results using a gravity drained multi-tank system. Our empirical results show that the mixing approach responds more quickly to faults than the switching approach. Further, we show that the speed and accuracy of fault isolation has a critical impact on fault tolerance.

1 Introduction

A fault-tolerant control system (FTC) is a system that is able to identify and recover from system faults and continue operation as normal or to maintain stability to a desired level of overall performance. The need for such systems in areas such as aerospace and industrial processes, where safety and reliability is paramount, has motivated significant research into the design and optimisation of these systems. Examples of such systems can be seen in aircraft [1], spacecraft [2], autonomous quad-copter systems [3], power plants, chemical reactors and ground vehicles, among many more.

FTCs can be divided into two types, passive and active. Passive fault tolerant controllers are designed off-line against predefined models for certain operating conditions and have no ability to react to unanticipated faults. Passive FTC enables fast adaptation to faults, within the predefined operating conditions. Active FTC uses on-line data to reconfigure the controller to stabilise the plant. They have a built in fault detection mechanism which allows them to react to pre-defined faults, but makes the controller reliant on the accuracy on this fault-detection unit. For a comprehensive study between the two approaches see [4].

Both active and passive FTC rely, to varying degrees, on specifying the space of faults that the system will encounter. For passive FTC, approaches such as mixing of controllers tuned to nominal

and failure modes are used to maintain system stability, e.g., [3]. Analogously, active FTC can rely on being able to detect pre-specified faults, such as using a bank of observers, with each observer tuned to a particular fault, e.g. [5]. For complex systems, it is impossible to pre-specify all faults, since there are too many fault combinations to consider, and it may be impossible to know all possible faults as *a priori*. As a consequence, it is imperative that a system designer understand the space of possible faults and their impact on a system. Creating a controller that can interpolate its control law from predefined edges of the fault space allows it to adapt to unknown/undefined faults by computing the optimal control policy for this fault state. Very little work has been conducted on exploring the space of faults and their impact on active versus passive FTC.

FTC extends traditional controllers with a method of detecting and tolerating faults. The main purpose of a control system is reference tracking and many different types of controllers including Model Predictive Controllers (MPCs), Proportional Integral Derivative (PID), Feedback Linearisation (FL), etc., have been designed for this purpose. We adopt an MPC approach, which is based on numerical optimization and is categorised as an optimal control strategy [6]. At every time step the MPC solves an optimization problem to obtain the optimal control input for the current system state. An MPC can be tuned to handle specific error models to make it a FTC. Several MPCs, each designed to handle a separate fault scenarios, can be combined to create a controller that is more robust than any individual MPC can be at maintaining the stability of the system.

We focus on hybrid systems control, where we assume a system that can operate in a set of N modes, with a different set of dynamics for each mode. We assume that we have a subset of N_f failure modes, with $N - N_f$ nominal modes. For each of the nominal modes we pre-compute a control setting. We also need to tolerate when certain failure modes occur, which is the focus of this article.

In this article we conduct a comparative study between two approaches for combining several MPCs designed for different fault scenarios. The first is MPC switching where a switch dictates which of the MPC controllers is currently active. The switching signal is generated by the Fault Detection Unit (FDU) so once a fault is identified the corresponding controller can be activated to handle this fault. The second is MPC mixing where all MPCs are running concurrently and their outputs are blended in proportion to the current fault state estimate generated by the FDU. We demonstrate results using a gravity drained multi-tank system.

2 Related Work

This work builds on extensive prior work in fault-tolerant control (FTC) and Fault Detection and Isolation (FDI).

In the area of FTC [7], a significant body of work has been developed and applied to real-world systems. [8] presents a recent overview of FTC, and [9] presents FTC with relation of system safety. Traditional methods for FTC employ a bank of observers coupled with dedicated controllers, and perform discrete switching. This approach enables designers to tune the system to dedicated faults, but the speed of the system hinges on the speed of FDI. More recent approaches use mixing controllers, e.g., [10, 11], which blend the outputs of multiple controllers and are less reliant on FDI.

Our work builds on research into the use of weighted multiple models for adaptive estimation and control. Early work, e.g., [12, 13] used multiple Kalman filter-based models to improve the accuracy of the state estimate in estimation and control problems. This early work was then extended to applications, e.g., real-world problems [14] and fault detection [15].

Blended or weighted multiple model adaptive control (WMMAC) has been used to make control more robust, e.g., [16, 17]. The use of multiple models within the context of MPC, denoted Multiple Models Predictive Control (MMPC), has been addressed by [18]. Beyond this focus on robustness, there have been few applications of multiple blended models to FTC outside of [3]. Our work is novel in its

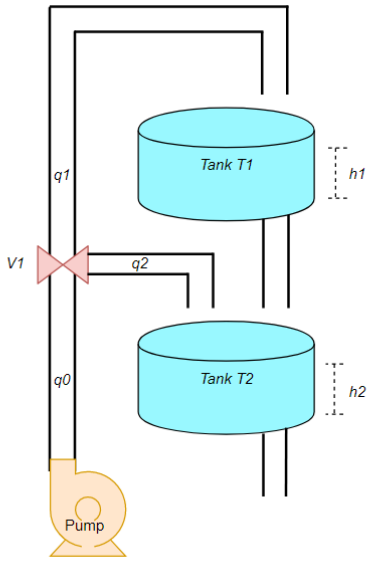

Figure 1: Two tank gravity drained system.

investigation of WMMAC for FTC. We are not aware of prior work that empirically compared active and passive FTC approaches, or that compared the impact of faults on switching vs. mixing supervisory FTC.

3 Running Example

We illustrate our approach using the two-tank system shown in Fig. 1. Tank T_i has area A_i and inflow q_{i-1}, for $i = 1, 2$. This system connects together two tanks, with a valve V_1 regulating the proportion of the flow q_0 and q_1 into tanks 1 and 2 respectively. The pump flow q_0 and value V_1 can be set between 0 and 1. A valve setting of 0 will cause the entire flow to go into tank 1 and a pump setting of 0 will turn the pump off completely. The control objective is to modify the pump and valve settings to maintain a reference height in both tanks. Water from tank 1 will flow into tank 2 at a constant rate and water will flow out of tank 2 at a constant rate. Sensors give us the current water level h_1 and h_2 in tank 1 and tank 2, respectively.
Both tanks T_1 and T_2 get filled from a pump, with measured flow q_0. The proportions of the flow into each tank are controlled by V_1, hence, our control inputs are $u = \{q_0, V_1\}$.

We assume that we do not directly measure any flows other than the inlet flows q_0 and valve setting V_1. As a consequence, we use the tank heights as a proxy for deriving flows through the two-tank system. We can control the valve settings where we assume a continuous-valued setting ranging over $[0, 1]$.

We can use basic physics to create a model of the 2-tank system by observing mass-balance requirements on each of the tanks, where tank T_i has area A_i, for $i = 1, 2$. The state equations are given by:

$$A_i \frac{dh_i}{dt} = q_{i-1} - q_i,$$

where q_{i-1} denotes the flow into tank i, and q_i denotes the flow out of tank i.

According to Torricelli's Law, flow q_i out of tank i, with liquid level h_i, into tank j, is given by:

$$q_i = \gamma \sqrt{2gh_i}, \tag{1}$$

where the coefficient γ is used to model the area of the drainage hole and its friction factor through the hole, and g is gravitational acceleration.

We can use equation 1 to derive the following equations, since the inflow into tank 2 equals the outflow of tank 1:

$$\begin{aligned} \dot{h}_1 &= q_0 - c_1 \cdot \sqrt{h_1} \\ \dot{h}_2 &= c_1 \cdot \sqrt{h_1} - c_2 \cdot \sqrt{h_2}, \end{aligned} \tag{2}$$

where the constants c_1, c_2 summarize the system parameters representing cross-sectional areas, friction factors, gravity, etc. Consequently, the parameter set is given by $\Theta = \{c_1, c_2, c_3, g\}$. We assume that we can measure the height of liquid in each tank. The set of state variables is $\{h_1,\ h_2,\ V_1, q_0\}$, and the set of controllable variables is $\{q_0, V_1\}$.

To formally transform a non-linear system into a linear one, we need to use techniques like small signal linearization or perturbation theory [19, 20]. For example, in small signal linearization, an equilibrium point x_0 of a fault-free non-linear function is first identified, about which the perturbed non-linear function is expanded: $x = x_0 + \delta x$. Then we can use a Taylor series expansion, neglecting the higher order terms, to obtain the linear function.

4 Fault Tolerant Control Schemes

A key aspect of FTC is the ability to adopt a control law that compensates for a system fault. Because having a single nominal-mode plant model does not allow us to simulate faulty behaviours, modern FTC uses multiple models that attempt to cover the space of failure behaviours [21]; many diagnostics approaches also use multiple models, e.g., [22].

However, in realistic problems, the dimension of the plant and of the unknown parameter vector are large. As a consequence, the number M of models needed to satisfy stability and controllability conditions given faults becomes prohibitively large, since M increases exponentially with the dimension of the unknown parameter vector. In addition, switching results in discontinuous control signals, and identification and control are coupled [21].

In the following, we study the impact of various design choices on the FTC schemes for two of the most advanced approaches, switching MPCs and mixing MPCs.

4.1 State-Space Model

Consider a linear time-invariant (LTI) discrete-time system of the form:

$$\begin{aligned} \dot{\boldsymbol{x}}(k+1) &= A_j \boldsymbol{x}(k) + B_j \boldsymbol{u}(k) + \boldsymbol{w} \\ \boldsymbol{y}(k) &= C\boldsymbol{x}(k), \end{aligned} \tag{3}$$

where $\boldsymbol{x},\ \boldsymbol{u},\ \boldsymbol{y}$ are the state, control and observation vectors, respectively, and $A_j,\ B_j$ are state matrices for mode j. Finally, C is the observation matrix. We assume that the system can operate in N possible modes $\mathcal{M} = \{\mathcal{M}_1, ..., \mathcal{M}_N\}$, with individual dynamics for mode j $(j = 1, \cdots, N)$, captured by the matrices $A_j,\ B_j$.

Since the hybrid system dynamics use a model for each distinct mode, we must also have a unique control law for each distinct mode. More precisely, if we have N modes, then we must have a set $\Lambda = \{\lambda_1, \cdots, \lambda_N\}$ of controllers, with a switching algebra to define when to switch from λ_i to λ_j for $i \neq j,\ i, j \in \{1, ..., N\}$.

We thus characterize the system's N modes by (i) a set of N controllers Λ, and (ii) a set of N parameter settings, i.e., we can partition the system's parameter space Ω into a set of sub-spaces $\{\Omega_1, \Omega_2, \cdots \Omega_N\}$, such that sub-space Ω_i corresponds to controller C_i, for $i = 1, ..., N$. In other words, we assume that we can define the parameter setting sub-spaces such that there exists a controller Λ_i that can guarantee a desired performance level for Ω_i, $i = 1, ..., N$.

4.1.1 Observer-Based Control

We assume that we control the system (in mode i) using a state (Luenberger) observer based on a state-space model with observer matrix L. Using the observed system with observed state and measurement, $\hat{\boldsymbol{x}}(k) \in \mathbb{R}^n$ and $\hat{\boldsymbol{y}}(k) \in \mathbb{R}^p$, respectively:

$$\begin{aligned} \hat{\boldsymbol{x}}(k+1) &= A_i\hat{\boldsymbol{x}}(k) + B_i\boldsymbol{u}(k) + \boldsymbol{w}(k); \\ \hat{\boldsymbol{y}}(k) &= C_i\hat{\boldsymbol{x}}(k); \end{aligned} \tag{4}$$

we obtain the observer equations:

$$\begin{aligned} \hat{\boldsymbol{x}}(k+1) &= A_i\hat{\boldsymbol{x}}(k) + B_i\boldsymbol{u}(k) + L_i(\boldsymbol{y}(k) - C_i\hat{\boldsymbol{x}}(k)); \\ \boldsymbol{r}(k) &= \boldsymbol{y}(k) - C_i\hat{\boldsymbol{x}}(k); \\ \boldsymbol{u}(k) &= -K_i\hat{\boldsymbol{x}}(k), \end{aligned}$$

where $\boldsymbol{r}(k) \in \mathbb{R}^p$ is the residual $\parallel \boldsymbol{y}(k) - \hat{\boldsymbol{y}}(k) \parallel$ for some norm $\parallel \cdot \parallel$. We tune the control matrix $K_i \in \mathbb{R}^{l \times p}$ and observer matrix $L_i \in \mathbb{R}^{n \times p}$ so that the closed-loop system and error dynamics are stable. We can rewrite these equations such that we obtain

$$\hat{\boldsymbol{x}}(k+1) = (A_i - B_iK_i)\hat{\boldsymbol{x}}(k) + L_i(\boldsymbol{y}(k) - C_i\hat{\boldsymbol{x}}(k)) \tag{5}$$

by substituting $-K_i\hat{\boldsymbol{x}}(k)$ for $\boldsymbol{u}(k)$ into the state equation.

4.1.2 Observer-Based Diagnosis

We partition our modes into nominal and fault modes, $\mathcal{M}_\emptyset$ and $\mathcal{M}_f$, respectively, such that $|\mathcal{M}_\emptyset| + |\mathcal{M}_f| = N$. We model (actuator) faults using a multiplicative fault model with parameter $0 \leq \gamma \leq 1$, where $\gamma = 0$ corresponds to nominal function and $\gamma = 1$ to total failure. For every failure mode $\mathcal{M}_i \in \mathcal{M}_f$ we have a corresponding fault parameter γ_i. Hence we obtain the state variable equation for failure mode j:

$$\dot{\boldsymbol{x}}(k+1) = A_j\boldsymbol{x}(k) + B_j(1 - \gamma_j)\boldsymbol{u}(k) + \boldsymbol{w}(k). \tag{6}$$

We use the residuals for mode identification. In this article we create a residual for every system mode, i.e., our observer indicates that mode i is active if the corresponding residual $r_i > \epsilon$ for some tunable threshold $\epsilon > 0$. If we have perfect tuning and no noise, then we have a monotonic relationship between γ_i and $r_i > \epsilon$: presence of fault i always leads to a detectable residual r_i corresponding to that fault.

4.2 Switching Model-Predictive Control

Given that our system operates in multiple possible modes, we use a controller that can switch between these modes. Fig. 2 depicts a generic framework that enables control switching of various types. In this article we examine two switching behaviours: (1) discrete switching (where one control regime is only used at any time), and (2) mixed switching, where we use a proportional mixture of multiple controllers at once. This section provides an overview of these two FTC controllers.

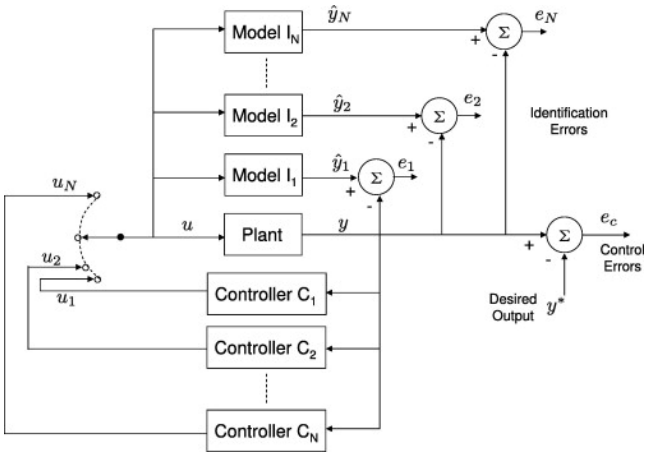

Figure 2: Use of multiple controllers for FTC

By using candidate controllers designed off-line, the switching architecture can use multiple controllers, either individually or in a mixed mode. The multi-controller is not only capable of generating any of the candidate control laws but also, by controller interpolation, a stable mix of candidate control laws. We can design the mixing controller to converge exponentially quickly to meet the control objective, provided certain conditions on the plant input are met [23].

The architecture, shown in Fig. 2, has two levels of control: (1) a low-level controller that generates finely-tuned candidate controls for each mode and (2) a high-level controller, called the supervisor, that influences the control by adjusting the low-level controller, typically by selecting or weighting candidate controllers, based on processed plant input/output data.

We now briefly outline the design of the supervisory controller that governs the switching behaviour.

4.2.1 Discrete Switching Controller

This section summarizes the MPC approach that we adopt. A traditional MPC controller includes a nominal operating point at which the plant model applies, such as the condition at which you linearize a non-linear model to obtain the LTI approximation. We generalize that operating point in terms of a parameter sub-space. When we consider a single fault, then we can define that sub-space in terms of the multiplicative fault parameter γ. Since the actuator effectiveness is in the range $0 \leq \gamma < 1$, we have *a priori* knowledge on the bounds of $\bar{\gamma}$:

$$\bar{\gamma} \in \Omega \subset (0, 1]. \tag{7}$$

We can then partition Ω into a set of sub-spaces $\{\Omega_1, \Omega_2, \cdots \Omega_N\}$, where $\Omega_i = [\bar{\gamma}_i^{min}, \bar{\gamma}_i^{max}]$ and $\bar{\gamma}_i^{min}$ and $\bar{\gamma}_i^{max}$ are the lower and upper bounds, respectively, of interval Ω_i. For multiple faults, we must consider the fault vector $\boldsymbol{\gamma} = \{\gamma_1, ..., \gamma_m\}$. We now define a multi-dimensional partition $\boldsymbol{\Omega}$ into a set of

sub-spaces $\{\mathbf{\Omega}_1, \mathbf{\Omega}_2, \cdots \mathbf{\Omega}_N\}$, where

$$\mathbf{\Omega}_i \subseteq \times_{j=1}^{m} \gamma_{ij},$$

and γ_{ij} denotes the i^{th} sub-space of the set of multiplicative fault parameter set indexed $j = 1, ..., m$. Given this partition, we design controller C_i to guarantee optimality when operating in sub-space $\mathbf{\Omega}_i$.

We then design a switching supervisor that, given an estimate of sub-space $\mathbf{\Omega}_i$, switches to the i^{th} controller.

We select the i^{th} controller if residual r_i is greatest, measured over all residuals that exceed a given threshold:

$$\Lambda_i(k+1) = \begin{cases} \Lambda_i(k) & \text{if } r_i < \epsilon_i \ \ \forall i \\ \Lambda_{i^*} | i^* = \text{argmax}_{r_i \in \hat{R}}\{r_i\} & \text{if } r_i \geq \epsilon_i \end{cases} \tag{8}$$

4.2.2 Mixing Controller

We aim to design a mixing scheme for regulating the state vector $\boldsymbol{x}$ to a chosen set-point, assuming the nominal system matrices A and B are known, but multiplicative actuation errors γ are unknown. Given the system in equation 3, sudden changes in γ will affect the system dynamics. In response, we tune the mixing scheme to blend the controllers appropriately. Given a set of controllers Λ_i, for $i = 1, ..., N$, each of which guarantees optimality when operating in sub-space $\mathbf{\Omega}_i$, we use a mixing supervisor for sub-optimal conditions. In other words, when sub-optimality occurs, i.e., parameters do not lie within some Ω_i, we must mix "appropriate" controllers.

In this article we use a linear combination of weighted controller inputs to restore stability given a pre-specified set of faults. In other words, given a set of N controllers $\Lambda = \{\Lambda_1, \cdots, \Lambda_N\}$ and corresponding probability distribution $\{\varphi_1, \cdots, \varphi_N\}$, our applied control is given by

$$\Lambda^* = \sum_{i=1}^{N} \varphi_i \Lambda_i. \tag{9}$$

We compute the distribution φ using the residuals for the N models, $r_i, \ i = 1, ..., N$:

$$\varphi_i : r_i \rightarrow [0, 1]. \tag{10}$$

We perform this mapping using the following three steps.

1. Discretization of Residuals We discretize the residual space for fault i into a set of intervals of the form

$$[0, \epsilon), [\epsilon, \epsilon + \delta), ..., [\epsilon + m\delta, r^{max}), [\geq r^{max}]$$

where ϵ and δ are appropriate thresholds and r^{max} is a maximal residual value that indicates the fault magnitude is significant.

2. Weight Assignment to Intervals Given these intervals, we compute a weight for the i^{th} residual interval as follows:

$$w_i(k+1) = \begin{cases} 0 & \text{if } r_i \in [0, \epsilon) \\ \alpha & \text{if } r_i \in [\epsilon, \epsilon + \delta) \\ \cdots & \cdots \\ k\alpha & \text{if } r_i \in [\epsilon, \epsilon + k\delta) \\ \cdots & \cdots \\ 1 & \text{if } r_i \geq r^{max} \end{cases} \tag{11}$$

where α is a [0,1] weight chosen by appropriate controller tuning, and $m \in \mathbb{Z}^+$.

3. Weight Normalization Given the weights, we need to convert them into a probability distribution so that our cumulative control signal (equation 9) remains stable. We assume that we start in a nominal mode, and our objective is to define a new blended controller given a residual $\boldsymbol{r}(k)$. We denote the current (nominal) controller at time step k as $\Lambda_j(k)$; we assume that this controller has probability $\varphi = 1$ assigned to a single mode. We denote alternative controllers as $\Lambda^* = \{\Lambda_l\}$, $l = 1, ..., N,\ l \neq j$. We denote our probability distribution for controllers $\Lambda_1(k), ..., \Lambda_N(k)$ using $\varphi(k) = \{\varphi_1(k), ..., \varphi_N(k)\}$. We compute a probability for time step $k+1$ from the weights through normalization. If all w_i are 0, then we maintain the current controller, $\Lambda_j(k)$; otherwise, if some $w_i > 0$, we update our weights as given below to generate a new blended controller:

$$\varphi_i(k+1) = \begin{cases} 1 - \mathcal{S} & \text{if } i = j \\ \frac{w_i}{\pi} & \text{if } j \neq i \end{cases} \tag{12}$$

where $\pi = \sum_i w_i$ and $\mathcal{S} = \sum_i \frac{w_i}{\pi}$.

Example:

Consider the running example of a two tank system described in Section 3. We have a single nominal mode Λ_1 and two failure modes Λ_2 and Λ_3, denoting pump and valve faults, respectively. Our weight intervals are $\{[0, .01),\ [.01, .02), [.02, .03), , \cdots, [\geq .1], \}$, i.e., $\epsilon = \delta = 0.01$ in equation 11. Assume that we are currently in the nominal mode and detect anomalous residuals for the pump in the intervals [.01,.02) and [.02,.03), corresponding to 1% error and 2% error, respectively. Taking a tuned value of $\alpha = 10$, that leads to the weights $w_2 = 0.1,\ w_3 = 0.2$. w_1 is calculated at $1 - \mathcal{S}$ where $\mathcal{S}$ is $\sum_i w_i$ and $i = [2, 3]$, which makes $w_1 = 0.7$. Then from Equation 9 we generate the new control signal to be:

$$\Lambda^*(k+1) = 0.7\Lambda_1(k) + 0.1\Lambda_2(k) + 0.2\Lambda_3(k) \tag{13}$$

4.3 Stability Properties

The weighting algorithm plays an important role in the control and stability properties of weighted multiple model control (WMMC) frameworks. The closed-loop stability of a WMMC system depends on three conditions [23]: (1) the model set includes the true model(s) of nominal operation of the plant (or the closest such models); (2) the weighting algorithm converges correctly; and (3) each local controller stabilizes its corresponding model. In addition, the convergence rate of a weighting algorithm will have effect on the transient process of the WMMC system.

Providing stability and optimality guarantees for controllers that use multiple models is known to be a difficult task, e.g., [17, 24]. The study of stability properties of switched and hybrid MPC systems has focussed on models defined using piecewise affine systems [25]. Proof of stability assumes that a supervisory controller would switch among low-level controllers with neighbouring, locally affine regions, such that each low-level controller is accurate within its own affine region [26].

Proving stability for FTC is beyond the scope of this article. Nevertheless, we can make a number of observations about this issue.

Incipient faults Incipient faults, which can be characterized in terms of gradual drift of a fault parameter γ, can be theoretically analysed using the piecewise affine system framework. In other words, using results from [25] we can define a nominal region Ω_{nom} with an adjacent fault region Ω_f such that blending of the controllers for the two regions in guaranteed to maintain stability and optimality of the control.

In this case, we can also show that mixing control will always outperform discrete switched control in terms of response times for fault tolerance. The mixing approach enables smooth

compensation of faults as the degree of the fault progresses (i.e., for any $\gamma > \epsilon$), whereas the discrete switched controller makes an abrupt switch from nominal to fault controllers once a fault magnitude threshold $\gamma^* \gg \epsilon$ is exceeded, ensuring a period of sub-optimal control with a fault whose magnitude is in the interval $[\epsilon, \gamma^*]$.

Abrupt faults Abrupt faults, which can be characterized in terms of a significant difference in at least one fault parameter γ, cannot be theoretically analysed using the piecewise affine system framework unless we can guarantee contiguity of nominal region Ω_{nom} and fault region Ω_f. Stability depends on the eigenvalues of the matrix $(A - BK)$ from a rewritten version of the observer equations, namely equation 5. We leave stability analysis as future work.

We have adopted a probabilistic approach based on residuals. In future work, we plan to compare this approach with those using Kalman filters for hypothesis testing, e.g., [17], which are computationally more complex, but may have different stability properties.

5 Experimental Design

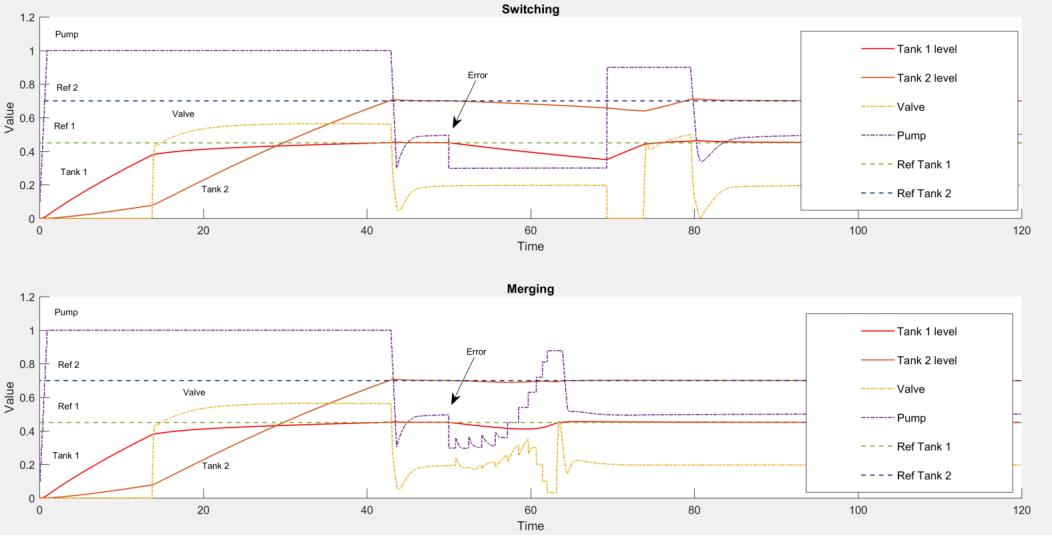

Figure 3: Comparison of controllers for FTC with Pump fault at $t = 50$s

5.1 Software Configuration

We implemented the running example described in section 2 in MATLAB/Simulink and ran all experiments in a simulation environment, where the real system and the model that the controller uses to estimate the system state are identical. We injected faults into the simulated real system by modifying the inputs and outputs of the model. We then computed residuals using the output $\hat{\boldsymbol{y}}$ of the model used by the controllers and the real system model output $\boldsymbol{y}$. This type of simulation environment allows for complete control of the fault injection, detection and controller output analysis.

5.2 System Fault Injection and Detection

We injected faults into the pump and valve using the multiplicative equation (6), with pump, valve and tank parameters $\gamma_1,\ \gamma_2,\ \gamma_3 \in [0, 1]$, respectively. γ_1 simulates the pump being blocked by only allowing a fraction of the water flow to go through the pipe. γ_2 simulates valve blockage (i.e., the valve not having full range of motion); we use a Simulink saturation block for this fault. We implemented leaks in both tanks by subtracting a constant value γ_3 of the real system model output for the two tank levels. We generated 4 residuals, for: valve, pump, tank 1 level and tank 2 level.

5.3 Controller Design

We designed three separate MPC controllers to handle different scenarios.

Nominal The first MPC is the nominal plant controller that is designed to handle the no-error state of the system. The plant model was linearised at initial model conditions with no errors present on in the system. The valve and pump output of the controller are both between 0 and 1 representing nominal working conditions.

Pump Fault The second controller is an MPC that has a plant model linearised when the pump was experiencing a fault of 40% reduction in throughput. The valve output range for this controller is also 1 but the pump output has an increased range of 0 to 1.5, since there is a blockage in the pipe the pump should be able to increase water flow to compensate for this. The reference signal coming into the second MPC is the original reference set point for the two tanks plus the estimated error in the tank levels. The justification for this is that if the error in the tank level and original reference is known then the required reference signal to stabilise the plant at the required set point can be computed.

Valve Fault The third controller is designed for valve fault scenarios, by linearising the model at a point where the valve setting is limited. The reference signal and controller output signal ranges are the same as for the nominal controller, but this controller favours the use of the pump flow over the valve setting to stabilise the plant output.

5.4 Discrete Switching and Merging implementation

We implemented discrete switching and merging approaches on identical systems using the three controllers described in section 5.3, but with different supervisory controllers.

Discrete Switching In the discrete switching implementation, higher level control mechanism analyses the current fault detection output and chooses which controller is the appropriate one for the current fault state. The decision is made by the current maximum error signal. The pump fault controller uses the two water levels in the tanks as the trigger signal. The valve controller uses the valve residual error as the trigger signal. The higher of the two tank error signals is compared to the valve error to decide which controller should get control. Error signals must be over 0.1 to make the system robust against false positive error signals and allow the nominal controller to control the plant under normal conditions.

Merging-Based Switching For the merging mechanism we implemented a look up table of weights, with error signals of the water tank error and valve error used as indexes into it. The weights allow for linear interpolation of the output signal and are defined as follows: {0.0 , 0.1 , 0.2 , ... 0.9, 1}. Every percent in the error signal will index into a higher weight, up to 10%. E.g. 0.01 valve error signal and 0.00 error signal in the tank levels will correspond to 10% of the output

signal being generated from the valve error controller, 0% from the pump fault controller and 90% from the nominal controller. If the two weighted signals added together are greater than 1 they are both incrementally reduced until their sum is equal to 1. This allows for the proportion of the signal to be preserved in extreme error cases. This interpolation mechanism is simple and has a corresponding controller merging configuration for every value in the error space.

6 Experimental Analysis

We ran experiments to compare the impact of faults on FTC designs based on discrete switching and merging approaches. We also examined the impact of time for fault isolation on FTC, since it is typically assumed that faults can be isolated instantly, even though the switching times do not have to be known *a priori*.

6.1 Switching vs. Merging

We ran experiments for single faults (pump/valve/leak) and double faults (pump and valve). Figure 3 shows the results for inducing a pump fault at $t = 50$s: the merging approach starts modifying the control signal immediately and restores set-point levels in tanks 1 and 2 by $t = 65$s; the switching approach does not perform a discrete switch until $t = 70$s, when the impact of the fault on tank levels exceeds the specified tolerance, and only restores set-point levels in tanks 1 and 2 by $t = 90$s.

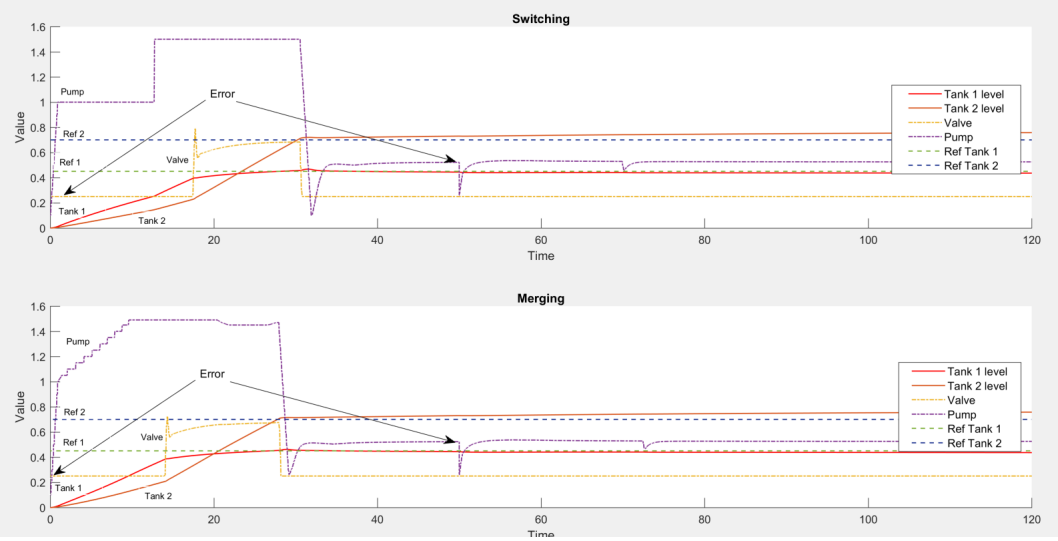

Figure 4: Comparison of controllers for FTC with Pump and Valve fault

Figure 4 shows the results for inducing a valve fault at $t = 0$s and a pump fault at $t = 50$s. The merging approach starts modifying the control signal immediately (given the valve fault) and restores set-point levels in tanks 1 and 2 by $t = 30$s; the impact of the pump fault is also quickly corrected.

In contrast, the switching approach does not perform a discrete switch to correct the valve fault until $t = 10$s, when the impact of the fault on tank levels exceeds the specified tolerance. The pump fault correction is also later than that for the mixing approach.

6.2 Impact of Delays and Fault Isolation Errors

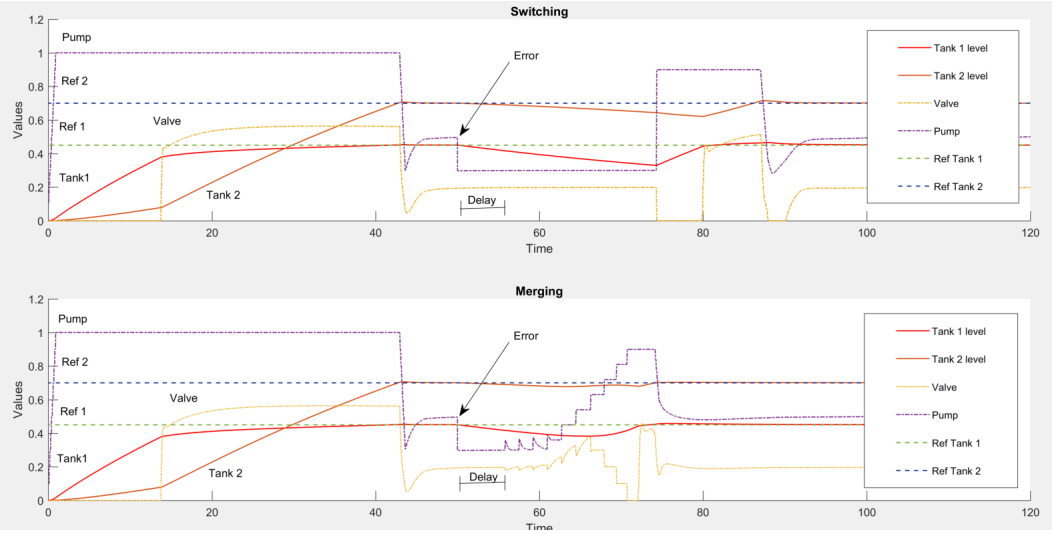

Figure 5: Impact of delay in FDI on switching and merging controllers for FTC with Pump fault.

Figure 5 shows the results of a delay of 5s on computing FDI results. This basically delays the possibility of switching and creates a greater impact on the system due to the fault.

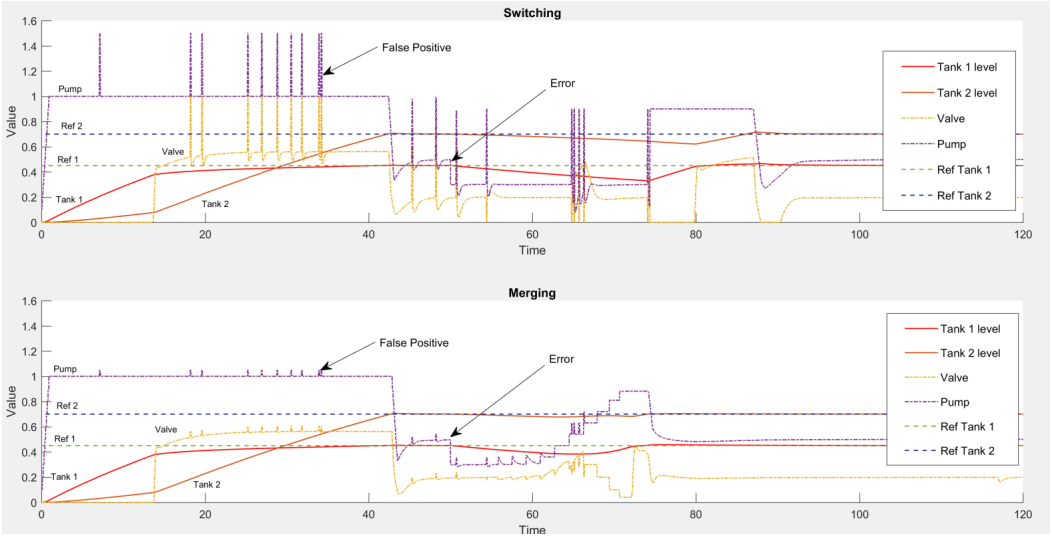

Figure 6: Impact of inaccuracy in FDI on switching and merging controllers for FTC with Pump fault.

Figure 6 shows the results of an inaccuracy in correctly isolating the fault on FDI results. Such inaccuracy must be taken into account since FDI is never perfect. These results have a large impact on the possibility of switching, which creates a significant impact on the system control due to the thrashing of the control. In contrast, the impact on mixing control is much less.

7 Discussion

Our results show a clear difference between discrete switching and mixing supervisory control in the presence of faults. The mixing supervisory control is more robust to faults and to errors in the FDI outputs. Further, the mixing supervisory control responds faster to faults than does the discrete switching approach. This improvement results from the dynamic blending of all controllers, such that even small errors are taken into account in the output signal. The supervisory controller is actively compensating for faults with a blend that is proportional to the faults seen, hence a small fault will be compensated immediately with a small portion of the output coming from the fault controller. Should the error persist the portion of the fault controller will become increasingly larger until it composes the entire output signal. However, in comparison to the switching approach, the change of controller is gradually increasing which means by the time the fault threshold has been reached the system is already moving towards a stable state and will reach this earlier than the switching approach. Even in the case where a big fault occurs which is greater than the threshold, the merging approach will simply act the same as the switching approach and let the dedicated fault controller take 100% of the output signal, so it will always perform better or equal to the switching approach.

We plan to extend this work in several ways. First, we plan to compare the active FTC methods presented with passive robust and adaptive MPC control. Second, we plan to use machine learning to adaptively tune our system to unseen anomalies/faults. We are also investigating more complex application domains, such as multi-copter drones.

References

[1] Philippe Goupil. AIRBUS state of the art and practices on FDI and FTC in flight control system. *Control Engineering Practice*, 19:524–539, 2011.

[2] Steven X. Ding Shen Yin, Bing Xiao and Donghua Zhou. "A review on recent development of spacecraft attitude fault tolerant control system". *Transactions on Industrial Electronis*, 63(5):3311–3320, 2016.

[3] Kemal Büyükkabasakal, Bariş Fidan, and Aydogan Savran. Mixing adaptive fault tolerant control of quadrotor UAV. *Asian Journal of Control*, 19(5):1–14, 2017.

[4] Jin Jiang and Xiang Yu. "Fault-tolerant control systems: A comparative study between active and passive approaches". *Annual Reviews in Control*, 36:60–72, 2012.

[5] Jan Lunze. From fault diagnosis to reconfigurable control: A unified concept. In *Control and Fault-Tolerant Systems (SysTol), 2016 3rd Conference on*, pages 413–421. IEEE, 2016.

[6] C.C. de Visser D. Molenkamp, E. van Kampen and Q.P. Chu. "Intelligent controller selection for aggressive quadrotor manoeuvring". *IAA Information Systems-AIAA Infotech @ Aerospace, AIAA SciTech Forum*, 2017.

[7] Mogens Blanke, Michel Kinnaert, Jan Lunze, Marcel Staroswiecki, and J Schröder. *Diagnosis and fault-tolerant control*, volume 691. Springer, 2006.

[8] Ron J Patton. Fault-tolerant control. *Encyclopedia of systems and control*, pages 422–428, 2015.

[9] Xiang Yu and Jin Jiang. A survey of fault-tolerant controllers based on safety-related issues. *Annual Reviews in Control*, 39:46–57, 2015.

[10] Matthew Kuipers and Petros Ioannou. Multiple model adaptive control with mixing. *IEEE Transactions on Automatic Control*, 55(8):1822–1836, 2010.

[11] Youmin Zhang and Jin Jiang. "Integrated active fault-tolerant control using IMM approach". *Transactions on Aerospace and Electronic Systems*, 37(4):1221–1235, 2001.

[12] D Magill. Optimal adaptive estimation of sampled stochastic processes. *IEEE Transactions on Automatic Control*, 10(4):434–439, 1965.

[13] M Athans, K-P Dunn, CS Greene, WH Lee, NR Sandell, I Segall, and AS Willsky. The stochastic control of the f-8c aircraft using the multiple model adaptive control (mmac) method. In *Decision and Control including*

the 14th Symposium on Adaptive Processes, 1975 IEEE Conference on, volume 14, pages 217–228. IEEE, 1975.

[14] Clement Yu, Rob J Roy, Howard Kaufman, and B Wayne Bequette. Multiple-model adaptive predictive control of mean arterial pressure and cardiac output. *IEEE Transactions on Biomedical Engineering*, 39(8):765–778, 1992.

[15] David W Lane and Peter S Maybeck. Multiple model adaptive estimation applied to the lambda urv for failure detection and identification. In *Decision and Control, 1994., Proceedings of the 33rd IEEE Conference on*, volume 1, pages 678–683. IEEE, 1994.

[16] Sajjad Fekri, Michael Athans, and Antonio Pascoal. Robust multiple model adaptive control (rmmac): A case study. *International Journal of Adaptive Control and Signal Processing*, 21(1):1–30, 2007.

[17] Vahid Hassani, Joao Pedro Hespanha, Michael Athans, and Antonio M Pascoal. Stability analysis of robust multiple model adaptive control. *IFAC Proceedings Volumes*, 44(1):350–355, 2011.

[18] Matthew Kuure-Kinsey and B Wayne Bequette. Multiple model predictive control of nonlinear systems. In *Nonlinear Model Predictive Control*, pages 153–165. Springer, 2009.

[19] Jian Sun. Small-signal methods for AC distributed power systems–a review. *Power Electronics, IEEE Transactions on*, 24(11):2545–2554, 2009.

[20] James H Taylor and Alfred J Antoniotti. Linearization algorithms for computer-aided control engineering. *Control Systems, IEEE*, 13(2):58–64, 1993.

[21] Kumpati S Narendra and Zhuo Han. The changing face of adaptive control: the use of multiple models. *Annual Reviews in Control*, 35(1):1–12, 2011.

[22] Peter D Hanlon and Peter S Maybeck. Multiple-model adaptive estimation using a residual correlation Kalman filter bank. *Aerospace and Electronic Systems, IEEE Transactions on*, 36(2):393–406, 2000.

[23] Weicun Zhang. Stable weighted multiple model adaptive control: discrete-time stochastic plant. *International Journal of Adaptive Control and Signal Processing*, 27(7):562–581, 2013.

[24] Petros A Ioannou. Cdc semi-plenary:âĂIJrobust adaptive control: The search for the holy grailâĂİ. In *Decision and Control, 2008. CDC 2008. 47th IEEE Conference on*, pages 12–13. IEEE, 2008.

[25] Lalo Magni, Riccardo Scattolini, and Mara Tanelli. Switched model predictive control for performance enhancement. *International Journal of Control*, 81(12):1859–1869, 2008.

[26] David Q Mayne. Model predictive control: Recent developments and future promise. *Automatica*, 50(12):2967–2986, 2014.