I&EC research Industrial & Engineering Chemistry Research

Article

Cite This: Ind. Eng. Chem. Res. 2020, 59, 2291−2306 pubs.acs.org/IECR

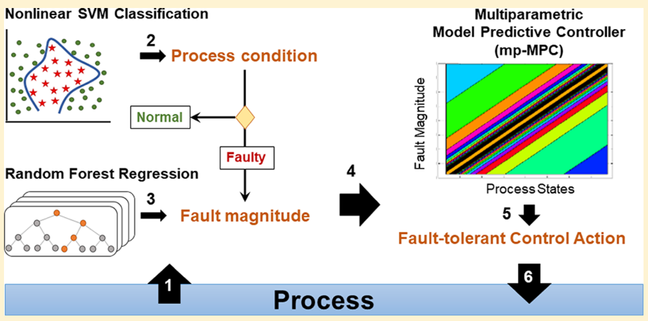
Integrated Data-Driven Process Monitoring and Explicit Fault-Tolerant Multiparametric Control

Melis Onel,[†,‡] Baris Burnak,[†,‡] and Efstratios N. Pistikopoulos*,[†,‡]

[†]Artie McFerrin Department of Chemical Engineering, Texas A&M University, College Station, Texas 77843, United States
[‡]Texas A&M Energy Institute, Texas A&M University, College Station, Texas 77843, United States

Supporting Information

ABSTRACT: We propose a novel active fault-tolerant control strategy that combines machine learning based process monitoring and explicit/multiparametric model predictive control (mp-MPC). The strategy features (i) data-driven fault detection and diagnosis models by using the support vector machine (SVM) algorithm, (ii) ranking via a nonlinear, kernel-dependent, SVM-based feature selection algorithm, (iii) data-driven regression models for fault magnitude estimation via the random forest algorithm, and (iv) a parametric optimization and control (PAROC) framework for the design of the explicit/multiparametric model predictive controller. The resulting explicit control strategies correspond to affine functions of the system states and the magnitude of the detected fault. A semibatch process, an example for penicillin production, is presented to demonstrate how the proposed framework ensures smart operation for which rapid switches between a priori computed explicit control action strategies are enabled by continuous process monitoring information.



INTRODUCTION

Achieving high process reliability and availability is of utmost importance and one of the major growing demands in process systems engineering.[1] As automation increases in industry with initiatives such as Smart Manufacturing and Industrie 4.0, process systems become more vulnerable to faults.[2] Deficiencies in sensors, actuators, controllers or disturbances in a process may cause fault, which can be amplified within a closed-loop control system and lead to a serious failure unless faulty operation is detected, recovered, and returned back to its normal condition.[3] Subsequently, rapid detection and diagnosis of faults play a key role in defining the necessary corrective actions in order to prevent fault propagation and further development of simple faults into failure. One way to approach this challenging problem is to build "fault-aware" control systems, which would understand the existence of faults in a process and adjust the controller actions accordingly, and rapidly to guarantee stability and satisfactory performance. Such control systems are referred as fault-tolerant control (FTC) systems in the literature and have been studied extensively for the last 40 years.

FTC has become an emerging research field in automatic control in the late 70s in order to overcome the limitations posed by conventional feedback control, in cases in which conventional feedback control design may end up performing poorly and lead to instabilities in the event of actuator, sensor, or another system component malfunctions.[4] The motivation in designing fault-tolerant systems has been driven by problems observed in aircraft control systems, for which particular automatic fault accommodation strategies are needed to guide pilots, and prevent the development of simple faults into severe failures which may lead to accidents.[5,6] FTC has been studied extensively in the literature;[7−10] however, interest spiked especially in the late 90s and early 2000s,[3,11−13] with applications starting to become prevalent especially in safety-critical systems with the increase in computational power and advancements in sensor technology.[14,15] Today, fault-tolerant systems are widely used in numerous fields including aircrafts,[6,16,17] mechatronics,[18] power plants,[19,20] spacecrafts,[21−24] and industrial plants producing hazardous materials such as nuclear plants.[25,26] The number of application areas is further increasing as the demand for higher process availability and profitability grows, and tolerance for process failures decreases in industries under smart manufacturing initiatives.[10]

The objective of constructing FTC systems is to increase process resilience by building a tolerance for unexpected events causing faults. FTC enables recovery to the original system performance by using the same control objective of the controlled system.[4,27] There are two different approaches in building a FTC system: one with active and the other one with passive fault-tolerance strategies (Figure 1). Passive approaches use robust control techniques to protect the system from

Special Issue: Christos Georgakis Festschrift

Received: July 31, 2019
Revised: November 20, 2019
Accepted: November 21, 2019
Published: November 21, 2019

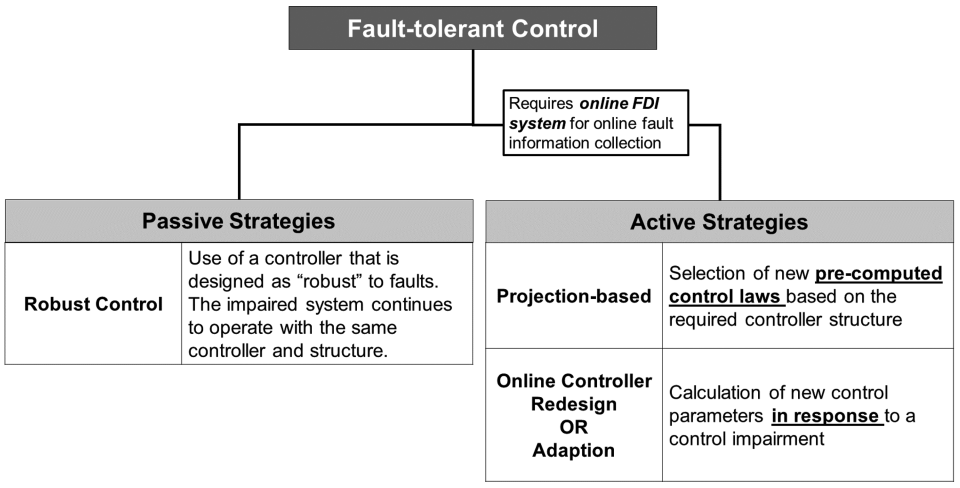

Fault-tolerant Control
Requires *online FDI system* for online fault information collection
Passive Strategies
Robust Control
Use of a controller that is designed as "robust" to faults. The impaired system continues to operate with the same controller and structure.
Active Strategies
Projection-based
Selection of new **pre-computed control laws** based on the required controller structure
Online Controller Redesign OR Adaption
Calculation of new control parameters **in response** to a control impairment

Figure 1. Categorization of fault-tolerant control strategies.

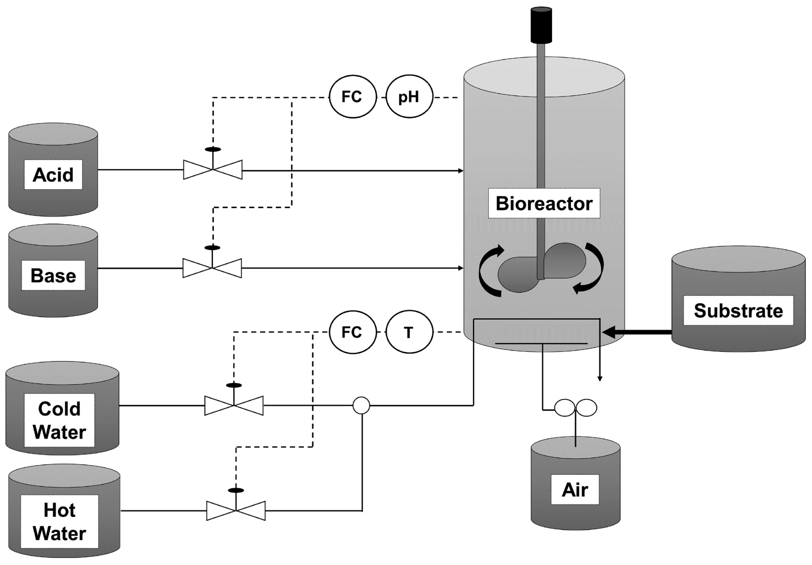


Figure 2. Fed-batch penicillin production flow diagram. (Adapted with permission from ref 28. Copyright 2018 Elselvier).

instabilities and ensure the closed-loop control system remains insensitive to certain faults by using the existing controller parameters. On the other hand, active fault-tolerant strategies do not necessarily use existing controller parameters. They use online fault detection and identification (FDI) mechanisms to monitor the process and collect information on faults when they occur for further accommodation. Active approaches are further grouped under two categories: (i) projection-based and (ii) online reconfiguration/adaptation. For projection-based FTC approaches, in addition to an accurate and robust FDI mechanism for getting online fault information, a priori knowledge on expected fault types is required to design controllers prior to possible faults that can be observed in the process. Hence, this technique necessitates offline calculation and storage of control laws. Once the information is received on a certain fault from an online FDI system, the corresponding projected controller actions are activated via one of three approaches: (i) model switching or blending, (ii) scheduling, and (iii) prediction.[2] Online reconfiguration/adaptation approaches benefit from adaptive control and reconfiguration/restructuring of the control signal distribution (i.e control allocation). Regardless of the active FTC category, reliability of online FDI mechanisms play a significant role in determining their effectiveness and robustness.

In this work, we integrate multiparametric model predictive control (mp-MPC) with a data-driven process monitoring framework[28,29] to introduce a novel parametric fault-tolerant control (FTC) design framework. The developed framework can replace the conventional approach, online controller parameter retuning, and be used as a novel corrective maintenance strategy that significantly minimizes the process downtime spent under faulty conditions by storing precalculated control laws. By using multiparametric programming,[30,31] we are able to establish the control actions for the faulty state explicitly and generate a priori, offline, maps of approximate control actions to be implemented in the online phase. This is an active fault-tolerant control strategy, specifically model-switching-based active FTC, where we need to use an online fault detection and identification (FDI) mechanism to monitor the process and get information on faults for further fault accommodation. Although switching-based active fault-tolerant control strategies that use multiparametric programming have been introduced in the literature,[32−34] the major challenge has remained to have a reliable and robust FDI system which can

Table 1. List of Process Variables. Online Measured Variables Are Marked with an Asterisk

	variable name	initial condition	measurement noise (σ)	type
1.	substrate concentration [mg/L]	17.5 ± 1	0.01	state variable
2.*	dissolved O_2 concentration [mg/L]	1.1601	0.004	state variable
3.	biomass concentration [g/L]	0.1250 ± 0.030	0.5	state variable
4.	pencillin concentration [g/L]	0	0.02	state variable
5.*	fermentation volume [m^3]	102.5 ± 5	0.002	state variable
6*.	dissolved CO_2 concentration [mg/L]	0.4487	0.12	state variable
7.*	pH [-]	5	0.02	state variable
8.*	reactor temperature [K]	298	0.01	state variable (controlled)
9.	reaction heat [cal]	0		state variable
10.*	feed rate [L/h]		0.01	input variable
11.	aeration rate [L/h]		0.01	input variable
12.*	agitator power [W]		0.01	input variable
13.*	feed temperature [K]		0.1	input variable
14.*	water flow rate [L/h]		0.01	input variable (manipulated)
15.	hot/cold switch [−]			input variable
16.*	base flow rate [mL/h]		0.01	input variable
17.*	acid flow rate [mL/h]		0.01	input variable
18.	feed substrate concentration [mg/L]			input variable
19.	cooling medium temperature [K]		0.1	input variable
20.	heating medium temperature [K]		0.05	input variable

provide accurate fault information and minimize the number of false alarms. Thus, we build data-driven fault detection and diagnosis models via the support vector machine (SVM)-based feature selection algorithm[28,35,36] and develop data-driven models for fault magnitude estimation via the random forest algorithm. The developed control strategies are affine functions of the system states and the magnitude of the detected fault which are transferred to the controller via the built machine learning-based fault detection and identification (i.e., magnitude estimation) mechanism. The premise of the presented framework is to increase process resilience and minimize process downtime while maintaining a safe and profitable operation by enabling rapid switches between a priori mapped control action strategies. The results are presented through a semibatch process for penicillin production. The paper is organized as follows: Section 1 introduces the adopted benchmark semibatch process. Section 2 describes the details of the parametric fault-tolerant control design framework. Section 3 reports the application of the framework on two distinct fault types. Finally Section 4 provides the conclusion of the presented work.

1. BENCHMARK SEMIBATCH PROCESS: PENICILLIN PRODUCTION

We adopt a fed-batch penicillin production process based on the PenSim benchmark model[37] (Figure 2). The process operates in two modes. First, it starts in batch mode with high substrate (glucose) concentrations stimulating biomass growth. After the initial glucose level is depleted in the bioreactor, the process switches to fed-batch mode where low but nonzero glucose concentration is provided. Then, under these stressful conditions penicillin production is triggered via biomass.[37,38] In this work, we simulate process data for fed-batch penicillin production by using the RAYMOND simulation package.[39] We produce 25 simulations for each fault magnitude and onset combinations in addition to the 200 simulations of normal operating condition (NOC) by using the RAYMOND software. Of note, fault direction is defined as *measured value − real value* within the RAYMOND simulator. In this work, a nominal feed rate of 0.06 L/h is chosen for the fed-batch phase of the simulations. A batch is terminated after a total of 30 L of substrate has been added. This corresponds to a total batch duration of approximately 549 h. The initial fermenter volume V_0, biomass concentration $C_{x,0}$, and substrate concentration $C_{s,0}$ are all independently sampled from normal distributions with mean μ and standard deviation σ. Values are limited to $\mu \pm 2.5\sigma$ in order to avoid outliers in the initial conditions. Measurements are collected from 20 process variables, where white noise is included into each of them (Table 1). Sensors are sampled every 0.2 h which has generated an average of 2745 sample points per batch. Please note that only a subset of these 20 process variables can be readily measured online in real-life and these are marked with an asterisk in Table 1. In fact, biomass, penicillin, and substrate concentrations are indicated to be measured only offline usually every 8−10 h.[37,38] We have utilized all 20 process variables in this work in order to demonstrate the capability of our FTC framework in handling large set of process variables.

In this work, we control the reactor temperature via fault-tolerant mp-MPC by manipulating water flow rate. We have studied two distinct fault types: (i) sensor fault, which introduces a bias in reactor temperature measurements, and (ii) actuator fault, which creates bias in water flow rate.

2. PARAMETRIC FAULT-TOLERANT CONTROL FRAMEWORK

To develop a parametric fault-tolerant control system, we design a fault-tolerant mp-MPC where we achieve offline optimal control strategies to be implemented for online control of the process, and build a mechanism for fault detection and magnitude estimation in the offline phase. The perspective is to obtain information on fault existence as well as magnitude of the detected fault in order to inform the mp-MPC with the corrected measurements. Fault is defined as the unpermitted deviation in at least one observed variable or computed parameter of the system where controllers cannot reverse it.[28] In this work, we assume that once a fault arises in the system, it does not fade out, therefore we need to inform the controller about the deviation in order to ensure smooth control actions.

The common first step is data acquisition. Data can be achieved via either historical operation data or process data simulations based on the dynamic model of the system, which is often readily available in industrial applications. For the offline design of the fault-tolerant mp-MPC, we use normal operation data. We collect both normal and faulty operation data for building fault detection, and diagnosis and magnitude estimation models.

For fault detection, we develop two-class classification models by following simultaneous fault detection and diagnosis (s-FDD) framework.[28,36] The major advantage of the s-FDD framework compared to the other fault detection and diagnosis (FDD) frameworks in the literature is the increased process monitoring efficiency by having one model that can detect and diagnose a fault. This creates a significant advantage during online process monitoring in terms of time efficiency where both detection and diagnosis can be achieved simultaneously with a unique function evaluation. In other words, s-FDD framework yields a classification model which recognizes process abnormalities, and marks them as process faults while it lists the major contributing process variables causing the detected fault, thus providing diagnosis. Furthermore, fault magnitude estimation models are achieved by developing regression models.

2.1. Offline Fault-Tolerant mp-MPC Design via PAROC Framework. We build a fault-tolerant mp-MPC by using the parametric optimization and control (PAROC) framework[30] (Figure 3), which provides a systematic methodology to design

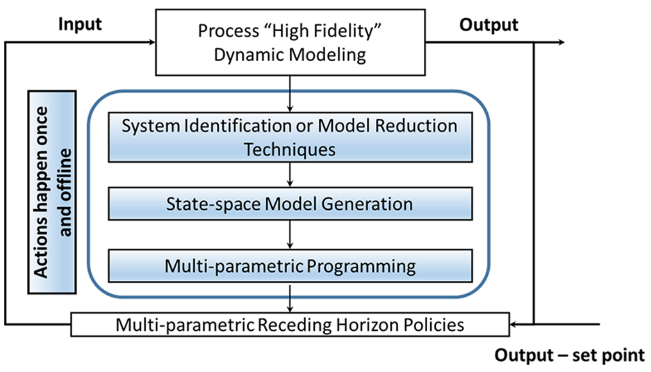

Input
Output
Process "High Fidelity" Dynamic Modeling
Actions happen once and offline
System Identification or Model Reduction Techniques
State-space Model Generation
Multi-parametric Programming
Multi-parametric Receding Horizon Policies
Output – set point

Figure 3. PAROC framework.

advanced model-based controllers via multiparametric programming. The PAROC framework presents an extensive environment for designing chemical processes, building controllers, and performing parameter estimation based on high-fidelity models while benefiting from the most recent advances in the field of multiparametric programming. Numerous applications of the PAROC framework are demonstrated in the literature for the integration of (i) process design and control,[40−42] (ii) process scheduling and control,[43] and (iii) process design, control, and scheduling.[44]

The initial step of the PAROC framework is high fidelity modeling and analysis in order to acquire a mathematical model that can describe the system of interest accurately. However, oftentimes the derived mathematical models are highly dimensional with a large number of variables and/or complex in nature posing a significant challenge during their optimization in terms of computational expense. This further hinders the direct use of these models for the development of model-based strategies and necessitates model approximation or reduction steps prior to the controller design. The reduced model is then used to build a model predictive control (MPC) scheme, and solved via multiparametric programming to obtain mp-MPC, which produces the offline map of optimal control actions under both normal and faulty operations. Here, the fault-tolerant control scheme is achieved by introducing the mismatch (fault) information as an additional dimension during the mp-MPC design. The final step is "closed-loop validation", where we implement the extracted offline map of fault-tolerant control actions to the original mathematical model of the process to observe the closed-loop behavior of the system. The PAROC framework has been applied to numerous fields successfully.[40−44] The details of each step are provided below.

2.1.1. High Fidelity Dynamic Modeling. A detailed and accurate representation of the system dynamics based on first principle dynamics and empirical correlations is used to simulate the open loop characteristics of the fed-batch penicillin production process. In this work, we employ the differential algebraic model (DAE) model presented by Birol et al.[37] as generalized below.

$$\dot{x} = f(x, u) \tag{1}$$

where x is the states of the system, u is the manipulated variables given in Table 1, and f is a generic function.

2.1.2. Model Approximation. The detailed model represented by eq 1 features complex and highly nonlinear dynamics among the manipulated variables and the observed outputs, rendering it inappropriate to develop advanced parametric controllers. Therefore, we develop an affine approximate model that accurately represents the high fidelity dynamics by model reduction or subspace identification techniques. In this work, we use the MATLAB System Identification Toolbox to capture the dynamics of eq 1 by the discrete time state space model, given by the equation below.

$$\begin{aligned} x_{t+1} &= Ax_t + Bu_t + Cd_t \\ \hat{y}_t &= Dx_t + Eu_t + Fd_t \end{aligned} \tag{2}$$

where subscript t denotes the discretized time step, and $\hat{y}$ is the output prediction. The state space matrices are developed based on the simulated process outputs y under randomized input profiles for u and d. The state space model used in this study is provided in eq 3. Note that **C**, **E**, and **F** are zero matrices since there are no measured disturbances or zeroth order inputs in the system.

$$\begin{aligned} \mathbf{A} &= \begin{bmatrix} 1.003 & -0.3722 \\ 0.009187 & 0.1688 \end{bmatrix} \\ \mathbf{B} &= \begin{bmatrix} 6.81e-5 \\ 1.52e-4 \end{bmatrix} \\ \mathbf{D} &= \begin{bmatrix} 4658 & -2069 \end{bmatrix} \end{aligned} \tag{3}$$

Note the following remarks:

(i) The identified states x do not represent the real system states.

(ii) The input−output data are generated without any sensor or actuator faults. The two fault types are accounted for analytically in the mp-MPC design phase, described in section 2.1.3.

(iii) The process faults directly affect the process dynamics, hence they are considered as added disturbances in the d_t

term. However, we only present the sensor and actuator faults in this study.

Acquiring satisfactory closed-loop performance relies heavily on developing accurate approximate models. Katz et al.[45] investigated the effects of approximating the high fidelity models by simpler models in the context of multiparametric programming, and introduced novel error metrics to evaluate open and closed-loop performances. In this work, we use the open and closed loop metrics introduced in Katz et al. to increase the confidence of the developed approximate models. Specifically, we evaluate the performance of the candidate approximate models in open loop via the standard integral time absolute error (ITAE) technique that shows the cumulative error against the high fidelity model, as well as the decision space volume comparison introduced by Katz et al.[45] After building the controller based on the developed approximate model, we further use ITAE to evaluate the tracking capacity of the developed controller for a given set point. These open loop and closed loop metrics provide relative criteria to assess the reliability of the developed approximate models. The details regarding the candidate approximate models and their evaluation are omitted here for brevity and to focus on the fault-tolerant explicit control scheme.

2.1.3. Designing the mp-MPC. The offline control strategy is designed to (i) track the output set points determined prior to the operation, and (ii) acquire smooth control actions to maintain the longevity of the processing equipment. Therefore, the objective function of the control problem is given by the following equation.

$$\sum_{t=1}^{N} \|y_t - y_t^{\mathrm{sp}}\|_{QR}^2 + \sum_{t=0}^{M} \|\Delta u_t - \theta^a\|_{R1}^2 \tag{4}$$

where N is the prediction horizon, M is the control horizon, θ^a is the magnitude of the fault acting on the corresponding actuator, $\|\cdot\|_\psi$ denotes a weighted vector norm with a weight matrix ψ, QR and $R1$ are the corresponding weight matrices, and the superscript sp denotes the set point. Hence, the quadratic objective function is minimized only if the process outputs track the designated set points y^{sp}, and the consecutive control actions are smooth in the existence of faulty actuators θ^a.

The developed objective function is subjected to the approximated process model, given by eq 2. However, using an approximate model to achieve closed-loop control creates a mismatch between the real process outputs, y, and the predicted output values, $\hat{y}$. We address this mismatch by including eq 5 in the mp-MPC formulation.

$$e = y_t - \hat{y}_t, \quad t = 0 \tag{5}$$

where the error term e denotes the mismatch magnitude between the real and predicted output values at the time of measurement, $t = 0$. The error term is carried over the entire prediction horizon, as given by the equation below.

$$y_t = \hat{y}_t + e - \theta^s, \quad \forall\, t \in \{1,2, ..., N\} \tag{6}$$

Note that apart from the mismatch term, we also incorporate a sensor bias term θ^s to account for the sensor faults in the mp-MPC. The path constraints are formulated as box constraints for the process variables to maintain certain product specifications, as presented by the equation shown below.

$$\begin{aligned} &\underline{x} \le x_t \le \overline{x} \\ &\underline{y} \le y_t \le \overline{y} \\ &\underline{u} \le u_t \le \overline{u} \\ &\underline{\Delta u} \le \Delta u_t - \theta^a \le \overline{\Delta u} \end{aligned} \tag{7}$$

Lastly, we define the set of parameters in the control problem as following:

$$\theta := [x_{t=0}^{\mathrm{T}}, u_{t=-1}^{\mathrm{T}}, y_{t=0}^{\mathrm{T}}, (y_t^{sp})^{\mathrm{T}}, d_{t=0}^{\mathrm{T}}, \theta^a, \theta^s]^{\mathrm{T}} \tag{8}$$

where θ is the vector of parameters. Therefore, we postulate the explicit control strategy described by eq 9.

$$\begin{aligned} &u_t(\theta) = \mathrm{argmin} \quad \text{eq 4} \\ &\text{s.t.} \quad \text{eqs 2, 5–8} \end{aligned} \tag{9}$$

Note that the control strategy formulated by eq 9 is a multiparametric optimization problem with a quadratic objective function and a set of linear constraints. This class of problems can be solved exactly by using the Parametric Optimization (POP) toolbox,[46] and the solution to these problems are expressed as a single piecewise affine function of the parameters. Therefore, the explicit control law is derived as given by the equation below.

$$\begin{aligned} &u_t(\theta) = K_i\theta + r_i, \forall\, t \in \{1,2, ..., M\}, \quad \forall\, \theta \in CR_i \\ &CR_i := \{\theta \in \Theta | CR_i^A \theta \le CR_i^b\} \end{aligned} \tag{10}$$

where CR denotes a polyhedral partition of the feasible parameter space, and Θ is a closed and bounded set.

Remark 1. Equation 10 explicitly maps the exact optimal control actions for any parameter realization in Θ, if a feasible solution exists. Therefore, inclusion of the monitored faults as parameters in the explicit control law identifies the range of recovery in the existence of faulty sensors and/or actuators prior to the operation.

2.1.4. Closed-Loop Validation. The proposed control problem is developed based on an approximate model. Therefore, the closed-loop strategy should be validated against the high-fidelity model by observing the set point tracking performance and path constraint violation by exhaustive simulations under numerous uncertainty scenarios. Note that due to the explicit nature of the closed-loop strategy, the control law can be embedded in the high-fidelity model exactly. Therefore, the closed-loop profiles can be simulated without the necessity of solving any online optimization problems.

In the case of insufficient or poor closed-loop performance, one can (i) adjust the weight matrices QR and $R1$ in the objective function given by eq 4, (ii) develop a new approximate model using a different technique, or (iii) develop multiple discrete time state space models that are used to govern different operating regions.

2.2. Offline Fault Detection and Reconstruction Mechanism Development. The fault detection and reconstruction mechanism is responsible for two main tasks: (i) precise and rapid fault detection and diagnosis, and (ii) accurate fault direction and magnitude estimation (a.k.a. fault reconstruction). We follow the main steps of the s-FDD framework to build fault and time-specific classification models for fault detection and diagnosis. Additionally, in order to predict the magnitude of the detected fault, we develop regression models

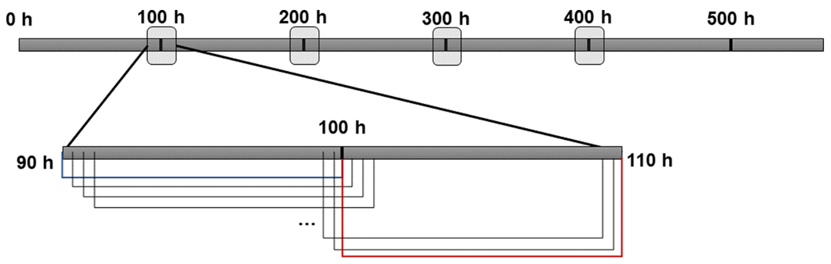


Figure 4. Schematic representation of targeted data collection for fault detection and diagnosis classifier development.

by adopting the random forest algorithm.[47] Specifically, we regress the water flow rate measurements for the actuator fault, and reactor temperature measurements for the sensor fault. The modeling procedure for both analyses is summarized in three main steps. The initial step is data preprocessing which includes targeted data collection, unfolding of 3-dimensional (3D) batch process data into 2D, extracting additional process descriptors when necessary, and data scaling, respectively. This is followed by parameter tuning and model building steps.

2.2.1. Data Preprocessing. Data preprocessing steps are necessary prior to model building in order to prevent bias and improve the performance of the model. Generally, data preprocessing comprises data formatting, scaling, and cleaning steps, where data cleaning includes both outlier removal and missing data handling. Below, we describe these three main pillars of data preprocessing in four steps: Data formatting, where we collect targeted process data; unfolding the 3D data into 2D; extracting further features when necessary to enrich the data set; and data scaling and cleaning.

Targeted Data Collection. We are building (i) fault and time specific two-class *C*-SVM classification models for fault detection and diagnosis, and (ii) regression models for fault magnitude estimation after fault onset time. Therefore, we need to gather process data around the fault onset time for both models. In this work, we have selected four different fault onset times, 100, 200, 300, and 400 h, where we introduce two different faults in various magnitudes. The details on the fault types and their magnitude are provided in section 2. In each batch, we consider the time periods that encompass the fault onset time and 10 h (50 sensor samples) afterward, where the sensor sampling frequncency is every 0.2 h.

During fault detection classifier building, we extract process data by following a sliding window approach in which we receive five samples per hour (sensor sampling frequency of every 0.2 h). At each sensor sample, we collect historical data in 10 h blocks. For instance, to build a classifier around 100 h, we consider the time period of 100−110 h of a batch. Next, starting from the fault onset time 100 h until the 110th h, we obtain process data in 10 h blocks: At hour 100, we collect data from the 90th to the 100th h. Similarly at the next sensor sample time, 100.2 h, we collect data from 90.2 to 100.2 h. We obtain the process data iteratively until the end of the considered time period, 110 h. The schematic representation of the targeted data collection is presented in Figure 4, wherein the gray boxes mark the fault onset time of the classification models being built. The blue line indicates the first and the red line indicates the last 10 h data block extracted from the 90−110 h time period for the 100 h fault detection classifier. Each data collection from the selected window adds a new instance in the data set. This approach yields a 3-dimensional (3D) data set with a size of 2500 × 20 × 50. The first dimension of the data set is obtained with 50 sliding window iterations in 50 batches (25 faulty and 25 normal operating). Furthermore, we observe 20 process variables that include both state and manipulated variables 1 in 50 sensor sample periods (i.e., 100−110 h for 100 h classifier building). The data set size is consistent for each fault and time-specific fault detection model building.

On the other hand, during fault magnitude regression development, we consider solely the process variables and do not extract any further process descriptors. Here, we collect a 10 h block for actuator fault, and 1 h for sensor fault magnitude estimation model development. We also combine all faulty operation data with varying fault magnitudes. Specifically, we have simulated six distinct fault magnitudes for sensor fault and eight for actuator faults. For each magnitude, we have simulated 25 batches. This yields 150 faulty batches with sensor fault and 200 faulty batches with actuator fault. Next, we include equal amounts of normal operating batches to our data sets. Thus, the size of the obtained data set is 300 × 20 × 5 for regression model development for sensor fault magnitude estimation, whereas the data set size for regression model development for actuator fault magnitude estimation becomes 400 × 20 × 50. Here, the first dimension belongs to the total number of batches (with equal number of faulty and normal operating batches), the second dimension is the 20 process variable measurements, and the last dimension indicates the 10 h (50 sensor sample) block, and 1 h (5 sensor sample) block examined after the fault onset time of actuator and sensor faults, respectively.

Unfolding 3D Batch Process Data into 2D. The collected 3D data needs to be unfolded into 2D prior to model building steps. The 3D data set can be unfolded in three ways by placing one out of three dimensions as rows, and grouping the other two dimensions as columns. In this work, we perform batch-wise unfolding for classification and measurement-wise unfolding for regression analysis.[48] In batch-wise unfolding, batches are the instances which are provided in the rows of the 2D data set, whereas in measurement-wise unfolding, we keep the process variable measurements as the features and place them to the columns of the 2D data set for regression analysis. The features that constitute the columns of the 2D data sets are time-specific process variable measurements for classification and time-specific-batch ID for regression models. After the unfolding step, the data set size becomes 2500 × 1000 for classification analysis. On the other hand, the unfolded data set size becomes 20000 × 20 for actuator fault and 1500 × 20 for sensor fault magnitude estimation.

Extracting Additional Features. This step is optional. We apply this step only during classification analysis. The aim of this step is to enrich the data set by including additional process descriptors to capture the process nonlinearity, which can then improve classification model performances. To do this, we calculate the (i) mean, (ii) standard deviation, and (iii) slope of 20 process measurements within each sliding time window and incorporate them into the unfolded data set. This increases the classification data set sizes to 2500 × 1060.

Data Normalization and Reduction. The final data-preprocessing step is scaling of the reconfigured data set and a priori dimensionality reduction to remove redundant features. This procedure is common to both classification and regression analysis. Each column of the 2D data set is scaled by a *z*-score calculation, in which the mean of the column is extracted from each value and then divided into the standard deviation of the column. Redundant features, where the standard deviation is less than 10^{-8}, are removed in order to decrease the computational cost during the offline model building phase.

2.2.2. Parameter Tuning. We are training (i) *C*-SVM (two-class) classification models by using the Gaussian radial basis function (RBF) as the nonlinear kernel function for fault detection and diagnosis, and (ii) regression models via random forest algorithm for fault magnitude estimation after fault detection. Note that any regression model can be used for fault magnitude estimation, yet nonlinear regression techniques are expected to be superior than linear techniques in terms of providing more accurate fault magnitude estimations due to the nonlinear relationship between the process variables. In this work, we investigated two advanced regression techniques, namely random forest regression and *C*-parametrized support vector regression (*C*-SVR). Specifically, we have trained *C*-SVR models by using the introduced feature selection algorithm in Onel et al.[28,36] The results from the *C*-SVR models are tabulated in Table S1 for actuator fault and Table S2 for sensor fault in the Supporting Information. The results provided in Tables S1 and S2 show that dimensionality reduction does not necessarily improve the model R^2 values. This is mainly due to there being a low ratio of the number of features to the number of instances in the process data set. Therefore, we use the entire process variables that remain after the data preprocessing step during regressor training. In this work, we prefer random forest algorithm over *C*-SVR due to the added benefit of the bagging technique of the random forest algorithm, which allows us to train more accurate regressors with the entire process variables for fault magnitude estimation. Regardless of the analysis, the initial step is parameter tuning which is required to achieve the optimal model performance.

Parameter Tuning for C-SVM Models. Here, we have two parameters to tune: (i) *C* (cost) parameter of *C*-SVM, and (ii) γ parameter of the Gaussian radial basis kernel function. The first parameter acts as a regularization parameter that controls the trade-off between low training error and low test error. In other words, this parameter regulates the balance between model complexity and model generalization. When the training error is lower, the model complexity is higher and the model generalizability is lower. On the other hand, when the testing error is lower, the model complexity is lower and the model generalizability is higher but with a higher training error. Finding an optimal balance is crucial to the development of an accurate classifier. Furthermore, the γ parameter determines the complexity of the Gaussian RBF kernel and affects the radius of influence of the samples selected as support vectors by the model.

In LIBSVM, the default value for the RBF kernel parameter, γ, is $1/n$, where n is the number of features. Thus, we tune parameter $\hat{\gamma}$ where

$$\gamma = \frac{2^{\hat{\gamma}}}{n} \tag{11}$$

Moreover, we tune parameter $\hat{C}$, where the relation between $\hat{C}$ and C is

$$C = 2^{\hat{C}} \tag{12}$$

According to the described iterative feature selection algorithm in our previous papers,[28,36] $\hat{\gamma}$ can be retuned after each feature elimination step with the available set of features:

$$\gamma = \frac{2^{\hat{\gamma}}}{z^{T}1} \tag{13}$$

We have performed the parameter tuning via a grid search and 10-fold cross-validation. In particular, we have used the values between −1:1 for $\hat{C}$, and −10:10 for $\hat{\gamma}$. We have performed the parameter tuning once in the beginning where we have the entire features in the data set. Although repeating the grid search for parameters tuning after each feature elimination would be ideal, we avoid the computational cost since the attained model performance has been observed to be satisfactory. If one obtains a poor-performing model, tuning can be repeated with each available feature subsets. Finally, the parameters that produce the highest average testing accuracy are chosen for the next steps. The optimal parameters for the fault-and-time specific *C*-SVM models are provided in Table 2.

Table 2. Optimal C and γ Parameters of the *C*-SVM Classifiers

fault type	fault onset time	optimal $\hat{C}$	optimal $\hat{\gamma}$
actuator	100 h	1	0
actuator	200 h	1	0
actuator	300 h	1	0
actuator	400 h	1	0
sensor	100 h	1	−2
sensor	200 h	−1	0
sensor	300 h	1	−2
sensor	400 h	1	−2

Parameter Tuning for Random Forest Regression Models. In regression analysis, we have one parameter to tune, which is the number of features that can be used in the training of each decision tree of the random forest model, *mtry*. This is performed via a grid search among the total number of features until 1 while training random forest models via 10-fold cross-validation. The optimal *mtry* parameters are obtained by using the "trainControl" function of the "caret" package of the R statistical software. The optimal *mtry* values for each time-specific regressor are provided in Table 3.

2.2.3. Model Building. Here, we address the model building steps separately for classification and regression analysis. We

Table 3. Optimal *mtry* Parameters of the Random Forest Regressors

fault type	fault onset time	optimal *mtry*
actuator	100 h	12
actuator	200 h	11
actuator	300 h	11
actuator	400 h	11
sensor	100 h	16
sensor	200 h	16
sensor	300 h	12
sensor	400 h	14

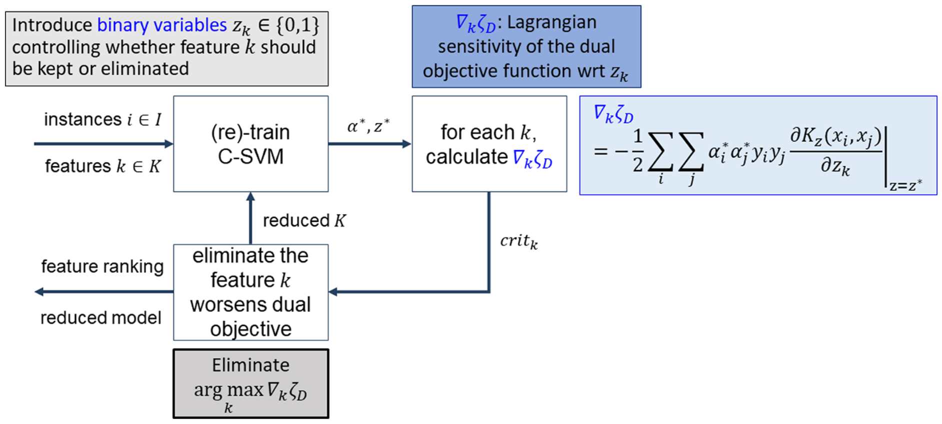

Introduce binary variables $z_k \in \{0,1\}$ controlling whether feature k should be kept or eliminated
$\nabla_k \zeta_D$: Lagrangian sensitivity of the dual objective function wrt z_k
instances $i \in I$
features $k \in K$
(re)-train C-SVM
α^*, z^*
for each k, calculate $\nabla_k \zeta_D$
$\nabla_k \zeta_D = -\frac{1}{2}\sum_i \sum_j \alpha_i^* \alpha_j^* y_i y_j \frac{\partial K_z(x_i,x_j)}{\partial z_k}\Big|_{z=z^*}$
reduced K
$crit_k$
feature ranking
reduced model
eliminate the feature k worsens dual objective
Eliminate $\arg\max_k \nabla_k \zeta_D$

Figure 5. Algorithmic solution procedure for simultaneous support vector machine-based feature selection and modeling.

follow the s-FDD framework[28,36] to build the *C*-SVM classifiers for fault detection and diagnosis. The application of the framework and data-specific details are provided below. Furthermore, we describe the model building steps for regression analysis via random forest algorithm.

Training C-SVM Models. The overall procedure for fault detection model development is summarized in Figure 5.

Step-1. Feature Ranking via C-SVM Modeling. The tuned parameters are incorporated into a simultaneous model-informed feature selection and classification algorithm via *C*-SVMs.[28,35,36] *C*-SVM binary classification models with the Gaussian radial basis function (RBF) kernel are trained iteratively with each feature subset as features being eliminated one by one. Features are eliminated based on the Lagrangian sensitivity of the dual objective function of the built *C*-SVM model with respect to the feature subset size at each iteration. This iterative process is performed with each of the 10 train−test data set pairs which produces 10 separate feature ranking lists. Next, we create an average feature rank list based on the statistical distribution of the feature ranks among the 10 ranking lists.

Step-2. Developing C-SVM Models for each Feature Subset. Here, we rebuild *C*-SVM models by using the optimal parameters and 10-fold cross validation, where we use the average feature rank list to guide the iterative feature elimination process. We start with the whole set of features and eliminate them one by one based on this final ranking list. This process produces 10 *C*-SVM classifiers for each feature subset due to the 10-fold cross-validation. The performance of each model is assessed via accuracy, area under the curve (AUC), fault detection rate, and false alarm rate. We average the performance of 10 classifiers and obtain one *C*-SVM model performance per feature subset. At the end of this step, we tabulate the performance of *C*-SVM models with each feature subset. Specifically, in this work, we have generated 1060 *C*-SVM models.

Step-3. Choosing the C-SVM Model with Optimal Feature Subset. This step determines the final *C*-SVM models to be implemented in the online phase for process monitoring. Here, we select the classifier that has provided the highest model accuracy and area under the curve with minimum number of features among the 1060 *C*-SVM models produced in Step 2. The selected feature subset is used in analyzing the root-cause of the detected fault. Therefore, selecting the minimum number of features is significant in order to facilitate the interpretation of the fault diagnosis. The performance of the selected fault-and-time specific *C*-SVM models are tabulated in Table 4.

Table 4. *C*-SVM Model Performances. (FDR: Fault Detection Rate, FAR: False Alarm Rate)

fault type	fault onset time	accuracy (%)	AUC	FDR	FAR (%)	optimal feature subset size
actuator	100 h	98.29	99.84	97.35	0.77	30
actuator	200 h	98.34	99.86	97.55	0.87	35
actuator	300 h	98.37	99.83	97.52	0.77	32
actuator	400 h	98.77	99.05	98.29	0.75	45
sensor	100 h	94.92	97.34	89.85	0.00	33
sensor	200 h	98.84	99.21	97.68	0.00	59
sensor	300 h	98.03	99.39	96.06	0.00	45
sensor	400 h	99.00	99.38	98.00	0.00	7

Training Random Forest Models. By using the optimal *mtry* parameters, we train random forest models with 500 decision trees. Training is performed via the "randomForest" function of the "randomForest" package of R statistical software. The performance of the fault-and-time specific random forest models are tabulated in Table 5.

Table 5. Random Forest Regressor Performances (RMSE: Root Mean Square)

fault type	fault onset time	R^2	RMSE
actuator	100 h	0.999	0.179
actuator	200 h	0.999	0.262
actuator	300 h	0.999	0.248
actuator	400 h	0.999	0.260
sensor	100 h	0.964	0.202
sensor	200 h	0.911	0.321
sensor	300 h	0.985	0.129
sensor	400 h	0.973	0.176

2.3. Closed-Loop Validation and Online Implementation. Prior to the online implementation, we have implemented the developed fault-tolerant mp-MPC, and fault detection and reconstruction mechanism to the RAYMOND simulator separately in order to validate their individual performances. The performance of fault-tolerant mp-MPC is assessed by providing the fault onset time and magnitude information to the controller. We have observed that the controller adapts to the faulty condition once it is provided with accurate information on the fault type, onset time, and magnitude. The accuracy of the fault detection and reconstruction mechanism is also tested and validated separately, where we have simulated a process with known fault onset and magnitude without incorporating any fault-tolerant control actions during the simulation. Finally, we implement the fault detection and reconstruction mechanism with the fault-tolerant mp-MPC in the RAYMOND simulator. During the online phase, the received signals on process variables are (i) initially collected and processed to detect the existence of any sensor or actuator faults, (ii) then reconstructed to determine the magnitude of the fault, and (iii) finally passed on to the controller for the optimal control action in the existence/absence of fault. The online procedure is illustrated in Figure 6.

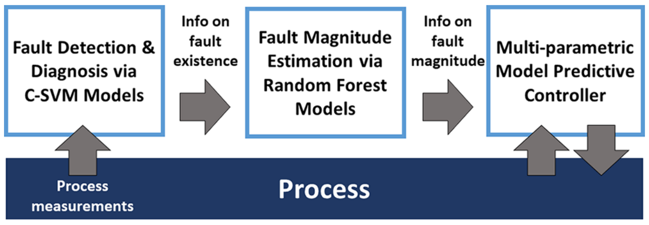

Fault Detection & Diagnosis via C-SVM Models
Info on fault existence
Fault Magnitude Estimation via Random Forest Models
Info on fault magnitude
Multi-parametric Model Predictive Controller
Process measurements
Process

Figure 6. Schematic description of the integrated online process monitoring and fault-tolerant control system.

3. RESULTS

In this work, we control reactor temperature by manipulating the water flow rate during penicillin production. We build a fault-tolerant control scheme that can tolerate both actuator and sensor fault. We introduce sensor bias in water flow rate measurements for actuator fault, whereas we introduce sensor bias in reactor temperature measurements to induce sensor fault. Numerous fault magnitudes and onset time are simulated for each fault type. Particularly, we select −2.5, −2.0, −1.5, +1.5, +2.0, +2.5, and −2.0, −1.5, −1.0, −0.5, +0.5, +1.0, +1.5, +2.0 for actuator and sensor fault magnitudes during the simulations, respectively. We have developed highly accurate fault and time-specific fault detection models and regression models for fault magnitude estimation (Tables 4 and 5) and implemented them for the fault detection and reconstruction mechanism of the established parametric fault-tolerant control system.

Figure 7 provides a comparison of the open and closed (via fault-tolerant mp-MPC) loop simulation, which signifies the importance of having accurate control actions on the reactor temperature by manipulating the water flow rate. The mp-MPC yields an offline, a priori, map of optimal control actions for the process. Figure 8 deliniates the distinct control laws for various magnitudes of sensor and actuator faults at the fixed parameters. The major advantage of the built fault-tolerant system is to gain a priori knowledge on the control actions for different fault magnitudes of actuator and sensor fault separately, as well as for different combinations of the two distinct fault types simultaneously. This map further draws the limits of the fault tolerance for each fault types. These limits indicate specific fault magnitudes for each fault type until the point at which the the designed fault-tolerant mp-MPC can recover the process back to the normal condition.

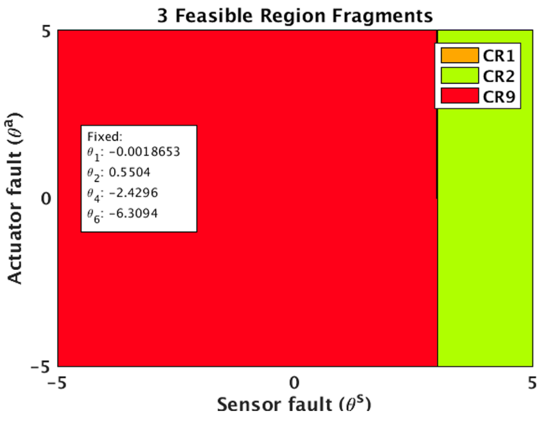


Figure 8. A demonstration of the offline map of the fault-tolerant mpMPC strategy projected to the actuator and sensor fault magnitudes at an arbitrary time in a closed-loop simulation. Each color contains a different explicit control law as a function of the parameters. The parameter θ_1 denotes the identified state, θ_2 is the normalized process output (reacture temperature), θ_4 is the output (reacture temperature) set point, and θ_6 denotes the previous control action.

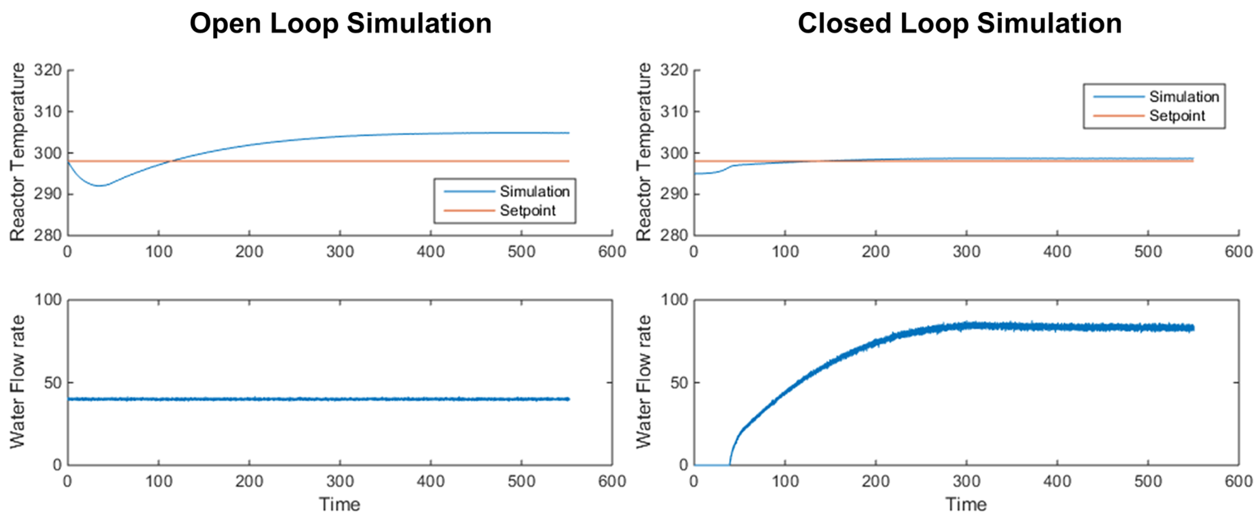


Figure 7. Simulated reactor temperature (controlled) and water flow rate (manipulated) profiles in open and closed loop (via mp-MPC).

From the beginning, we monitor the process with the fault tolerant mp-MPC and acquire information on the existence of any fault within the system from the fault detection classifier models. In this work, the adopted alarm policy is the generation of three consecutive alarms. In other words, we conclude on the fault existence when we obtain three consecutive positive responses from the fault detection classifiers. Once the fault is detected, we initiate to regress the magnitude and direction of the fault. The random forest models use the online process variable measurements to estimate the amount of deviation from the reactor temperature of the normal operating condition. Here, early detection of the faults is crucial to initiate the fault estimation process. If the fault detection latency is high, that is when fault is detected late during the operation, the controller may not be able to return the process back to the normal condition. The reason can be 2-fold: (i) the validity of the regressor may expire, thus accuracy of the fault estimation deteriorates, and (ii) there may be significant damage on the process which is irreparable. Table 6 presents the average fault detection latency of each fault and time specific classifier among the entire simulations with varying fault magnitudes.

Table 6. Average Fault Detection Latency of the Fault and Time-Specific *C*-SVM Models

fault type	fault onset time	average latency (h)
actuator	100 h	0.5
actuator	200 h	1.5
actuator	300 h	0.04
actuator	400 h	0.16
sensor	100 h	0.38
sensor	200 h	1.27
sensor	300 h	0.64
sensor	400 h	6.17

Achieving low latency with the fault and time-specific *C*-SVM models indicates early fault detection. When we compare the two different fault types, the average latency is lower for the actuator fault models. The main reason for this can be the fact that changes in water flow rate may affect the other process variables in a more definite way. This may lead to sudden changes not only in one but numerous process variables, thus facilitating the fault detection. Furthermore, the process nonlinearity affects the detection latency in distinct ways for different fault types. Specifically, we observe that we detect the actuator fault more rapidly in the later stages of the batch process, namely 300 and 400 h models. On the other hand, the separation in the average latency is not that clear among the sensor fault detection models. Here, the high latency can be linked to the low number of process variables used in the fault magnitude estimator models, which may not be adequate to capture the process behavior in the specific process time.

We are building fault and time-specific regression models for fault magnitude estimation. Therefore, it is crucial to assess the accuracy of the fault reconstruction performance after the fault onset time. During the online operation, we use the regressors that are trained around the simulated fault onset time. As the operation progresses after the fault onset time, where the process is kept under normal condition thanks to the fault-tolerant mp-MPC, the regressor model continues to use the online process data at every new sampling point. However, as the sampling time moves away from the fault onset time, the process data characteristics can significantly change, which renders the regressor inaccurate for fault estimation. Fault estimation may not be performed as accurate as it is done near the fault onset time, which hinders the controller's learning about the process condition. This, in turn, may lead to insufficient control actions to recover the process back to the normal condition. Note that the extended validity of the regressor accuracy heavily depends on the amount of deviation of the process data characteristics. As a result, it is significant and necessary to assess the time-sensitivity of the fault estimators and identify when we need new models for accurate fault reconstruction. Furthermore, the limit of each regressor determines the targeted data collection location for the next regression model training.

Tables 7 and 8 tabulate the extent of the validity of the time-specific fault detection classifiers and magnitude estimation regressors for two sets of thresholds around the reactor temperature set point being ±0.5 and ±0.75 K. The complete set of reactor temperature and water flow rate profiles with ±0.5 K threshold on the set point for each time-specific model is provided in the Supporting Information. In particular, we assess the extent of the validity of each time-specific model until the next time-specific model territory (i.e., the 100th h models are tested until the 200th h, etc.). The results for the actuator fault case show that the models that are built at the 200th and 300th h have successfully provided necessary control actions until the target process time is the 300th and 400th h, respectively. Similarly, models built at the 400th h have enabled satisfactory control actions until the end of the operation. The results for the models built for the 100th h show that the models are valid on average for the next 73.5 and 75.4 h for ±0.5 and 0.75 K thresholds around the reacture temperature set point, respectively. This highlights that we need to have additional

Table 7. Extent of Time-Specific Fault Detection and Magnitude Estimation Model Validity for Actuator Fault

		validity limit (h)	
fault onset time (h)	fault magnitude	0.5 K threshold	0.75 K threshold
100	−2.5	143.2	146.6
100	−2	147.3	150.7
100	−1.5	151.2	155.0
100	1.5	199.4	200.0
100	2	200.0	200.0
100	2.5	200.0	200.0
200	−2.5	200.0	200.0
200	−2	300.0	300.0
200	−1.5	300.0	300.0
200	1.5	300.0	300.0
200	2	300.0	300.0
200	2.5	300.0	300.0
300	−2.5	400.0	400.0
300	−2	400.0	400.0
300	−1.5	400.0	400.0
300	1.5	400.0	400.0
300	2	400.0	400.0
300	2.5	400.0	400.0
400	−2.5	through the end	through the end
400	−2	through the end	through the end
400	−1.5	through the end	through the end
400	1.5	through the end	through the end
400	2	through the end	through the end
400	2.5	through the end	through the end

Table 8. Extent of Time-Specific Fault Detection and Magnitude Estimation Model Validity for Sensor Fault

fault onset time (h)	fault magnitude	validity limit (h)	
		0.5 K threshold	0.75 K threshold
100	−2.0	187.9	200.0
100	−1.5	187.4	200.0
100	−1.0	187.2	200.0
100	−0.5	187.2	200.0
100	0.5	187.6	200.0
100	1.0	187.6	200.0
100	1.5	187.6	200.0
100	2.0	187.4	200.0
200	−2.0	201.3	300.0
200	−1.5	201.3	300.0
200	−1.0	201.3	300.0
200	−0.5	201.3	300.0
200	0.5	300.0	300.0
200	1.0	300.0	300.0
200	1.5	300.0	300.0
200	2.0	300.0	300.0
300	−2.0	358.1	358.1
300	−1.5	356.4	356.4
300	−1.0	359.9	359.9
300	−0.5	358.0	358.0
300	0.5	369.9	369.9
300	1.0	360.5	360.5
300	1.5	360.4	360.4
300	2.0	363.1	363.1
400	−2.0	401.4	401.4
400	−1.5	401.4	401.4
400	−1.0	401.4	401.4
400	−0.5	401.5	401.5
400	0.5	Through the end	Through the end
400	1.0	Through the end	Through the end
400	1.5	Through the end	Through the end
400	2.0	Through the end	Through the end

models for accurate fault detection and magnitude estimation between the 100th and 200th h of the batch operation.

On the other hand, for the sensor fault case, we note that the models built for the 200th and 400th h are not valid for an extended process time when negative fault magnitudes are observed. On average, the models are valid for another 1.3 and 1.5 h after the fault is introduced in the 200th and 400th h, respectively, when the reactor temperature deviation threshold is set to 0.5 K. When we increase the threshold to 0.75 K around the set point, we observe that the models built at 200th h can maintain a smooth operation for the entire targeted operation range, which is the next 100 h, because the latency in fault detection has caused a deviation that is higher than 0.5 but lower than 0.75 K. However, this does not apply to the models for the 400th h. The threshold increase does not extend the validity of the 400th h models since the maximum deviation observed is as high as 2.1 K (Figure S52). The limited model validity for the two time-specific models at 200 and 400 h is due to the high fault detection latency. In other words, by the time we detect the fault occurring at the 200th and 400th h, the deviation from the reactor temperature set point already exceeds the predetermined thresholds (Figure S34−S37 for the 200th models and Figure S52−S55 for the 400th models). Therefore, required control actions are not provided by the controller as it has not been notified of the existence and magnitude of the fault. To overcome this problem, fault detection latency must be improved. This can be achieved by increasing the frequency of the fault detection classifiers between 200 and 400 h of the batch operation.

To provide a comparison between the two fault types, we provide the reactor temperature and water flow rate profiles for 100 h models. Particularly, we display the profiles of the simulations in which we introduce actuator faults with −2.5 and +2.5 fault magnitude in Figures 9 and 10, respectively.

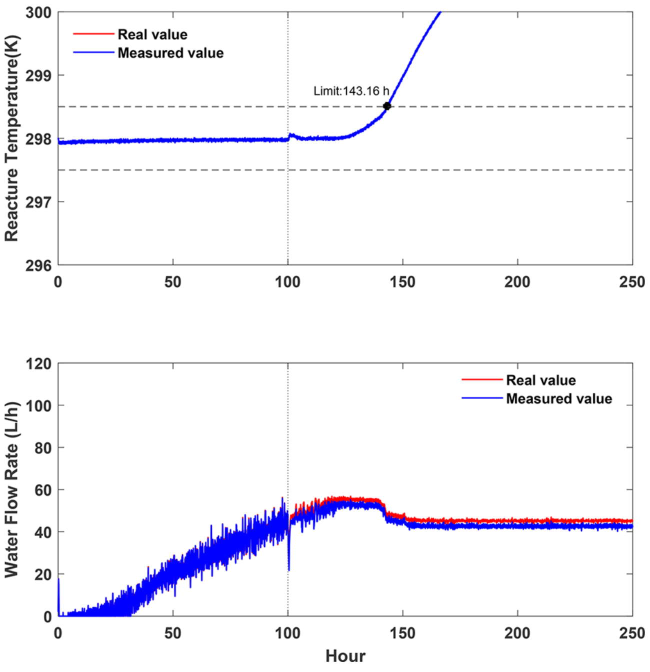


Figure 9. Reactor temperature and water flow rate profiles for a process with actuator fault. Fault onset, 100 h; fault magnitude, −2.5.

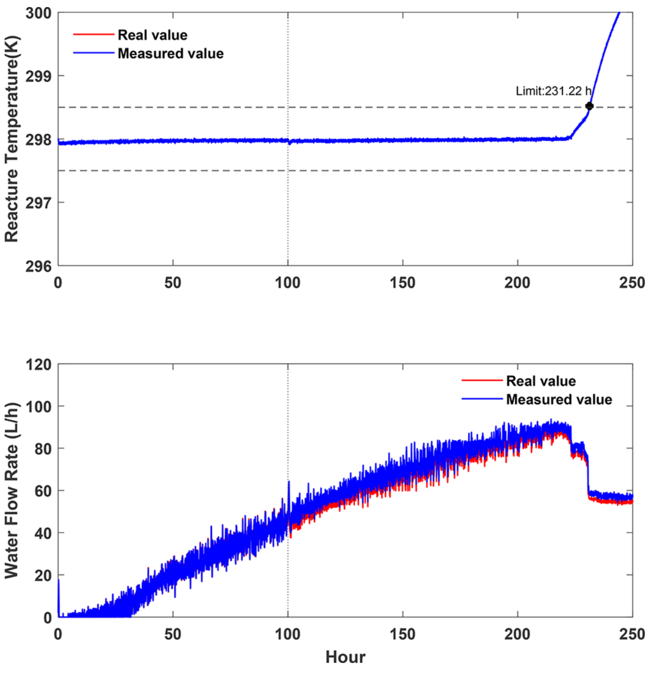


Figure 10. Reactor temperature and water flow rate profiles for process with actuator fault. Fault onset, 100 h; fault magnitude, +2.5.

Additionally, Figures 11 and 12 demonstrate the profiles of the simulations in which we introduce sensor faults with −2 and +2 fault magnitude. The profiles with actuator fault show that once the regressor model validity expires with the altering dynamics of the batch process, the correction in the faulty water flow rate disrupts and deteriorates. This leads to a significant increase in the reactor temperature that leads to a possible system failure. On the other hand, early capture of the sensor fault leads to rapid and necessary changes in the water flow rate which enable a fast process recovery back to the normal condition. Of note, in order to ensure smooth control actions, one needs to switch to the next valid model once the validity of the previous model expires. This is necessary in order to capture dynamic process characteristics and detect any possible faults. The presented simulation profiles with actuator and sensor faults prove that the designed fault-tolerant mp-MPC provides smooth control actions successfully.

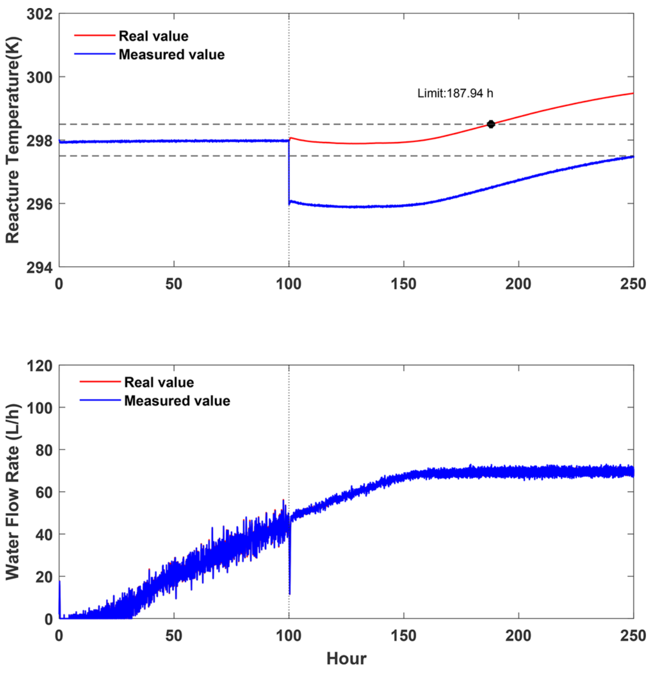


Figure 11. Reactor temperature and water flow rate profiles for process with sensor fault. Fault onset, 100 h; fault magnitude, −2.0.

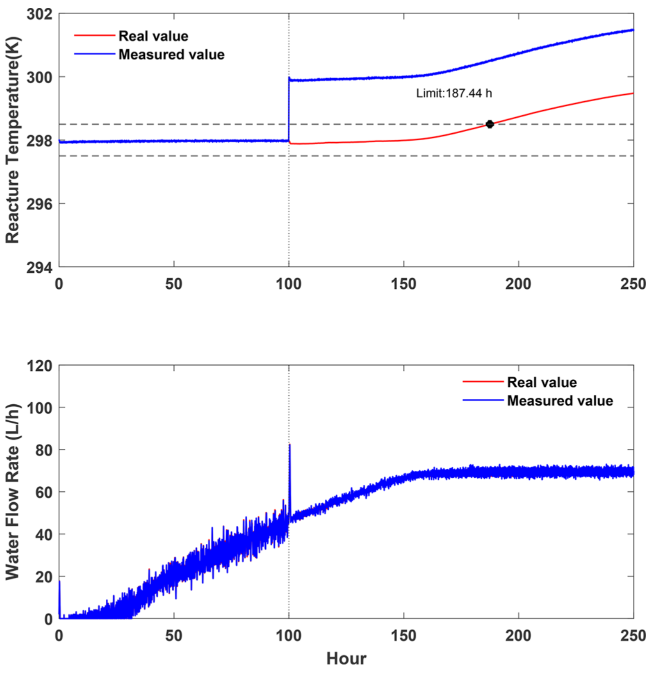


Figure 12. Reactor temperature and water flow rate profiles for process with sensor fault. Fault onset, 100 h; fault magnitude, +2.0.

Finally, we perform a sensitivity analysis with the time-specific fault detection and magnitude estimation models built at 100th and 200th h in order to determine the perimeter of the model effectiveness. To this end, we use the time-specific models for ±30 h perimeter of their corresponding process time. Particularly, the *C*-SVM model for fault detection and random forest model for the fault magnitude estimation are utilized for every 5 h fault onsets between 70th and 130th h with the models built at 100th h and between 170th and 230th h with the models built at 200th h (Figure 13). We adopt the ±0.5 K threshold around the reactor temperature set point and only utilize the extreme negative and positive fault magnitudes simulated in this work (−2.5 and +2.5 for actuator and −2 and +2 for sensor fault) for the analysis. The results reveal that actuator fault models have more limited range compared to sensor fault models. In particular, the models built at 100th have successfully been used between the 90th and 100th h of the batch operation. The validity range for the models built at the 200th reaches to 15 and 20 h for negative and positive fault magnitudes, respectively. On the other hand, the analysis yields that the models built at the 100th and 200th h for the sensor fault were able to perform the required control actions for the analyzed 30 h perimeter except for the analysis performed with a negative fault magnitude with models built at the 200th h. This is again due to the fact that by the time the fault is detected the raise in the reactor temperature exceeds the allowed region (Figure S34). When the deviation threshold is raised to ±0.75 K, the time-specific models are shown to be valid for the entire analyzed 30 h perimeter (Figure 14). This analysis elucidates the effectiveness limit of the time-specific models which is required to determine the model switching frequency during online monitoring. Overall, the results demonstrate the need for additional models during 100−200 h of the operation if a strict deviation threshold (i.e., 0.5 K) is preferred during the operation. Yet for a 0.75 K deviation threshold, the presented time-specific models have successfully provided satisfactory control actions under faulty conditions.

4. CONCLUSIONS

As the effect of smart manufacturing revolution propagates and influences the vision of numerous industrial operations, the development of a fault-tolerant control system becomes one of the major factors in achieving high process resilience. Traditional corrective maintenance strategies include controller retuning which leads to longer process downtime that may adverse the end-product quality and cause higher operation cost. This work proposes a novel parametric fault-tolerant control framework that enables rapid and accurate switches within the offline map of control actions to eliminate process downtime and maximize process reliability. This further enables attaining higher product quality which leads to higher profit from the operation.

In this work, we present a novel active fault-tolerant strategy and corrective maintenance strategy which benefits from multiparametric programming and machine learning-based process monitoring. Particularly, we have designed a multiparametric model predictive controller by following the PAROC

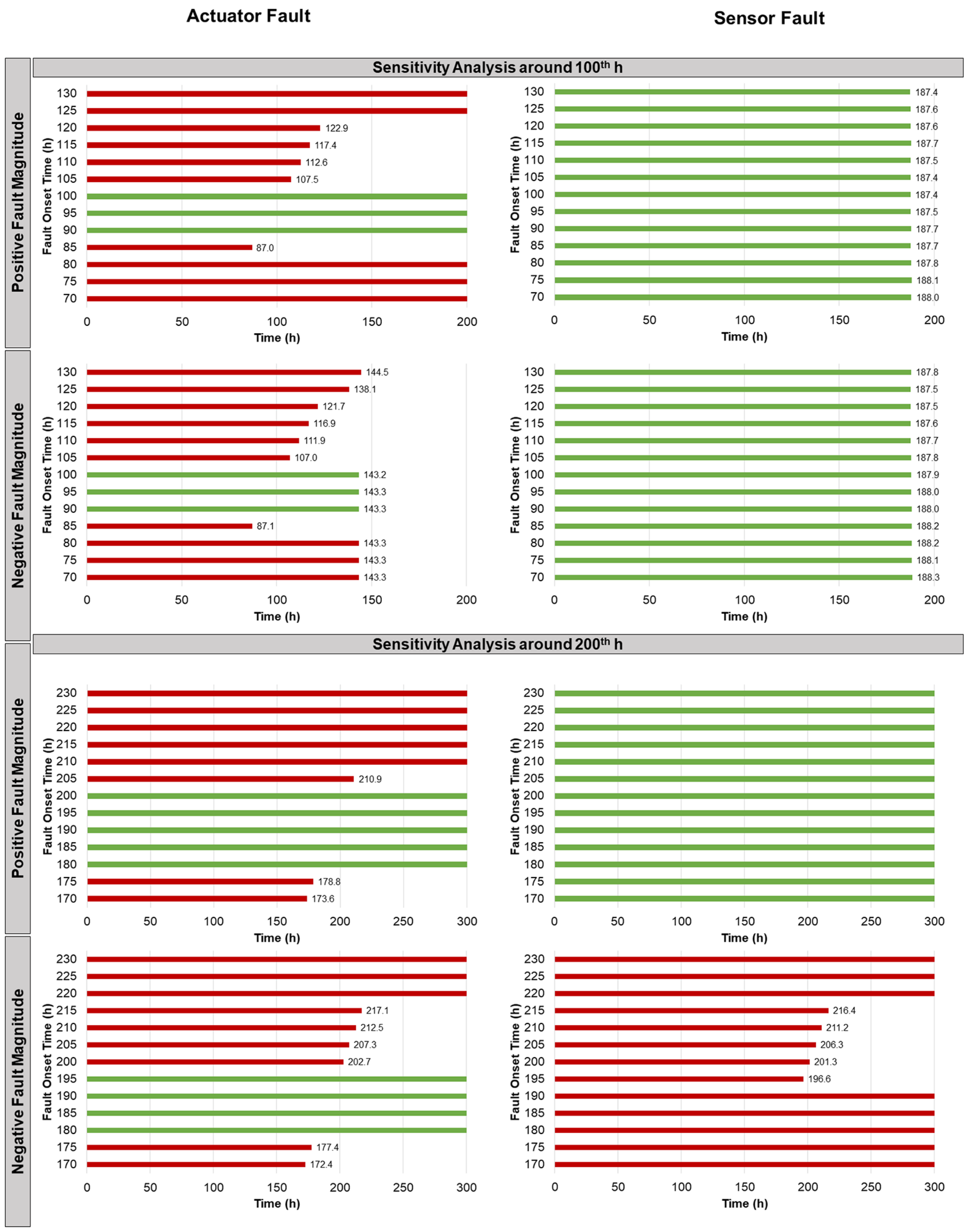


Figure 13. Sensitivity analysis of time-specific models built at the 100th and 200th h for actuator and sensor faults. The set point deviation threshold is ±0.5 K. The green bars highlight that the model is satisfactorily valid. The red bars belong to limited time model validity cases. Note that once a red bar is assigned, the further hours are automatically assigned with red.

framework[30] and the s-FDD framework. The s-FDD framework is used to formulate the fault detection and reconstruction mechanism of the fault-tolerant system, where the built classifiers provide the information on fault existence and regressors yield the fault magnitude and direction estimation. The trained *C*-SVM models with the optimal feature subset further enable the rapid diagnosis of the detected fault. The average accuracy of the classifiers is 98.44%, and 97.70% for the actuator and sensor faults, respectively. Moreover, the average R^2 of the trained regressors is 0.999 and 0.958 for the actuator and sensor faults, respectively. The presented approach formulates as a novel active fault-tolerant strategy in which an accurate and robust fault detection and reconstruction mechanism is ensured via the s-FDD framework and multi-parametric MPC enables rapid switches between fault-tolerant control actions. Please note that the presented fault-tolerant strategy is agnostic to any fault types, thus it can be extended to process faults by treating them as measured disturbances. Finally, we note that the design of the fault-tolerant mp-MPC can further enable the handling of simultaneous faults as it includes the deviation in both process variables (i.e., reactor temperature and water flow rate) as additional parameters.

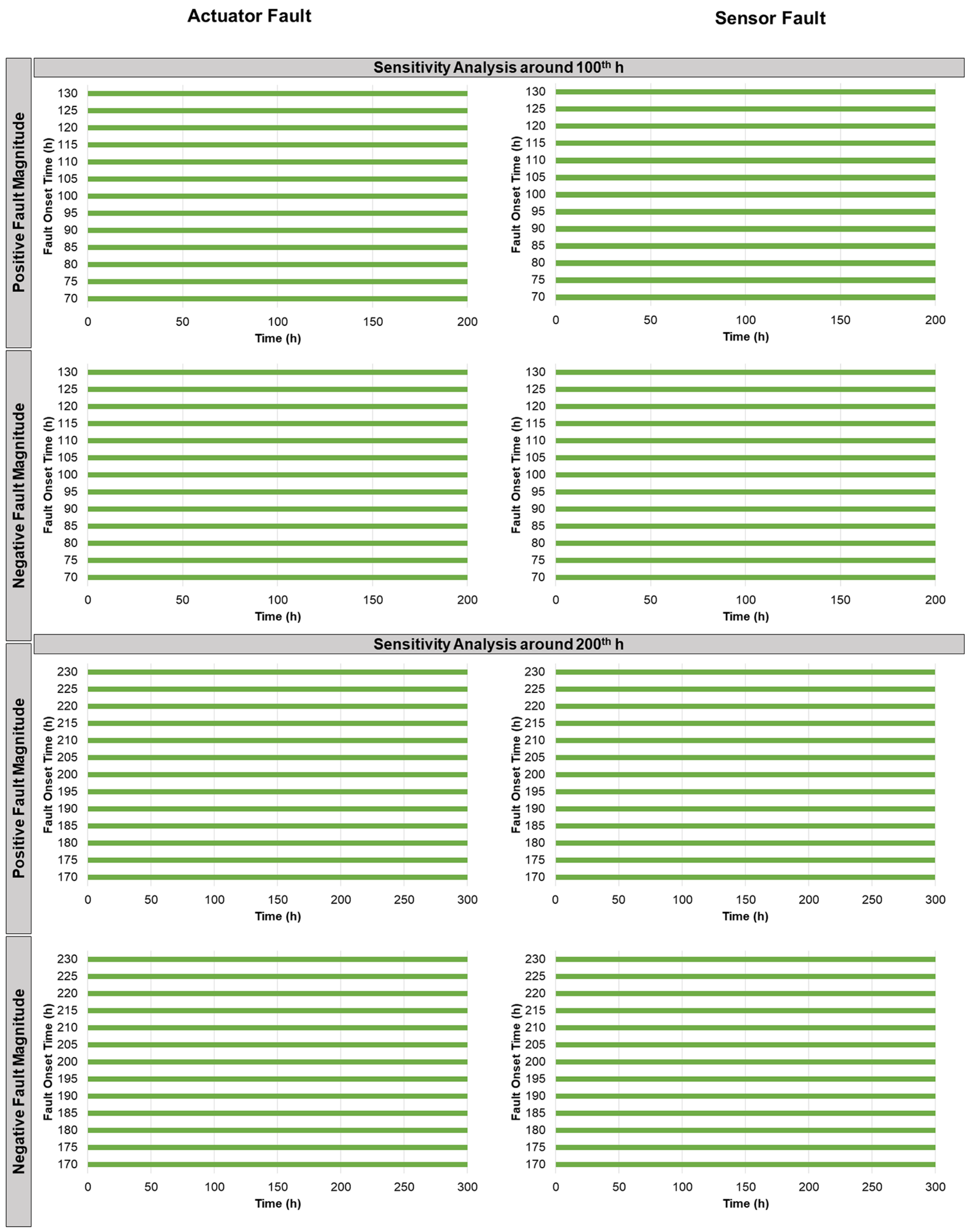


Figure 14. Sensitivity analysis of time-specific models built at the 100th and 200th h for actuator and sensor faults. The set point deviation threshold is ±0.75 K. The green bars highlight that the model is satisfactorily valid. The red bars belong to limited time model validity cases. Note that once a red bar is assigned, the further hours are automatically assigned with red.

■ ASSOCIATED CONTENT

Supporting Information

The Supporting Information is available free of charge at https://pubs.acs.org/doi/10.1021/acs.iecr.9b04226.

Results from *C*-SVR models, which are built for fault magnitude estimation after fault detection; complete set of reactor temperature and water flow rate profiles with ±0.5 K threshold on the set point under actuator and sensor faults for each time-specific model (PDF)

■ AUTHOR INFORMATION

Corresponding Author

*Email: stratos@tamu.edu.

ORCID

Baris Burnak: 0000-0001-6118-8711
Efstratios N. Pistikopoulos: 0000-0001-6220-818X

Notes

The authors declare no competing financial interest.

■ ACKNOWLEDGMENTS

This research was funded by U.S. National Institute of Health (NIH) Grant P42 ES027704, U.S. National Science Foundation (NSF) Grant No. 1705423, and Texas A & M Energy Institute. The content of this publication does not necessarily represent the official views of the NIH or NSF. Furthermore, portions of this research were conducted with the advanced computing resources provided by Texas A&M High Performance Research Computing.

■ REFERENCES

(1) Kusiak, A. Fundamentals of smart manufacturing: A multi-thread perspective. *Annu. Rev. Control* **2019**, *47*, 214.
(2) Alwi, H.; Edwards, C.; Tan, C. P. *Fault detection and fault-tolerant control using sliding modes*; Springer Science & Business Media, 2011.
(3) Blanke, M.; Staroswiecki, M.; Wu, N. E. Concepts and methods in fault-tolerant control. *Proc. 2001 Am. Control Conf.* **2001**, *4*, 2606−2620.
(4) Patton, R. J. Fault-tolerant control: the 1997 situation. *IFAC Proceedings Volumes* **1997**, *30* (18), 1029−1051.
(5) McMahan, J. Flight 1080. *Air Line Pilot* **1978**, *47* (7), 6−11.
(6) Edwards, C.; Lombaerts, T.; Smaili, H.; et al. Fault tolerant flight control. *Lecture Notes in Control and Information Sciences* **2010**, *399*, 1−560.
(7) Himmelblau, D. M. *Fault detection and diagnosis in chemical and petrochemical processes*; Elsevier Science Ltd, 1978; Vol. *8*.
(8) Zhao, Q.; Jiang, J. Reliable state feedback control system design against actuator failures. *Automatica* **1998**, *34* (10), 1267−1272.
(9) Jiang, J. Fault-tolerant control systems−an introductory overview. *Automat. SINCA* **2005**, *21* (1), 161−174.
(10) Zhang, Y.; Jiang, J. Bibliographical review on reconfigurable fault-tolerant control systems. *Annual reviews in control* **2008**, *32* (2), 229−252.
(11) Wong, W.; Lee, L. W. *Fault-resilient automobile control system*. US Patent 5,957,985, Sept. 28, 1999.
(12) Jeong, Y.-s.; Sul, S.-K.; Schulz, S. E.; Patel, N. R. Fault detection and fault-tolerant control of interior permanent-magnet motor drive system for electric vehicle. *IEEE Trans. Ind. Appl.* **2005**, *41* (1), 46−51.
(13) Noura, H.; Theilliol, D.; Ponsart, J.-C.; Chamseddine, A. *Fault-tolerant control systems: Design and practical applications*; Springer Science & Business Media, 2009.
(14) Christofides, P. D.; Davis, J. F.; El-Farra, N. H.; Clark, D.; Harris, K. R.; Gipson, J. N. Smart plant operations: Vision, progress and challenges. *AIChE J.* **2007**, *53* (11), 2734−2741.
(15) Davis, J.; Edgar, T.; Graybill, R.; Korambath, P.; Schott, B.; Swink, D.; Wang, J.; Wetzel, J. Smart manufacturing. *Annu. Rev. Chem. Biomol. Eng.* **2015**, *6*, 141−160.
(16) Yu, B.; Zhang, Y. Fault-tolerant control of a boeing 747-100/200 based on a laguerre function-based mpc scheme. *IFAC-PapersOnLine* **2016**, *49* (17), 58−63.
(17) Hallouzi, R.; Verhaegen, M. Reconfigurable fault tolerant control of a boeing 747 using subspace predictive control. In *AIAA Guidance, Navigation and Control Conference and Exhibit*; AIAA, 2007; pp 6665.
(18) Caccavale, F.; Villani, L. *Fault diagnosis and fault tolerance for mechatronic systems: Recent advances*; Springer Science & Business Media, 2002; Vol. *1*.
(19) Ye, L.; Wang, S.; Bing, F.; Malik, O.; Zeng, Y. Control/maintenance strategy fault tolerant mode and reliability analysis for hydro power stations. *IEEE Transactions on Power Systems* **2001**, *16* (3), 340−345.
(20) Noura, H.; Sauter, D.; Hamelin, F.; Theilliol, D. Fault-tolerant control in dynamic systems: Application to a winding machine. *IEEE control systems magazine* **2000**, *20* (1), 33−49.
(21) Pirmoradi, F.; Sassani, F.; De Silva, C. Fault detection and diagnosis in a spacecraft attitude determination system. *Acta Astronaut.* **2009**, *65* (5−6), 710−729.
(22) Xiao, B.; Hu, Q.; Zhang, Y. Adaptive sliding mode fault tolerant attitude tracking control for flexible spacecraft under actuator saturation. *IEEE Transactions on Control Systems Technology* **2012**, *20* (6), 1605−1612.
(23) Jiang, Y.; Hu, Q.; Ma, G. Adaptive backstepping fault-tolerant control for flexible spacecraft with unknown bounded disturbances and actuator failures. *ISA Trans.* **2010**, *49* (1), 57−69.
(24) Yin, S.; Xiao, B.; Ding, S. X.; Zhou, D. A review on recent development of spacecraft attitude fault tolerant control system. *IEEE Transactions on Industrial Electronics* **2016**, *63* (5), 3311−3320.
(25) Eryurek, E.; Upadhyaya, B. R. Fault-tolerant control and diagnostics for large-scale systems. *IEEE Control Systems Magazine* **1995**, *15* (5), 34−42.
(26) Deng, Z.; Shi, X.; Xia, G.; Fu, M. Fault tolerant control for steam generators in nuclear power plant. *Nuclear Power Engineering* **2010**, *31* (1), 107−111.
(27) Jiang, J.; Yu, X. Fault-tolerant control systems: A comparative study between active and passive approaches. *Annual Reviews in control* **2012**, *36* (1), 60−72.
(28) Onel, M.; Kieslich, C. A.; Guzman, Y. A.; Floudas, C. A.; Pistikopoulos, E. N. Big data approach to batch process monitoring: Simultaneous fault detection and diagnosis using nonlinear support vector machine-based feature selection. *Comput. Chem. Eng.* **2018**, *115*, 46−63.
(29) Onel, M.; Kieslich, C. A.; Guzman, Y. A.; Pistikopoulos, E. N. Simultaneous fault detection and identification in continuous processes via nonlinear support vector machine based feature selection. In *13th International Symposium on Process Systems Engineering (PSE 2018)*; Computer Aided Chemical Engineering, Vol. 44; Eden, M. R., Ierapetritou, M. G., Towler, G. P., Eds.; Elsevier, 2018; pp 2077−2082.
(30) Pistikopoulos, E. N.; Diangelakis, N. A.; Oberdieck, R.; Papathanasiou, M. M.; Nascu, I.; Sun, M. Paroc−−an integrated framework and software platform for the optimisation and advanced model-based control of process systems. *Chem. Eng. Sci.* **2015**, *136*, 115−138.
(31) Pistikopoulos, E. N.; Georgiadis, M. C.; Dua, V. *Multi-parametric programming*; Wiley, 2011; Vol. *1*.
(32) Marseglia, G. R.; Raimondo, D. M. Active fault diagnosis: A multi-parametric approach. *Automatica* **2017**, *79*, 223−230.
(33) Schneider, T.; Ducard, G.; Rudin, K.; Strupler, P. Fault-tolerant control allocation for multirotor helicopters using parametric programming. *Proc. Intl. Micro Air Vehicle Conf. Flight Compet.* **2012**, *1*, r2.
(34) Spjøtvold, J.; Johansen, T. A. Fault tolerant control allocation for a thruster-controlled floating platform using parametric programming. *Proc. 48h IEEE Conf. Decision. Control (CDC) held jointly with 2009 28th Chin. Control Conf.* **2009**, 3311−3317.
(35) Guzman, Y. *Theoretical Advances in Robust Optimization, Feature Selection, and Biomarker Discovery*. Ph.D. Thesis, Princeton University, Princeton, NJ, 2016.
(36) Onel, M.; Kieslich, C. A.; Pistikopoulos, E. N. A nonlinear support vector machine-based feature selection approach for fault detection and diagnosis: Application to the tennessee eastman process. *AIChE J.* **2019**, *65* (3), 992−1005.
(37) Birol, G.; Ündey, C.; Çinar, A. A modular simulation package for fed-batch fermentation: Penicillin production. *Comput. Chem. Eng.* **2002**, *26* (11), 1553−1565.
(38) Van Impe, J.; Gins, G. An extensive reference dataset for fault detection and identification in batch processes. *Chemom. Intell. Lab. Syst.* **2015**, *148*, 20−31.
(39) Gins, G.; Vanlaer, J.; Van den Kerkhof, P.; Van Impe, J. F. The raymond simulation package−−generating raypresentative monitoring data to design advanced process monitoring and control algorithms. *Comput. Chem. Eng.* **2014**, *69*, 108−118.
(40) Diangelakis, N. A.; Burnak, B.; Katz, J.; Pistikopoulos, E. N. Process design and control optimization: A simultaneous approach by multi-parametric programming. *AIChE J.* **2017**, *63* (11), 4827−4846.

(41) Ogumerem, G. S.; Pistikopoulos, E. N. Parametric optimization and control toward the design of a smart metal hydride refueling system. *AIChE J.* **2019**, e16680.

(42) Papathanasiou, M. M.; Burnak, B.; Katz, J.; Shah, N.; Pistikopoulos, E. N. Assisting continuous biomanufacturing through advanced control in downstream purification. *Comput. Chem. Eng.* **2019**, *125*, 232−248.

(43) Burnak, B.; Katz, J.; Diangelakis, N. A.; Pistikopoulos, E. N. Simultaneous process scheduling and control: A multiparametric programming-based approach. *Ind. Eng. Chem. Res.* **2018**, *57* (11), 3963−3976.

(44) Burnak, B.; Diangelakis, N. A.; Katz, J.; Pistikopoulos, E. N. Integrated process design, scheduling, and control using multiparametric programming. *Comput. Chem. Eng.* **2019**, *125*, 164−184.

(45) Katz, J.; Burnak, B.; Pistikopoulos, E. N. The impact of model approximation in multiparametric model predictive control. *Chem. Eng. Res. Des.* **2018**, *139*, 211−223.

(46) Oberdieck, R.; Diangelakis, N. A.; Papathanasiou, M. M.; Nascu, I.; Pistikopoulos, E. N. Pop − parametric optimization toolbox. *Ind. Eng. Chem. Res.* **2016**, *55* (33), 8979−8991.

(47) Breiman, L. Random forests. *Machine learning* **2001**, *45* (1), 5−32.

(48) Nomikos, P.; MacGregor, J. F. Monitoring batch processes using multiway principal component analysis. *AIChE J.* **1994**, *40* (8), 1361−1375.