processes

MDPI

Article

Goal-Oriented Tuning of Particle Filters for the Fault Diagnostics of Process Systems

Éva Kenyeres and János Abonyi *

ELKH-PE Complex Systems Monitoring Research Group, Department of Process Engineering, University of Pannonia, Egyetem u. 10, H-8200 Veszprém, Hungary
* Correspondence: janos@abonyilab.com

Abstract: This study introduces particle filtering (PF) for the tracking and fault diagnostics of complex process systems. In process systems, model equations are often nonlinear and environmental noise is non-Gaussian. We propose a method for state estimation and fault detection in a wastewater treatment system. The contributions of the paper are the following: (1) A method is suggested for sensor placement based on the state estimation performance; (2) based on the sensitivity analysis of the particle filter parameters, a tuning method is proposed; (3) a case study is presented to compare the performances of the classical PF and intelligent particle filtering (IPF) algorithms; (4) for fault diagnostics purposes, bias and impact sensor faults were examined; moreover, the efficiency of fault detection was evaluated. The results verify that particle filtering is applicable and highly efficient for tracking and fault diagnostics tasks in process systems.

Keywords: state estimation; particle filtering; intelligent particle filter; fault detection; carbon-removal wastewater treatment process; cascade reactors

check for updates

Citation: Kenyeres, É.; Abonyi, J. Goal-Oriented Tuning of Particle Filters for the Fault Diagnostics of Process Systems. *Processes* **2023**, *11*, 823. https://doi.org/10.3390/pr11030823

Academic Editor: Wei Sun

Received: 25 January 2023
Revised: 26 February 2023
Accepted: 5 March 2023
Published: 9 March 2023

1. Introduction

This paper offers an insight into the topic of state estimation with a particle filter algorithm and its practical applications in process systems by considering the fault diagnostics of a wastewater treatment cascade reactor benchmark in particular.

The monitoring and fault diagnostics of complex process systems is extremely challenging for engineers. In practice, not every state variable is available from measurements. However, knowing their values is desired, e.g., in process control or fault diagnostics. Applying state estimation is an excellent solution to the problem.

State observers can estimate unmeasured state variables based on measured ones. The simplest and most widely used state observer is the Kalman filter (KF). However, it is only usable if the system can be described with a linear state space model, and the model and measurement noise follow a Gaussian distribution. In other cases, the extended Kalman filter (EKF) is promising. This method is based on the local linearization of nonlinear model equations. However, if the nonlinearity of the model is high, it does not work well as a state observer [1].

Complex process systems are usually nonlinear and environmental noise is not Gaussian. In this case, particle filtering is suggested to solve state estimation problems [2]. While KF and EKF are based on the covariance matrix of the estimation error and solve an optimization problem, a particle filter is a Monte Carlo simulation. As this method not only yields the estimated states but their probability density, the technique can be quite well applied for fault detection purposes [3].

Although particle filter algorithms are widely used in robotics and electronics [4,5], their utilization in chemical processes is far less. The method has been used for single continuous stirred tank reactors (CSTRs) a couple of times [6,7], mainly for the purposes of process control [8,9], but rarely applied in cascade reactors and fault diagnostics. A recently

developed version of the algorithm is intelligent particle filters (IPFs) [10], but they have yet to be applied for the state estimation of reactors.

The aim of this research is to test the applicability of PF in complex process systems for state monitoring and fault diagnostics purposes. The present study examines a benchmark wastewater treatment process that well represents the aforementioned issues.

The novelty of this paper is the provision of an efficient state estimation method for cascade reactors, which is usable in fault diagnostics as well. The PF algorithm is implemented and a method suggested for optimal sensor placement according to the state estimation performance. An extensive sensitivity analysis is also proposed to determine the best settings of PF and increase the efficiency. Since intelligent particle filters (IPFs) in the field of process engineering are a research gap in the literature, one was also implemented in this cascade reactor system and its performance compared to general PF. A PF-based fault detection method was also implemented and evaluated [3]. This method was tested to determine whether it is applicable in dynamic systems or not. Bias as well as impact sensor faults were examined and a tuning method proposed.

The contributions of this paper follows the points below:

- The interpretations of particle filter and intelligent particle filter algorithms are given in detail. It is shown why they are convenient for achieving the set state estimation and fault detection goals.
- A dynamic analysis of the wastewater treatment technology benchmark is proposed and PF implemented.
- According to the state estimation performance, a method is suggested for determining optimal sensor placement.
- A sensitivity analysis is conducted and a technique given for tuning the PF.
- The performances of the general PF and IPF are compared.
- The estimation results are used for the purposes of fault detection of bias and impact sensor faults.

Beyond all these achievements, a MATLAB toolbox is provided that covers the content of this paper as well as promotes reproduction of the results and application of the method.

As for the road map of this paper, in Section 2, the methodological background of the research is introduced. In Section 2.1, the state estimation problem is formalized. The particle filter algorithm is presented in Section 2.2, and its application in fault diagnostics is introduced in Section 2.3. In Section 2.4, some ideas are given about the necessity of tuning the algorithms. The experimental results are provided in Section 3. The benchmark wastewater treatment process is introduced in Section 3.1 and its dynamic analysis proposed in Section 3.2. In Section 3.3, the implementation of PF and different case studies are shown, including a sensor placement and a parameter tuning problem; moreover, the performance of PF and IPF are compared. The fault diagnostics results are presented in Section 3.4. Finally, in Section 4, the main conclusions are summarized and possible future research directions determined.

2. Methodological Background

In this section, the methodological background of this research is introduced. The state estimation and fault detection problems are defined, the PF and IPF algorithms described in detail; moreover, the applied fault detection technique is explained.

2.1. Formalization of the State Estimation Problem

In complex process systems, not every state variable is measured because they are immeasurable or the minimum number of sensors is applied to reduce costs. Furthermore, measured data are usually noisy because every measurement has an element of uncertainty. However, the exact values of the state variables must be known, e.g., in process control. State observers are able to solve these outlined problems.

The algorithms they use can estimate the unmeasured states according to the measured ones. For the systems where model equations are nonlinear, the application of a particle

filter state observer is suggested. In contrast to the more commonly used Kalman filter, the PF does not require the model equations to be linear nor supposes a Gaussian model and measurement noises. Preliminarily, only the initial state and state space model of the process system are needed.

Since PF estimates states according to the model of the system, besides measurement noise, the uncertainty of the model is also a factor that has to be considered. PF can handle these uncertainties, as these two types of noises are incorporated in the applied state space model:

$$\mathbf{x}_k = f(\mathbf{x}_{k-1}, \mathbf{u}_{k-1}) + \mathbf{v}_k \quad (1)$$

$$\mathbf{y}_k = h(\mathbf{x}_k, \mathbf{u}_k) + \mathbf{w}_k \quad (2)$$

where $\mathbf{x}_k$ denotes the state vector, $\mathbf{y}_k$ stands for the measurement vector, $\mathbf{u}_k$ represents the input vector, $\mathbf{v}_k$ and $\mathbf{w}_k$ are the model and measurement noises, respectively, at time k, $f()$ refers to the state transfer function and $h()$ represents the measurement function.

Since PF estimates the unmeasured states according to the measured data, it is essential to determine which state variable is most suitable to measure because the accuracy of the estimation depends on it. Otherwise, the parameters of the particle filter algorithm itself have to be tuned. These questions have to be answered before using the PF to achieve the best estimation performance.

In process systems, different faults can occur. One of the most common are sensor faults which is a key issue in PF-based state estimation. This means that the information gained from the process is incorrect. As the operation of the observer is based on the measured data, the estimations of the state variables will also be faulty. However, for the same reason, this is not perceptible without implementing a fault diagnostics algorithm.

The probabilistic approach of the PF allows the faults to be perceived. PF not only yields the most probable values but the probability density of the states too. With a carefully chosen fault decision function, incorrect operation of the sensor can be detected in time from this additional information collected. Therefore, it is possible to repair or replace the faulty sensor and accurately continue the state estimation.

The aforementioned concept is shown in Figure 1.

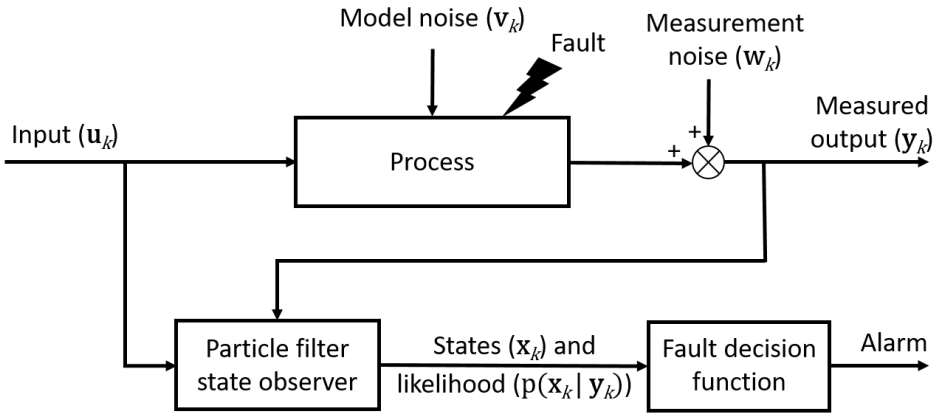

Figure 1. Particle filter state observer-based fault detection scheme of a process. The particle filter state observer computes the unmeasured states in light of the measured sensor data and the inputs of the process. According to the fault decision function, the faults can be detected and alarms generated. Faults can be technological faults, parameter uncertainties or sensor faults.

2.2. Particle Filter-Based State Estimation

A particle filter (PF) is a kind of Monte Carlo simulation in which a set of particles represents the different states of the system.

The particles are distributed according to the system and measurement noises in the state space. The coordinates of the particles are the state variables, and every particle has a weight that represents the probability of the particle. At every time step, the PF computes

the probability of each particle according to the given new measurement data, thereby calculating the estimated states.

The algorithm of the general particle filter consists of three parts: the prediction step, the correction step, and the resampling step. The method is described in more detail in this section of the paper. An intelligent particle filter (IPF) is an improved version of the general particle filter that uses a genetic algorithm in the resampling step to achieve a higher level of efficiency. This subject is covered below as well.

2.2.1. Description of the PF Algorithm

In order to use a particle filter, the state-space model of the process (Equations (1) and (2)), the initial states and the measurement data of at least one state variable must be known.

The initial state is defined by a set of randomly drawn particles from the state space around the real initial state. This set of particles is defined by their localization at the state space and their weights as $\{\mathbf{x}_0^i, \omega_0^i\}_{i=1}^{N_s}$, where N_s denotes the number of particles. Similarly, the set of particles at time k is represented by $\{\mathbf{x}_k^i, \omega_k^i\}_{i=1}^{N_s}$. The initial weights of the particles are equal.

The posterior probability density function, the so-called posterior pdf, is sought at every time step during the estimation. The prior pdf is given at every k. It is represented by the initial set of particles at time $k = 1$. If $k > 1$, the prior pdf at k equals the estimation result of the last time step, that is, the posterior pdf at $k - 1$. The prior pdf can be approximated to a sampled density:

$$p(\mathbf{x}_k \mid \mathbf{y}_{1:k}) \approx \sum_{i=1}^{N_s} \omega_k^i \delta(\mathbf{x}_k - \mathbf{x}_k^i) \equiv \{\mathbf{x}_k^i, \omega_k^i\}_{i=1}^{N_s} \tag{3}$$

where $\mathbf{x}_k$ refers to the estimated state vector, ω_k^i represents the normalized weights of the particles and $\mathbf{y}_{1:k}$ stands for the measured data points from the first to the kth time step. In order to determine the posterior pdf, the task is to estimate these weights at every time step.

The first step of the particle filter algorithm is the prediction step. Every particle progresses forward by one time step; thus, the following pdf is calculated:

$$p(\mathbf{x}_{k+1} \mid \mathbf{y}_{1:k}) = \{\mathbf{x}_{k+1}^i, \omega_k^i\}_{i=1}^{N_s} \tag{4}$$

The new $\mathbf{x}_{k+1}^i$ state vectors can be determined using the state transfer function and the ω_k^i weights remain the same in this step of the algorithm.

The second step of the algorithm is the correction step. The new ω_{k+1}^i weights are estimated from the previous ones (ω_k^i) recursively according to the new measurement data and normalized as:

$$\omega_{k+1}^i = \frac{\omega_k^i \cdot p(\mathbf{y}_k \mid \mathbf{x}_k^i)}{\sum_{i=1}^{N_s} \omega_k^i \cdot p(\mathbf{y}_k \mid \mathbf{x}_k^i)} \tag{5}$$

Since $\mathbf{x}_{k+1}^i$ remains the same in this step of the algorithm, the posterior pdf is defined as:

$$p(\mathbf{x}_{k+1} \mid \mathbf{y}_{1:k+1}) = \{\mathbf{x}_{k+1}^i, \omega_{k+1}^i\}_{i=1}^{N_s} \tag{6}$$

The prediction and correction steps are repeated iteratively. Although these two steps provide the core of the algorithm, different problems can occur during the estimation, so additional steps are required. These improved algorithms will be introduced in the following subsections.

The derivations of the equations above are available in [1,11]. The prior pdf was used as an importance function and a Markov process assumed.

2.2.2. Resampling-Based Improvement of the PF Algorithm

Using only the prediction and correction steps iteratively, a degeneracy problem is faced after a few time steps. One or a few particles will have large weights and all the others will be negligibly small. This phenomenon leads to less accurate estimation results with an unnecessarily high computational effort. To avoid this problem, a resampling step in the algorithm must be used.

The set of particles can be evaluated in terms of degeneracy with the effective sample size:

$$N_{eff} = \frac{1}{\sum_{i=1}^{N_s} \left(\omega_k^i\right)^2} \tag{7}$$

If N_{eff} is below an N_t threshold after the correction step, resampling takes place. In other cases, the current particle set is good enough to continue the estimation with the prediction step. N_t is conveniently chosen as half the value of N_s. While more than half of the particles are effective, the estimation is sufficiently robust.

Resampling means that a new particle set is produced. The new particles are chosen from the old ones. Every $\mathbf{x}_k^i$ is drawn with the probability of ω_k^i-normalized weight, so the probability density of the particle set remains the same as before. The weights of the new particles are identical.

Four main resampling methods exist: multinomial, stratified, systematic, and residual resampling. Systematic resampling was used in the present research, which is shown as the most favorable in [12]. A detailed description of this method is given in Appendix A.

2.2.3. A New Variant of PF: The Intelligent Particle Filter (IPF)

Besides the degeneracy problem, sample impoverishment is another inconvenience of general particle filters. Therefore, the variance in the particle states is relatively small, so many particles will be located in almost the same place. This phenomenon can mislead the estimation and reduce the level of accuracy. A good solution to the problem is to apply an intelligent particle filter (IPF).

This algorithm implements a genetic algorithm between the correction and resampling steps in order to increase the diversity and broaden the range of particles. A description of the algorithm is given in Algorithm 1.

As can be seen in Algorithm 1, the genetic algorithm in the PF consists of two main steps: crossover and mutation. During the crossover step, the small-weight particles are modified into larger ones. The α parameter determines how strong the effect of small-weight particles is during the modification.

On the other hand, the mutation step extends the range of the particles and further increases the level of diversity. It has a p_M parameter that defines the probability of modification.

The two parameters of Algorithm 1 (α and p_M) have to be tuned before using the method, depending on the system to be estimated.

In this paper, the performance of the PF and IPF are compared. It is shown that by using an IPF, the accuracy of the estimation is somewhat increased. The metric we used to evaluate the performances of the two methods is the one proposed in [10]:

$$e_{av} = \frac{1}{N_r \cdot T} \sum_{r=1}^{N_r} \sum_{k=1}^{T} e_{rk} \tag{8}$$

where N_r denotes the number of simulations, T refers to the time steps in one simulation and e_{uk} stands for the absolute value of the difference between the estimated and real states. We calculated e_{av} for each state variable and made our conclusions in this manner.

Algorithm 1 Genetic algorithm-based part of the intelligent particle filter [10]

(1) Calculate N_{eff} from the normalized particle weights after the correction step.
(2) Set the particles in descending order according to their weights.
(3) Choose the weight that belongs to the N_{eff}th particle and classify the particles into the "large weight" group if $\omega_k^i > \omega_k^{N_{eff}}$ and "small weight" group in all other cases. $\mathbf{x}_{kH}^l$ denotes large-weight particles and $\mathbf{x}_{kL}^j$ the small-weight ones.
(4) Crossover step: modify the small-weight particles and make them more efficient by an arithmetic crossover:

$$\mathbf{x}_{kC}^j = \alpha \mathbf{x}_{kL}^j + (1-\alpha)\mathbf{x}_{kH}^l \tag{9}$$

where $\mathbf{x}_{kC}^j$ refers to the new particle that $\mathbf{x}_{kL}^j$ is replaced by and $\alpha \in [0,1]$. For every small-weight particle, a large-weight one is randomly selected.
(5) Mutation step: further modify the small-weight particles and broaden their range:

$$\mathbf{x}_{kM}^j = \begin{cases} 2\mathbf{x}_{kH}^l - \mathbf{x}_{kC}^j & if\ r_L \leq p_M \\ \mathbf{x}_{kC}^j & if\ r_L > p_M \end{cases} \tag{10}$$

where $r_L \in [0,1]$ is drawn from the uniform distribution and p_M denotes the mutation probability.
(6) Reevaluate the ω_k^i particle weights.

2.3. Incorporation of the PF Method in Fault Diagnostics

One of the main application areas of particle filtering is fault diagnostics, which forms the main subject of this paper.

The main steps of the proposed method are summarized in Figure 2. After the dynamic analysis of the benchmark system, the PF is implemented. Now, the measured state variable must be chosen and the PF algorithm appropriately tuned. After these preliminary steps, the method can be tested for fault detection problems.

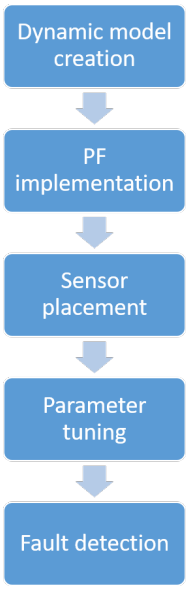

Figure 2. Outline of the main steps of the proposed method. The proposed approach covers the tasks from the creation of the dynamic model through the implementation of the PF to its application in fault detection.

Traditional fault detection techniques are based on forming residuals. These methods calculate the difference between the estimated and measured states as well as detect faults according to this value. However, these methods require multiple sensors (e.g., simplified

observer scheme (SOS)) or even multiple state observers (e.g., generalized observer scheme (GOS)), leading to increases in computational and instrumentation costs [13].

Knowing the probability distribution of the states, i.e., the weights of the particles at every time step, enables faults to be detected. Some practical examples can be seen in a three-tank and a helicopter system in [10,14], respectively. However, very few papers in the literature have dealt with the chemical engineering application in operational units.

In the present work, the method proposed in [3] was applied, which is based on the sum of the log-likelihood of particle weights which decreases if a fault occurs.

First of all, the measurement function has to be evaluated for each particle, i.e., the likelihood of the measurement, assuming that the state represented by the particle is the real state. The next step is to determine its average in terms of particles:

$$L_k = \frac{1}{N_s} \sum_{i=1}^{N_s} p(\mathbf{y}_k \mid \mathbf{x}_k^i) \tag{11}$$

The model parameters are assumed to be constant.

The normalized decision function is defined as:

$$d_k = \frac{1}{M} \sum_{j=k-M+1}^{k} ln(L_j) \tag{12}$$

where M denotes the so-called sliding window and k stands for the time at that moment.

If $-d_k$ is higher than a threshold h, a fault is detected and an alarm generated. In this paper, the method introduced in [10] is used to determine h, which selects the $-d_k$ that belongs to the 98% confidence interval of the fault-free operation as a threshold. Selecting a proper threshold is a trade-off. The lower h is, the more false alarms are generated. However, if h is too high, faults are detected later after their occurrence, and some faults even can be missed. Thereby, 98% confidence interval of $-d_k$ in fault-free operation seems an appropriate choice as false alarms (type I error) are generated only the 2% of fault-free cases; moreover, faults are detected quite early after the occurrence.

A log-likelihood-based fault detection method is also used for chemical engineering examples in [15]. However, this work only examines the processes and the fault detection technique in steady-state cases using general PF. We tested the method simultaneously with the dynamic behavior of the benchmark system by applying the PF and IPF state estimation algorithms and proposed an analysis in the cases of impact and bias additive sensor faults.

2.4. Challenges of Tuning the Algorithms

From the aspect of increasing the efficiency of estimation and fault detection, it is essential that the algorithms work with the proper parameters. In the present case, the PF and even the fault decision function have tunable parameters.

On the one hand, the number of particles (N_S) in the PF is crucial. As PF is a Monte Carlo simulation, one of its main disadvantage is the high simulation time that mainly depends on this parameter. Therefore, a smaller value of N_S is preferable. However, reducing N_s decreases the accuracy of the estimation. In Section 3.3.2, a tuning method is proposed concerning this topic.

On the other hand, it is also important to tune M in the fault decision function that refers to the sliding window. A higher value of M yields a more filtered $-d_k$ signal but can cause detrimental effects on the fault detection efficiency. Section 3.4.2 gives a detailed analysis of this subject.

3. Results and Discussion

In this section, some ideas about the technology that was chosen to implement the aforementioned experiments are given. Results are also presented.

Firstly, the dynamic analysis of the system is presented and the test ranges determined. Afterwards the implementation of PF and its sensitivity analysis are conducted. Some ideas are also given about the optimal location of sensors according to the state estimation performance. The estimation efficiency is increased by implementing an IPF and its performance is also compared with that of the PF in this section. At the end of this study, the results of PF-based fault detection experiments are shown and a method given for tuning the sliding window parameter.

The above-mentioned experiments were executed on the dynamic model of an organic-carbon-removal wastewater treatment process in which nonlinear behavior excludes the use of the Kalman filter.

3.1. Introduction of the Wastewater Treatment Process

In our research, we examined the behavior of the standard reactor cascade (SRC) model of the organic-carbon-removal wastewater treatment technology. Since the reactions that take place in the reactors can be modeled by Monod growth kinetics, the process accurately represents the nonlinear behavior of complex systems [16].

The task of this process is to remove the carbon content of sewage using microorganisms. The system consists of two bioreactors in series and a settling unit, as is seen in Figure 3. In the reactors, biological organic carbon removal from the wastewater takes place. The feed flows into Reactor I with the recycled stream. The effluent of this unit forms the influent of Reactor II. The purified wastewater flows into the settling unit, which separates the light and heavy phases. A part of the heavy phase is recycled to enhance efficiency. The reactors are assumed to be well stirred and aerated; moreover, their volumes are equal.

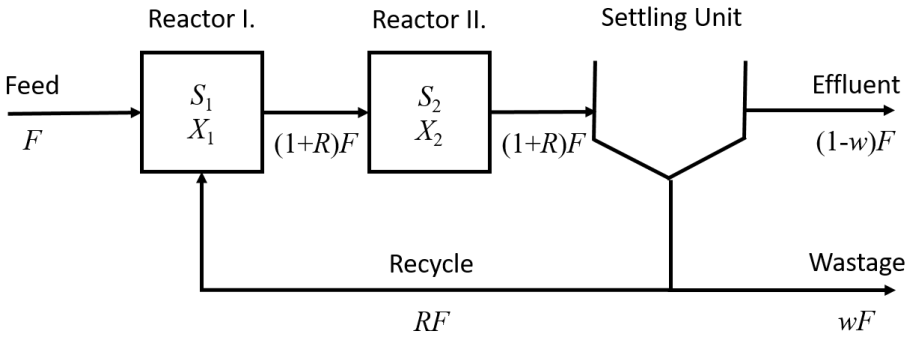

Figure 3. Block diagram of the standard reactor cascade (SRC) of the wastewater treatment process. The figure contains the names of the flows above the arrows and the flowrates below the arrows. F refers to the flowrate, R denotes the recycle ratio, S_i and X_i represent the concentrations of the substrate and microorganisms (i stands for the number of the reactors), and w is the waste fraction.

Four state variables can be determined during the process, namely the concentrations of the substrate and microorganisms in both reactors. However, measuring all these concentrations is expensive. This problem can be avoided by using a state observer. In this case, it is sufficient to install only one sensor and estimate the other three concentrations, thereby reducing investment costs. As the wastewater treatment process is nonlinear, a particle filter is suggested as a state observer.

Although this wastewater treatment process was thoroughly analyzed under steady-state conditions, a dynamic analysis was not carried out [16]. As the temporal behavior of the system was examined, firstly, its dynamic behavior needed to be analyzed and its physical constraints determined.

Since the biochemical reactions in the process can be described by Monod growth kinetics, the four model equations of the system can be defined as:

$$V\frac{\mathrm{d}S_1}{\mathrm{d}t} = F(S_0 - S_1) + RF(S_2 - S_1) - V\frac{\mu_m S_1 X_1}{K_s + S_1} \cdot \frac{1}{\beta} \tag{13}$$

$$V\frac{dX_1}{dt} = -FX_1 + RF(CX_2 - X_1) + V\frac{\mu_m S_1 X_1}{K_s + S_1} - Vk_d X_1 \quad (14)$$

$$V\frac{dS_2}{dt} = F(1+R)(S_1 - S_2) - V\frac{\mu_m S_2 X_2}{K_s + S_2} \cdot \frac{1}{\beta} \quad (15)$$

$$V\frac{dX_2}{dt} = F(1+R)(X_1 - X_2) + V\frac{\mu_m S_2 X_2}{K_s + S_2} - Vk_d X_2 \quad (16)$$

where bottom indexes refer to the parameters of the first (1) and second (2) reactors, V denotes the reactor volume, and S and X stand for the concentrations of the substrate and microorganisms. The input variables of the system are the feed concentration S_0, flowrate F, and recycle ratio R. The other parameters of Equations (13)–(16) along with their names and values can be seen in Table 1.

Table 1. Constant parameters of the model equations. We used the parameter values in [16]. The reactor volume used is from [17].

Label	Name	Value
μ_m	maximum specific growth rate [day^{-1}]	1
β	yield factor [mg MLSS/mg COD]	0.5
K_s	Monod constant [mg COD/L]	100
C	concentration factor [-]	1.5
k_d	death coefficient [day^{-1}]	0.028
w	waste fraction [-]	0.1
V	reactor volume [m^3]	125

C denotes the so-called concentration factor that refers to the performance of the settling unit. Its minimum value is $C = 1$, which means that the unit functions as a recycling unit. For the experiments, an intermediate value for C was chosen which determines the maximum value of R as follows:

$$R_{max} = \frac{1 - wC}{C - 1} \quad (17)$$

where w denotes the waste fraction, which is an empirical constant. $R_{max} = 1.7$ in the present case.

3.2. Dynamic Analysis of the System

The system has three input variables: S_0, R, and F. The reason why S_0 changes is the fluctuation in the composition of the sewage over time. R and F are manipulated variables that the operators in the plant can handle to achieve the required level of performance. However, these variables have physical constraints to preserve the stability of the washout solution and avoid process failures [16].

The maximum value of F can be calculated from the critical residence time which is a function of R and S_0, as introduced in [16]. The detailed equations can be found in Appendix B.

F_{max} determines the boundary of the washout phenomenon. The values of F_{max} belong to the R and S_0 values that were examined and are shown in Table 2.

As can be seen in Table 2, F_{max} depends very little on S_0. However, R has a great effect on it which must be considered when choosing the setpoints.

Table 2. Maximum values of F given different R and S_0 values. The same feed concentration $S_0 = 4000$ mg COD/L was used as in [16]. Another two values above and below this value were chosen. Regarding R, the same values were used as in [16].

S_0 [mg COD/L]	$R = 0.5$ [-]	$R = 1.0$ [-]	$R = 1.5$ [-]
3000	267	438	916
4000	270	442	923
5000	271	444	928

After examining the constraints, the dynamic model and simulation of the system were created in MATLAB. Equations (13)–(16) were discretized and solved. As a first step, validation of the model was necessary. With parameters $C = 1.5$ and $R = 1.5$, the system achieved 90% carbon removal when the residence time was about 0.1545 days, which is in line with the results in the literature [16]. The setpoint changes and results are shown in Figure 4:

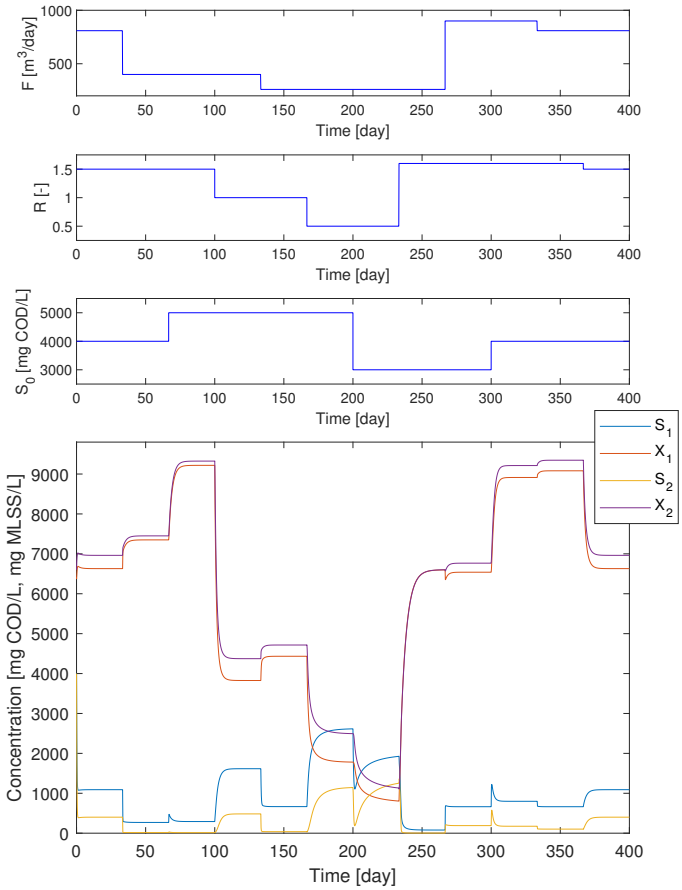

Figure 4. The input and state variables over time.According to Table 2, $F \leq F_{max}$ in every period of time. Equations (13)–(16) were discretized and solved iteratively with a sampling time of $dt = 0.01$ days to acquire the model outputs. The initial state vector was $\mathbf{x_0} = [S_{1,0}\ X_{1,0}\ S_{2,0}\ X_{2,0}] = [4000\ 6400\ 4000\ 6600]$.

It can be concluded that the system is underdamped and yields a first-order response following the changes to R, F, and S_0. The time constants of the process are a few days long as wastewater treatment is a slow process. They are higher in the periods when F is close to the value of F_{max}. Another interesting observation is that changes to S_0 cause sudden,

impulse-like responses in S_1 and S_2, but the stationery values of S_2 stay almost unchanged. This can be explained by the fact that S_0 directly affects the substrate concentrations; however, once the microorganism concentration adapts to this change, the process can achieve nearly the same level of performance as before.

Regarding the dynamic experiments above, the model and the set of input variables are suitable for further investigations.

3.3. Implementation of the Particle Filter

After analyzing the dynamic behavior of the system, a particle filter state observer was implemented to see if the state estimation task was feasible. We used a MATLAB implementation based on [1] and modified it according to the current research goals.

First of all, since the program had to be validated, it was tested to see if the estimated states followed the real states. State S_2 was measured first, as it is the most important of the four, being the one to be controlled. A total of 50 particles was used for the estimation.

The particle paths and the median of the particle states are in line with the real state represented by blue lines, as can be seen in Figure 5. Although only the first 40 days were illustrated due to visibility, conclusions are the same throughout the entire time period.

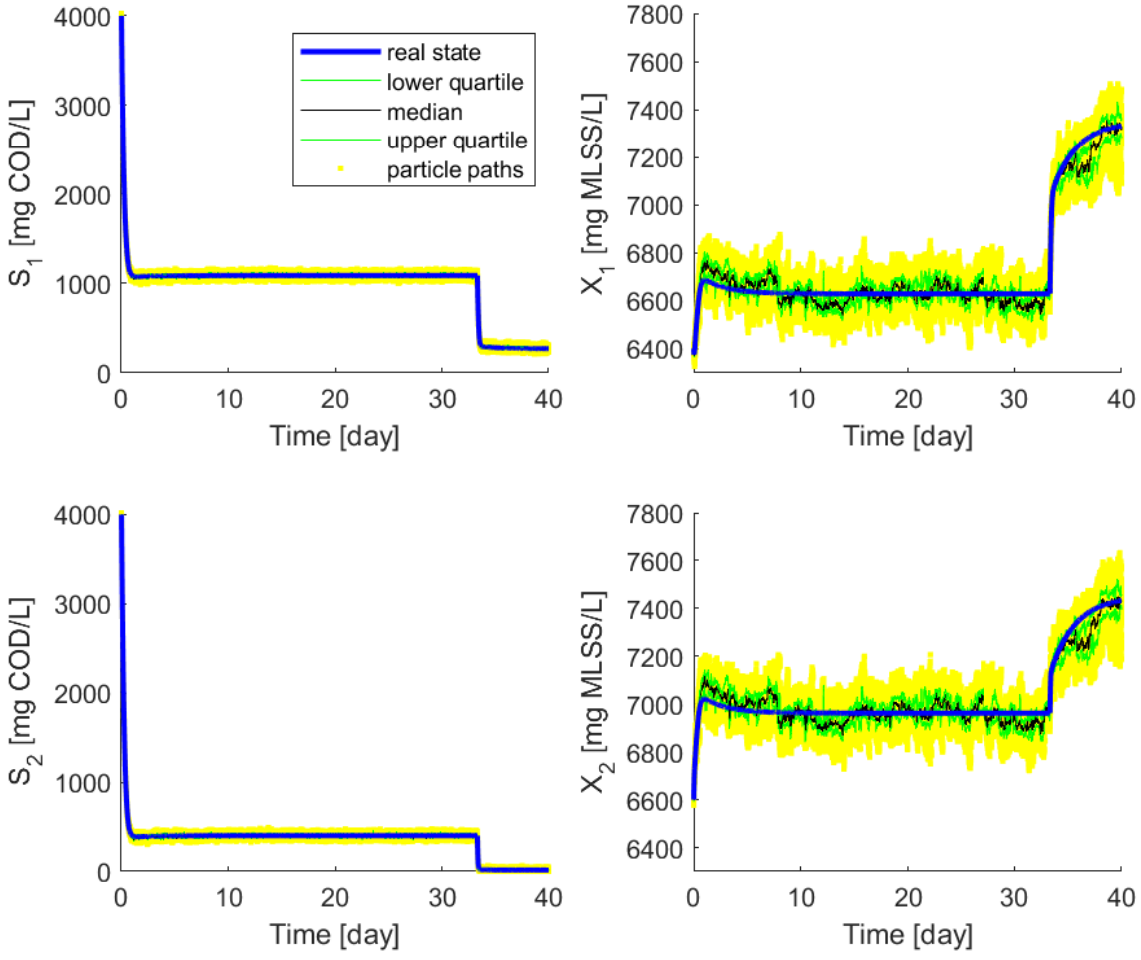

Figure 5. Comparison of the estimated and real states. The concentrations of the substrate are shown in mg COD/L and of the microorganisms in mg MLSS/L. Deviations of $\sigma_v = 10$ and $\sigma_w = 25$ were used for noises.

Regarding the performed experiments, this MATLAB implementation of the PF works well and is suitable for further examinations.

3.3.1. Method of Sensor Placement

The next emerging issue is to determine which state variable is the most suitable to measure. Despite the fact that an explicit cost function is not available, the different configurations are comparable according to the estimation errors. Therefore, PF is used, the basic performance of which is satisfactory as introduced before. The placement of the sensor is determined by these measures before the PF is fine-tuned in Section 3.3.2.

Online COD and suspended solids sensors are also used in wastewater treatment, such as in [18], of different accuracies, namely: 5% for the former and about 2% for the latter. These values were used to define the standard deviations of the measurement noise in the simulation. Different measuring ranges in the two cases were also considered, so $\sigma_w = 25$ was used for the COD and $\sigma_w = 140$ for the suspended solids sensors.

Regarding the above-mentioned issues, the estimation performance was examined in all four cases by using 50 particles as well. For the evaluation, Equation (8) was used to calculate the estimation error.

According to Table 3, the cases when the concentration of microorganisms was measured produced much lower estimation errors because they are more sensitive to changes in the input variables, as seen in Figure 4. Therefore, if the measured state is chosen from them, PF can provide a more accurate estimation. Another reason is that the measurement noises are relatively lower in these cases. The estimation performances are almost the same when comparing X_1 and X_2 as measured states. However, since the accuracy of S_2 is the most crucial state in terms of the process control, X_1 is measured in this wastewater treatment model.

Table 3. Error of the estimated states (e_{av}) at different sensor locations. The error values were calculated as an average of $N_r = 10$ simulations over the first 40 days (with a sampling time of $dt = 0.01$ days corresponding to $T = 4000$ time steps). $N_S = 50$.

Measured State	$e_{av}(S_1)$	$e_{av}(X_1)$	$e_{av}(S_2)$	$e_{av}(X_2)$
S_1 [mg COD/L]	7.17	25.45	4.99	26.44
X_1 [mg MLSS/L]	4.43	17.37	4.23	17.36
S_2 [mg COD/L]	5.67	25.58	6.80	26.49
X_2 [mg MLSS/L]	4.42	16.52	4.31	17.62

Another interesting observation is that by considering the rows in Table 3 separately, the same types of concentrations are almost identical. Moreover, any variation is always for the benefits the unmeasured state, e.g., when measuring S_1, the estimation of S_2 is more accurate than that of S_1.

It is important to note that the results described above pertain to the actual process model with the current parameters. For example, the value of the measurement noise has a significant impact on them. However, besides the given parameters and state-space model, the optimal sensor location can be determined by the aforementioned method.

3.3.2. Sensitivity Analysis of the PF

As the previous example illustrated, some assumptions and parameters of the process significantly affect the effectiveness of state estimation. On the other hand, the parameters of the particle filter itself are also worth tuning. It was stated in Section 2.4 that N_S exhibits an opposite effect on the estimation accuracy and simulation time. Therefore, this section aims to determine its optimal value for the present system.

Regarding the results of Section 4, we chose X_1 as a measured state and executed the estimation using four different values of N_s. The results are depicted in Figure 6.

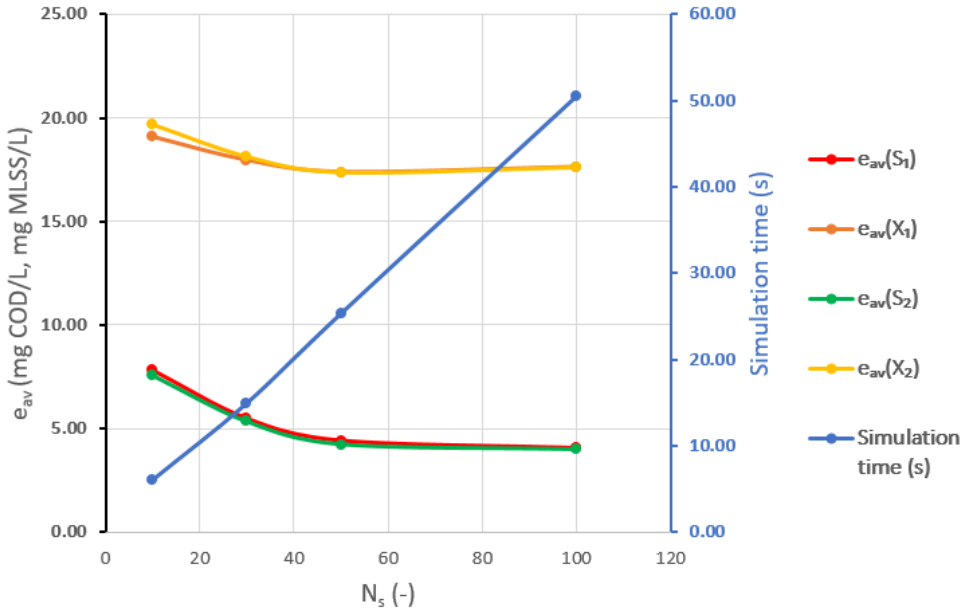

Figure 6. The average estimation (absolute) errors (e_{av}) and the simulation time for different numbers of particles (N_s). $N_S = 10,\ 30,\ 50$, and 100 were simulated. The presented simulation times refer to the duration of one simulation ($T = 4000$). The number of simulations is $N_r = 10$.

During the evaluation, estimation errors were considered against the simulation time. The estimation errors decrease exponentially and become nearly constant beyond a determined value of N_S. On the contrary, the simulation time increases linearly as the number of particles rises. According to the curves of Figure 6, the optimal N_S is about 30. Below this value, estimation errors increase rapidly, while above it, they remain nearly constant. For further investigations, this value will be used.

In the presented case study, the simulation time is not a crucial factor as the time constant of the system is a few days. However, for the state estimation of more complex systems with a shorter time constant, hundreds of particles may be needed and the simulation time can be a constraint, since it should always be lower than the sampling time of the system.

According to these principles, the N_s parameter of the PF can be tuned in a similar manner, even for other processes.

3.3.3. PF and IPF Comparison

The simulation time and estimation errors are conflicting factors, as was seen in Section 3.3.2. The question arises whether estimation accuracy could be increased without significant growth in the simulation time. A good opportunity in this regard is to test IPF in the present environment, which improves the level of efficiency by handling the phenomenon of sample impoverishment of general PF. IPF has not yet been applied in biochemical systems nor reactors. Its performance is tested in this section, besides setpoint changes.

The MATLAB program was coded according to the algorithm shown in [10]. In order to be used, firstly, the two parameters (α, p_M) of the algorithm had to be tuned. This task was executed in a similar manner to the above-mentioned issues: by keeping in mind the average error, the optimal set of the parameters was determined. With this method, $\alpha = 0.2$ and $p_M = 0$ were derived. Therefore, for the purpose of increasing the accuracy of the estimation, the particles only need to be better distributed in the same interval and it is not necessary to broaden it. From this result, it can be concluded that sample impoverishment is not a severe problem in this system. However, the level of efficiency could also be increased in this case by distributing the particles better.

The tuned parameters were used to test the performance of the IPF compared to that of the PF.

The results are shown in Table 4. It can be seen that the IPF performs better in all states but particularly in X_1 and X_2. Although the IPF contains one more calculation step, the simulation times are almost identical because of the number of resamplings in the two cases. Investigating this issue, the average number of resamplings was 170 in the case of the PF and 158 in the case of the IPF ($N_r = 10$). Therefore, if the IPF is used, resampling is needed less frequently.

Table 4. Comparison of the PF and IPF with regard to the average error (e_{av}) and simulation time. $N_r = 10$, $N_S = 30$, $T = 4000$.

	PF	IPF
$e_{av}(S_1)$ [mg COD/L]	5.50	5.09
$e_{av}(X_1)$ [mg MLSS/L]	17.96	15.55
$e_{av}(S_2)$ [mg COD/L]	5.34	4.88
$e_{av}(X_2)$ [mg MLSS/L]	18.12	15.72
Simulation time [s]	14.89	14.26

Regarding the results of this section, it is recommended to use IPF instead of general PF to enhance the estimation accuracy.

As Figures 7 and 8 present, the aforementioned conclusions also have pertinence in the case of different N_s values. Furthermore, it can be concluded that $N_s = 30$ is still the optimal value of the number of particles, even if IPF is applied.

To check if IPF's performance is better than the PF's even besides more setpoint changes, a longer part of the simulation was even used to compare the two algorithms in the case of the optimal $N_s = 30$. It contained three setpoint changes so that the dynamic behavior could be examined more profoundly. The conclusions were the same as above, namely the estimation error and number of resamplings were also lower in the case of IPF.

With this method, differences in the performances of the two algorithms can be investigated and the optimal value of the N_s parameter determined.

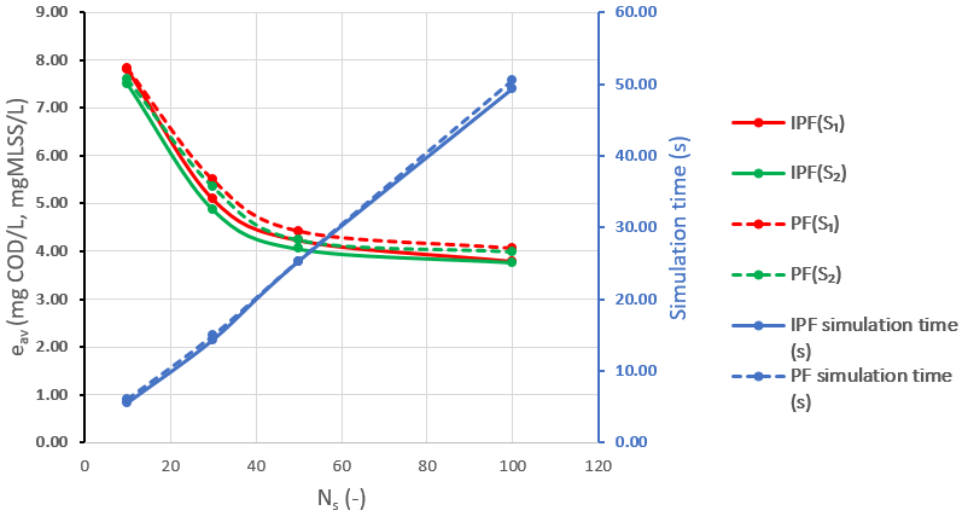

Figure 7. Comparison of PF and IPF: the estimation errors (e_{av}) of the substrate concentrations and the simulation time regarding the different numbers of particles. The curves belonging to PF are represented by dashed lines and those belonging to IPF by continuous ones. The red and green colors refer to the S_1 and S_2 state variables, respectively. The other parameters are the same as before ($N_S = 10,\ 30,\ 50, 100$; $T = 4000$; $N_r = 10$).

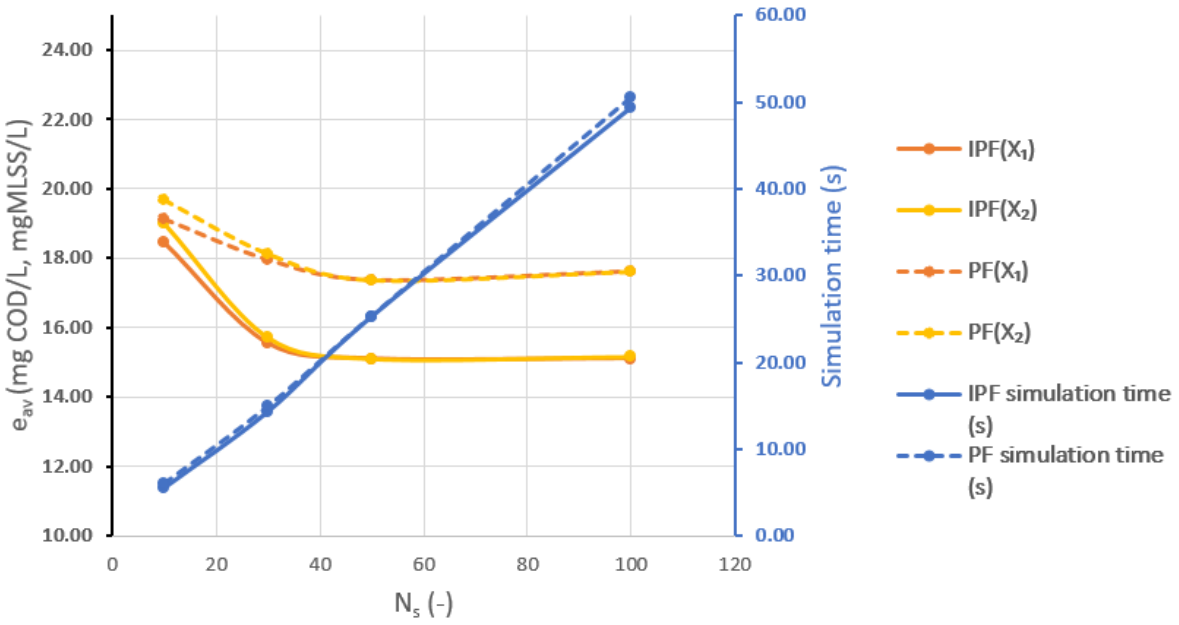

Figure 8. Comparison of PF and IPF: the estimation errors (e_{av}) of the concentrations of microorganisms and the simulation time regarding the different numbers of particles. The curves belonging to PF are represented by dashed lines and those belonging to IPF by continuous lines. The orange and yellow colors refer to the X_1 and X_2 state variables, respectively. The other parameters are the same as before ($N_S = 10,\ 30,\ 50, 100$; $T = 4000$; $N_r = 10$).

3.4. Application of PF and IPF in Fault Diagnostics

With the carefully chosen and tuned MATLAB implementation of the particle filter, different fault detection problems can be solved. General PF and the IPF were also tested for this aim, and a comparison is presented in this section.

As state estimation with particle filters is based on the information gained from the process, it is essential to have fault-free measurement data. If the sensor is faulty, the estimation will be inaccurate, and it is not observable according to the estimation results. However, the particle filter is able to indicate the faults by following the method described in Section 2.3.

Fault occurrence is evaluated according to the $-d_k$ value which is the negative sum of logarithms of the particle weights. If a faulty measurement occurs, the sum of the particle weights decreases, so $-d_k$ increases. With a carefully chosen threshold, the fault can be detected.

First of all, the fault-free operation was tested, and the threshold belonging to its 98% confidence interval determined. Results are similar in the cases of general PF and IPF. The distribution of $-d_k$ values along with the threshold are shown in Figure 9 in the case of the IPF.

3.4.1. Analysis of the Fault-to-Signal Ratio

A sensor fault can be modeled by an additive constant in the measurement function:

$$\mathbf{y}_k = h(\mathbf{x}_k, \mathbf{u}_k) + \mathbf{w}_k + \mathbf{f} \tag{18}$$

where $\mathbf{f}$ represents the fault. One of the main factors in fault detection is the fault-to-signal ratio. Therefore, the lowest value of $\mathbf{f}$ that can already be detected must be determined.

Four different values of $\mathbf{f}$ were examined: 5, 10, 15, and 20% of the value of measured state X_1 at the first stationary state. The fault occurred on the 25th day and lasted until the 55th day, as is shown in Figure 10. Furthermore, two setpoint changes were modeled

around the 33rd and 66th days, one within the faulty time interval and the other after the fault had been rectified.

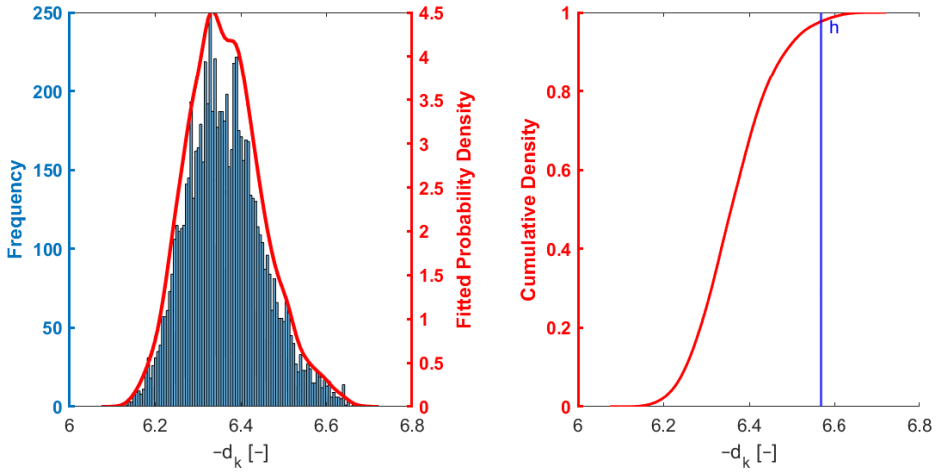

Figure 9. Fault-free operation with the IPF: the histogram, empirical probability density function (pdf), and cumulative distribution function (cdf) of $-d_k$. Threshold h is denoted by a vertical line. The sliding window is $M = 50$. $T = 8000$.

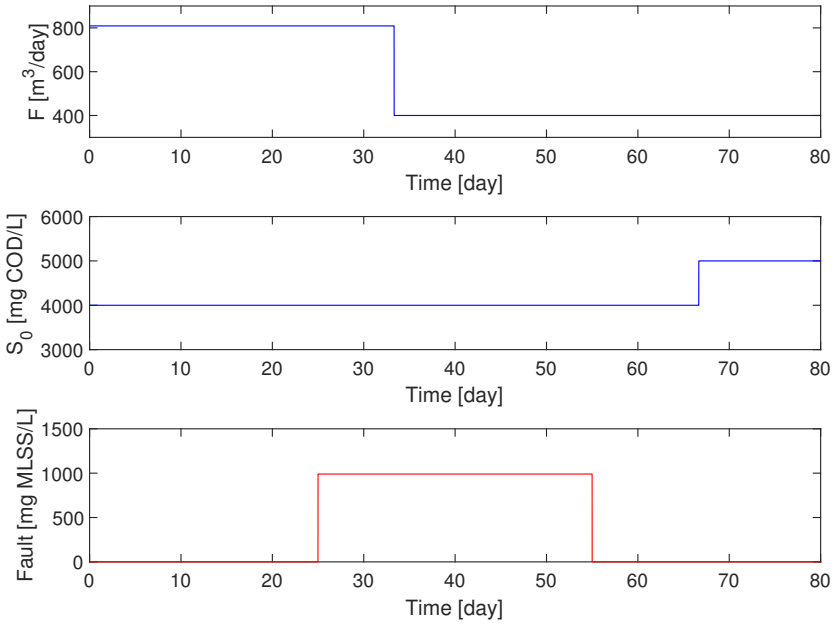

Figure 10. Operation with a fault: setpoint changes and the fault. The fault refers to the case of a 15% fault which is 990 mg MLSS/L. $T = 8000$.

The results are illustrated in Figure 11. It can be seen that $-d_k$ is quasi-constant during fault-free operations, so setpoint changes do not have an effect on it and the dynamic behavior of the process does not influence the efficiency of fault detection. Otherwise, the moment of fault occurrence is detected in all cases. However, $-d_k$ is once again quickly below the threshold in terms of the 5% and 10% faults. In the case of the 15% and 20% faults, it is above the threshold for almost the whole duration of the fault. As can be seen in Figure 11, results are similar in the cases of general PF and IPF.

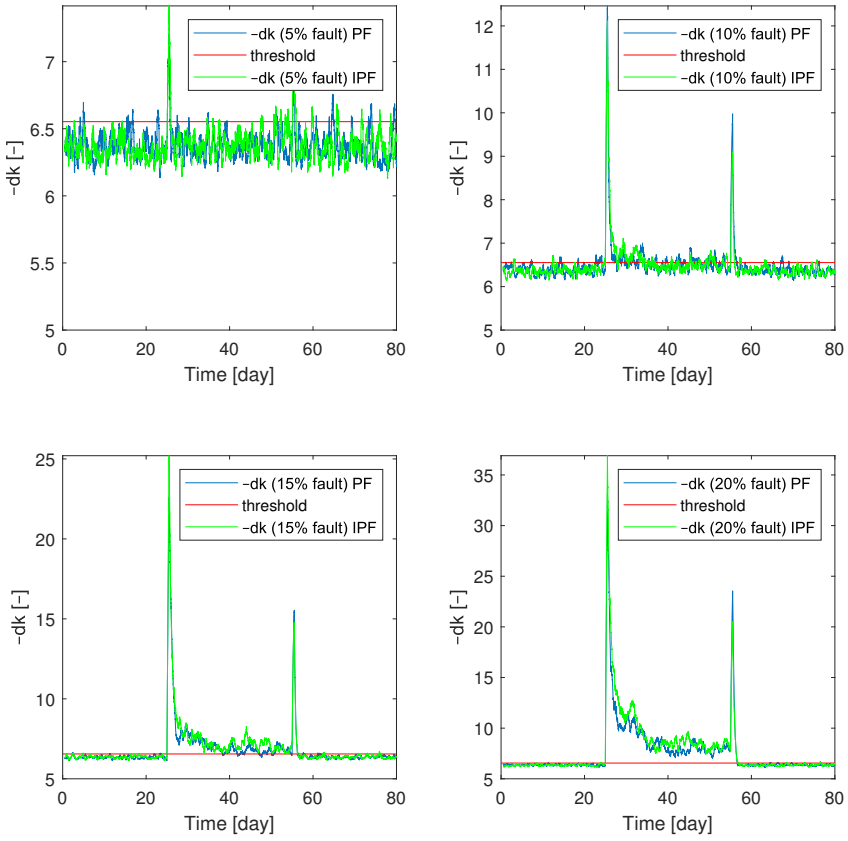

Figure 11. Operation with a fault: results according to different fault amplitudes belonging to general PF and IPF. One of the setpoint changes occurs within the fault duration and one once it has been resolved, as is seen in Figure 10. The threshold is $h = 6.57$, calculated by the 98% confidence interval of the fault-free operation. Sliding window is $M = 50$. $T = 8000$.

The aforementioned difference between the small and large fault amplitudes is explained by Figures 12 and 13. These results show that in the case of minor faults, the estimated state is in line with the measured values after the fault occurs. However, in the case of the 20% fault, a residual error is recorded not only between the estimated and real states but even between the estimated and measured states. General PF and IPF behave similarly even in this case.

On the one hand, the reason for this phenomenon is that during the resampling step, the PF chooses new particles from the set of the prior particles. If the change is very significant, resampling cannot follow it. On the other hand, the particles try to converge with the real state as the state space model is deterministic.

For the aforementioned reasons, $-d_k$ does not return to below the threshold while the fault lasts.

Once the fault has ceased, $-d_k$ returns to its initial value in all the cases in Figure 11; moreover, the rectification of the fault is indicated by a spike.

It was noticed that in the case of minor faults, only the beginning and the end of the faulty time interval could be detected. However, considering the number of resamplings, information could be gained about the existence of the fault. As is shown in Table 5, since the average number of resamplings per day is higher while the fault lasts, the duration of the fault can be unambiguously determined by following the tendency of $-d_k$ and the number of resamplings.

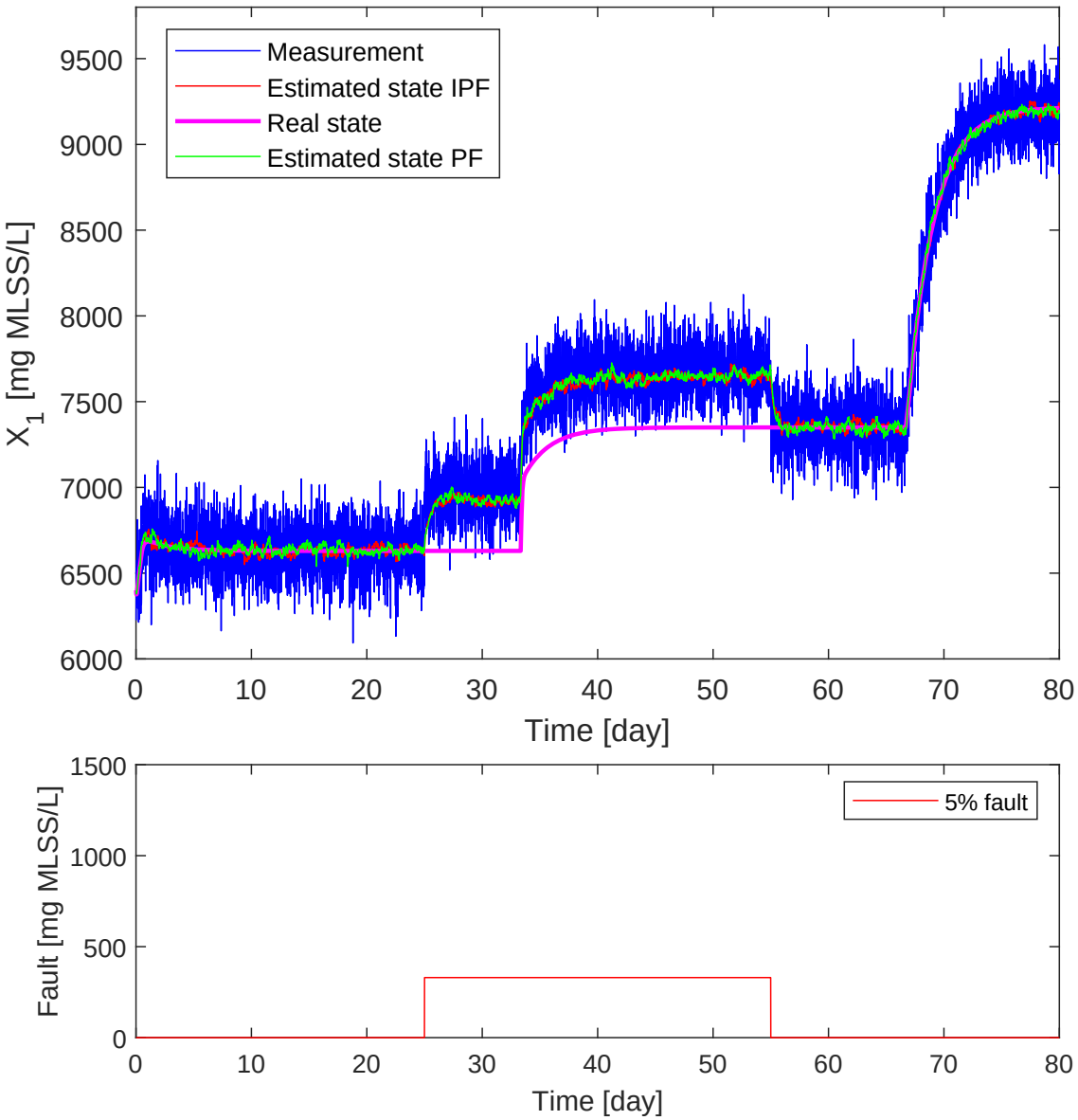

Figure 12. Real, measured, and estimated values of the state variable X_1 by the 5% fault. The value of the 5% fault is 330 mg MLSS/L. $T = 8000$.

Table 5. Average number of resamplings per day by different values of the faults in case of the IPF. $T = 8000$.

Fault [%]	Before the Fault [#/Day]	During the Fault [#/Day]	After the Fault [#/Day]
5%	3.64	5.70	4.56
10%	3.84	10.73	4.68
15%	3.64	18.93	5.04
20%	3.80	27.43	5.64

The effect of the setpoint changes is insignificant. In the case of the fault-free interval (setpoint change in S_0), the setpoint change is not perceptible in Figure 11. The setpoint change in F while the fault occurs is also negligible. For further investigations, the case of the 15% fault was chosen as it is the lowest that gives an alarm throughout the entire duration of the fault according to $-d_k$. As general PF and IPF showed similar behavior, henceforth the IPF is used to present the results.

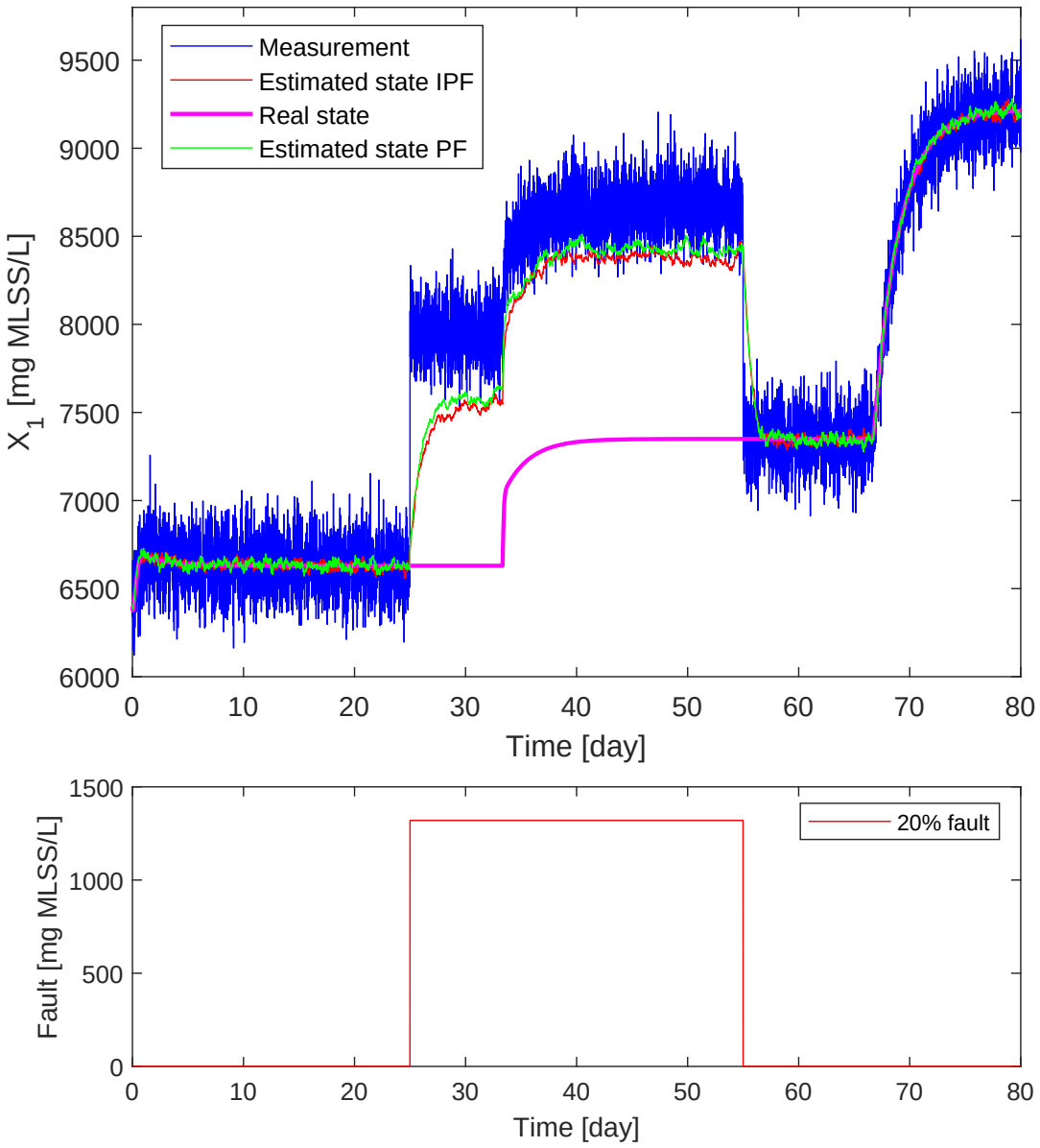

Figure 13. Real, measured, and estimated values of the state variable X_1 by the 20% fault. The value of the 20% fault is 1320 mg MLSS/L. $T = 8000$.

3.4.2. Tuning the Sliding Window

In the fault decision function (Equation (12)), the M sliding window parameter functions as a smoothing parameter. The effect of this parameter on the noisiness of the $-d_k$ signal is shown in Figure 14. If M is small, the fluctuations in $-d_k$ are large, while if M is larger, $-d_k$ is more filtered.

To determine the optimal value of M, impact faults were also considered. The results are shown in Figure 15. It can be concluded that the larger M is, the less acute the spikes are and the intensity of the fault signal decreases.

According to these aspects, if the aim is to rapidly detect impact faults and sudden changes, a smaller M is suggested. However, if the duration of the bias fault is wanted to be indicated, a larger M is preferred. In the present case, $M = 50$ is optimal to detect both bias and impact faults.

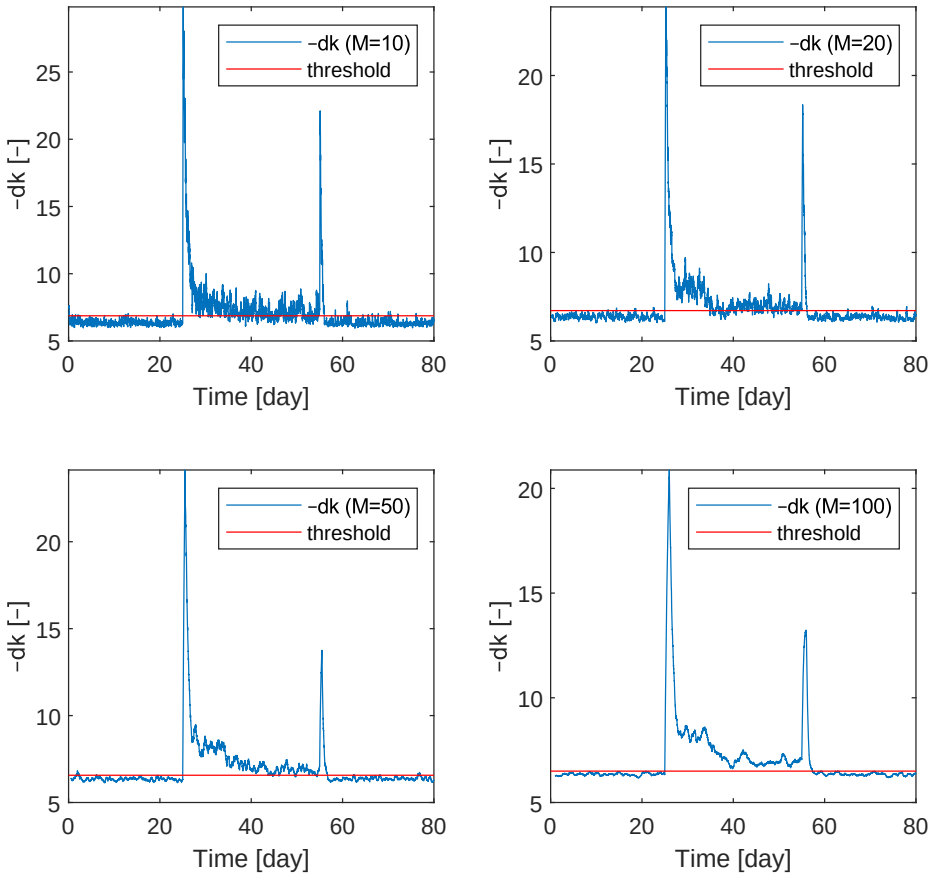

Figure 14. Sensitivity analysis by the sliding window (M): bias faults. Bias sensor faults are defined in the same way as above. $M = 10$, 20, 50, and 100 time steps. The 15% fault was used, that is, 990 mg MLSS/L. $T = 8000$.

Besides the conclusions already made, the aforementioned experiments also point out that the implementation of IPF instead of general PF does not require changes to be made to the fault detection method. It can also be applied as in the case of general PF and is able to detect both bias and impact faults with high efficiency.

The threshold values used above, calculated by the 98% confidence interval, are summarized in Table 6. As the M sliding window functions as a smoothing parameter, the larger M is, the narrower the distribution of $-d_k$ that results. Therefore, h decreases as M increases.

Table 6. Summary of h threshold values.

M **[#]**	***h*** **[-]**
10	6.87
20	6.71
50	6.57
100	6.50

In this section, a PF-based fault detection technique was introduced. Based on the given method, it is feasible to determine the h threshold of alarm generation as well as tune the value of the M sliding window according to how the system responds to bias and impact sensor faults. To detect the existence of bias faults more accurately, the number of resamplings should also be considered.

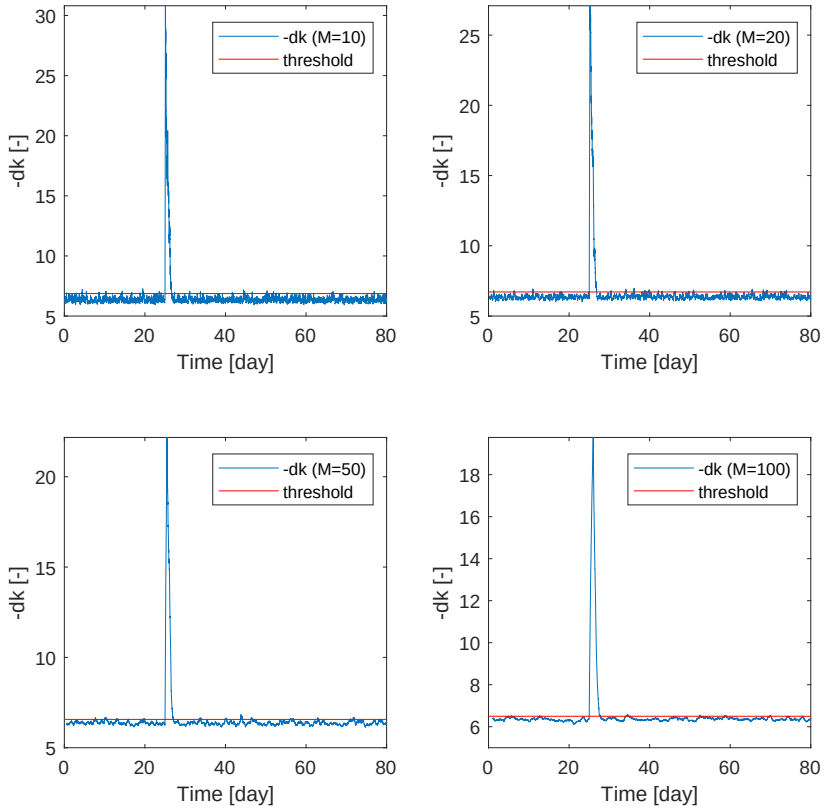

Figure 15. Sensitivity analysis by the sliding window (M): impact faults. The fault occurs from the 25th to the 26th day. $M = 10$, 20, 50, and 100 time steps. The 15% fault was used, that is, 990 mg MLSS/L. $T = 8000$.

4. Conclusions

In complex process systems, a state observer is required to estimate the unmeasured states. As these systems are often nonlinear, a particle filter (PF) is suitable for the task.

The present research work aimed to investigate the applicability of a particle filter state observer in process systems for the purposes of state estimation and fault diagnostics tasks through a case study. The PF was validated for and implemented in the system as a state observer; moreover, unmeasured states were estimated based on the measured one.

One contribution of the paper is a method obtained to determine the optimal sensor placement. Based on the principles declared in this paper, the method is usable in all cases by applying the actual state-space model and parameters.

Another novelty is a technique for tuning the PF and a proposed sensitivity analysis. To increase the estimation accuracy, a new variant of the PF, the genetic algorithm-based IPF, was implemented and tested under setpoint changes. A case study was presented to compare the performances of the two algorithms.

The completed state estimation algorithm was also tested for fault detection problems. A PF-based fault detection technique was implemented and evaluated. The effect of the setpoint changes was examined and a tuning method presented for the sliding window by also taking into consideration the bias and impact faults.

As applications of PF in process engineering are scarce, this area formed the subject of our research. The aforementioned experiments were executed in a carbon-removal wastewater treatment process that consists of a two-element reactor cascade and a settling unit. After validating the wastewater treatment model, according to the dynamic analysis of the system, a proper set of setpoint changes was determined considering the physical constraints of the process. This scenario was used for further investigations.

The PF performed well in the benchmark system since the estimated states closely followed the real ones. According to the average values of the estimation errors (e_{av}), the

preferable measured state variable was chosen. Considering the simulation time and e_{av} values, the optimal value of the N_s parameter of the PF was determined. An IPF was implemented in the system and compared with the general PF. This improved method reduced the estimation errors and even the number of resamplings.

As for fault detection, bias and impact sensor faults were extensively analyzed. The thresholds of alarm generation were determined according to the $-d_k$ output of the decision function and sensor faults with various amplitudes were examined to evaluate the efficiency of fault detection. A comparison was also made between general PF and IPF from fault detection aspects. The M parameter of the decision function was also tuned; moreover, it was declared that the number of resamplings is also worthwhile following. With the applied method, bias and impact faults were also detected highly efficiently.

In the future, several research directions are plausible, as the field of particle filtering is very diverse and understudied. For example, fault detection performance could be examined for actuator and process faults. It would also be interesting to investigate the applicability of the technique with regard to estimating inputs or parameters. In terms of chemical engineering, other operational units could be studied or PF could be tested with regard to process control.

Author Contributions: Conceptualization and supervision, J.A.; writing and visualization, É.K. All authors have read and agreed to the published version of the manuscript.

Funding: This work has been implemented by the OTKA 143482 (Monitoring Complex Systems by goal-oriented clustering algorithms) and the TKP2021-NVA-10 projects with the support provided by the Ministry for Innovation and Technology of Hungary from the National Research, Development and Innovation Fund, financed under the 2021 Thematic Excellence Programme funding scheme. Éva Kenyeres was supported by the Foundation of the University of Pannonia.

Data Availability Statement: The MATLAB codes that were used during the experiments are accessible at https://github.com/abonyilab/ParticleFilter_in_FDI (accessed on 19 January 2023), written by Éva Kenyeres.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviation

The following symbols are used in this manuscript:

$\mathbf{x}_k$	estimated state vector at time k
$\mathbf{y}_k$	measurement vector at time k
$\mathbf{u}_k$	input vector at time k
$\mathbf{v}_k$	model noise at time k
$\mathbf{w}_k$	measurement noise at time k
$f()$	state transfer function
$h()$	measurement function
N_S	number of particles
$\mathbf{x}_0^i$	initial state vector of the ith particle
ω_0^i	initial weight of the ith particle
$\mathbf{x}_k^i$	estimated state vector of the ith particle at time k
ω_k^i	weight of the ith particle at time k
$\mathbf{y}_{1:k}$	measurement data points from time 1 to time k
$\delta()$	Dirac delta function
N_{eff}	effective sample size
N_t	threshold of N_{eff}
$\mathbf{x}_{kH}^j$	large-weight particles before crossover step at time k
$\mathbf{x}_{kL}^j$	small-weight particles before crossover step at time k
$\mathbf{x}_{kC}^j$	new particles after crossover step at time k
$\mathbf{x}_{kM}^j$	new particles after mutation step at time k
α	parameter of the IPF's crossover step
p_M	parameter of the IPF's mutation step

N_r	number of simulations
T	number of time steps
e_{uk}	absolute error at time k in the uth simulation
e_{av}	average absolute error
L_k	average likelihood of the measurement by particles at time k
M	sliding window
d_k	normalized fault decision function at time k
h	fault threshold
F	flowrate
R	recycle ratio
S_i	substrate concentration in the ith reactor
X_i	concentration of microorganisms in ith reactor
S_0	substrate concentration of the feed entering the first reactor
w	waste factor
t	time
V	reactor volume
μ_m	maximum specific growth rate
β	yield factor
K_S	Monod constant
C	concentration factor
k_d	death coefficient
n	volume ratio of the two reactors
b_{SRC}	parameter for calculating critical residence time
τ^*_{min}	dimensionless critical residence time
$S_{i,0}$	initial substrate concentration in the ith reactor
$X_{i,0}$	initial concentration of microorganisms in the ith reactor
σ_v	deviation of the model noise
σ_w	deviation of the measurement noise
f	fault
#	count of occurences

Appendix A. Systematic Resampling

After the correction step, the ω_i-normalized weights of the particles become available. As they are normalized, the cumulative sum of the weights is:

$$Q^{(m)} = \sum_{i=1}^{m} \omega_i \ \in [0,1] \tag{A1}$$

where $1 \leq m \leq N_S$.

During the systematic resampling, the $[0,1]$ interval is divided into N_S subintervals equidistantly. From every subinterval, a z_j number is chosen. z_1 is randomly drawn from the uniform distribution on $(0,1]$, and the others are determined by z_1:

$$z_j = z_1 + \frac{j-1}{N_S}, \quad j = 2,3,\ldots,N_S \tag{A2}$$

These z_j $(j = 1,2,\ldots,N_S)$ values are used to select the new particles from the old ones. $\mathbf{x}_m$ is chosen when:

$$Q^{m-1} < z_j \leq Q^m \tag{A3}$$

Therefore, the particles are selected with a probability equal to their weights. The new particles are considered to be of equal weights.

Appendix B. Derivation of the Critical Residence Time and F_{max}

The critical residence time can be calculated in the following way:

$$b_{SRC} = (4Cn + (1-n)^2)R^2 + 2(2Cn + (1-n)^2)R + (1-n)^2 \tag{A4}$$

$$\tau^*_{min} = \frac{1+\frac{S_0}{K_S}}{2(\frac{S_0}{K_S}-(1+\frac{S_0}{K_S})\frac{k_d}{\mu_m})} \cdot \frac{(1+n)(1+R)-\sqrt{b_{SRC}}}{n} \tag{A5}$$

where τ^*_{min} refers to the dimensionless critical residence time and n denotes the volume ratio of the two reactors. As the reactor volumes are assumed to be equal, $n = 1$ in the present case, therefore, Equations (A4) and (A5) are simplified to the following forms:

$$b_{SRC} = 4CR^2 + 4CR \tag{A6}$$

$$\tau^*_{min} = \frac{1+\frac{S_0}{K_S}}{2(\frac{S_0}{K_S}-(1+\frac{S_0}{K_S})\frac{k_d}{\mu_m})} \cdot (2(1+R)-\sqrt{b_{SRC}}) \tag{A7}$$

F_{max} can be calculated from τ^*_{min}:

$$F_{max} = \frac{V\mu_m}{\tau^*_{min}} \tag{A8}$$

References

1. Arulampalam, M.; Maskell, S.; Gordon, N.; Clapp, T. A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking. *IEEE Trans. Signal Process.* **2002**, *50*, 174–188. [CrossRef]
2. Stelzer, I.; Kager, J.; Herwig, C. Comparison of Particle Filter and Extended Kalman Filter Algorithms for Monitoring of Bioprocesses. In *Computer Aided Chemical Engineering*; Elsevier: Amsterdam, The Netherlands, 2017. [CrossRef]
3. Kadirkamanathan, V.; Li, P.; Jaward, M.H.; Fabri, S.G. Particle filtering-based fault detection in non-linear stochastic systems. *Int. J. Syst. Sci.* **2002**, *33*, 259–265. [CrossRef]
4. Elfring, J.; Torta, E.; van de Molengraft, R. Particle Filters: A Hands-On Tutorial. *Sensors* **2021**, *21*, 438. [CrossRef] [PubMed]
5. Hongliang, L.; Yannian, H.; Jianglei, Q.; Haocheng, S. Fault diagnosis using particle filter for MEA typical components. *J. Eng.* **2018**, *2018*, 603–606. [CrossRef]
6. Anandhakumar, K.; Ali, I.S.M.K.; Selvakumar, K.; Raja, K. State estimation of a nonlinear system using particle filter. In Proceedings of the 2014 IEEE International Conference on Computational Intelligence and Computing Research, Coimbatore, India, 18–20 December 2014; pp. 1–4. [CrossRef]
7. Mansouri, M.; Nounou, H.; Nounou, M. State estimation of a chemical reactor process model—A comparative study. In Proceedings of the 10th International Multi-Conferences on Systems, Signals & Devices 2013 (SSD13), Hammamet, Tunisia, 18–21 March 2013; pp. 1–6. [CrossRef]
8. Ikonen, E. Particle Filtering for Open-Loop Process Control. Technical Report. 2006, Volume 30. Available online: http://sun3.oulu.fi/~iko/MGDT/MGDT-UsersGuide.pdf (accessed on 2 March 2023).
9. Dias, A.C.S.R.; da Silva, W.B.; Dutra, J.C.S. Propylene polymerization reactor control and estimation using a particle filter and neural network. *Macromol. React. Eng.* **2017**, *11*, 1700010. [CrossRef]
10. Yin, S.; Zhu, X. Intelligent particle filter and its application to fault detection of nonlinear system. *IEEE Trans. Ind. Electron.* **2015**, *62*, 3852–3861. [CrossRef]
11. Rawlings, J.; Mayne, D. *Model Predictive Control: Theory and Design*; Nob Hill Publishing: Madison, WI, USA, 2009.
12. Hol, J.D.; Schon, T.; Gustafsson, F.K. On Resampling Algorithms for Particle Filters. In Proceedings of the 2006 IEEE Nonlinear Statistical Signal Processing Workshop, Cambridge, UK, 13–15 September 2006; pp. 79–82.
13. Frank, P.M. Fault Diagnosis in Dynamic Systems via State Estimation—A Survey. In *System Fault Diagnostics, Reliability and Related Knowledge-Based Approaches*; Tzafestas, S., Singh, M., Schmidt, G., Eds.; Springer: Dordrecht, The Netherlands, 1987; pp. 35–98.
14. Kanthalakshmi, S.; Raghappriya, M. Active fault diagnosis of 2 DoF helicopter using particle filter-based log-likelihood ratio. *Int. J. Control* **2022**, *98*, 3148–3165. [CrossRef]
15. Alrowaie, F.; Gopaluni, R.; Kwok, K. Fault detection and isolation in stochastic non-linear state-space models using particle filters. *Control. Eng. Pract.* **2012**, *20*, 1016–1032. [CrossRef]
16. Nelson, M.I.; Bradshaw-Hajek, B.H. An analysis of organic carbon removal in a two-reactor cascade with recycle and a two-reactor step-feed cascade with recycle. *Asia-Pac. J. Chem. Eng.* **2020**, *15*, e2392. [CrossRef]

17. Grady, C.P.L. Environmental science and pollution control. In *Biological Wastewater Treatment*, 2nd ed.; Grady, C.P.L., Jr., Daigger, G.T., Lim, H.C., Eds.; Marcel Dekker: New York, NY, USA, 1999.
18. Dablowski, B. Controlling Biomass Quantity, Quality at MBR Plant Requires Accurate, Real-Time MLSS Measurement. In *Membranes Supplement*; PennWell Corporation: Nashville, TN, USA, 2009. Available online: https://in.hach.com/asset-get.download.jsa?id=7639984593 (accessed on 2 March 2023).