Computers and Chemical Engineering 180 (2024) 108469

Contents lists available at ScienceDirect

Computers and Chemical Engineering

journal homepage: www.elsevier.com/locate/compchemeng

On the use of machine learning to generate *in-silico* data for batch process monitoring under small-data scenarios

Luca Gasparini [a,b], Antonio Benedetti [c], Giulia Marchese [d], Connor Gallagher [e], Pierantonio Facco [a], Massimiliano Barolo [a,*]

[a] CAPE-Lab – Computer-Aided Process Engineering Laboratory, Department of Industrial Engineering, University of Padova, via Marzolo 9, 35131 Padova PD, Italy
[b] INSTM – Consorzio Interuniversitario Nazionale per la Scienza e la Tecnologia dei Materiali, via Giusti 9, 50121 Firenze FI, Italy
[c] Process Engineering & Analytics, Medicine Development and Supply, GSK R&D, Park Rd Ware, SG12 0DP, United Kingdom
[d] Process Engineering & Analytics, Medicine Development and Supply, GSK R&D, Gunnels Wood Rd, Stevenage, SG1 2NY, United Kingdom
[e] Process Engineering & Analytics, Medicine Development and Supply, GSK R&D, 1250 S. Collegeville Road, Collegeville, PA 19426, United States

ARTICLE INFO

Keywords:
Batch processes
Small data
Big data
Machine learning
Process monitoring
Biopharmaceutical industry
Pharmaceutical engineering

ABSTRACT

Batch process monitoring using principal component analysis requires sufficient historical manufacturing data to model the normal operating conditions of the process. However, when a new product is to be manufactured for the first time in a given facility, very limited historical data are available, thus entailing a small-data scenario. We thoroughly investigate and improve a data-driven methodology, previously reported in the literature (Tulsyan, Garvin & Ündey (2019). *J. Process Control*, **77**, 114–133), that enables batch process monitoring under such type of scenarios. The methodology exploits machine learning algorithms (based on Gaussian process state-space models) to generate *in-silico* batch trajectory data from the few available historical ones, and then uses the overall pool of real and *in-silico* data to build a process monitoring model. We develop automatic procedures to tune the values of several parameters of this machine-learning framework, in such a way that the generation of consistent *in-silico* batch trajectory data can be streamlined, thus facilitating the deployment of the framework at an industrial level. Furthermore, we develop indicators and a metric to assist the *in-silico* data generation activity from a process monitoring-relevant perspective. Finally, using datasets from a benchmark simulated semi-batch process for the manufacturing of penicillin, we thoroughly investigate the appropriateness of the *in-silico* generated data for the purpose of process monitoring.

1. Introduction

In batch and semi-batch manufacturing, reproducibility (or consistency) across batches is required to guarantee that the end-product quality targets are met after every batch. Whether or not a new batch conforms to a set of "normal" batches that were run in the past can be assessed by using a data-driven process monitoring framework, where the most widely used one exploits multivariate statistical techniques, such as principal component analysis (PCA; (Jackson, 1991; Wise and Gallagher, 1996; Kourti, 2003)). The rationale behind this method is quite simple: (*a*) collect a set of historical batches that were run satisfactorily ("normal" batches); (*b*) build a PCA model on the trajectories of all measured variables across all normal batches; (*c*) using statistical control charts built through the PCA model, test whether a new batch can be considered normal; (*d*) if it is not, raise an alarm (fault diagnosis may then follow). This approach is very effective (Kourti et al., 1995; Reis and Gins, 2017), but suffers from a strong limitation: the required number of historical batches, which are needed to identify the set of normal operating conditions (NOC), is usually large (big-data scenario) (Chiang et al., 2022). This prevents it from being used when a new product is to be manufactured in a given facility for the first time. In fact, in this case, the number of historical batches is usually very small, because almost no history of past manufacturing of that particular product in that particular facility is available. This small-data scenario is also referred to as a low-*N* one, *N* being the number of available historical batches for the product being manufactured. As an industrially relevant example, low-*N* scenarios are frequently encountered by biopharmaceutical industries, for which first-principles modeling (Rato et al., 2020) is often impractical or impossible. In fact, for a new biotechnological product that is manufactured at a clinical or commercial scale (from 2k to 20k liters), the number of available historical

* Corresponding author.
E-mail address: max.barolo@unipd.it (M. Barolo).

https://doi.org/10.1016/j.compchemeng.2023.108469
Received 12 June 2023; Received in revised form 11 September 2023; Accepted 17 October 2023
Available online 18 October 2023
0098-1354/© 2023 The Authors. Published by Elsevier Ltd. This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/).

List of symbols

$\|\|*\|\|$	euclidean norm of a given vector
$* \mid *$	conditional density of the left-hand side random vector with respect to the right hand-side one
$\mathbf{a},\ \mathbf{b}$	scaled versions of $\widetilde{\mathbf{a}}, \widetilde{\mathbf{b}}$ respectively
$\widetilde{\mathbf{a}}, \widetilde{\mathbf{b}}$	vectors extracted from the $\underline{\mathbf{Z}}$ matrix
A	number of principal components
ceil	ceiling operator, $\text{ceil}(x) = \min\{x' \in \mathbb{Z} \mid x' \geq x\}$
$c^{(s)}$	cardinality of $\mathscr{Z}^{(s)}$ normalized with respect to the one of $\mathscr{Z}$
$c^{(v)}(t,\tau)$	normalized estimation of the correlation for the variable v between the time points t and $t-\tau$
$\widetilde{c}^{(v)}(t,\tau)$	estimation of the correlation for the variable v between the time points t and $t-\tau$
$C(J,w,h)$	auxiliary coefficient for the computation of $P(J,\ w)$
$D(J,h)$	number of sequences of J elements that contain exactly h different (specific) ones
$\widetilde{D}(J,w,h)$	number of sequences of J elements that contain exactly h different ones, choosing them from w different elements
$\mathscr{D}_{v,t}$	training dataset with reference to the variable v and time-point t
floor	floor operator, $\text{floor}(x) = \max\{x' \in \mathbb{Z} \mid x' \leq x\}$
F	symbol denoting an F-type continuous probability density
$F_{*,*}$	F density with specified degrees of freedom
$\underline{\mathbf{G}}$	generic 3D matrix
$\left[\underline{\mathbf{G}}\right]_{i_1,i_2,i_3}$	entry of the generic 3D matrix $\underline{\mathbf{G}}$ in position (i_1,i_2,i_3)
h	generic number of different high-frequency time points caught in a resampling interval
i, i_1, i_2, i_3	generic indexes to identify an element in a column vector or a matrix (either 2D or 3D)
i_{NOC}	indicator of coverage
$i_{\text{var}}, i_{\text{mean}}$	auxiliary variables for the computation of i_{NOC}
I_t	set of all the indexes of the J high-frequency time points from the t-th resampling interval
$I_{\mathscr{Z}}$	interval in which the fitting of a probability density function must be performed
$\mathbf{I}_{K\times K}$	$K \times K$ identity matrix
$\underline{\mathbf{I}}$	3D matrix containing *in-silico* batches
j	generic index identifying a batch from the stratified resampling among J ones
$j_\tau^{\min}(n,v,t),\ j_\tau^{\max}(n,v,t)$	auxiliary variables for the partitioning along J
J	number of resampled trajectories for each variable
$\mathscr{J}(n,v,t)$	set containing the indexes of retained trajectories after the partitioning along J with reference to (n,v,t)
$\mathscr{J}_\tau(n,v,t)$	auxiliary set for the partitioning along J
k	low-frequency resampling interval
K	generic number of retained trajectories for each variable with the partitioning
$K(v,t)$	number of retained trajectories with the partitioning for the v-th variable and t-th time point
$K_n(v,t)$	number of retained trajectories with the partitioning for the n-th real batch, the v-th variable and t-th time point
$\mathscr{K}(\cdot,\cdot)$	kernel function
$\mathscr{K}(\mathbf{Z}_{t-L,\dots,t-1})$	kernel covariance matrix
L	memory parameter
$L(v,t)$	memory parameter for the t-th low-frequency time point of the v-th variable
m	high-frequency sampling interval
$\mathbf{M}$	generic 2D matrix
$[\mathbf{M}]_{i_1,i_2}$	entry of the generic 2D matrix $\mathbf{M}$ in position (i_1,i_2)
n	generic index identifying a real batch
$\widetilde{n}$	generic index identifying a row in $\mathbf{X}^{\text{cal}}$
N	number of available real batches
$\widetilde{N}$	total number of batches among the real ones and the *in-silico* ones
$\mathscr{N}$	Gaussian density (possibly multivariate)
$\mathscr{N}(*,*)$	$\mathscr{N}$ density with specified mean and covariance
$\mathbb{N}$	set of the natural numbers
p_J	required probability that every high-frequency time point is caught
$P(J,h)$	probability that by randomly resampling J times one achieves h different points in total
$\widetilde{P}(J,w)$	lower-bound for $P(J,h)$
q	generic index identifying a high-frequency time point (ranging from 1 to $\bar{T}$)
$\widetilde{\mathbf{q}}^{(j)}$	T-uple containing the high-frequency time points selected for the j-th resampled trajectory
Q	residual statistics
Q_{lim}	statistical limit for Q
r	generic index identifying an *in-silico* batch
R	number of generated *in-silico* batches
$\bar{R}^2$	lower bound for the variance to be explained when calibrating a PCA model with $\mathbf{X}^{\text{cal}}$
$\mathbb{R}$	set of the real numbers
s	generic index identifying a subinterval in $I_{\mathscr{Z}}$
S	number of rectangles for the fitting of a probability density function (subintervals of $I_{\mathscr{Z}}$)
t	generic index identifying a low-frequency time point (ranging from 1 to T)
T	number of low-frequency time points
T^2	Hotelling statistics
T^2_{lim}	statistical limit for T^2
$\bar{T}$	number of high-frequency time points
u	sample from the discrete uniform density in $\{1,\dots,w\}$
v	generic index identifying a process variable
$\mathbf{v}$	generic column vector
$[\mathbf{v}]_i$	entry of the generic column vector $\mathbf{v}$ in position i
V	number of process variables
w	number of high-frequency time points per resampling interval
x, x'	generic numbers
$\widehat{\mathbf{x}}$	vector extracted from $\mathbf{X}^{\text{cal}}$
$\mathbf{x}, \mathbf{x}', \mathbf{x}''$	auxiliary vectors derived from $\widehat{\mathbf{x}}$ for the computation of i_{NOC}
$\mathbf{x}_i, \mathbf{x}_j$	generic arguments for the $\mathscr{K}$ function
$\underline{\mathbf{X}}$	3D matrix of the real batches
$\mathbf{X}^{\text{cal}}$	calibration matrix containing down-sampled real batches and *in-silico* ones
$\mathbf{y}$	auxiliary vector derived from $\widehat{\mathbf{x}}$ for the computation of i_{NOC}
$\mathbf{y}_t$	measured values of a specific variable among the K considered trajectories at time point t
$z, z_1, \dots, z_{\widetilde{N}}$	generic values representing either T^2 or Q values
z', z''	extreme values of $I_{\mathscr{Z}}$
$z^{(1)}_{t-\tau}, \dots, z^{(K)}_{t-\tau}$	values of the K retained trajectories respectively at the time point $t-\tau$ for a specific variable
$\mathbf{z}^{(i)}_{t-L,\dots,t-1}$	vector containing the values of the i-th retained trajectory across the L time points for a specific variable
$\underline{\mathbf{Z}}$	3D matrix of the resampled trajectories
$\mathbf{Z}_{t-L,\dots,t-1}$	matrix containing the values of all the retained trajectories across the L time points for a specific variable
$\mathscr{Z}$	whole set of either T^2 or Q statistics for $\mathbf{X}^{\text{cal}}$
$\mathscr{Z}^{(s)}$	subset of $\mathscr{Z}$ containing the elements belonging to $I^{(s)}_{\mathscr{Z}}$
$\mathbb{Z}$	set of the integer numbers

$\mathbf{0}_K$	column vector with all its K elements equal to 0
$\mathbf{1}_J$	column vector with all its J elements equal to 1

Greek letters

α	hyperparameter of the $\mathscr{K}$ function
ε	sample from the Gaussian density $\mathcal{N}(0,1)$
$\delta_L^{(v)}$	threshold for the estimation of L for the v-th variable
δ_L	threshold for the estimation of L for a generic variable
$\delta_{\tilde{\phi},\ \phi}$	reconstruction error of $\tilde{\phi}$ with respect to ϕ
δ_F^*	threshold for the reconstruction error of the F density
$\delta_{\chi^2}^*$	threshold for the reconstruction error of the χ^2 density
η	hyperparameter of the $\mathscr{K}$ function
$\mu_{\tilde{\mathbf{a}}}$	mean of the vector $\tilde{\mathbf{a}}$
μ_Q	mean of the whole set of Q statistics of $\mathbf{X}^{\text{cal}}$
$\mu_{\mathbf{y}}$	mean of the vector $\mathbf{y}$
ρ	number of *in-silico* batches generated at each iteration
σ	standard deviation of the measurement noise in a GP-SS model
$\sigma_{\tilde{\mathbf{a}}}$	standard deviation of the vector $\tilde{\mathbf{a}}$
σ_Q	standard deviation of the whole set of Q statistics of $\mathbf{X}^{\text{cal}}$
$\sigma_{\mathbf{y}}$	standard deviation of the vector $\mathbf{y}$
τ	number of shifting time points
ϕ	generic probability density function to be reconstructed
$\tilde{\phi}$	approximation from samples of ϕ
ϕ_{T^2}	nominal PDF of T^2 values
ϕ_Q	nominal PDF of Q residuals
χ^2	symbol denoting a χ^2-type continuous probability density
χ^2_*	χ^2 density with specified degrees of freedom

Acronyms

GP-SS	Gaussian process state-space
NOC	normal operating conditions
PDF	probability density function
PC	principal component
PCA	principal component analysis

batches may be as small as two to four. A PCA model built on such a small number of batches would result in unsatisfactory monitoring performance, because the process data would be insufficient to derive any statistical inference about the process. In batch process monitoring, this is a longstanding problem that is typically tackled by downgrading it from multivariate to univariate, until enough batches are collected to enter a large-*N* (big-data) scenario. Recently, Tulsyan et al. (2018, 2019) proposed a machine-learning methodology to effectively transition from low-*N* to large-*N* scenarios, with no need to downgrade the monitoring problem to univariate. Basically, the methodology uses machine learning to build a virtual large-*N* scenario from a low-*N* one. More specifically, it exploits a block-learning method for a Bayesian non-parametric model of the process under investigation, which enables modeling the process using a Gaussian process state-space (GP-SS) model. Then, it uses probabilistic programming to generate an arbitrarily large number of *in-silico* batch datasets, which can be used to build a PCA model until a sufficient number of historical batches has been accumulated to enter a large-*N* scenario. The authors test the methodology on proprietary datasets, and show that the resulting monitoring models are able to detect a faulty batch when as few as two or three historical batches are available. Whereas this methodology elegantly addresses the low-*N* problem in a systematic way for the first time, it suffers from some limitations that can hinder deployment at the industrial level.

From the algorithmic side, the machine-learning framework proposed by Tulsyan et al. (2018, 2019) requires assigning the values to several parameters, among which the number of trajectories to be resampled for each process variable, the memory (lag) of each process variable, and the number of resampled trajectories to be retained for each variable for GP-SS model training. These parameters affect the computational cost and performance of the *in-silico* data generation, and eventually also the performance of the process monitoring model. However, no systematic methodology was proposed to tune them.

From the process monitoring side, there is a need of investigating more systematically the role of not only the *number*, but also the *characteristics* of the available low-*N* batches that cause the *in-silico* data generation procedure to return trajectories that are indeed useful for process monitoring. Additionally, how many *in-silico* batches should be generated to build the process monitoring model is still to be investigated. Finally, we note that the results reported by Tulsyan et al. (2018, 2019) only refer to the ability of a monitoring model to detect a *faulty* batch. However, whether or not a *normal* batch is detected as such is also crucial in batch process monitoring, because false alarms can lead to rejecting batches that are actually normal, thus causing an economic penalty.

In this study, we address all the above issues systematically. Our contributions can be summarized as follows. (*i*) We develop automatic (or semiautomatic) procedures to tune the values of the machine-learning framework parameters, in such a way that the generation of consistent *in-silico* batch trajectory data can be streamlined, thus facilitating the deployment of this framework at an industrial level. (*ii*) We develop indicators and a metric to assist the *in-silico* data generation exercise from a process monitoring-relevant perspective. (*iii*) Using datasets from a benchmark simulated semi-batch process for the manufacturing of penicillin, we thoroughly investigate the appropriateness of the data generated *in-silico* to the purpose of process monitoring.

The remainder of this paper is organized as follows. Section 2 provides a general overview of the *in-silico* data generation methodology proposed by Tulsyan et al. (2018, 2019), with emphasis on the improvements proposed in this paper. Section 3 focusses on the tuning of parameters. A discussion on *in-silico* data generation in a process monitoring perspective is provided in Section 4. Section 5 introduces the benchmark process. Results are provided and discussed in Section 6. Conclusions are drawn in a separate section. A set of Appendices provide some theoretical insights, and a guideline to the generation of historical datasets through the benchmark simulator.

2. Overview of the *in-silico* batch generation methodology

This Section presents the methodology for *in-silico* batch generation adopted in this study. Details about the methodology have been presented by Tulsyan et al. (2019), and we will not report all of them here. Rather, we focus on the improvements we propose with respect to the original methodology. Software and data are made available as illustrated in the Data availability section.

Throughout this paper, a boldface character denotes a vector (if lowercase) or a matrix (if uppercase). A boldface character within square brackets and with subscripts denotes the element of a vector or a matrix: namely, the i-th element of the generic column vector $\mathbf{v}$ is denoted as $[\mathbf{v}]_i$; notation $[\mathbf{M}]_{i_1,i_2}$ is used for the element in position (i_1,i_2) of the two-dimensional (2D) matrix $\mathbf{M}$; the element in position (i_1,i_2,i_3) of the three-dimensional (3D) matrix $\underline{\mathbf{G}}$ is denoted as $\left[\underline{\mathbf{G}}\right]_{i_1,i_2,i_3}$.

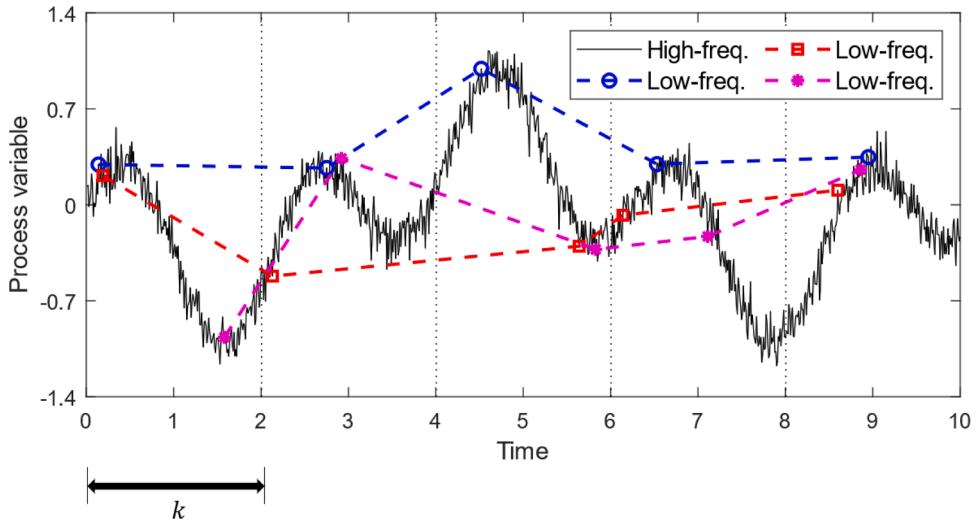

Fig. 1. Stratified resampling. In this example, we have: $N = 1$ real batch, $V = 1$ variable, $J = 3$ resampled trajectories, $T = 5$ low-frequency time points, $k = 2$ time units resampling interval. The original sampling interval is very small and it is not shown for ease of reading.

Furthermore, both the floor and the ceil operator are used, i.e., $\text{floor}(x) = \max\{x' \in \mathbb{Z} | x' \leq x\}$ and $\text{ceil}(x) = \min\{x' \in \mathbb{Z} | x' \geq x\}$ for any $x \in \mathbb{R}$, where $\mathbb{Z}$ is the set of the integer numbers, and $\mathbb{R}$ the one of the real numbers.

2.1. Data from real batches

The inputs to the *in-silico* data generation methodology consist of high-frequency real batch data collected in the 3D matrix $\underline{\mathbf{X}}$ with size $N \times V \times \bar{T}$, where N is the number of real batches (N typically being a "small" number), V is the number of measured variables, and $\bar{T}$ is the number of high-frequency time points along which each variable is measured. We denote with m the high-frequency sampling interval, which is assumed to be the same for all variables. Note that, in the context of this study, the high and low frequency attributes are denoted on a relative basis, rather than on an absolute one; namely, a sampling frequency is considered high when it is much greater than the one (denoted as low) required for process monitoring.

2.2. Stratified resampling of the available data

Data resampling is a strategy for data augmentation (Agarwal et al., 2019). Here, we consider the stratified resampling approach proposed by Tulsyan et al. (2019). For a given real batch, J/N trajectories are generated for each measured variable by low-frequency resampling of the relevant high-frequency trajectory. Therefore, for each variable, the overall number trajectories created by resampling from the entire set of real batches is $(J/N) \cdot N = J$. Essentially, stratified resampling of a variable trajectory for a given real batch consists first in partitioning its time profile into a number of subintervals. A resampled trajectory is then obtained by randomly collecting a sample from each of these subintervals (which are the actual "strata"), and by repeating this operation J/N times. Whereas J needs to be sufficiently large in order to achieve sufficient data augmentation, too large a value of J increases the computational cost with possibly no benefits in terms of process monitoring performance. How to determine a reasonable value for J is not discussed by Tulsyan et al. (2019). Later in this paper (Section 3.1), we consider this issue in detail.

Loosely speaking, stratified resampling generates J "artificial" batches from the N available real ones; however, since the resampled trajectories are tethered around the original (real) ones, stratified resampling does not change the coverage of NOC. Therefore, the resampled trajectories cannot be used by themselves to augment the real data in order to build a reliable model for batch process monitoring.

For a given resampling interval k, with $k \gg m$ and multiple of m, the resampled trajectories are collected in the 3D matrix $\underline{\mathbf{Z}} \in \mathbb{R}^{J \times V \times T}$, with $T = \text{floor}(m\bar{T}/k) \in \mathbb{N}$, and $\mathbb{N}$ being the set of the natural numbers. If $m\bar{T}/k > T$ is found, the remaining part of batches is simply discarded; for simplicity, the assumption $T = m\bar{T}/k \in \mathbb{N}$ is then made here. For compactness, define w as the number of high-frequency time points contained in a resampling interval, formally $w = k/m \in \mathbb{N}$; it follows that $T = \bar{T}/w$.

The stratified resampling operation is formally carried out as follows:

$$n = \text{ceil}\left(\frac{j}{J/N}\right), \tag{1}$$

$$\begin{aligned}&\left[\underline{\mathbf{Z}}(j,1,t) \ldots \underline{\mathbf{Z}}(j,V,t)\right] \\ &\quad = \left[\underline{\mathbf{X}}(n,1,w\cdot(t-1)+u) \ldots \underline{\mathbf{X}}(n,V,w\cdot(t-1)+u)\right]\end{aligned} \tag{2}$$

where u comes from the discrete uniform density in $\{1,\ldots,w\}$. The full matrix $\underline{\mathbf{Z}}$ is built by doing (1) and (2) for $j = 1,\ldots,J$ and $t = 1,\ldots,T$. An example of such operation is shown in Fig. 1 for $J = 3$ resampled low-frequency trajectories (broken lines) from one single high-frequency trajectory (solid line).

2.3. Two-stage exploitation of the resampled data

Dataset $\underline{\mathbf{Z}}$ is then used to train a GP-SS model, which will eventually allow generating *in-silico* batches to be used for process monitoring. In the context of machine-learning techniques, GP modeling represents a powerful modeling methodology that allows non-parametric model learning; namely, no assumptions about model structure are required, which is very favorable for process modeling (Rasmussen and Williams, 2006).

GP-SS model training is accomplished according to a two-step procedure: data partitioning is done first (Section 2.3.1), in order to save computational time during model training; then, actual model training is carried out (Section 2.3.2).

2.3.1. Data partitioning

The learning phase for a GP-SS model is known to be computationally intensive (Moore et al., 2016). Specifically, using the entire $\underline{\mathbf{Z}}$ matrix as a training dataset for the GP-SS model can make the computational problem at hand prohibitive. To tackle this issue, we adopt the same strategy originally proposed by Tulsyan et al. (2019). First, the overall

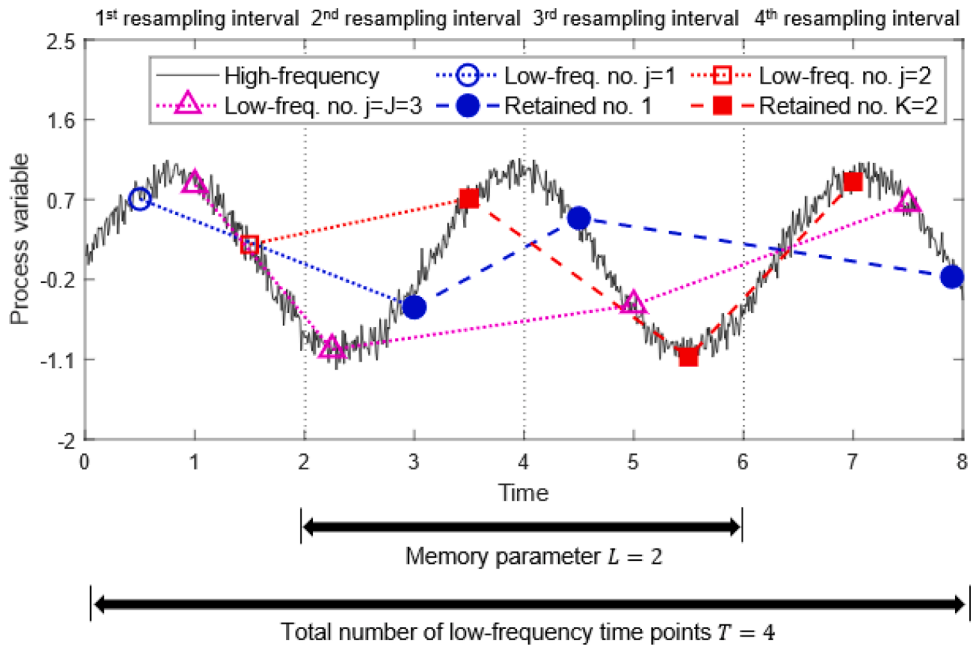

Fig. 2. Schematic representation of data partitioning along two directions (time and number of resampled trajectories) for one single variable in one single batch when the memory is $L = 2$ time points for that variable; a set of $J = 3$ resampled trajectories are available from stratified resampling. With reference to the fourth resampling interval: first, only the $L = 2$ preceding resampling intervals are considered for all the J resampled trajectories; then, only $K = 2 < J$ resampled trajectories (each one containing only $L + 1 = 3$ time points) are retained out of the entire set. The dotted segments connect the resampled time points eventually retained for GP-SS model training.

dataset is sliced along the time direction into smaller datasets, which are built by assuming that, for each variable and each time point, the sample for that variable at that time point depends only on the $L \geq 0$ preceding ones; stated differently, we assume that each variable trajectory is a Markov process with order L. The value of L for a given variable and time point sets the "memory" (or lag) of the process for that particular variable at that particular time point. Parameter L is assumed to be known *a priori* by Tulsyan et al. (2019), and identical for each variable and each time point. However, none of the two conditions may occur in practice. Later in this paper (Section 3.2), we propose a semiautomatic methodology to estimate an appropriate value for L for each variable and each low-frequency time point.

Then, to further reduce the computational cost, only $K \ll J$ resampled trajectories for each variable and time point are retained for model training. Each one of these retained trajectories has length $L + 1$, since this partitioning occurs right after the one along the time direction. An example of such framework is illustrated in Fig. 2 for a generic process variable and low-frequency time point. Tulsyan et al. (2019) do not provide indications about how to choose K. We address this issue in Section 3.3, where we propose a method that automatically tunes K for each variable and each time point.

According to these operations, a reduced training dataset $\mathscr{D}_{v,t}$ can eventually be related to each pair (v,t), i.e., to each variable v and low-frequency time point t. Namely, the training dataset contains the K retained trajectories for variable v at time point t, each one having length $L + 1$; these trajectories span the time points from $t - L$ to t itself. To provide a formal representation of $\mathscr{D}_{v,t}$, we simplify the notation by omitting the variable index v as well as the reference to the last time-point t the trajectories include. Then, let us denote as $z^{(1)}_{t-\tau}, \ldots, z^{(K)}_{t-\tau}$ the values that the K retained trajectories respectively take at time point $t - \tau$ for $\tau = 0, \ldots, L$ (assuming that $L > 0$) for variable v. Then, define

$$\mathbf{z}^{(i)}_{t-L,\ldots,t-1} = \begin{bmatrix} z^{(i)}_{t-L} & \ldots & z^{(i)}_{t-1} \end{bmatrix}^\top, i = 1, \ldots, K, \tag{3}$$

$$\mathbf{Z}_{t-L,\ldots,t-1} = \left[\mathbf{z}^{(1)}_{t-L,\ldots,t-1} \ldots \mathbf{z}^{(K)}_{t-L,\ldots,t-1}\right]^\top. \tag{4}$$

Define the vector

$$\mathbf{y}_t = \begin{bmatrix} z^{(1)}_t \\ \vdots \\ z^{(K)}_t \end{bmatrix} \tag{5}$$

representing the measured values of the process variables. The reduced training dataset is then $\mathscr{D}_{v,t} = \{\mathbf{Z}_{t-L,\ldots,t-1}, \mathbf{y}_t\}$. If $L = 0$, then the computation of (3) and (4) is meaningless, and it is simply $\mathscr{D}_{v,t} = \{\mathbf{y}_t\}$.

2.3.2. GP-SS model training against the partitioned data

Once that data partitioning is obtained, a GP-SS model is trained against the reduced dataset $\mathscr{D}_{v,t}$, for each variable v and for each low-frequency time point t. This allows one to build a probabilistic framework for the generation of *in-silico* batches. Next, we discuss how training is carried out.

Let us consider a given variable in $\underline{\mathbf{Z}}$. If $L = 0$ for that variable at a given time point, then its value at that time point is considered to be Gaussian distributed, with mean and covariance being determined by the K retained artificial trajectories for that variable (prior learning); to find them, one can simply calculate the mean and standard deviation of vector $\mathbf{y}_t$ defined in (5). If $L > 0$, then a squared exponential kernel is used to model the covariance matrix of the transition density from the value at the current time point and the L preceding ones; in practice, we assume that the variable evolves through a certain dynamics that is to be identified, i.e., a "dynamics learning" has to be carried out. The idea of using such type of kernel was originally proposed by Tulsyan et al. (2019). Here, we use the simplified version

$$\mathscr{K}(\mathbf{x}_i, \mathbf{x}_j) = \alpha^2 \exp\left(\frac{-||\mathbf{x}_i - \mathbf{x}_j||^2}{2\eta^2}\right), \tag{6}$$

because it involves a smaller number of hyperparameters. In (6), $\mathscr{K}(\cdot,\cdot)$ is the actual kernel function, $\mathbf{x}_i$ and $\mathbf{x}_j$ are vectors of L components, containing values of the process variable at issue, while α and η are

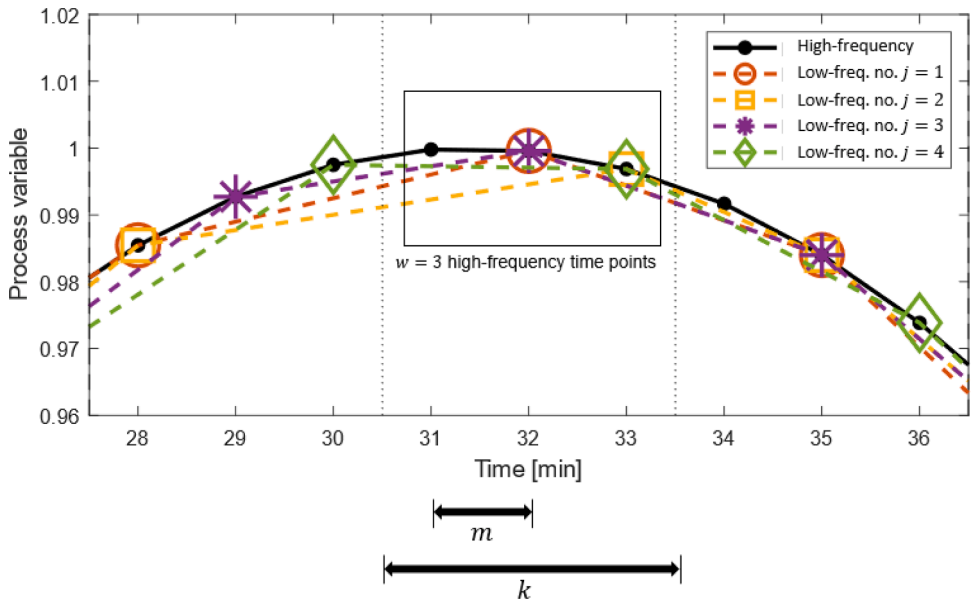

Fig. 3. Schematic representation of the outcomes of the stratified resampling operation over three resampling intervals for a given variable. For both the central resampling interval and the rightmost one, not all the w high-frequency time points are captured by the resampled trajectories.

hyperparameters (η is the characteristic length scale, and α is the signal standard deviation).

Following Tulsyan et al. (2019), we consider additive white Gaussian noise $\mathcal{N}(0, \sigma^2)$ as measurement noise. A formal description of the GP-SS model with reference to a given variable and low-frequency time point is reported in Appendix A.

The training operations need to be repeated for each time point $t \in \{1, \ldots, T\}$ and for each variable $v \in \{1, \ldots, V\}$. Each (v, t) pair can then be referred to either a prior learning operation or a dynamics learning one. When training is concluded, a probabilistic model of the process is available for each variable and low-frequency time point.

2.4. In-silico batch generation

In-silico trajectories can be generated for all variables using the trained GP-SS model, thus generating a set of *in-silico* batches that can then be used to build a process monitoring model.

Let $\underline{\mathbf{I}}$ indicate the 3D matrix collecting the *in-silico* data to be generated, and let $\left[\underline{\mathbf{I}}\right]_{r,v,t}$ be the value of variable v at time point t for *in-silico* batch r. If $L = 0$, then $\left[\underline{\mathbf{I}}\right]_{r,v,t}$ is generated according to the prior Gaussian density, using the mean and the covariance calculated during GP-SS model training. If $L > 0$, then $\left[\underline{\mathbf{I}}\right]_{r,v,t}$ is generated by using the samples from $t - L$ to $t - 1$, that is $\left[\underline{\mathbf{I}}\right]_{r,v,t-L}, \ldots, \left[\underline{\mathbf{I}}\right]_{r,v,t-1}$, according to the mathematics of GPs. The rationale we use in this study is the same presented as Algorithm 4 in Tulsyan et al. (2019), and will not be discussed further here.

The *in-silico* generation is continued until the *in-silico* dataset is large enough, i.e., until the monitoring model being built is deemed reliable. No indications about a suitable size of the *in-silico* dataset is provided by Tulsyan et al. (2019). We address this issue in detail in Section 4.1.

3. Tuning of the GP-SS model parameters

The original study of Tulsyan et al. (2019) proved groundbreaking in addressing the low-N problem in a process monitoring context by means of a GP modeling methodology. Yet, no guidelines were provided on how to assign appropriate values for the model parameters. This Section is meant to fill this gap.

3.1. Tuning of J

The stratified resampling operation involves the risk that a certain amount of information about the original (real) batches gets lost upon resampling. Particularly, the resampled trajectories of a given variable may show less variability than the original ones in $\underline{\mathbf{X}}$ (this may occur, for example, if stratified resampling fails from capturing enough peaks/valleys in the real trajectories). Hence, the number J of resampled trajectories should be determined in such a way as to preserve the variability of the real data.

3.1.1. Formalisation of stratified resampling

As already discussed, stratified resampling consists in obtaining J/N low-frequency trajectories for each high-frequency one measured in a given real batch for each variable. Next, we assume that only one measured variable exists and one real batch is available, i.e., $N = V = 1$; the generalization to several batches and several variables is provided in Section 3.1.4.

Recall that w is the number of high-frequency time points per resampling interval (Section 2.2). Every resampled trajectory (indexed by $j = 1, \ldots, J$) can be identified by a T-uple $\tilde{\mathbf{q}}^{(j)}$, containing the high-frequency time points that have been selected for that trajectory; denote with $[\tilde{\mathbf{q}}^{(j)}]_t$, $t \in \{1, \ldots, T\}$, the t-th element of $\tilde{\mathbf{q}}^{(j)}$. It follows that

$$\left[\tilde{\mathbf{q}}^{(j)}\right]_t \in \{(t-1)\cdot w + 1, \ldots, t\cdot w\}. \tag{7}$$

To ensure that the resampled trajectories cover the same variability as the real ones, it is required that each high-frequency time point appears at least once in any of the J resampled trajectories. Formally, it is required that

$$\forall q \in \{1, \ldots, \bar{T}\}\ \exists\, j \in \{1, \ldots, J\} \mid q = \left[\tilde{\mathbf{q}}^{(j)}\right]_t,\ t = \mathrm{ceil}(q\,/\,w). \tag{8}$$

In other words, let $I_t = \{[\tilde{\mathbf{q}}^{(1)}]_t, \ldots, [\tilde{\mathbf{q}}^{(J)}]_t\}$ be the set all of the high-frequency time points captured in the resampling interval with index t through the J random selections. Each high-frequency time point in $\{(t-1)\cdot w + 1, \ldots, t\cdot w\}$ is required to appear at least once in I_t; equivalently, I_t must contain exactly w different elements among all the J ones considered, and this must hold for all $t \in \{1, \ldots, T\}$.

A situation violating this condition is illustrated in Fig. 3. In this case, we have $w = 3$ high-frequency time points within each resampling interval, and $J = 4$ resampled batches; for both the central resampling interval and the rightmost one, only $2 < w$ different high-frequency time-points are captured by the resampled trajectories, despite $J > w$ trajectories were resampled.

We propose the following two-step procedure to estimate J:

Step 1. Determine the probability $P(J, w)$ that, by doing J repeated random selections from a pool of w different high-frequency time points within a given resampling interval, each point is extracted at least once; or, equivalently, that all the w different points do appear in the outcome of the J random selections;

Step 2. Determine the minimum value of J such that $P(J, w)$ is greater than a given threshold.

Step 1 is done by determining a lower bound for $P(J, w)$ in analytical form. This is addressed in Section 3.1.2.

3.1.2. Determination of a lower bound for $P(J, w)$

Let $D(J, h)$ be the number of sequences made by J elements that contain exactly h different (specific) elements, for each $h = 1, \ldots, w$, where h indicates a high-frequency time point. It can be shown that the following equation holds:

$$D(J, w) = w^J - \sum_{h=1}^{w-1} \binom{w}{h} D(J, h). \tag{9}$$

This fact is proved in Appendix B. Upon division by w^J, the previous equation can be rewritten as

$$\frac{D(J, w)}{w^J} = 1 - \sum_{h=1}^{w-1} \binom{w}{h} \frac{D(J, h)}{w^J}, \tag{10}$$

and then, by defining $P(J, h) = D(J, h)/h^J$, $h = 1, \ldots, w$, one can write

$$P(J, w) = 1 - \sum_{h=1}^{w-1} \underbrace{\binom{w}{h} \frac{h^J}{w^J}}_{=C(J,w,h)} P(J, h). \tag{11}$$

Being $P(J, h)$ a probability value, then $P(J, h) \in [0, 1]$ and hence the previous equation yields

$$P(J, w) \geq \underbrace{1 - \sum_{h=1}^{w-1} C(J, w, h)}_{=\widetilde{P}(J,w)}. \tag{12}$$

The right-hand side quantity $\widetilde{P}(J, w)$ in (12) is actually a lower bound for $P(J, w)$, and this is sufficient for our purpose because we require $P(J, w)$ to be greater than a threshold.

3.1.3. Determination of a suitable J

It is convenient to assign a-priori the probability $p_J \in (0, 1)$ that I_t contains w different elements for each resampling interval $t = 1, \ldots, T$. Since the resampling operations are independent across the resampling intervals, the probability for a single interval is $p_J^{1/T}$, and this is the actual lower bound for $P(J, w)$. Therefore, one can evaluate $\widetilde{P}(J, w)$ defined in (12) for $J = w$, and then increase the guess of J until $\widetilde{P}(J, w) \geq p_J^{1/T}$.

3.1.4. Generalization for $N > 1$ and $V > 1$

If more than one real batch is considered (i.e., $N > 1$), then the algorithm discussed above is used to estimate J/N rather than J, and the reference threshold for $P(J/N,\ w)$ becomes $p_J^{1/(N \cdot T)}$; hence J is increased until $\widetilde{P}(J/N,\ w) \geq p_J^{1/(N \cdot T)}$. Each one of the N real batches leads to J/N resampled trajectories, in such a way that the total number of resampled trajectories is still indicated as J. The generalization to the case with $V > 1$ does not require any modifications, since the resampling is done with reference to all the variables for each time point. Note that this algorithm uses only the number w of time points within a resampling interval as an input, and w is same for the all variables in all batches. Therefore, the same value of J/N for all variables and all batches will be found.

3.2. Tuning of L

Parameter L has a critical role, as it tunes the memory of each process variable. The issue of estimating the order in Markov processes has been addressed by several researchers in different contexts (Nakahama, et al., 1977; Ku et al., 1995; Peres and Shields, 2005; Merhav et al., 1989; Morvai and Weiss, 2005). In this study, we propose a methodology that is a compromise between computational burden, mathematical rigor and easiness to interpret the results in the specific context it is applied to. The methodology returns a value of L that is adjusted for each variable at each low-frequency time point; therefore, notation $L(v, t)$ will be used.

Consider the following column vectors derived from matrix $\underline{\mathbf{Z}}$:

$$\widetilde{\mathbf{a}} = \left[\left[\underline{\mathbf{Z}}\right]_{1,v,t} \ldots \left[\underline{\mathbf{Z}}\right]_{J,v,t} \right]^\top, \tag{13}$$

$$\widetilde{\mathbf{b}} = \left[\left[\underline{\mathbf{Z}}\right]_{1,v,t-\tau} \ldots \left[\underline{\mathbf{Z}}\right]_{J,v,t-\tau} \right]^\top \tag{14}$$

for a given value of $\tau \in \{0, \ldots, t-1\}$. They represent the entire set of values of a given variable v at time point t and at time point $t - \tau \leq t$ (respectively) across all resampled batches in $\underline{\mathbf{Z}}$. Let $\mu_{\widetilde{\mathbf{a}}}$ and $\sigma_{\widetilde{\mathbf{a}}}$ be the mean and standard deviation of $\widetilde{\mathbf{a}}$ respectively, and let $\mathbf{1}_J$ be the column vector with all its J elements equal to 1. With these values, we compute

$$\mathbf{a} = \frac{\widetilde{\mathbf{a}} - \mu_{\widetilde{\mathbf{a}}} \cdot \mathbf{1}_J}{\sigma_{\widetilde{\mathbf{a}}}}, \tag{15}$$

$$\mathbf{b} = \frac{\widetilde{\mathbf{b}} - \mu_{\widetilde{\mathbf{a}}} \cdot \mathbf{1}_J}{\sigma_{\widetilde{\mathbf{a}}}}. \tag{16}$$

Thus, $\widetilde{\mathbf{a}}$ has been auto-scaled, while this has not been the case for $\widetilde{\mathbf{b}}$. The correlation between $\mathbf{a}$ and $\mathbf{b}$ (i.e., the correlation between the set of values of variable v at time point t and the set of the values it takes at the preceding time point $t - \tau$) is defined as

$$\widetilde{c}^{(v)}(t, \tau) = \frac{\mathbf{a}^\top \mathbf{b}}{J - 1}. \tag{17}$$

According to the Cauchy-Schwarz inequality (Wu and Wu, 2009), we can write

$$c^{(v)}(t, \tau) = \frac{|\mathbf{a}^\top \mathbf{b}|}{\sqrt{(\mathbf{a}^\top \mathbf{a}) \cdot (\mathbf{b}^\top \mathbf{b})}} \in [0, 1], \tag{18}$$

provided that both $\mathbf{a}$ and $\mathbf{b}$ have norm different from 0. Since $\mathbf{a}$ is the result of an autoscaling operation, it is $\mathbf{a}^\top \mathbf{a} = J - 1$ if $\mathbf{a}$ has norm different from 0.

If $\mathbf{a}$ or $\mathbf{b}$ has norm equal to 0, then $c^{(v)}(t, \tau)$ in (18) is not computable, and we address the issue as follows. We set $c^{(v)}(t, \tau) = 1$ if $\tau = 0$, that is any value of v at a specific time point is maximally correlated with itself. For $\tau \geq 1$, we set $c^{(v)}(t, \tau) = 0$. This means that either $\widetilde{\mathbf{a}}$ or $\widetilde{\mathbf{b}}$ contains J equal values; in practice, the value taken by variable v at either time point t or $t - \tau$ is deterministic, and therefore it is uncorrelated with the values at any other time points.

A threshold $\delta_L^{(v)} \in [0, 1]$ is then set on $c^{(v)}(t, \tau)$: if $c^{(v)}(t, \tau)$ is smaller than the threshold, then the values of v at the two involved time points (i.e., t and $t - \tau$) are considered uncorrelated, and $L(v, t)$ is reduced of

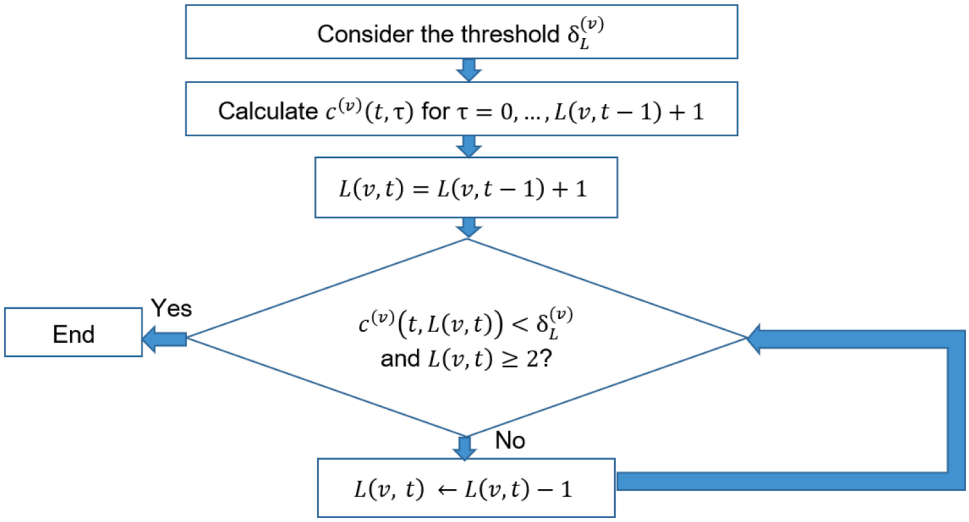

Fig. 4. Flow-chart describing the procedure for the estimation of L for variable v at time point t. Note that the threshold $\delta_L^{(v)}$ must be set for each variable, but does not vary across the time points. This is why for the estimation of $L(v,t)$ it is not necessary to set again the threshold $\delta_L^{(v)}$, but just to keep it into account.

one unit (i.e., the estimated memory is reduced of one resampling interval) with respect to its current guess.

We suggest setting the same $\delta_L^{(v)}$ for all the variables at the beginning; each value can then be adjusted if results for the case study at hand are not satisfactory. Tentative values for $\delta_L^{(v)}$ can be assigned as follows. A value of $\delta_L^{(v)} = 0.1$ can be used to obtain large values of L (this may be appropriate for process variables displaying flat time profiles); a value of $\delta_L^{(v)} = 0.5$ may fit trajectories showing significant dynamics; greater values of $\delta_L^{(v)}$ (e.g., $\delta_L^{(v)} \geq 0.9$) can be used to obtain small L's, which may be useful for very noisy trajectories, where autocorrelation is harder to notice by inspection. An example of how to perform this adjustment is provided in Section 6.3.1.

The procedure to estimate L is summarized in the flowchart of Fig. 4 for a generic variable v at a time point $t \geq 2$. Clearly, it is $L(v,1) = 0$ for any v. For $t \geq 2$, the initial guess for $L(v,t)$ assumes that the memory at a certain time point is not greater than the memory at the previous time point plus one sample, that is $L(v,t) \leq L(v,t-1)+1$; clearly, this is only an initial guess, and the estimation algorithm may end up even with a decrease of the memory, that is $L(v,t) < L(v,t-1)$. Moreover, $L(v,t)$ is not decreased below 1 for $t \geq 2$; this represents the fact that any time point after the first one does have memory of at least one sample.

3.3. Tuning of K

The number K of resampled trajectories to be retained for each variable is the parameter that eventually sets the size of training dataset for the GP-SS model. The tuning procedure we propose sets the value of K for each variable at each time point; hence notation $K(v,t)$ will be used. Whereas Tulsyan et al. (2019) assign K, and use the K-furthest neighbors algorithm to determine the indices of the resampled trajectories to be retained, we propose a different approach, that simultaneously returns both a suitable value of K and the indices of the trajectories to be retained. The proposed approach is completely automatic, does not require initialization, and is computationally efficient. The underlying idea is to retain the smallest number of trajectories that can preserve the same variability as the entire set of trajectories in $\underline{\mathbf{Z}}$. To this purpose, for a given variable, we retain all and only the trajectories that, for any of the considered time points, include either maximum or minimum values of that variable. Next, a formal discussion of the proposed algorithm is presented.

Consider the triplet (n,v,t) indicating one real batch, one variable and one low-frequency time point. Furthermore, consider the current value of L, i.e., $L(v,t)$; this allows identifying the set of time points $t-\tau$, for $\tau = 0, \ldots, L(v,t)$. Then, enumerate the variable trajectories coming from the resampling of the variable v for the batch n, namely $j = (n-1)J/N+1, \ldots, nJ/N$, and define $\mathcal{J}_\tau(n,v,t) = \{j_\tau^{\min}(n,v,t), j_\tau^{\max}(n,v,t)\}$ with

$$j_\tau^{\min}(n,v,t) = \text{argmin}_{j \in \{(n-1)J/N+1,\ldots,nJ/N\}} \left[\underline{\mathbf{Z}}\right]_{j,v,t-\tau}, \tag{19}$$

$$j_\tau^{\max}(n,v,t) = \text{argmax}_{j \in \{(n-1)J/N+1,\ldots,nJ/N\}} \left[\underline{\mathbf{Z}}\right]_{j,v,t-\tau}. \tag{20}$$

This operation is done for every $\tau \in \{0,\ldots,L(v,t)\}$. Finally, one can write

$$\mathcal{J}(n,v,t) = \cup_{\tau=0}^{L(v,t)} \mathcal{J}_\tau(n,v,t). \tag{21}$$

The latter set contains the indices of the resampled trajectories to be retained with reference to the selected real batch-variable-time point triplet $(n,\ v,\ t)$, and its size can be denoted by $K_n(v,t)$.

The above operation is repeated for all the N involved real batches, thus providing the final number of retained trajectories for the pair (v,t), namely

$$K(v,t) = \sum_{n=1}^{N} K_n(v,t)\ . \tag{22}$$

4. Role of the *in-silico* batches in a monitoring perspective

The set of batches generated *in-silico* are used to complement the (few) available real ones in such a way that a PCA model can be built upon them to monitor the performance of a new real batch. This is done until a sufficient number of real batches have been run to transition from a low-N to a large-N scenario, where *in-silico* batches are no longer needed to build the monitoring model. Selecting the number of batches to be generated in silico is not trivial. Loosely speaking, if too few batches are generated *in-silico*, the resulting monitoring model may not capture sufficient variability of the true data, and therefore be prone to errors in monitoring new real batches. On the other hand, generating too many *in-silico* batches may increase the computational burden without significant benefits. On a different perspective, the adequacy of the monitoring model built using the *in-silico* batches should be assessed in advance, to make the user more confident about the monitoring performance expected from the model. These issues are addressed in the following.

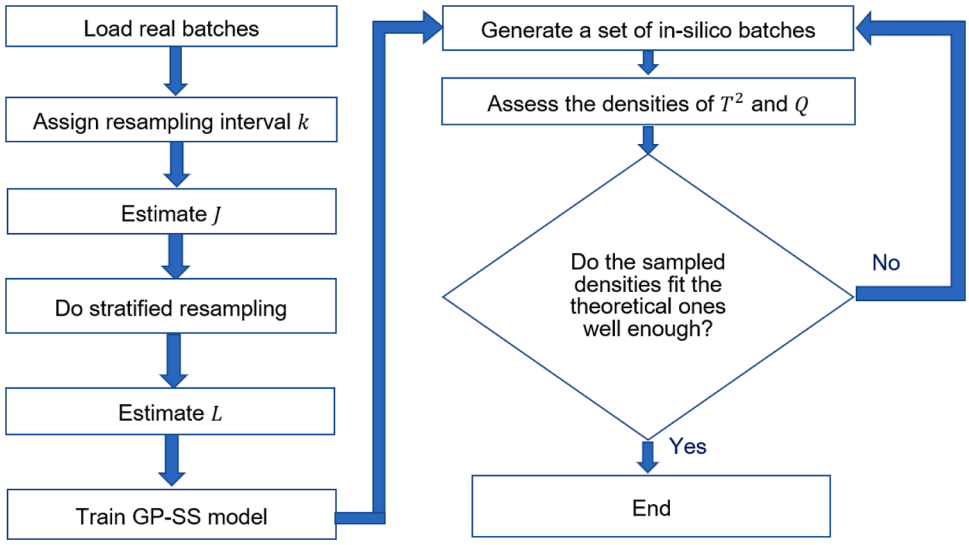

Fig. 5. Simplified flowchart illustrating the stopping criterion for *in-silico* batch generation.

4.1. Determining a suitable number of in-silico batches

We propose a procedure to determine an appropriate number of batches to be generated *in-silico*. The idea is fairly simple: since, for a properly built monitoring model, the monitoring statistics are known to follow given probability density functions (PDFs), the number of batches to be generated in silico is chosen as the smallest one that enables the reconstructed densities to follow the expected ones to a reasonably close extent. The density reconstruction errors can then be used as a metric to assess the adequacy of the monitoring model (Section 4.1.2).

The procedure for determining the number of batches to be generated *in-silico* is sketched in Fig. 5. When a new *in-silico* batch (or a set of new *in-silico* batches) has been generated using the GP-SS model, a calibration 2D matrix $\mathbf{X}^{\text{cal}}$ is built by batch-wise unfolding (Nomikos and MacGregor, 1995) the 3D matrix collecting the (down-sampled) real batches together with the batches generated *in-silico*. A PCA model is then built on $\mathbf{X}^{\text{cal}}$, and the monitoring statistics (namely, Hotelling T^2 and Q residuals; (Nomikos and MacGregor, 1995)) values are calculated for all the batches in $\mathbf{X}^{\text{cal}}$. Further to that, the theoretical density of each statistics is approximated through the resulting sampled one, and the reconstruction error is evaluated. When the errors on both statistics fall below assigned thresholds, the generation of new *in-silico* batches can be stopped.

4.1.1. Defining the target densities for the monitoring statistics

Let A be the number of principal components (PCs) of the PCA model calibrated on $\mathbf{X}^{\text{cal}}$ (whose columns are assumed to be autoscaled). Nomikos and MacGregor (1995) showed that, for a proper monitoring model, the T^2 values are F-distributed, whereas the Q residuals are χ^2-distributed.

In this study, we calculate the PDF of the T^2 values as suggested by Wise and Gallagher (1996). Namely, denoting such PDF with $\phi_{T^2}(z)$ for any $z \in \mathbb{R}$, we have

$$\phi_{T^2}(z) = \frac{\widetilde{N}-A}{A(\widetilde{N}-1)} F_{A,\widetilde{N}-A}\left(\frac{\widetilde{N}-A}{A(\widetilde{N}-1)} z\right). \tag{23}$$

The density of the Q residuals is calculated as indicated by Nomikos and MacGregor (1995). Namely, if such PDF is denoted with ϕ_Q, we have

$$\phi_Q(z) = \frac{2\mu_Q}{\sigma_Q^2} \chi^2_{2\mu_Q^2/\sigma_Q^2}\left(\frac{2\mu_Q}{\sigma_Q^2} z\right). \tag{24}$$

In (23) and (24), the following notation is used: $\widetilde{N}$ is the number of rows of $\mathbf{X}^{\text{cal}}$; $F_{A,\widetilde{N}-A}$ is an F density with A and $\widetilde{N}-A$ degrees of freedom; μ_Q is the mean of the Q residuals; σ_Q is the standard deviation of the Q residuals; $\chi^2_{2\mu_Q^2/\sigma_Q^2}$ is a χ^2 density with $2\mu_Q^2/\sigma_Q^2$ degrees of freedom.

The continuous PDFs (23) and (24) are set as references against which the densities reconstructed from sampled data are checked. While modifications of these statistics have been developed to handle critical aspects in batch process monitoring (e.g., Reis, et al. (2021) for the Q statistics), we prefer to use the standard statistics to frame our research in a more general context.

4.1.2. Reconstructing a continuous density from sampled values

An algorithm to reconstruct a tentative density starting from available samples is presented. The underlying idea has some similarities with other approaches discussed in the literature, such as the one by Kropotov et al. (2018); further discussions on this topic are provided by Berg and Harris (2008) and Hernandez (2018).

Recall that, if N is the number of real batches and R the number of batches generated *in-silico*, then $\mathbf{X}^{\text{cal}}$ has $\widetilde{N} = N + R$ rows and $V{\cdot}T$ columns. Suppose $\mathscr{Z} = \{z_1, z_2, \ldots, z_{\widetilde{N}}\}$ is the whole set of either the T^2 statistics or the Q statistics of $\mathbf{X}^{\text{cal}}$. Consider then an interval $I_{\mathscr{Z}} = [z', z'']$ containing all of them in such a way that $[\min(\mathscr{Z}), \max(\mathscr{Z})] \subset (z', z'')$; suitable choices are $z' = \max\{x \in \mathbb{Z} \mid x < \min(\mathscr{Z})\}$, $z'' = \min\{x \in \mathbb{Z} \mid x > \max(\mathscr{Z})\}$. Divide $I_{\mathscr{Z}}$ in S equal subintervals indexed by $s = 1, \ldots, S$, that is

$$I_{\mathscr{Z}}^{(s)} = \left[z' + (s-1)\cdot\frac{z''-z'}{S}, z' + s\cdot\frac{z''-z'}{S}\right). \tag{25}$$

Then, for each s, define $\mathscr{Z}^{(s)}$ as the subset of $\mathscr{Z}$ built with all and only the elements of $I_{\mathscr{Z}}^{(s)}$, namely

$$\mathscr{Z}^{(s)} = \{z \in \{z_1, \ldots, z_{\widetilde{N}}\} \mid z \in I_{\mathscr{Z}}^{(s)}\} = \mathscr{Z} \cap I_{\mathscr{Z}}^{(s)}. \tag{26}$$

In practice, $\mathscr{Z}^{(s)}$, for $s = 1, \ldots, S$, makes a partition of $\mathscr{Z}$. Moreover, compute $c^{(s)}$ as the cardinality (i.e., the number of elements) of $\mathscr{Z}^{(s)}$ divided by the cardinality of $\mathscr{Z}$ (and multiplied by a normalization factor):

$$c^{(s)} = \frac{|\mathscr{Z}^{(s)}|}{|\mathscr{Z}|}\cdot\frac{S}{z''-z'} = \frac{|\mathscr{Z}^{(s)}|}{\widetilde{N}}\cdot\frac{S}{z''-z'}. \tag{27}$$

A piecewise-constant probability density function (PDF) $\widetilde{\phi}$ can be associated to $\mathscr{Z}$ through

$$\widetilde{\phi}(z) = \begin{cases} 0 \text{ if } z \notin I_{\widetilde{Z}}, \\ c^{(s)} \text{ if } z \in I_{\widetilde{Z}}^{(s)} \text{ for a unique } s \end{cases}. \tag{28}$$

If $\widetilde{\phi}$ attempts to approximate a target density ϕ, the total reconstruction error can be evaluated according to the Kullback-Leibler divergence (Kullback, 1968)

$$\delta_{\widetilde{\phi},\,\phi} = \int_{-\infty}^{\infty} \widetilde{\phi}(z) \log\left(\frac{\widetilde{\phi}(z)}{\phi(z)}\right) \mathrm{d}z, \tag{29}$$

considering the term inside the integral equal to zero if $\widetilde{\phi}(z) = 0$. The number S of the subintervals in which $I_{\widetilde{Z}}$ is divided is selected in order to minimize the reconstruction error defined by (29).

With reference to Fig. 5, the generation of new *in-silico* batches can be stopped when the reconstruction error is sufficiently small for both densities. This ensures that both T^2 and Q adequately follow the relevant theoretical density, and therefore that reliable confidence limits can be defined for both statistics as in Nomikos and MacGregor (1995) or in Wise and Gallagher (1996). It should be remarked that obtaining reliable confidence limits is only a necessary condition for a monitoring model to be deemed as reliable. However, the resulting performance of the monitoring model may be insufficient even if both monitoring statistics follow the relevant theoretical densities. This fact will be discussed in Section 6.1.

4.2. Assessing the in-silico batches from a monitoring perspective

Next, we propose two tests that can be carried out on the pool of real and *in-silico* batches to assess their adequacy for process monitoring purposes.

4.2.1. Positions of the Q residuals of the real batches

Once the *in-silico* batch generation activity is concluded (Fig. 5), it is convenient to plot the Q residuals of the (few) available real batches onto the PDF plot of the reconstructed χ^2 density. Remind that calibrating a PCA model means defining a low-dimensional hyperspace of PCs in the whole space of the columns of the calibration matrix; if the residuals for the real batches are small, this means that they are very close to such hyperspace, and that they can be considered as references for the monitoring model. Therefore, for the monitoring model to be deemed adequate, these residuals should be sufficiently small. Stated differently, building a PCA model in which the real batches are recognized as largely different from the *in-silico* ones would denote model inadequacy.

With reference to the Hotelling T^2 statistic, we found that the location of T^2 for the real batches in the relevant PDF plot does not provide useful information in the context of *in-silico* batch generation. Experience suggests to avoid extreme values.

The final remark done in Section 4.1.2 applies here too: whereas suitable values of the Q residuals of the real batches may provide useful indications about the quality of the *in-silico* batches with respect to the real ones, this does not necessarily guarantee satisfactory monitoring performance.

4.2.2. Univariate indicator of the coverage

Next, we develop a univariate indicator i_{NOC} to quantitatively assess the coverage of the NOC by the *in-silico* batches (and the few original real ones), time point by time point, and variable by variable (i.e., with reference to the pair (v,t)). The proposed indicator allows highlighting the process variables for which the coverage is poor after the *in-silico* generation, and at which time points this occurs. This occurrence may suggest re-considering some steps of the *in-silico* data-generation procedure, most likely by adjusting either parameter L or the number of batches generated *in-silico*.

The idea for the calculation of i_{NOC} with reference to the (v,t) pair is to consider the values of variable v at time point t for all batches (both real and *in-silico*), and to quantify how well these values comply with the following empirical requirements, that are typically met under large-N monitoring scenarios: (*i*) they are close to each other; (*ii*) they provide uniform coverage of the real batch variability.

First, define vector $\widehat{\mathbf{x}} = [[\mathbf{X}^{\mathrm{cal}}]_{1,(t-1)V+v} \cdots [\mathbf{X}^{\mathrm{cal}}]_{\widetilde{N},(t-1)V+v}]^{\top}$; namely, $\widehat{\mathbf{x}}$ is the set of values taken by variable v across all calibration batches at time point t. Starting from $\widehat{\mathbf{x}}$, define vector $\mathbf{x}$ by doing the following operations:

- sort $\widehat{\mathbf{x}}$ in ascending order, thus obtaining $\mathbf{x}''$;
- shift $\mathbf{x}''$ in such a way that its first element (i.e., the smallest one) is 0, thus obtaining $\mathbf{x}'$;
- scale $\mathbf{x}'$ in such a way that its last element (i.e., the largest one) is 1, thus obtaining $\mathbf{x}$.

Essentially, vector $\mathbf{x} = [[\mathbf{x}]_1 \ldots [\mathbf{x}]_{\widetilde{N}}]^{\top}$ is built with the elements of $\widehat{\mathbf{x}}$ by sorting in ascending order, shifting and rescaling, in such a way that

$$0 = [\mathbf{x}]_1 \leq \ldots \leq [\mathbf{x}]_{\widetilde{N}} = 1. \tag{30}$$

Then, take

$$\mathbf{y} = [[\mathbf{x}]_2 - [\mathbf{x}]_1 \ldots [\mathbf{x}]_{\widetilde{N}} - [\mathbf{x}]_{\widetilde{N}-1}]^{\top} \tag{31}$$

and denote by $[\mathbf{y}]_{\tilde{n}}$, $\tilde{n} = 1, \ldots, \widetilde{N} - 1$ the elements of $\mathbf{y}$. Note that $\mathbf{y}$ contains the distance of each value in $\mathbf{x}$ from its subsequent one. The mean of $\mathbf{y}$ can be proved to be simply

$$\mu_{\mathbf{y}} = \frac{1}{\widetilde{N}-1} \cdot \sum_{\tilde{n}=1}^{\widetilde{N}-1} [\mathbf{y}]_{\tilde{n}} \tag{32}$$

$$= \frac{1}{\widetilde{N}-1}. \tag{33}$$

The variance of $\mathbf{y}$ is

$$\sigma_{\mathbf{y}}^2 = \frac{1}{\widetilde{N}-2} \cdot \sum_{\tilde{n}=1}^{\widetilde{N}-1} \left([\mathbf{y}]_{\tilde{n}} - \mu_{\mathbf{y}}\right)^2 \tag{34}$$

$$= \frac{1}{(\widetilde{N}-2)(\widetilde{N}-1)} \left(\left((\widetilde{N}-1) \cdot \sum_{\tilde{n}=1}^{\widetilde{N}-1} [\mathbf{y}]_{\tilde{n}}^2 \right) - 1 \right), \tag{35}$$

where (35) is obtained from (34) by substituting (33) into (34). Define the auxiliary indicator i_{var} as

$$i_{\mathrm{var}} = 1 - (\widetilde{N}-1)\sigma_{\mathbf{y}}^2 \tag{36}$$

$$= \frac{\widetilde{N}-1}{\widetilde{N}-2} \left(1 - \sum_{\tilde{n}=1}^{\widetilde{N}-1} [\mathbf{y}]_{\tilde{n}}^2 \right) \tag{37}$$

by substituting (35) into (36). Define also the auxiliary indicator i_{mean} as

$$i_{\mathrm{mean}} = -\mu_{\mathbf{y}} + 1 \tag{38}$$

$$= \frac{\widetilde{N}-2}{\widetilde{N}-1} \tag{39}$$

by using (33). After that, introduce the i_{NOC} indicator by defining

$$i_{\mathrm{NOC}} = i_{\mathrm{var}} \cdot i_{\mathrm{mean}} \tag{40}$$

$$= 1 - \sum_{\tilde{n}=1}^{\widetilde{N}-1} [\mathbf{y}]_{\tilde{n}}^2 . \tag{41}$$

Table 1
Pensim simulator variable numbering used in this study.

Variable no.	Variable name	Units
1	Aeration rate	L/h
2	Agitator power	W
3	Glucose feed rate	L/h
4	Glucose feed temperature	K
5	Glucose concentration	g/L
6	Dissolved O_2 concentration	g/L
7	Biomass concentration	g/L
8	Penicillin concentration	g/L
9	Culture volume	L
10	CO_2 concentration	mmol/L
11	pH	–
12	Temperature	K
13	Generated heat	kcal/h
14	Acid flow rate	mL/h
15	Base flow rate	L/h
16	Cooling/Heating flow rate	L/h

Table 2
Parameters used for the generation of *in-silico* batches using the GP-SS model.

k	p_J	$\delta_L^{(v)}$	$\bar{R}^2$	δ_F^*	$\delta_{\chi^2}^*$	ρ
600 min	0.999	0.99 for $v = 2, 3$ 0.97 for $v = 4$ 0.1 for $v = 7$ 0.5 for $v = 8$ 0.95 for $v = 12$ 0.7 for $v = 13$ 0.99 for $v = 14$ 0.55 for $v = 16$ 0.9 for any other v	50 %	0.08	0.05	25

Eq. (41) is achieved by inserting (37) and (39) into (40).

It can be proved that i_{NOC} is bounded in $[0,\ 1)$. Note that i_{NOC} is defined in such a way that it is close to 1 if the *in-silico* values for variable v at time point t uniformly cover the variability of the real batches and are close to each other; conversely, i_{NOC} is close to 0 if the *in-silico* trajectories are tethered around the real ones. In fact, i_{NOC} is the product between i_{var} and i_{mean}; the former is close to 1 when the values in $\mathbf{x}$ uniformly cover their variability, that is the variance of $\mathbf{y}$ is close to 0; the latter is close to 1 when the values in $\mathbf{x}$ are close to each other, that is the mean of $\mathbf{y}$ is close to 0. Therefore, the closer to 1 i_{NOC} is, the better the coverage for the v-th variable at the t-th time point is.

Proofs of (33) and of the statements above are reported in Appendix C. Incidentally, if $\hat{\mathbf{x}}$ is a constant vector, then the aforementioned procedure is unfeasible, and $i_{\text{NOC}} = 1$ by default.

The i_{NOC} indicator allows assessing the coverage of the NOC basing on the knowledge available from the few real batches. However, it must be stressed that the actual NOC, i.e., those encountered under a large-N scenario, are unknown when a low-N scenario is in place. Therefore, whereas a low value of i_{NOC} indicates that the data generated *in-silico* are not entirely appropriate for process monitoring, a large value of i_{NOC} does not necessarily mean that satisfactory monitoring performance will be achieved.

5. Benchmark process

As a testbed to assess the *in-silico* data generation methodology and the performance of the related process monitoring model, we consider a simulated fed-batch fermentation process for the manufacturing of penicillin. The detailed mechanistic model, also known as the Pensim simulator, has been proposed by Birol et al. (2002), and is extensively used for process monitoring studies. Here, we will not provide a description of the process and of the software; the interested reader can find details in the original reference (Birol et al., 2002). In remainder of the discussion, this model will be referred to as the real process. Table 1 summarizes the variable numbering used in this study. The listed variables represent those for which process measurements are assumed to be available, and whose profiles are generated *in-silico* under low-N scenarios. The batch length is fixed, and set to 300 h.

The process monitoring model is based on the variables listed in Table 1. Since we are not interested in assessing the specific ability in real-time monitoring, but rather the general effectiveness of process monitoring using *in-silico* data, we carry out process monitoring only at the end of a batch (postmortem or retrospective analysis). To do this, we use multiway principal component analysis, namely, by batchwise unfolding the three-way array where the real and *in-silico* data are stored (Nomikos and MacGregor, 1994). The sampling interval for the process measurement profiles is $m = 10$ min (0.167 h); the sampling interval for the profiles generated *in-silico* is set to $k = 600$ min (10 h). Before being stored in the three-way array, the process measurements are down-sampled to match the frequency at which the *in-silico* data are made available.

6. Results and discussion

Results from the *in-silico* batch generation methodology are discussed here in a threefold direction. First (Section 6.1), we assess the monitoring performance of the process monitoring models calibrated with (few) available real batches together with the batches generated *in-silico*. Then (Section 6.2), we draw our attention to the *in-silico* batch generation activity itself, namely we consider the effect of the number of batches generated *in-silico* on the monitoring performance. Finally (Section 6.3), we discuss the impact of the GP-SS model parameters (assigned through the algorithms proposed in Section 3) on the data generated *in-silico*. Throughout Sections 6.1–6.3, we also discuss the appropriateness of the metrics we have proposed to assess the quality of the data generated *in-silico* for process monitoring purposes.

To benchmark the monitoring results, we consider a pool of 180 real batches under NOC (Appendix D); these batches are collectively denoted as the large-N batches. In a way, the large-N batches represent those that one would collect after a long production campaign, i.e., when a low-N scenario is not in place anymore. To build a monitoring model under a low-N scenario, we draw a few (4 or 3) batches from the large-N pool; these batches are denoted as the low-N batches. A different pool of 19 real batches, not included in the large-N one, is used to test the monitoring performance; these batches are denoted as the validation batches. Four of them (namely, batches nos. 16, 17, 18 and 19) are intentionally generated as faulty ones; the remaining validation batches are normal. The modeling conditions characterizing the large-N batches and the validation ones are reported in Appendix D.

A PCA model with $A = 5$ PCs (explaining ~69 % of the overall variance of the data) is built using the large-N dataset. All validation batches are correctly labeled as either normal or faulty by this model (results are not shown for conciseness); in particular, all faulty batches show Q residuals greater than the confidence limit. When building a PCA model with the *in-silico* batches together with the few low-N ones, the number of PCs is chosen in such a way to explain an assigned amount $\bar{R}^2$ of variance. Although this is not necessarily the optimal way to set the number of PCs, it nevertheless simplifies the presentation of results, while still ensuring they are reliable. Confidence limits for all monitoring charts are set to 95 %.

For *in-silico* data generation, we use the algorithms discussed in Sections 2, 3 and 4, except when noted. With reference to the GP-SS model, the parameters used to generate the *in-silico* data are listed in Table 2. Symbols δ_F^* and $\delta_{\chi^2}^*$ indicate the thresholds of the reconstruction errors for the F density and the χ^2 one, respectively. Parameter ρ is the number of *in-silico* batches generated for each single iteration (see Fig. 5). Possible modifications to the parameters will be considered throughout the discussion.

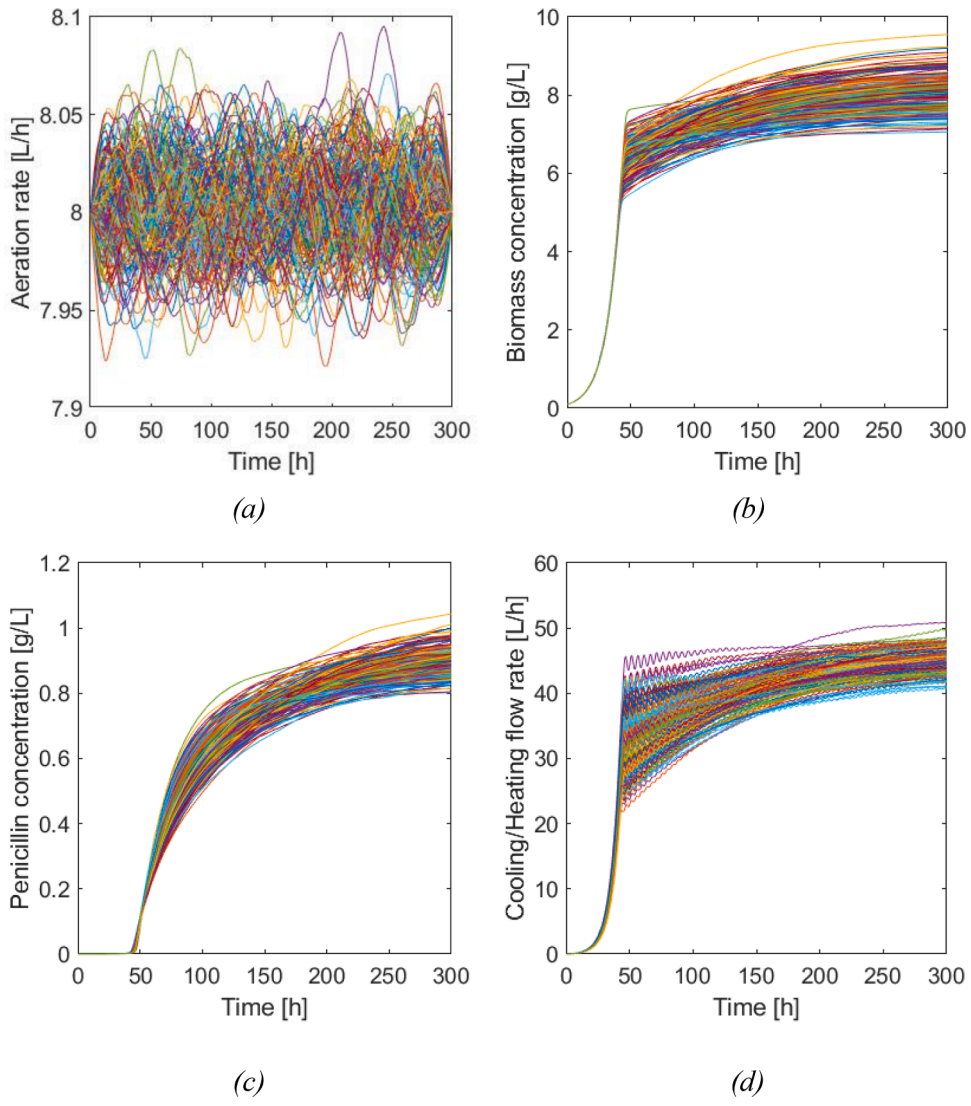

Fig. 6. Time profiles of (*a*) aeration rate, (*b*) biomass concentration, (*c*) penicillin concentration, (*d*) cooling/heating water flow rate for the large-*N* batches.

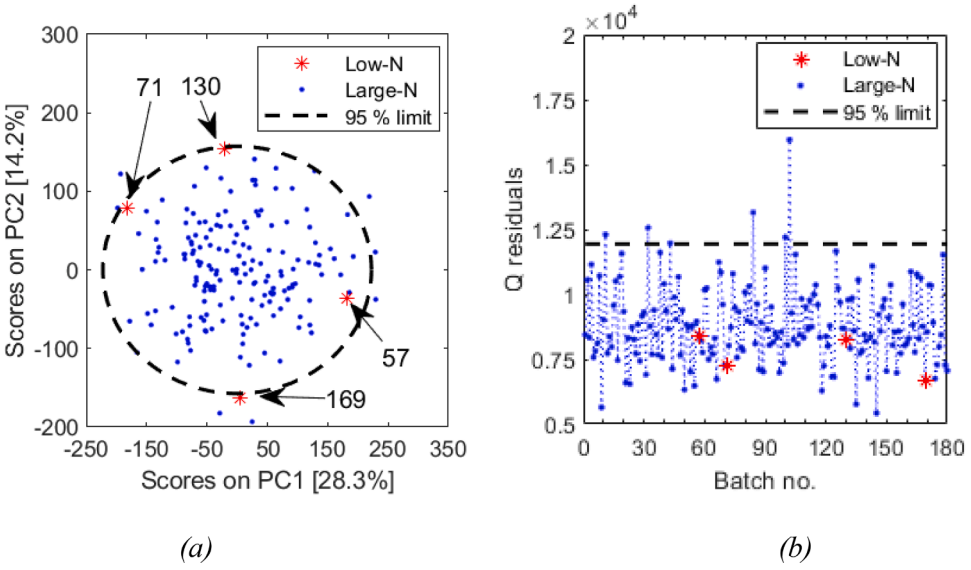

Fig. 7. Monitoring model built on the large-*N* batches: (*a*) score plot, (*b*) *Q* residuals plot. The four low-*N* batches selected as convenient for generating *in-silico* data (Case 1) are indicated by red stars.

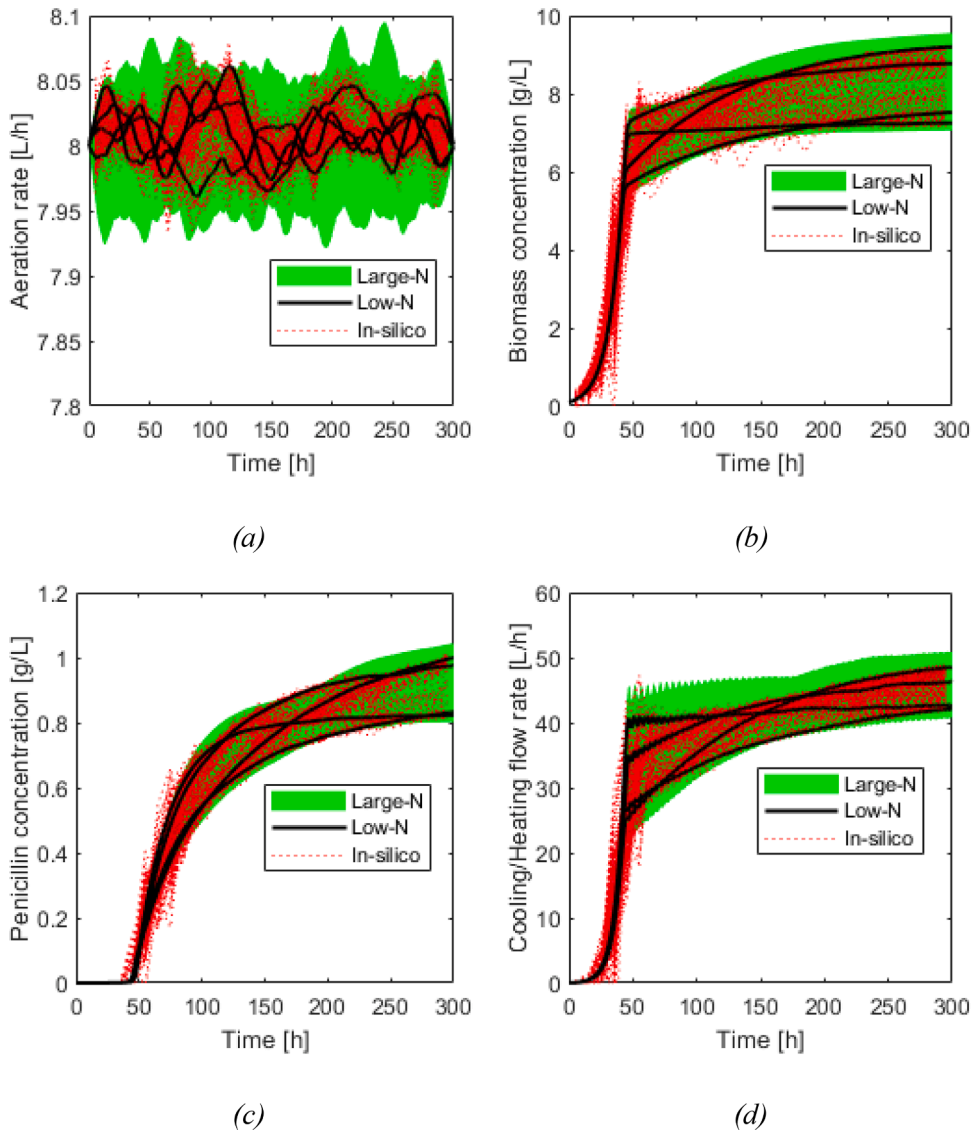

Fig. 8. Time profiles of (*a*) aeration rate, (*b*) biomass concentration, (*c*) penicillin concentration, (*d*) cooling/heating water flow rate for the same four low-*N* batches as in Fig. 7 (Case 1), and for the 200 *in-silico* batches generated from them. For convenience, the variability of the large-*N* dataset is shown as a green-shaded area.

Table 3
Explained variance and cumulative explained variance for the monitoring model built on the same four low-*N* batches as in Fig. 7 (Case 1) and the 200 *in-silico* batches generated from them. For comparison, the variance explained by the model built using the large-*N* dataset is also reported.

	Low-*N* scenario		Large-*N* scenario	
PC no.	Explained variance [%]	Cumulative explained variance [%]	Explained variance [%]	Cumulative explained variance [%]
1	7.85	8.04	28.31	28.31
2	6.37	14.22	14.18	42.50
3	5.82	20.04	11.14	53.63
4	5.31	25.34	8.98	62.62
5	4.95	30.30	6.35	68.97
6	4.85	35.15	1.62	70.59
7	3.98	39.14		
8	3.41	42.55		
9	3.04	45.49		
10	2.72	48.30		
11	2.57	50.87		

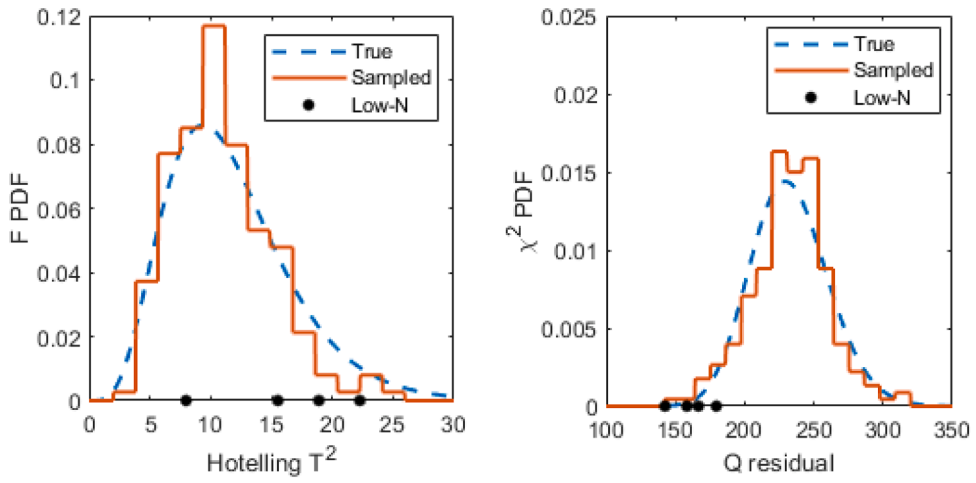

Fig. 9. Reconstructed (*a*) *F*-density, (*b*) χ^2-density for the same four low-*N* batches as in Fig. 7 (Case 1) and for the 200 *in-silico* batches generated from them.

Table 4
Mean and standard deviation of the coverage indicator i_{NOC} across all time points for the variables of Fig. 6, using the same four low-*N* batches as in Fig. 7 (Case 1) and the 200 *in-silico* batches generated from them.

	Aeration rate	Biomass concentration	Penicillin concentration	Cooling/Heating flow
Mean of i_{NOC}	0.9759	0.9775	0.9758	0.9824
Standard deviation of i_{NOC}	0.0089	0.0112	0.0130	0.0078

In the sequel, when speaking about "coverage of the NOC", we refer to the NOC defined by the low-*N* batches, unless otherwise stated. In fact, as already noted, the actual coverage of the NOC achieved in a large-*N* scenario is impossible to know in advance.

6.1. Assessing the monitoring performance under low-N and large-N scenarios

In this Section, we compare the process monitoring results from two monitoring models: one is built using the low-*N* batches together with the batches generated *in-silico*, and the other is built using the large-*N* batches. We first consider two low-*N* scenarios, namely one with $N = 4$ (Section 6.1.1), and one with $N = 3$ (Section 6.1.2). The purpose is to show that *in-silico* batch generation cannot lead to satisfactory monitoring results if too small a number of low-*N* batches are available. Then, for the case study with $N = 4$, we consider two additional sets of low-*N* batches from which the *in-silico* ones are generated (Sections 6.1.3 and 6.1.4). We show that it is not only the number *N* of real batches that can make the difference in the monitoring performance; in fact, also the characteristics of the available real batches are critical.

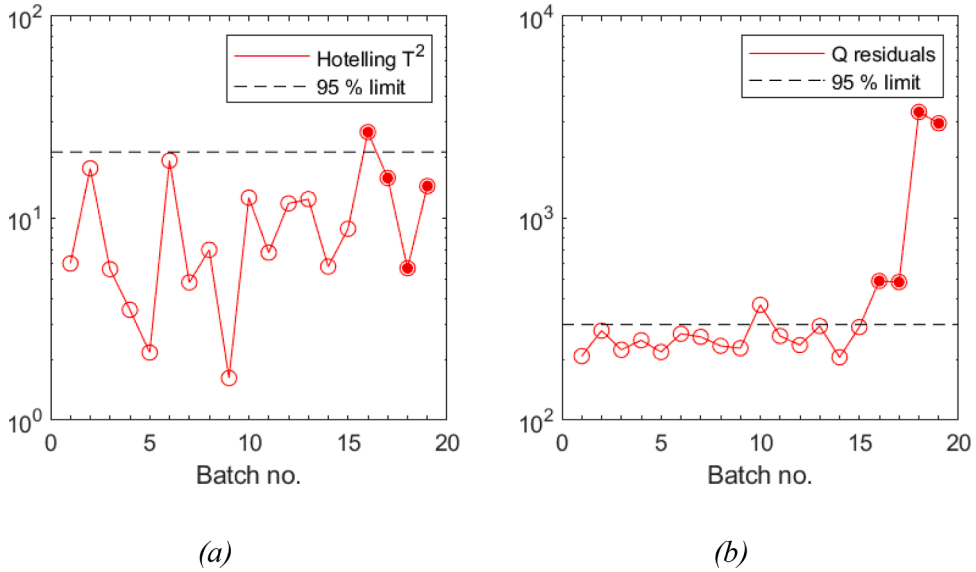

Fig. 10. Monitoring results for the validation dataset in terms of (*a*) Hotelling T^2, and (*b*) *Q* residuals for the model built on the same four low-*N* batches as in Fig. 7 (Case 1) and the 200 *in-silico* batches generated from them. The true faulty batches are denoted with closed symbols.

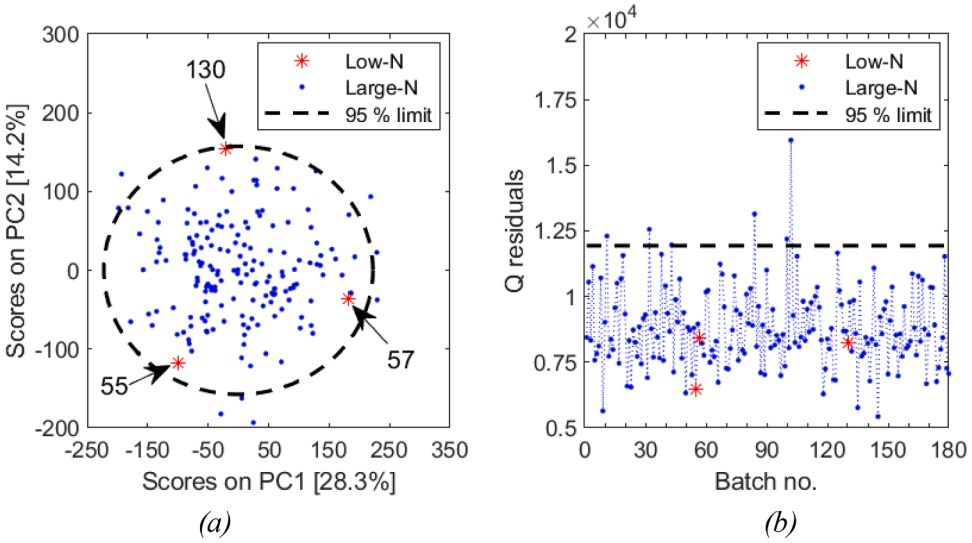

Fig. 11. Monitoring model built on the large-*N* batches: (*a*) score plot, (*b*) *Q* residuals plot. The three low-*N* batches selected as convenient for generating *in-silico* data for process monitoring (Case 2) are indicated by red stars.

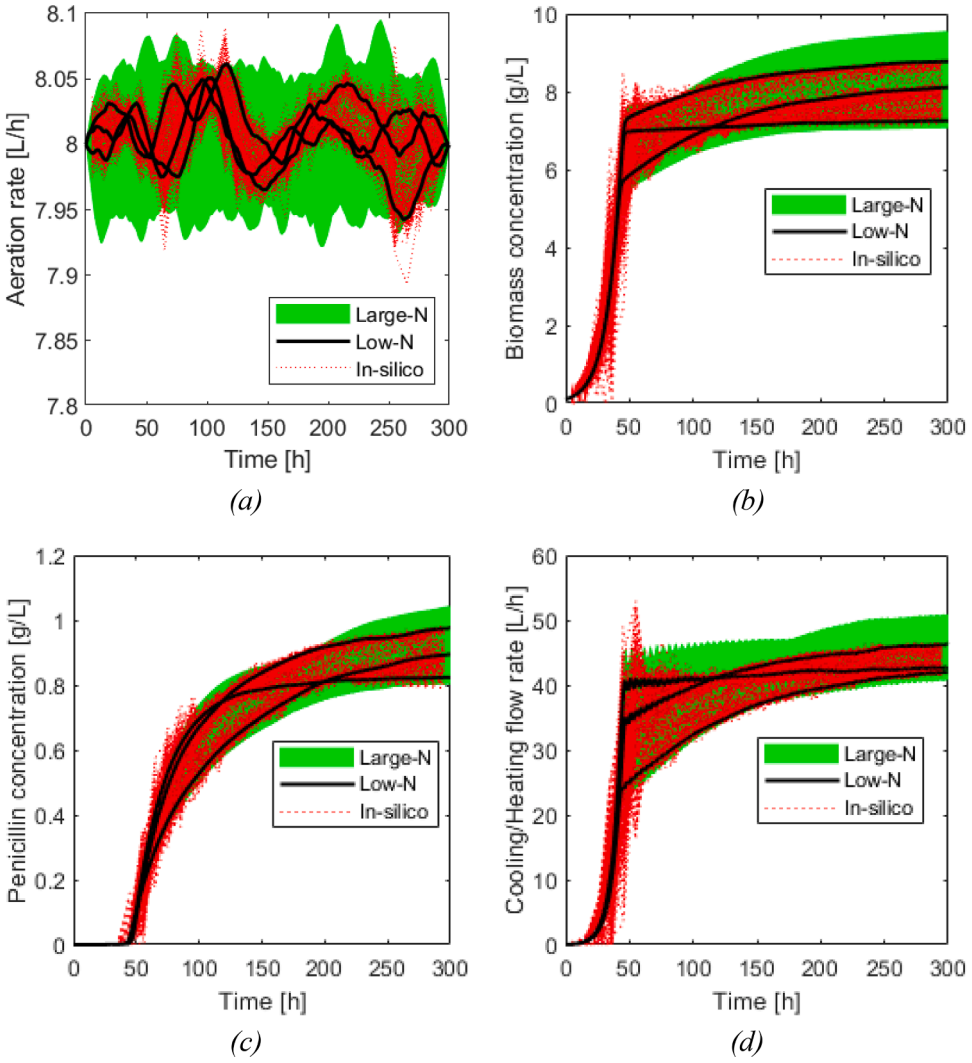

Fig. 12. Time profiles of (*a*) aeration rate, (*b*) biomass concentration, (*c*) penicillin concentration, (*d*) cooling/heating water flow rate for the same three low-*N* batches as in Fig. 11 (Case 2), and for the 350 *in-silico* batches generated from them. For convenience, the variability of the large-*N* dataset is shown as a green-shaded area.

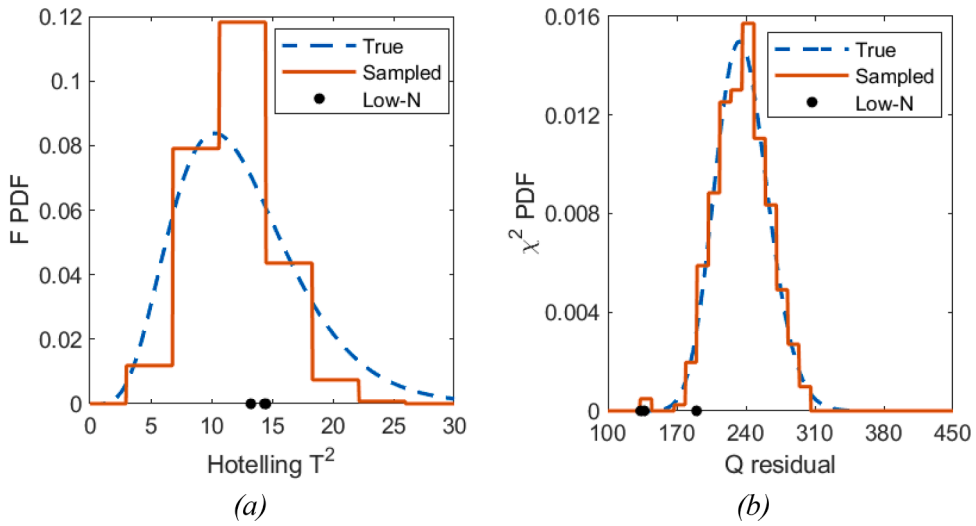

Fig. 13. Reconstructed (*a*) *F*-density, (*b*) χ^2-density for the same three low-*N* batches as in Fig. 11 (Case 2) and for the 350 *in-silico* batches generated from them.

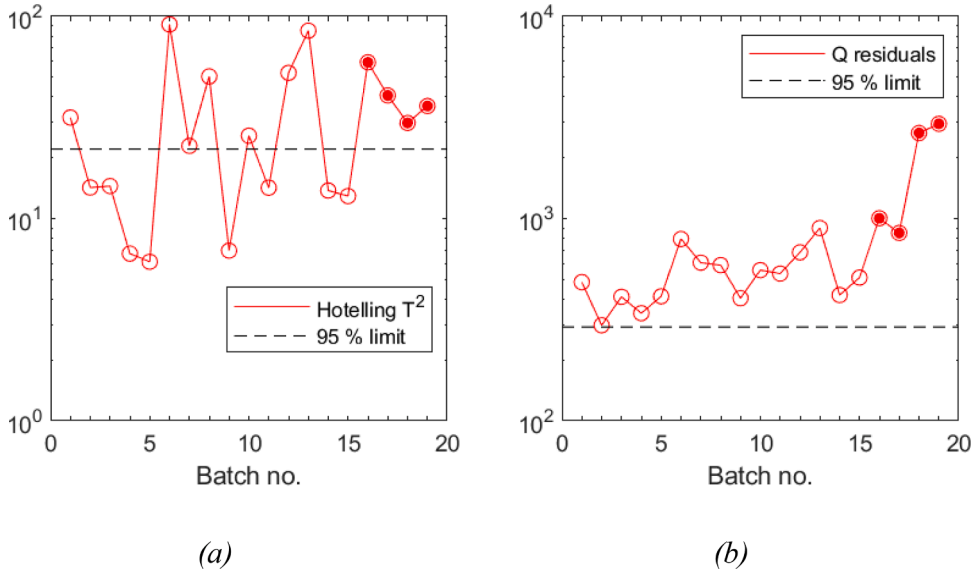

Fig. 14. Monitoring results for the validation dataset in terms of (*a*) Hotelling T^2, and (*b*) *Q* residuals for the model built on the same three low-*N* batches as in Fig. 11 (Case 2) and the 350 *in-silico* batches generated from them. The true faulty batches are denoted with closed symbols.

Fig. 6 shows the time profiles of four representative variables from the large-*N* batches. For *in-silico* batch generation under a process monitoring perspective, a convenient situation is one in which the low-*N* batches fulfill both the following conditions: *i*) they have small *Q* residuals, and *ii*) they span a large variability in the (unknown) NOC of the large-*N* scenario. The first condition is required because the *in-silico* batches are expected to be generated around the hyperspace defined by the low-*N* ones; therefore, if the low-*N* batches are lying on the hyperspace of the NOC of large-*N* scenario (i.e., $Q \cong 0$ for the low-*N* batches), the *in-silico* batches are likely to lie around such hyperplane. The second condition is required because the variability of the *in-silico* batches is approximately bounded by the one of the low-*N* batches; therefore, if the low-*N* batches span too small a variability in the NOC of the large-*N* scenario, the resulting monitoring model will be prone to labeling as abnormal a batch that is actually normal but simply shows a greater variability with respect to the low-*N* ones. Unfortunately, fulfillment of the above two conditions cannot be tested in a low-*N* scenario, because this would require knowing in advance the NOC space of the large-*N* scenario, which is impossible.

6.1.1. Case 1: sufficient number ($N = 4$) and appropriate characteristics of the real batches

A low-*N* case with $N = 4$ real batches fulfilling both conditions discussed above is considered. To identify the low-*N* batches, the PCA model built over the large-*N* dataset is considered. Let us assume that the low-*N* batches are those indicated by the red stars in Fig. 7 (namely, batches nos. 57, 71, 130 and 169): it can be seen that they span a large portion of the large-*N* NOC space (Fig. 7*a*), and are characterized by *Q* values that are well below the confidence limit (Fig. 7*b*).

The *in-silico* batch generation procedure returns 200 *in-silico* batches. Fig. 8 shows the time profiles of the same variables of Fig. 6, for the low-*N* batches and for the batches generated *in-silico*. A green-shaded area is reported for convenience to map the unknown true variability, as spanned by the large-*N* batches. In principle, this is also the variability that one would like to reconstruct with the *in-silico* batches generated from the low-*N* ones. It can be seen that the *in-silico* batch generation methodology works well: the *in-silico* trajectories cover the operating space bracketed by the low-*N* batches very satisfactorily, and the *in-silico* trajectories are consistent with the original ones (i.e., similar patterns

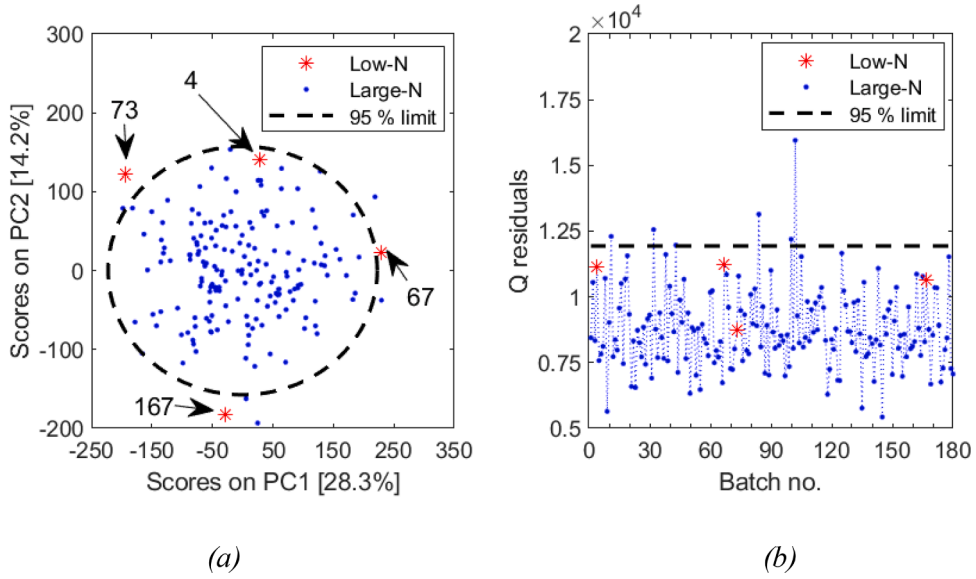

Fig. 15. Monitoring model built on the large-*N* batches: (*a*) score plot, (*b*) *Q* residuals plot. The four low-*N* batches deemed as unfit (because of large *Q* residuals) for generating *in-silico* data for process monitoring (Case 3) are indicated by red stars.

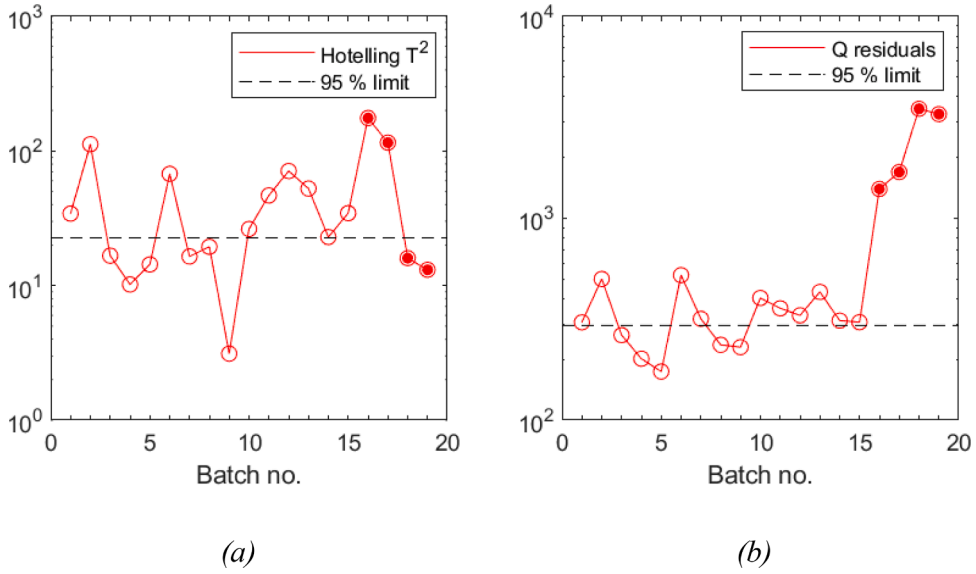

Fig. 16. Monitoring results for the validation dataset in terms of (*a*) Hotelling T^2, and (*b*) *Q* residuals for the model built on the same four low-*N* batches as in Fig. 15 (Case 3) and the 225 *in-silico* batches generated from them. The true faulty batches are denoted by closed symbols.

are reproduced, and no significant noise is generated). As expected, the variability of the *in-silico* batches is bounded by the one of the low-*N* batches; this is particularly clear for the plots in Fig. 8*a* and *d*.

A PCA monitoring model is then built using the low-*N* batches together with the batches generated in silico from them; 11 PCs are used, capturing $\bar{R}^2 \cong 50$ % of the variance. The number of PCs is somewhat greater than typically encountered in PCA modeling, and can be interpreted as an indirect indication that an "artificial" large-*N* scenario (as the one obtained through *in-silico* batches) embeds less systematic variance than a real one. This is particularly clear when the model built under the low-*N* scenario is compared to the one built under the large-*N* scenario (Table 3).

The fittings of the *F* density and of the χ^2 density are illustrated in Fig. 9, together with the T^2 and *Q* values of the low-*N* batches. The sampled T^2 density (Fig. 9*a*) satisfactorily approximates the theoretical continuous one, thus confirming that the coverage of NOC provided by the *in-silico* data is proper (i.e., the score points are multi-normally distributed). The T^2 values of the low-*N* batches are well within the extremes of the *F* density, indicating that the *in-silico* batches are within the variability of the low-*N* ones. With respect to the *Q* residuals (Fig. 9*b*), they are small for the low-*N* batches (left-hand side of the χ^2 density), indicating that they are close to the hyperspace of the low-*N* PCA model; moreover, the fitting of the χ^2 density is suitable.

With respect to the coverage indicator i_{NOC}, the average value and standard deviation along all the low-frequency time points for the four considered variables are summarized in Table 4. The four mean values are very close to 1, and the standard deviations are very small; this indicates that, for the four involved variables, the indicator i_{NOC} is very close to 1 across all the time points. Altogether, the results indicate that the necessary conditions for the monitoring model to be reliable are fulfilled.

The validation dataset is then considered, and results are illustrated

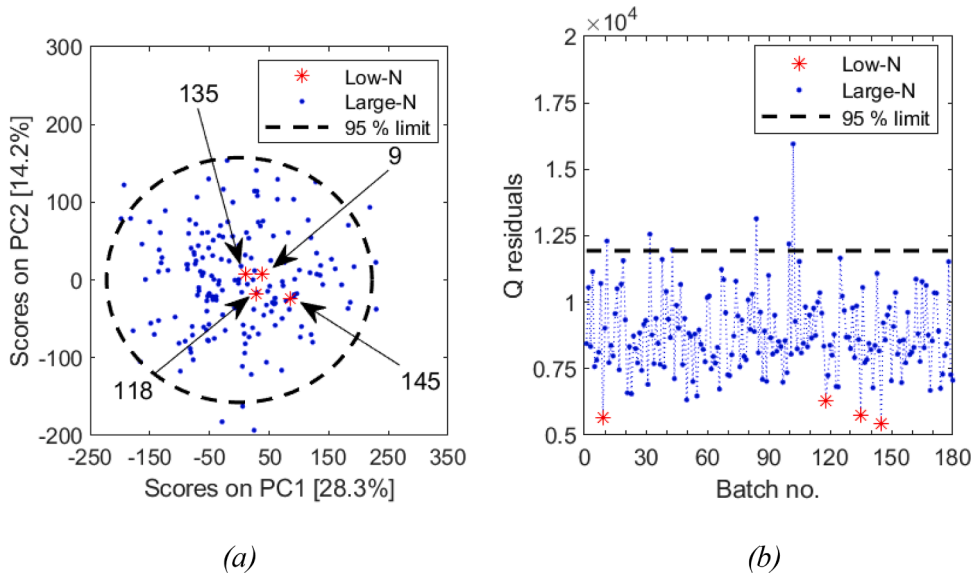

Fig. 17. Monitoring model built on the large-*N* batches: (*a*) score plot, (*b*) *Q* residuals plot. The four low-*N* batches deemed as unfit (because of low variability) for generating *in-silico* data for process monitoring (Case 4) are indicated by red stars.

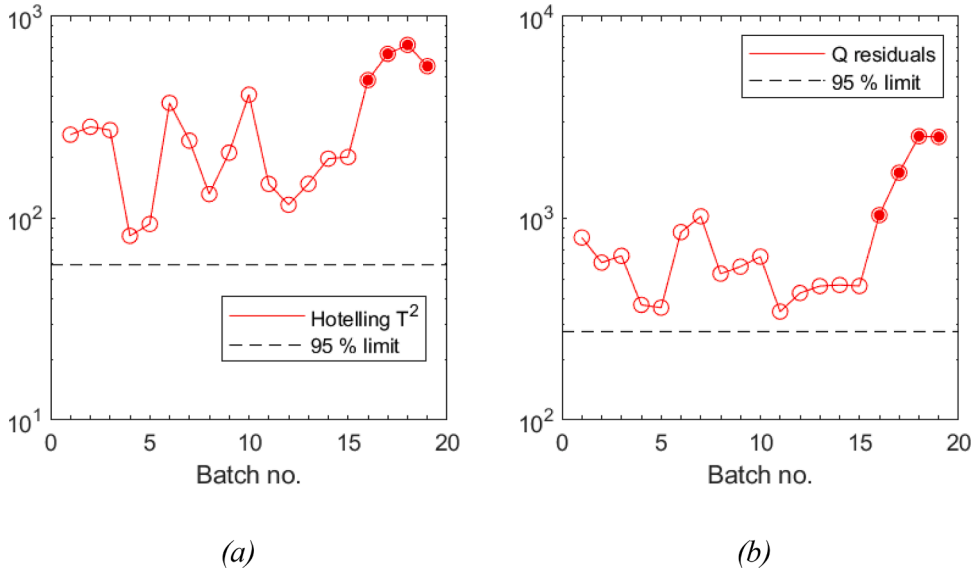

Fig. 18. Monitoring results for the validation dataset in terms of (*a*) Hotelling T^2, and (*b*) *Q* residuals for the model built on the same four low-*N* batches as in Fig. 17 (Case 4) and the 1500 *in-silico* batches generated from them. The true faulty batches are denoted by closed symbols.

in Fig. 10. We notice that the monitoring model calibrated with low-*N* and *in-silico* batches provides a good monitoring performance: all faulty batches (no. 16 through no. 19) are labeled as such in the *Q* residuals plot, and only one false alarm (batch no.10) is raised in the same plot. We remark that the performance of the monitoring model should be benchmarked also against false alarms. In fact, although a false alarm does not represent a concern for batch normality, it can lead to batch rejection, therefore representing an economic penalty associated to ineffective monitoring.

6.1.2. Case 2: insufficient number ($N = 3$) and appropriate characteristics of the real batches

We consider a low-*N* scenario with $N = 3$ real batches fulfilling both conditions discussed above. The score plot and the *Q* residuals plot are shown in Fig. 11. Again, the low-*N* batches (namely, batches nos. 55, 57 and 130) are selected in such a way that they span sufficient variability in the score plot and have small *Q* residuals.

For this simulation, the *in-silico* batches are generated using a threshold value $\delta_F^* = 0.16$ (rather than $\delta_F^* = 0.08$) for the reconstruction error of the *F* density, in order to improve convergence. A total of $R = 350$ *in-silico* batches are returned by the *in-silico* batch generation procedure (note that *R* is much greater than obtained in Case 1), and results are plotted in Fig. 12 in terms of process variable profiles. The coverage of NOC by the *in-silico* data is still good for all variables; however, since now the low-*N* batches span less variability than in Case 1, also the *in-silico* trajectories do.

A monitoring model using the available real batch datasets together with the *in-silico* ones is built. The fittings of the *F* density and of the χ^2 density are shown Fig. 13: both of them are satisfactory, as are the values of the *Q* residuals for the low-*N* batches. However, as discussed earlier, this provides only a necessary condition to obtain a reliable monitoring model, but sufficiency is not guaranteed. Indeed, Fig. 14 clarifies that the monitoring performance against the validation dataset is poor.

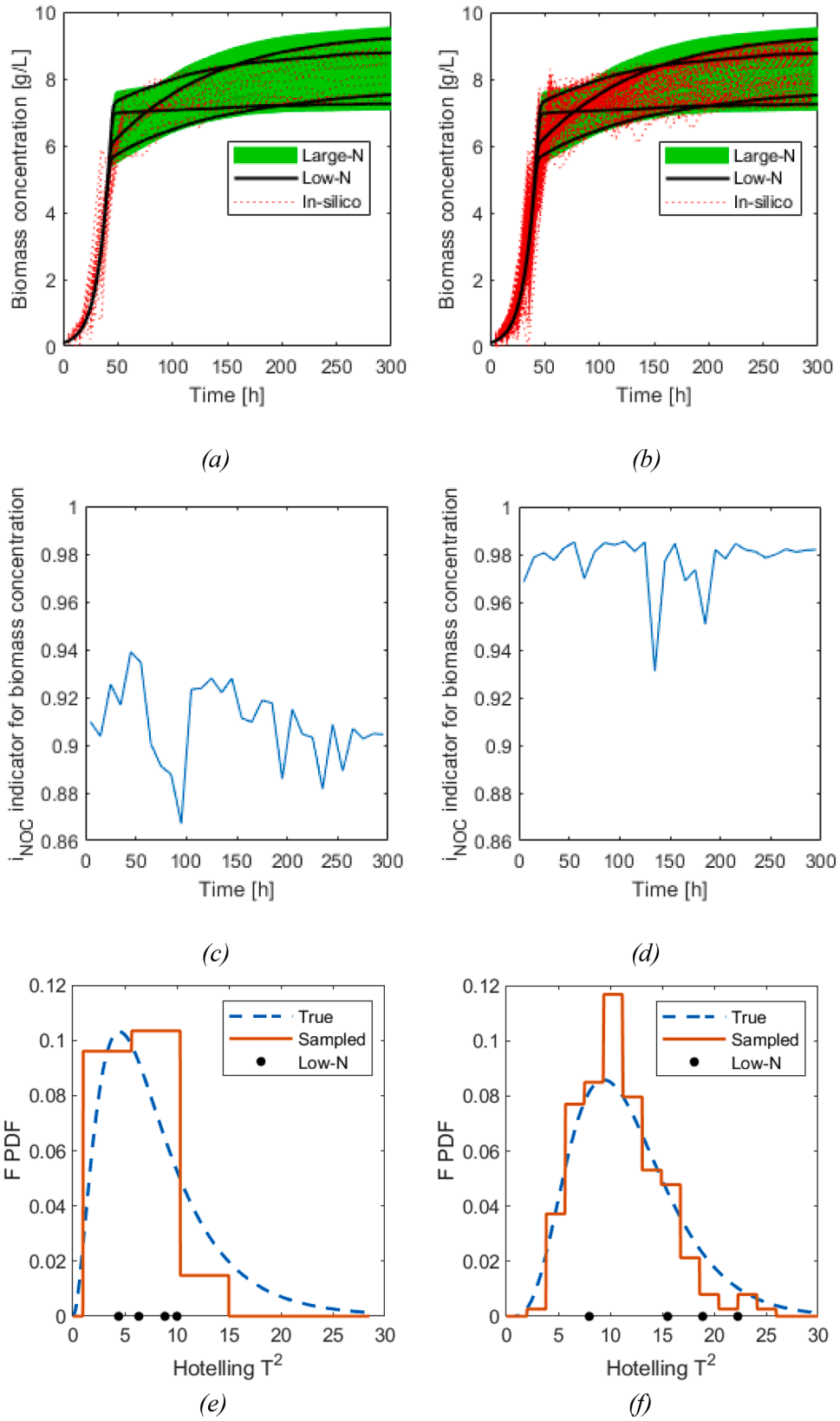

Fig. 19. Generation of *in-silico* batches using the same four low-*N* batches as in Fig. 7 (Case 1). Time profile of biomass concentration for (*a*) 25 *in-silico* batches, (*b*) 200 *in-silico* batches (for convenience, the variability of the large-*N* dataset is shown in (*a*) and (*b*) as a green-shaded area). Coverage indicator for biomass concentration for (*c*) 25 *in-silico* batches, (*d*) 200 *in-silico* batches. Fitting of *F*-density with (*e*) 25 *in-silico* batches, (*f*) 200 *in-silico* batches.

Table 5
Reconstruction errors for the F density, mean across all the time points and standard deviation across all the time points of the coverage indicator i_{NOC} for biomass concentration at different numbers of batches generated *in-silico* from the same four low-N batches as in Fig. 7 (Case 1).

No. of batches generated *in-silico*	F density reconstruction error	Coverage indicator i_{NOC}	
		Mean	Std. dev.
25	0.2000	0.9088	0.0164
50	0.1675	0.9416	0.0172
75	0.1341	0.9550	0.0177
100	0.1045	0.9626	0.0185
125	0.0941	0.9691	0.0141
150	0.0843	0.9738	0.0111
175	0.0683	0.9758	0.0111
200	0.0663	0.9775	0.0112

Whereas all validation faulty batches are correctly labeled as such by the low-N monitoring model, almost all normal batches are wrongly identified as faulty. This indicates that when too few low-N batches are available, the monitoring performance can be inadequate, even if the low-N batches have adequate characteristics and the *in-silico* batch generation step returns reasonable trajectories for the process variables (Reis, et al., 2021).

6.1.3. Case 3: sufficient number ($N = 4$) and inappropriate characteristics (large Q residuals) of the real batches

We assume that enough real batches are available, and that they span a large variability in the score plot (namely, $N = 4$; batches nos. 4, 67, 73 and 167; Fig. 15*a*). However, three of these batches are characterized by large Q residuals (Fig. 15*b*).

After generating 225 *in-silico* batches and building the relevant monitoring model, the monitoring results shown in Fig. 16 are obtained. Despite the number of real batches is the same as in Case 1, the monitoring performance is inadequate (yet better than in Case 2), given that several normal validation batches are erroneously labeled as faulty. Note that, also in this case, all true fault batches are correctly labeled as faulty by the monitoring model.

6.1.4. Case 4: sufficient number ($N = 4$) and inappropriate characteristics (insufficient variability) of the real batches

In this case, the available low-N batches do have small Q residuals, but they span too low a variability in the hyperplane of the PCs (Fig. 17). The *in-silico* data generation algorithm returns the profiles for $R = 1500$ batches. This number is meaningless from a practical perspective, and provides indirect indication of the very small variability covered by the available real batches. The monitoring results on the validation dataset are shown in Fig. 18, and are clearly unacceptable: all batches are labeled as faulty both in the Q residuals chart and in the T^2 chart.

Cases 3 and 4 support the conclusion that, if the ultimate goal of *in-silico* data generation is building a dataset to be used for process monitoring, the *number* of available real batches is not the only parameter that matters. The *characteristics* of the available real batches are also crucial. Inadequate low-N data characteristics may not prevent the *in-silico* data generation exercise to complete satisfactorily. However, the resulting dataset will be inappropriate for process monitoring purposes, because the monitoring model will likely rise several false alarms.

6.2. Impact of the number of batches generated in-silico

The purpose of this Section is to discuss how the number of batches generated *in-silico* affects the coverage of the NOC; namely, reference is made to the F density function and to the i_{NOC} coverage indicator. We assume to deal with a low-N scenario where the same $N = 4$ real batches with favorable characteristics as in Case 1 are available. We compare a situation, in which 25 *in-silico* batches are generated, to one where the number of *in-silico* batches (namely, 200) is returned automatically by the procedure proposed in Section 4.1. Among all variables, only the biomass concentration is considered; results for the other variables are qualitatively similar, and are omitted for conciseness.

Fig. 19*a* and *b* visually clarify that, when only 25 batches are generated *in-silico*, the coverage is much smaller than when 200 *in-silico* batches are generated; this issue is also quantitatively confirmed by the coverage indicator i_{NOC} (Fig. 19*c* and *d*). This affects the reconstruction of the F density, which is quite rough when only 25 batches are generated (Fig. 19*e*), thus indicating that the scores are far from being multinormally distributed; the situation is clearly better when 200 batches are generated *in-silico* (Fig. 19*f*).

The reconstruction error for the F density is calculated according to (29) at several numbers of *in-silico* batches, and outcomes are reported in Table 5 together with summary values for i_{NOC}. As expected, as the number of *in-silico* batches increases, the reconstruction error decreases, and i_{NOC} gets closer to the reference. We conclude that the coverage indicator and the density reconstruction errors are useful metrics that can guide the user in the determination of the appropriate number of batches to be generated in silico for process monitoring purposes.

6.3. Impact of the GP-SS model parameters

The main parameters of the GP-SS model are the memory parameter L and the number J of resampled trajectories that are obtained for each variable from the stratified resampling step. Next, we discuss how the tunings of L (through $\delta_L^{(v)}$) and of J affect the *in-silico* batch generation results, and whether the methodologies we have proposed to estimate appropriate values for these parameters return meaningful results. We refer to the same $N = 4$ favorable low-N batches considered in Case 1.

6.3.1. Impact of parameter L

In Section 3.2, we have proposed a methodology to estimate L for a given variable v at a given time point t, based on a threshold value $\delta_L^{(v)}$ for the correlation between the set of values of the variable v at t, and the set of the values that v takes at a preceding time point. Next, we discuss the impact of $\delta_L^{(v)}$ (hence of L) on the *in-silico* generation results. For convenience, we restrict our discussion to one single process variable, namely the penicillin concentration, and we write δ_L instead of $\delta_L^{(v)}$.

Fig. 20 compares the generation results for three values of the threshold, namely $\delta_L = 0.999$, $\delta_L = 0.5$ and $\delta_L = 0.001$. For better comparability, 200 *in-silico* batches are intentionally generated in all cases, regardless of the PDF reconstruction errors.

When δ_L is very large ($\delta_L = 0.999$), the coverage is poor in the final part of the batch (Fig. 20*a*), as also confirmed by the declining value of i_{NOC} (Fig. 20*b*). This is because the *in-silico* trajectories are tethered around the low-N ones; in such a situation, parameter L is very small ($L = 1$ for $t \geq 2$; Fig. 20*b*), and the inherent correlation structure of the variable (i.e., its autocorrelation) is therefore destroyed. This also explains why the *in-silico* trajectories are so tethered around the real ones.

The situation improves with $\delta_L = 0.5$ (this is the value used throughout all simulations discussed elsewhere in this paper for this specific variable); the proposed methodology returns values for L that slowly grow in time (Fig. 20*d*), and this determines a much improved

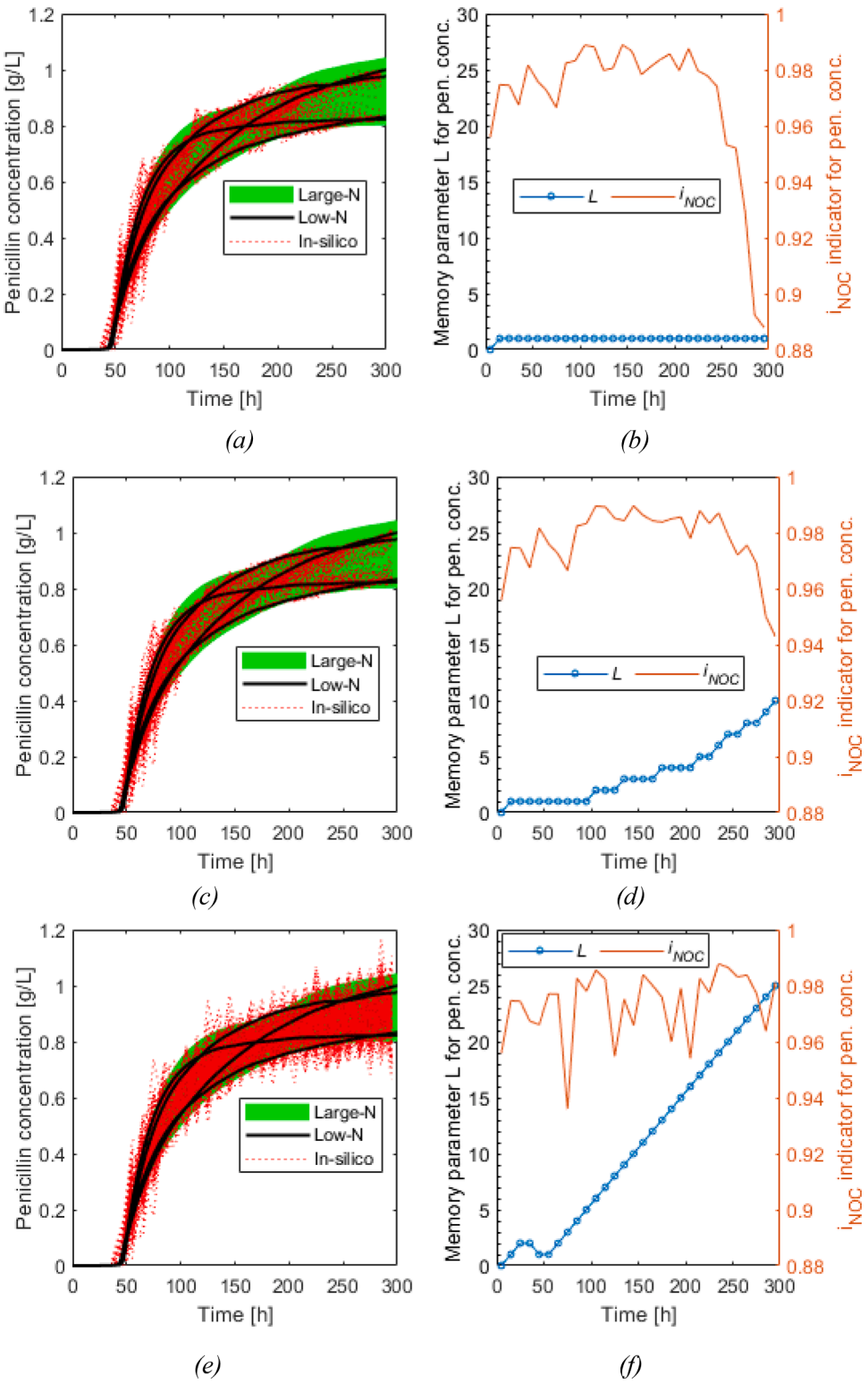

Fig. 20. Generation of 200 *in-silico* batches using the same four low-*N* batches as in Fig. 7 (Case 1). Time profile of the penicillin concentration for (*a*) $\delta_L = 0.999$, (*c*) $\delta_L = 0.5$, and (*e*) $\delta_L = 0.001$. Returned values for the memory parameter L and coverage indicator i_{NOC} for penicillin concentration for (*b*) $\delta_L = 0.999$, (*d*) $\delta_L = 0.5$, and (*f*) $\delta_L = 0.001$. For convenience, the variability of the large-*N* dataset is shown in (*a*), (*c*) and (*e*) as a green-shaded area.

coverage (Fig. 20*c* and *d*).

When $\delta_L = 0.001$, the value of L by the end of the batch is too large (Fig. 20*f*). This may explain why the *in-silico* trajectories are very noisy (Fig. 20*e*), an occurrence that makes i_{NOC} increase with respect to the case with $\delta_L = 0.5$ (Fig. 20*f*).

Unfortunately, fine tuning of δ_L is a matter of trial-and-error. Two opposing effects need compromising: *i*) decreasing δ_L improves the description of autocorrelation (and possibly of coverage); *ii*) increasing δ_L reduces noise.

6.3.2. *Impact of parameter J*

Parameter J sets the number of resampled trajectories to be obtained

Table 6
Monitoring errors for different values of the number of resampled trajectories J using the same four low-N batches as in Fig. 7 (Case 1).

	$J = 3760$ (calculated automatically)	$J = 1200$	$J = 600$	$J = 40$
Returned number of *in-silico* batches	200	175	375	200
Number of labeling errors	1/19	2/19	2/19	4/19

by stratified resampling for each variable. A methodology to tune the value of J was proposed in Section 3.1, and its effectiveness is discussed here. We consider several scenarios, where J is either estimated automatically according to the proposed methodology (using the parameters of Table 2), or it is intentionally set to a smaller value. For each scenario, the *in-silico* batch generation is carried out automatically, and the relevant PCA monitoring model is built. The monitoring models are then assessed against the validation dataset. Results are summarized in Table 6. We note that the value of J returned automatically leads to the smallest number of labeling errors. This is probably due to the fact that too small a value for J is likely to cause a smaller variability of the resampled trajectories (hence of the *in-silico* batches) with the respect to the low-N ones. Whereas this confirms the appropriateness of the methodology we have proposed for the tuning of J, it must nevertheless be noted that a six-fold decrease of the number of resampled trajectories ($J = 600$) over the value returned automatically does not cause a dramatic loss of monitoring performance.

7. Conclusions

In this paper, we have thoroughly investigated a methodology, already proposed in the literature, that enables batch process monitoring under low-N (i.e., small data) scenarios; namely, we have addressed the issue of batch process monitoring when limited historical manufacturing data are available. The problem is particularly meaningful when a new product is to be manufactured for the first time in a given facility, or when an old product is to be transferred to a facility wherein it has never been manufactured before. The methodology exploits machine learning algorithms (based on Gaussian process state-space models) to generate *in-silico* batch datasets from the few available historical ones, and then uses the overall pool of real and *in-silico* data to build a process monitoring model for the batch process. The progress made by the research in this paper is twofold: on the algorithmic side, and on the process monitoring side.

From an algorithmic perspective, we have developed automatic (or semiautomatic) procedures to tune the values of several parameters required by the machine-learning framework, namely: *i*) the number of trajectories to be resampled for each process variable, *ii*) the memory (or lag) of each variable at each time point, and *iii*) the number of resampled trajectories to be retained for each variable at each time point for GP-SS model training. The proposed procedures facilitate the development, interpretation and fine tuning of the GP-SS model, thereby streamlining the generation of consistent *in-silico* batch trajectory data.

From a process monitoring perspective, we have clarified that, to target the development of a reliable process monitoring model under a low-N scenario, it is not only the number of available historical batches that matters. In fact, the variability they cover across the (unknown) domain of normal operating conditions of the process is also central. Insufficient coverage does not prevent the *in-silico* batch data generation process to complete satisfactorily; however, the resulting dataset will be unfit for process monitoring purposes, typically resulting in a large number of false alarms. Additionally, we have proposed: *iv*) a method, based on the fitting of probability density functions of the monitoring model statistics, to determine the appropriate number of batches to be generated *in-silico*, and *v*) a set of indicators, based on Q-residuals plots and on a new univariate coverage indicator, to assess the fulfillment of necessary conditions that the overall pool of real and *in-silico* data should possess in order to be exploited for process monitoring purposes.

We believe that further improvements of the *in-silico* batch data generation methodology should be directed to better address the issue of correlation in the data. Strictly speaking, correlation between variables is not considered during GP-SS model training. This can significantly downgrade the performance of a process monitoring model (e.g., one based on principal component analysis), because it is precisely by analyzing how each variable co-vary with the others (i.e., by modeling the cross-correlation between variables) that abnormal operating conditions can be discriminated from normal ones. Therefore, being able to generate *in-silico* trajectories that maintain the same cross-correlation structure as the real ones is expected to further enhance the potential for exploitation of this machine-learning methodology in a process monitoring perspective.

Data availability

The Matlab software code (bGen) that was developed to carry out the research described in this paper is available at the following GitHub link: https://github.com/antoniobenedetti-pmh/BGen

Funding

This study was funded by GSK R&D, Medicine Development & Supply, Drug Substance Development (U.S.).

CRediT authorship contribution statement

Luca Gasparini: Methodology, Software, Validation, Formal analysis, Investigation, Data curation, Writing – original draft, Visualization. **Antonio Benedetti:** Conceptualization, Resources, Writing – review & editing, Project administration, Software, Funding acquisition. **Giulia Marchese:** Resources, Software, Writing – review & editing. **Connor Gallagher:** Resources, Writing – review & editing. **Pierantonio Facco:** Conceptualization, Formal analysis, Writing – review & editing. **Massimiliano Barolo:** Conceptualization, Formal analysis, Resources, Writing – review & editing, Supervision, Project administration, Funding acquisition.

Declaration of Competing Interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests:

Luca Gasparini reports financial support was provided by GSK R&D. Massimiliano Barolo reports a relationship with GSK R&D that includes: funding grants. Antonio Benedetti, Giulia Marchese and Connor Gallagher are employees of the GSK group of companies.

Acknowledgments

A preliminary investigation on this topic was carried out in 2021 by Mr. Alberto Marchetto as a final thesis project of the Master's Degree program in Chemical and Process Engineering at University of Padova. We gratefully acknowledge his contribution.

Appendix A - Formal description of the model training procedure for dynamics learning

This Appendix contains a formal description of the model training procedure for dynamics learning presented in Section 2.3.2.

The kernel matrix $\mathscr{K}(\mathbf{Z}_{t-L,\dots,t-1})$ is defined through the kernel function $\mathscr{K}(\cdot,\cdot)$ at (6) in such a way that

$$\left[\mathscr{K}\left(\mathbf{Z}_{t-L,\dots,t-1}\right)\right]_{i_1,i_2} = \mathscr{K}\left(\mathbf{z}^{(i_1)}_{t-L,\dots,t-1}, \mathbf{z}^{(i_2)}_{t-L,\dots,t-1}\right),\ i_1, i_2 = 1,\dots,K \tag{A.1}$$

The GP-SS model of the process is then

$$\mathbf{z}_t\ |\mathbf{Z}_{t-L,\dots,t-1} = \mathscr{N}\left(\mathbf{0}_K, \mathscr{K}\left(\mathbf{Z}_{t-L,\dots,t-1}\right)\right), \tag{A.2}$$

$$\mathbf{y}_t\ |\mathbf{z}_t = \mathscr{N}\left(\mathbf{z}_t, \sigma^2\mathbf{I}_{K\times K}\right), \tag{A.3}$$

with $\mathscr{N}$ indicating a multi-variate Gaussian density, the vector $\mathbf{z}_t$ representing the internal state of the process among the K considered trajectories at time point t, $\mathbf{0}_K$ representing the column vector with all its K elements equal to 0 and $\mathbf{I}_{K\times K}$ the $K \times K$ identity matrix. Eq. (A.2) explains how the value of the variable v at time point t (randomly) depends on its preceding L ones.

Parameters $\{\eta,\ \alpha,\ \sigma\}$ are estimated through a Maximum Likelihood estimator, using the training dataset $\mathscr{D}_{v,t}$ (as in Tulsyan et al. (2019)). Initial guesses can be determined following Ulapane et al. (2020); if numerical issues prevent from doing this, then no initial guesses are provided, and the task is automatically executed by a given solver (e.g., MATLAB® can be used to this purpose).

Appendix B - Proof about $D(J,h)$

This Appendix contains a proof for $D(J,h)$, which was introduced in Section 3.1.2 with reference to Eq. (9).

The framework is as follows:

- w different elements are available (in our specific context, the elements are the high-frequency time points themselves);
- J random samplings of these elements are done.

If the order of the samplings is meaningful, that is, if sequences (and not combinations) are actually considered, the total number of the possible outcomes is w^J; indeed, each one of the J samplings has w possible different outcomes. Among these w^J sequences, a fraction of them does not contain exactly w different points, but fewer; this is because some sequences present repetitions. Clearly, each one of the possible w^J outcomes has h different elements, where h may take the values 1, 2, …, w. For convenience, denote by $\widetilde{D}(J,h,\ w)$ the number of sequences, with length equal to J, containing exactly h different elements, and these h different elements are chosen from a set of w different ones. The number of sequences containing w different elements is found by subtracting the number of sequences with fewer than w different elements from the total w^J, hence

$$D(J,w) = w^J - \sum_{h=1}^{w-1}\widetilde{D}(J,w,h). \tag{B.1}$$

The number $\widetilde{D}(J,w,h)$ of sequences of J elements containing h different ones, with these ones coming from a set of w different elements, can be calculated as

$$\widetilde{D}(J,w,h) = \binom{w}{h} D(J,h). \tag{B.2}$$

Indeed, the number of combinations of h elements chosen from w ones is $\binom{w}{h}$. To each combination of h elements, one can associate $D(J,h)$ sequences containing these specific h elements (and none else). Then, (B.2) follows. By substituting it into (B.1), Eq. (9) is obtained.

Appendix C - Proofs about the univariate coverage indicator i_{NOC}

This Appendix gathers some technical proofs concerning the univariate coverage indicator i_{NOC} introduced in Section 4.2.2.

C.1 Calculation of the mean of $\mathbf{y}$

The aim here is to justify the existence of (33). First, recall (30) and (31); by substituting them into (32), we get

$$\mu_{\mathbf{y}} = \frac{1}{\widetilde{N}-1}\sum_{\tilde{n}=1}^{\widetilde{N}-1}\left([\mathbf{x}]_{\tilde{n}+1} - [\mathbf{x}]_{\tilde{n}}\right) \tag{C.1}$$

$$= \frac{1}{\widetilde{N}-1}\left(\sum_{\tilde{n}=2}^{\widetilde{N}}[\mathbf{x}]_{\tilde{n}} - \sum_{\tilde{n}=1}^{\widetilde{N}-1}[\mathbf{x}]_{\tilde{n}}\right) \tag{C.2}$$

$$= \frac{[\mathbf{x}]_{\tilde{N}} - [\mathbf{x}]_1}{\tilde{N} - 1} \tag{C.3}$$

$$= \frac{1}{\tilde{N} - 1} \tag{C.4}$$

which is the same as (33). Moreover, this proves that

$$\sum_{\tilde{n}=1}^{\tilde{N}-1} [\mathbf{y}]_{\tilde{n}} = 1. \tag{C.5}$$

C.2. Bounds for i_{NOC}

A further goal is to prove that 0 is a lower bound for i_{NOC}. Using (C.5), one can write

$$\mathbf{y} = \left[1 - \sum_{\tilde{n}=2}^{\tilde{N}-1} [\mathbf{y}]_{\tilde{n}} \quad [\mathbf{y}]_2 \quad \dots \quad [\mathbf{y}]_{\tilde{N}-1} \right]^\top. \tag{C.6}$$

Then, we have

$$\sum_{\tilde{n}=1}^{\tilde{N}-1} [\mathbf{y}]_{\tilde{n}}^2 = \left(1 - \sum_{\tilde{n}=2}^{\tilde{N}-1} [\mathbf{y}]_{\tilde{n}} \right)^2 + \sum_{\tilde{n}=2}^{\tilde{N}-1} [\mathbf{y}]_{\tilde{n}}^2 \tag{C.7}$$

$$= 1 - 2 \cdot \sum_{\tilde{n}=2}^{\tilde{N}-1} [\mathbf{y}]_{\tilde{n}} + \left(\sum_{\tilde{n}=2}^{\tilde{N}-1} [\mathbf{y}]_{\tilde{n}} \right)^2 + \sum_{\tilde{n}=2}^{\tilde{N}-1} [\mathbf{y}]_{\tilde{n}}^2 \tag{C.8}$$

$$\leq 1 - 2 \cdot \sum_{\tilde{n}=2}^{\tilde{N}-1} [\mathbf{y}]_{\tilde{n}} + 2 \cdot \sum_{\tilde{n}=2}^{\tilde{N}-1} [\mathbf{y}]_{\tilde{n}}^2 \tag{C.9}$$

$$= 1 + 2 \cdot \sum_{\tilde{n}=2}^{\tilde{N}-1} [\mathbf{y}]_{\tilde{n}} \cdot \underbrace{([\mathbf{y}]_{\tilde{n}} - 1)}_{\leq 0} \tag{C.10}$$

$$\leq 1 \tag{C.11}$$

and so $i_{\mathrm{NOC}} = 1 - \sum_{\tilde{n}=1}^{\tilde{N}-1} [\mathbf{y}]_{\tilde{n}}^2 \geq 0$. The step from (C.8) to (C.9) is justified by the Cauchy-Schwarz inequality, which guarantees that $(\sum_{\tilde{n}=2}^{\tilde{N}-1} [\mathbf{y}]_{\tilde{n}})^2 \leq \sum_{\tilde{n}=2}^{\tilde{N}-1} [\mathbf{y}]_{\tilde{n}}^2$. The proof of the upper bound is trivial, since it is $\sum_{\tilde{n}=1}^{\tilde{N}-1} [\mathbf{y}]_{\tilde{n}}^2 > 0$, and it easily follows that $i_{\mathrm{NOC}} = 1 - \sum_{\tilde{n}=1}^{\tilde{N}-1} [\mathbf{y}]_{\tilde{n}}^2 < 1$.

C.3 Upper and lower bounds for i_{NOC}

Reconsider (41). The upper bound for i_{NOC} corresponds to the situation in which the coverage is excellent. This can be represented mathematically by the fact that the vector $\mathbf{x}$ is composed of equally spaced elements, that is $[\mathbf{x}]_{\tilde{n}} = (\tilde{n} - 1)/(\tilde{N} - 1)$, which leads to $[\mathbf{y}]_{\tilde{n}} = 1/(\tilde{N} - 1)$ for each $\tilde{n}$; then, (41) becomes

$$i_{\mathrm{NOC}} = \frac{\tilde{N} - 2}{\tilde{N} - 1}, \tag{C.12}$$

which is close to 1 provided that $\tilde{N}$ is sufficiently large, which occurs if a sufficient number of *in-silico* batches have been generated.

The lower bound for i_{NOC} occurs when all the $\tilde{N}$ values in $\mathbf{x}$ are either 0 or 1. This can be the case if $N = 2$, that is only 2 real batches are available, and all the generated *in-silico* trajectories are tethered around the real ones for variable v at time point t. In such case, vector $\mathbf{x}$ contains 0′s as the left elements and 1′s as the right ones (regardless of their repartitioning). This implies that $\mathbf{y}$ has only one entry equal to 1 and the remaining ones equal to 0; clearly, from (41), it follows that $i_{\mathrm{NOC}} = 0$.

Appendix D - Generation of the historical datasets

We provide information on how to generate the large-N and validation datasets discussed in Section 6 using the first-principles penicillin fermentation model presented in Section 5 (Birol et al., 2002).

The NOC are characterized by acceptable variability around the nominal initial conditions (Table D.1) and the nominal operating variables (Table D.2) of the process. Acceptable variability is obtained by sampling parameter ε (see the above Tables) from the Gaussian density $\mathcal{N}(0,1)$; this operation is done independently for each variable subject to variability. Note that NOC characterize the entire set of batches in the large-*N* dataset, and 15 (out of 19) validation batches.

Table D.1
Nominal initial conditions, and acceptable variability around them, for the generation of large-*N* and validation batches from the penicillin simulation model (ε is a sample from the Gaussian density $\mathcal{N}(0,1)$).

Initial condition	Initial condition
Substrate concentration [g/L]	$15+\varepsilon$
Dissolved O_2 concentration [g/L]	1.16
Biomass concentration [g/L]	0.1
Penicillin concentration [g/L]	0
Culture volume [L]	$150+10\varepsilon$
CO_2 concentration [mmol/L]	$0.75+0.05\varepsilon$
Hydrogen ion concentration H^+ [mol/L]	$10^{-5+0.1\varepsilon}$
Substrate feed temperature [K]	$297+0.2\varepsilon$
Reactor temperature [K]	298
Generated heat [kcal/h]	0

Table D.2
Nominal operating variables, and acceptable variability around them, for the generation of large-*N* and validation batches from the penicillin simulation model (ε is a sample from the Gaussian density $\mathcal{N}(0,1)$).

Operating variable	Value
Aeration rate [L/h]	8
Agitator power input [W]	$30+4\varepsilon$
Substrate feed rate [L/h]	$0.04+0.001\varepsilon$

The four faulty validation batches are characterized by nominal conditions similar to those of the NOC ones. However, unexpected changes in the agitator power input or in the aeration rate occur at given times, as detailed in Table D.3.

Table D.3
Characterization of faulty conditions in the validation dataset.

Batch nos.	Faulty variable	Fault type	Start time [h]	End time [h]	Fault magnitude
16, 17	Agitator power input	Step	90	250	−40 %
18, 19	Aeration rate	Ramp	150	300	-2.0×10^{-3} (L/h)/h

The reactor temperature controller settings and pH controller settings are the same as in the original reference.

References

Agarwal, P., Tamer, M., Sahraei, M.H., Budman, H., 2019. Deep learning for classification of profit-based operating regions in industrial processes. Ind. Eng. Chem. Res. 59 (6), 2378–2395.

Berg, B.A., Harris, R.C., 2008. From data to probability densities without histograms. Comput. Phys. Commun. 179 (6), 443–448.

Birol, G., Ündey, C., Cinar, A., 2002. A modular simulation package for fed-batch fermentation: penicillin production. Comput. Chem. Eng. 26 (11), 1553–1565.

Chiang, L.H., Braun, B., Wang, Z., Castillo, I., 2022. Towards artificial intelligence at scale in the chemical industry. AIChE J. e17644.

Hernandez, H., 2018. Comparison of methods for the reconstruction of probability density functions from data samples. ForsCh. Res. Rep. 12.

Jackson, J.E., 1991. A User's Guide to Principal Components. John Wiley & Sons s.l.

Kourti, T., 2003. Multivariate dynamic data modeling for analysis and statistical process control of batch processes, start-ups and grade transitions. J. Chemom. 17 (1), 93–109.

Kourti, T., Nomikos, P., MacGregor, J.F., 1995. Analysis, monitoring and fault diagnosis of batch processes using multiblock and multiway PLS. J. Process Control 5 (4), 277–284.

Kropotov, Y.A., Belov, A.A., Proskuryakov, A.Y., 2018. Estimation of the Distribution Probability Density Acoustic Signals and interferences, the Reconstruction Methods. IEEE, pp. 1–6 s.l.

Kullback, S., 1968. Information Theory and Statistics. Dover Publi. Inc., NY.

Ku, W., Storer, R.H., Georgakis, C., 1995. Disturbance detection and isolation by dynamic principal component analysis. Chemom. Intell. Lab. Syst. 30 (1), 179–196.

Merhav, N., Gutman, M., Ziv, J., 1989. On the estimation of the order of a Markov chain and universal data compression. IEEE Trans. Inf. Theory 35 (5), 1014–1019.

Moore, C.J., Chua, A.J., Berry, C.P., Gair, J.R., 2016. Fast methods for training Gaussian processes on large datasets. R. Soc. Open Sci. 3 (5), 160125.

Morvai, G., Weiss, B., 2005. Order estimation of Markov chains. IEEE Trans. Inf. Theory 51 (4), 1496–1497.

Nakahama, H., et al., 1977. Dependency as a measure to estimate the order and the values of Markov processes. Biol. Cybern. 25 (4), 209–226.

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

Nomikos, P., MacGregor, J.F., 1995. Multivariate SPC charts for monitoring batch processes. Technometrics 37 (1), 41–59.

Peres, Y., Shields, P., 2005. Two new Markov order estimators. arXiv preprint math/0506080. https://doi.org/10.48550/arXiv.math/0506080.

Rasmussen, C.E., Williams, C.K.I., 2006. Gaussian Processes for Machine Learning. MIT Press, Cambridge, Massachusetts.

Rato, T.J., Delgado, P., Martins, C., Reis, M.S., 2020. First principles statistical process monitoring of high-dimensional industrial microelectronics assembly processes. Processes 8 (11), 1520.

Reis, M.S., Gins, G., 2017. Industrial process monitoring in the big data/industry 4.0 era: from detection, to diagnosis, to prognosis. Processes 5 (3), 35.

Reis, M.S., et al., 2021. Improving the sensitivity of statistical process monitoring of manifolds embedded in high-dimensional spaces: the truncated-Q statistic. Chemom. Intell. Lab. Syst. 215, 104369.

Tulsyan, A., Garvin, C., Ündey, C., 2018. Advances in industrial biopharmaceutical batch process monitoring: machine-learning methods for small data problems. Biotechnol. Bioeng. 115 (8), 1915–1924.
Tulsyan, A., Garvin, C., Ündey, C., 2019. Industrial batch process monitoring with limited data. J. Process Control 77, 114–133.
Ulapane, N., Karthic, T., Sarath, K., 2020. Hyper-parameter Initialization For Squared Exponential Kernel-Based Gaussian Process Regression. IEEE, pp. 1154–1159 s.l.
Wise, B.M., Gallagher, N.B., 1996. The process chemometrics approach to process monitoring and fault detection. J. Process Control 6 (6), 329–348.
Wu, H.H., Wu, S., 2009. Various proofs of the Cauchy-Schwarz inequality. Octogon Math. Mag. 17 (1), 221–229.