Chemical Engineering Science 302 (2025) 120780

Contents lists available at ScienceDirect

Chemical Engineering Science

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

MLAPI: A framework for developing machine learning-guided drug particle syntheses in automated continuous flow platforms

Arun Pankajakshan [a], Sayan Pal [a], Nicholas Snead [a], Juan Almeida [b], Maximilian O. Besenhard [a], Shorooq Abukhamees [c], Duncan Q.M. Craig [d], Asterios Gavriilidis [a], Luca Mazzei [a], Federico Galvanin [a,*]

[a] Department of Chemical Engineering, University College London, Torrington Place, London, WC1E 7JE, United Kingdom
[b] Perceptive Engineering, Applied Materials, Vanguard House, Cheshire, WA4 4AB, United Kingdom
[c] Department of Pharmaceutics and Pharmaceutical Technology, Faculty of Pharmaceutical Sciences, The Hashemite University, Zarqa, 13115, Jordan
[d] Faculty of Science, University of Bath, Claverton Down, Bath, BA2 7AY, United Kingdom

ARTICLE INFO

Keywords:
Machine learning
Drug Active Pharmaceutical Ingredient (API)
Continuous flow
Automation

ABSTRACT

Recently, machine learning (ML) models are increasingly being used in process analytical technology (PAT) frameworks for pharmaceutical manufacturing. Yet, the applications of ML-integrated PAT frameworks are limited by big data requirements. This work introduces a computational framework to develop data-efficient ML models to guide drug particle synthesis in an automated continuous flow precipitation platform. The framework incorporates classification algorithms to identify feasible (fouling-free) operating regions of the precipitation platform, a multiple-output Gaussian process (GP) regression model to relate key process parameters to the drug particle size, and active learning to optimally generate new data for training and validation of the GP model. The usefulness of the proposed framework is demonstrated on the synthesis of ibuprofen microparticles in an automated flow precipitation platform. We envision that properly trained GP models developed using the proposed framework can be employed to fine tune the drug particle size, targeting desired particle bioavailability and processability.

1. Introduction

Poor bioavailability of solid formulations of pharmaceutical drugs caused by low aqueous solubility and dissolution rate of Active Pharmaceutical Ingredients (APIs) is a persistent problem within the pharmaceutical industry. Among several methods to address this problem, nanoparticle or microparticle API formulations are an attractive solution (Mohammadi et al., 2023). Preparing drugs as nano or microsuspensions markedly increases their specific surface area, speeding up drug dissolution and improving their bioavailability. However, making drugs as sub-micron particles is challenging. It requires significant fine-tuning to control particle size, stop particles from clumping together, and prevent the formation of unwanted structures. These challenges have impeded widespread adoption of this technology within the pharmaceutical sector (Jia et al., 2022).

Continuous flow antisolvent precipitation (Sinha et al., 2013) is an energy-efficient and scalable approach for the continuous synthesis of drug particles at the nano or micro scale (Jia et al., 2022). From a process standpoint, the main engineering obstacle in producing drug particles continuously is to manage suspensions in flow effectively. Conversely, when considering product quality, the primary engineering challenge in manufacturing drug particles in the sub-micron range is achieving precise control over the size distribution of the drug particles (Benyahia et al., 2012; Lakerveld et al., 2015). This control is vital for ensuring efficient drug delivery. Moreover, the size distribution plays a critical role in various subsequent pharmaceutical procedures, such as filtration, washing, drying, milling, mixing, and tableting, underscoring its importance for overall process efficacy and drug product quality (Manee et al., 2022). An additional significant hurdle in synthesizing drug particles in flow is establishing efficient online particle size distribu-

* Corresponding author.
E-mail addresses: a.pankajakshan.16@ucl.ac.uk (A. Pankajakshan), sayan.pal@ucl.ac.uk (S. Pal), n.snead@ucl.ac.uk (N. Snead), Juan_Almeida@amat.com (J. Almeida), m.besenhard@ucl.ac.uk (M.O. Besenhard), sh.abukhamees@hu.edu.jo (S. Abukhamees), dqmc21@bath.ac.uk (D.Q.M. Craig), a.gavriilidis@ucl.ac.uk (A. Gavriilidis), l.mazzei@ucl.ac.uk (L. Mazzei), f.galvanin@ucl.ac.uk (F. Galvanin).

https://doi.org/10.1016/j.ces.2024.120780
Received 16 June 2024; Received in revised form 16 September 2024; Accepted 28 September 2024
Available online 11 October 2024

tion (PSD) analysis methods, typically conducted offline (Talicska et al., 2022).

To improve the API particle development, manufacturing, and quality assurance by overcoming the aforementioned challenges, pharmaceutical industries are relying on the application of Process Analytical Technology (PAT), a combination of online data aquisition, automation and chemometrics tools (U.S. Food and Drug Administration, 2004) to continuous flow technology (Hinz, 2006; Mascia et al., 2013; Gutmann et al., 2015; Talicska et al., 2022). As a result, PAT combined with online feedback control has emerged as a promising approach for manufacturing drug particles in the submicron range, allowing for precise control over the PSD (Wu et al., 2007). This, along with the advent of smart manufacturing (also referred to as industry 4.0), in the pharmaceutical industry has resulted in the development of automated precipitation platforms for API particle development and manufacturing (Zhou et al., 2013; Zhang et al., 2018). Such platforms can generate an increased amount of real time quality data. Naturally, Machine learning (ML) models are increasingly being used in PAT frameworks for pattern recognition and information extraction from data. Moreover, autonomous decision-making using ML methods does not require expert knowledge to build the process models. Therefore, less human intervention is required to achieve ML-informed autonomous decision-making in PAT frameworks. Consequently, there is a rising interest in incorporating ML models within PAT frameworks for pharmaceutical manufacturing (Xiouras et al., 2022).

Notable applications of ML models within PAT frameworks for precipitation encompass inferring particle size and three-dimensional shape from images, identifying parameters of kinetic models, and formulating hybrid predictive models for key thermodynamic properties (Boobier et al., 2020; Wyttenbach et al., 2020). ML-based statistical models have proved to be capable of accurately depicting solution and solid-state properties in real-time during precipitation (Salami et al., 2021). Input-output ML models have been utilized to relate critical quality attributes of particulate products to adjustable process parameters, in order to simulate and control the complex nonlinear dynamics of precipitation (Nagy et al., 2013; Griffin et al., 2016; Liu et al., 2020; Fiordalis and Georgakis, 2013). Such models were also employed as predictive tools for control applications (Rohani et al., 1999a,b; Garg and Rathore, 2021; Öner et al., 2020; Manee et al., 2022). Recently, reinforcement learning methods have also proved to be efficient for optimal control of crystallization processes (Meng et al., 2023). ML models were also shown to be effective in predicting supersaturation during the antisolvent precipitation of L-histidine (Coliaie et al., 2022). In the realm of polymorphism and particle structure prediction (Woodley and Catlow, 2008; Price, 2014), ML techniques have demonstrated their ability to expedite the discovery of novel crystalline materials and structures (Schmidt et al., 2019). Lastly, in high-throughput precipitation experiments (Morissette et al., 2004) for developing complex biomolecules like proteins and oligonucleotides, ML helps to find quickly promising operating conditions for precipitation. This makes the automated development of new treatments (Roberts et al., 2020) easier. A holistic review of ML applications in precipitation is available in Xiouras et al. (2022).

Despite notable advancements brought by integrating ML methods into PAT frameworks for API precipitation, several significant challenges persist. In many studies, the development of ML models, including training and refinement, was conducted offline rather than in an online sequential manner. This approach limits the utility of ML models and the overall potential of PAT frameworks for achieving autonomous operations. Furthermore, numerous previous works have relied on data-hungry ML algorithms such as Artificial Neural Networks (ANNs), Deep Neural Networks (DNNs), Convolutional Neural Networks (CNNs), and Recurrent Neural Networks (RNNs) (Vamathevan et al., 2019; Nagy et al., 2022). These models demand substantial amounts of data for effective training, rendering training and refinement challenging and expensive. Data requirement is a challenge in PAT frameworks operating solely on actual experimental data. In this context, the use of pre-trained ML models can be also risky if the actual process data has correlations different to the data used for training. Additionally, the majority of previous studies aimed at developing predictive ML models for precipitation processes have largely overlooked the uncertainty in the model predictions. A related important issue is the uncertainty linked to inference, which if critical results in poor decisions (Lele, 2020). Gaussian processes (GPs) offer a promising solution to address both of these challenges. GPs are data-efficient ML algorithms capable of learning effectively from smaller datasets. GPs being probabilistic mappings also return prediction uncertainties which is key to assess model reliability. Additionally, GP being a non-parametric method does not require the definition of the model architecture, unlike ANNs. GPs are also efficient at overcoming over-fitting issues (Rasmussen and Williams, 2006). Any prior domain knowledge can be also introduced into GPs to improve model predictions. Despite these advantages, GPs also suffer from limitations such as substantial computational cost and storage cost and difficulty to model structured data. Variants of GP models are proposed in literature to overcome these limitations (Jakkala, 2021). More details about the pros and cons of GPs are available in Jakkala (2021). Due to the advantages stated, GPs are well-suited for integration into automated precipitation platforms, to streamline the development of API particles.

This study introduces MLAPI, a ML-driven computational framework designed to assist drug particle synthesis in continuous flow precipitation platforms, with the ultimate goal of autonomously synthesising drug particles in desired size ranges. This paper focuses on the development of the MLAPI framework. In the proposed framework, data-efficient ML classification models play a pivotal role, aiding in the identification of the feasible (that is, fouling- and clogging-free) operating space for the precipitation platform. This is particularly crucial in scenarios where limited prior information is available, as it is often the case in the early stages of API particle development. Once the feasible operating space is determined, the next step in the framework involves designing experiments within this space to generate data required to train and calibrate GP regression models that relate process parameters to moments of the PSD. In the framework, this is done first by Design of Experiments (DoE) methods and then using active learning (AL) methods. In this way, properly calibrated GP regression models, capable of providing reliable particle size predictions, can be developed using real-time data. Such models will be quite useful as decision support tools in developing self-optimized precipitation platforms for autonomous API particle development. The application of the framework is demonstrated on antisolvent precipitation of ibuprofen in an automated continuous flow precipitation platform.

2. Materials and methods

2.1. The experimental platform

The experimental studies involved in this work were carried out in an automated continuous flow precipitation platform, designed to produce micro or nano-sized drug particles (Pal et al., 2024). These studies were aimed to investigate the impact of critical process parameters (referred to as process inputs henceforth) on particle quality attributes through the particle size analysis (referred to as process outputs henceforth). The process inputs are antisolvent flow rate, antisolvent/solvent volumetric flow rate ratio and additive concentration, while the measured process outputs are moments of a volume weighted PSD: $\bar{D}_V$, $D_V(10)$, $D_V(50)$, and $D_V(90)$. Online laser diffraction (LD) was used as the analytical tool to measure the PSDs. The LD determines PSDs by measuring the angular variation in intensity of light scattered as a laser beam traverses a dispersed sample of particles. Because the light intensity recorded by the detector is proportional to the volume of particles, LD yields volume-weighted PSDs. Consequently, the PSD obtained through LD reflects the volume of particles present within distinct size classes. Hence, $\bar{D}_V$ represents the volume-weighted mean value of the particle size, while $D_V(10)$

signifies the size point in the volume-weighted PSD where 10% of the total volume of material in the sample is encompassed. For example, if $D_V(10)$ is 844 nm, this indicates that 10% of the sample consists of particles with a size of 844 nm or smaller. Similarly, $D_V(50)$ represents the size point below which 50% of the material is contained, serving as the median value of the PSD. Additionally, $D_V(90)$ denotes the size below which 90% of the material is contained.

The key technology within the experimental platform involves sending real-time input data (values of process inputs) and output data (values of process outputs) to a cloud resource. Accessing these data in real-time allowed developing and validating ML models rapidly. The platform ensures automated processes for data acquisition, management, and communication, reducing the need for human involvement. Additionally, the platform was designed to run periodic replication of experiments to assess the reliability and consistency of the measurements. This periodic check is essential in automated platforms to ensure their robust operation. Fig. 1 (a) visually depicts the experimental setup, while Fig. 1 (b) presents a schematic overview of the experimental configuration.

2.1.1. Process overview

In the experimental platform, a confined impinging jet reactor (CIJR) was used to carry out the antisolvent precipitation process. As shown in Fig. 1 (b), three pumps, labelled as Pump 1, 2, and 3, were employed to deliver the API, the antisolvent and the additive solutions to the CIJR, respectively. The resulting suspension with precipitates from the CIJR was transferred to the collection vessel SV1. Once SV1 reached its capacity (detected by the level sensor LS1), Pump 4 was activated to transfer the suspension to a waste bottle, while maintaining a constant level in SV1. Next, Pump 7 was utilized to fill the vessel SV2 with a saturated solution of API until the level sensor LS2 indicated that it was full. Valve V1 was then employed to transfer the precipitation product (API suspension) from SV1 to SV2 where it was diluted and stirred with a magnetic stirrer to ensure a uniform mixture. The diluted sample was then pumped through the flow cell of the LD (Beckman Coulter, LS320), using Pump 5. The flow sensor FS2 at the outlet of the flow cell detected when the flow cell was full, triggering the start of LD measurements. On completion of the PSD measurement of the diluted sample followed by emptying the flow cell and SV2 (detected by the flow sensor FS2), Pump 5 was stopped. Any remaining solution in SV2 was removed using Pump 10 from the bottom of the vessel. To clean the SV2, ethanol water mixture (8:1) was used as cleaning liquid. Pump 6 filled SV2 with the cleaning solution; this was then circulated through the LD flow cell using Pump 5. After the cleaning cycle was completed, the next LD measurement cycle could begin.

The inlet and outlet connections of the LD flow cell were modified to fit standard Swagelok fittings (1/8 inch). To connect the flow cell to the sample vessel SV2, the standard 10 mm diameter PTFE tubing was replaced with 1.58 mm inner diameter PTFE tubing (VICI Jour). This change significantly reduced the volume of diluted sample needed for each laser diffraction measurement cycle, thereby speeding up the measurements. To precipitate ibuprofen, ethanol served as solvent, while deionized water (15 m-Ω) acted as antisolvent. A solution of ibuprofen® (BASF) (1 wt% of equilibrium solubility) was prepared by dissolving 3.95 g of ibuprofen in 100 ml of ethanol. Soluplus® (BASF), a polymeric solubilizer, was used as the additive to inhibit particle growth and stabilize the particles. More details about the crystallization process and particle size analysis are reported in a previous study (Pal et al., 2024).

2.1.2. Process automation and data communication

To automate the process, LabVIEW software (NI, 2024) was employed. LabVIEW (2023 Q3 version) is a user-friendly system-design platform with a visual programming language (Jeffrey and Jim, 2006). In this work, LabVIEW programs called virtual instruments (VIs) were created to interface and automatically control instruments such as pumps, valves, magnetic stirrers, flow sensors, and level sensors. A secure and efficient cloud-based data communication protocol was established for data exchange between different devices in the experimental platform. The architecture of this protocol is illustrated in Fig. 2. This protocol was based on Open Platform Communications (OPC) standards. OPC is a set of standards and specifications for industrial communication (Foundation, 2024). The most widely used specification within OPC is the Open Platform Communications Data Access (OPC DA), which facilitates reading and writing of real-time data.

In this work, to establish the OPC DA data communication protocol, essential data were defined as network-published shared variables (NPSVs) within the LabVIEW project library. When a LabVIEW VI with NPSVs is executed, LabVIEW automatically launches its OPC server facility called Shared Variable Engine (SVE) and deploys the project library containing the NPSVs on the server. The SVE (Mahmoud et al., 2015) creates OPC tags for the NPSVs. These tags act as addresses for the variables, allowing their communication with any OPC DA client connected to the server. A pseudo-algorithm listing the main steps involved in developing LabVIEW OPC communication protocol is provided in Algorithm 1.

In this work, we utilised PharmaMV, an advanced process control software from Perceptive Engineering (Engineering, 2024), to communicate with LabVIEW via an OPC DA protocol. The PharmaMV software has an OPC DA client for communication, which is an in-built tool of the software. Besides providing OPC DA client for communication, the PharmaMV software was utilized to automate processes through the implementation of event flow algorithms, ensuring sequential execution of the process overview described in Section 2.1.1. Additionally, PharmaMV was used to create the Graphical User Interface (GUI) for the precipitation platform.

Algorithm 1: OPC data communication protocol.

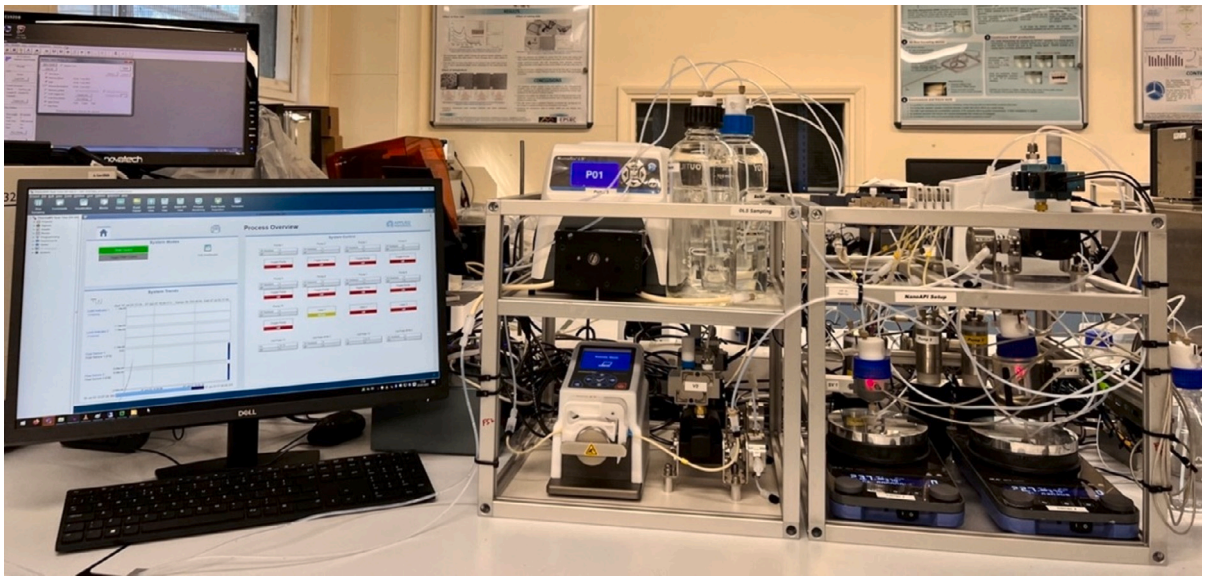
Requirements: LabVIEW software, OPC client (for example, OPC Quick Client from KEPServerEX)
Result: OPC communication between LabVIEW server and the OPC client
Step 1: Create LabVIEW VIs within a LabVIEW project file
```
/* When creating the LabVIEW VIs, define the data to
   be transferred across the network as NPSVs      */
```
Step 2: Run LabVIEW VIs
```
/* This launches the LabVIEW OPC server and deploys
   NPSVs on the server                             */
```
Step 3: Launch OPC client and connect it to the LabVIEW OPC server
Step 4: Write to or read from the NPSVs, either from the client side or from the LabVIEW side
```
/* This enables a secure and fast two way
   communication between the LabVIEW OPC server and
   any clients connected to the server             */
```

2.2. The computational framework

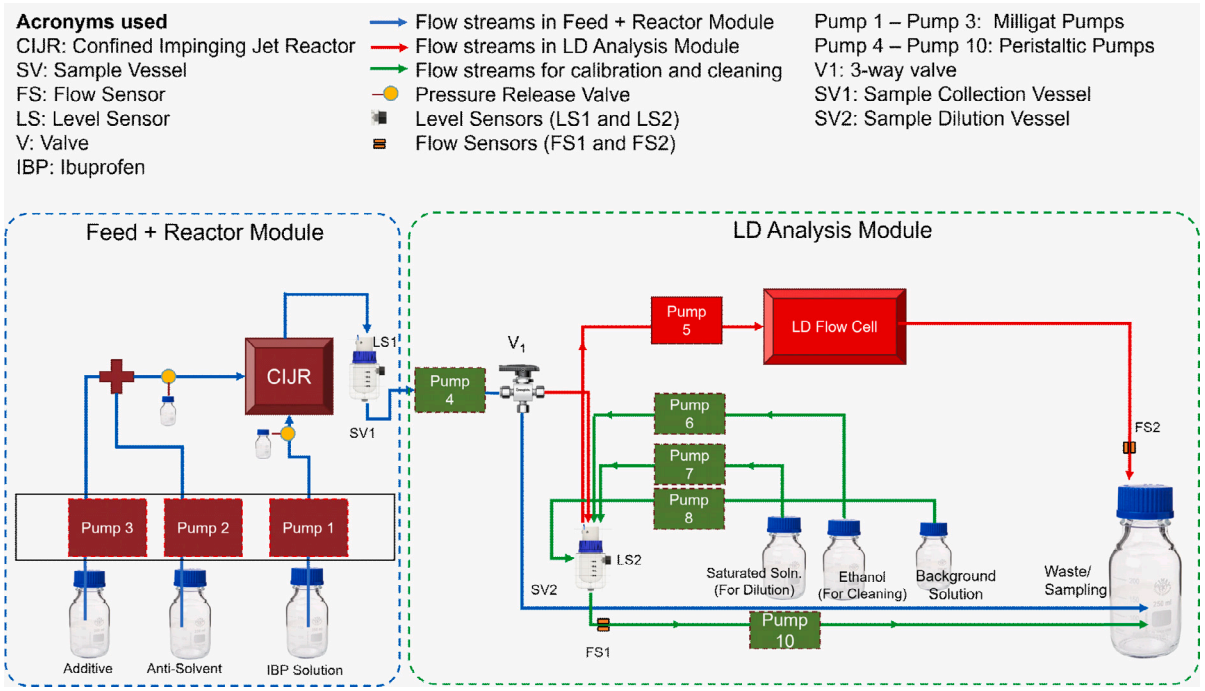
The ML-driven computational framework was designed with the ultimate goal of establishing a self-optimizing continuous flow precipitation platform for API particle development. The framework incorporates four modules: 1) a DoE module, implementing factorial design methods; 2) a classification module, to define the feasible region of operating conditions; 3) a model building module, implementing development, training and posterior analysis of ML models; 4) an optimal experimental design module based on AL methods to generate new training datasets. The flowchart of the computational framework is depicted in Fig. 3.

2.2.1. Generation of initial labelled dataset using DoE methods

DoE methods are statistical approaches used to systematically explore the cause-effect factors in a process. When experiments aim to support deductive learning, such as verifying an existing theory, factorial DoE methods are often a preferred starting point. This preference



(a)



(b)

Fig. 1. The experimental platform. - (a) a photograph showing the major components of the experimental setup, (b) schematic diagram of the experimental setup. The background solution is a mixture of water and ethanol in the same ratio as the antisolvent/solvent volumetric flow rate ratio.

is not only due to their various advantages, including greater efficiency and comprehensiveness (Fisher, 1935), but also because, in deductive learning problems, the nonlinear relationship between process inputs and outputs is already framed within the postulated theory or model. Hence, factorial DoE methods can provide valuable information to validate the theory with minimum experimental trials.

The situation differs in the case of ML models, which are data-driven models developed through an inductive learning procedure. The relationship between cause-effect factors that influence a system behaviour can be modelled using supervised ML methods. Supervised learning involves training a model on labelled data, where both inputs and corresponding outputs are known. This enables the model to predict outputs for new, unseen inputs. When the predicted outputs are discrete (for example, categorical variables), the task is called classification. When the outputs are continuous variables, the task is known as regression. On the other hand, unsupervised learning methods are used to identify hidden patterns within the input data, which are then used to group or cluster the data into meaningful categories (Vamathevan et al., 2019). Often, the input-output data supplied to supervised ML models are a subset of a transformed form of the actual process inputs and outputs. These transformed data are also known as input features and output labels or collectively as labelled dataset.

The criteria for generating initial labelled datasets for training supervised ML models should not only be based on efficiency and comprehensiveness in sampling but should also be based on the complexity or correlation between input features and output labels (Narayanan et al., 2021). Unfortunately, this complexity is not known in the absence of any historical dataset; in such situations, space filling DoE methods are appropriate for designing the initial experiments. However, in the present study, among the three process inputs: 1) antisolvent flow rate (ml/min), 2) antisolvent/solvent flow rate ratio (-) and 3) additive concentration (wt.%), the second input was allowed to have only discrete values while

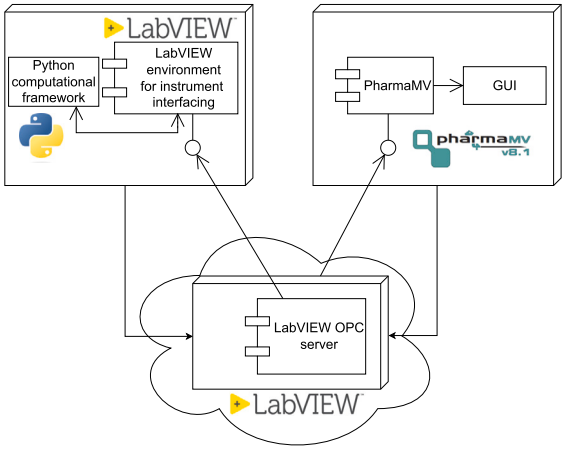


Fig. 2. The architecture of the cloud-based data communication system of the automated precipitation platform.

the other inputs were allowed to vary continuously. This non-ideal situation rendered space filling design methods, such as Latin Hypercube Sampling (LHS), unusable, because these methods treat the input variables as continuous. But, most importantly, the interactions among the three process inputs were expected to affect the precipitation process significantly, and this can be best captured by factorial DoE methods. For these reasons, factorial DoE was chosen as the DoE method to generate the initial dataset. The factorial DoE methods were implemented utilizing the Python experimental design package pyDOE2 (Sjoegren, 2023).

2.2.2. Search of feasible operating space, formulated as a multi-class classification problem

Recognizing fouling as a significant challenge in the development of continuous flow precipitation processes (Lapkin et al., 2017; Besenhard et al., 2023), it becomes crucial to pinpoint a feasible operating space for the precipitation process where fouling does not happen. In this study, we intended to determine the boundaries separating the operating space into feasible, partially feasible and infeasible regions. We formulated this task as a multi-class (since more than two possible outcomes are present) classification problem.

The classification dataset is formed of N data points, with a $N \times N_u$ matrix $\mathbf{U}$ of process inputs and a $N \times 1$ vector $\mathbf{c} = (c_1, c_2, \ldots, c_N)^{\mathrm{T}}$ of labels. Here, N_u represents the number of process inputs, N denotes the number of experiments, and $c_i \in C$ represents the class label for the i-th data point or experiment. Each row of the matrix $\mathbf{U}$ features a N_u dimensional vector $\mathbf{u}$ of inputs. In the context of the antisolvent precipitation of ibuprofen, the i-th row of the matrix $\mathbf{U}$ represents the conditions of the i-th experiment, encompassing antisolvent flow rate ($u1$), antisolvent/solvent flow rate ratio ($u2$) and additive concentration ($u3$). The labels c_i represent the possible classes: 0, 1 and 2, i.e., $C = \{0, 1, 2\}$. The class labels - 0, 1, 2 - signify respectively infeasible experiments (leading to system clogging without allowing PSD measurements), partially feasible experiments (some deposition in the flow channels with PSD measurements possible) and fully feasible experiments (no clogging, successful PSD measurements). A matrix notation of the classification dataset is provided in Eq. (1) and an illustration of this dataset is provided in Fig. 4.

$$\underbrace{\begin{bmatrix} u1_1, u2_1, u3_1 \\ \vdots \\ u1_i, u2_i, u3_i \\ \vdots \\ u1_N, u2_N, u3_N \end{bmatrix}}_{\mathbf{U}}, \quad \underbrace{\begin{bmatrix} c_1 \\ \vdots \\ c_i \\ \vdots \\ c_N \end{bmatrix}}_{\mathbf{c}} \tag{1}$$

A classification algorithm is a decision function $f_c(\mathbf{u}, \theta_c)$ parameterized by a set of parameters θ_c together with a loss function $L_c(f_c(\mathbf{u}, \theta_c), \mathbf{U}, \mathbf{c})$. The loss function measures the discrepancy between the predicted outputs $f_c(\mathbf{u}, \theta_c) \mid \mathbf{U}$ and the true class labels $\mathbf{c}$. The training of the classifier model consists of a hyper-parameter optimization problem that aims to minimize the loss function over the training dataset.

$$\arg\min_{\theta_c} L_c(f_c(\mathbf{u}, \theta_c), \mathbf{U}, \mathbf{c}) \tag{2}$$

We utilized two fundamental approaches to define f_c and solve the classification problem: i) discriminative, and ii) generative approaches (Rasmussen and Williams, 2006; Rigollet, 2015). Discriminative approaches provide direct predictions of class labels, while generative approaches involve probabilistic classification, where model predictions are expressed as class probabilities.

In the discriminative approach, a Support Vector Machine (SVM) model (Smola and Schölkopf, 2004; Bishop and Nasrabadi, 2006) was implemented, and in the generative approach, a multi-class GP model (Rasmussen and Williams, 2006) was employed. In the SVM classifier, the decision function for each class m is defined as

$$f^{\mathrm{svm}}_{c_m}(\mathbf{u}) = \sum_{i=1}^{N_{\mathrm{SV}}} \alpha_{im} c_i k^{\mathrm{svm}}_c(\mathbf{u}, \mathbf{u}_i) + w_m \tag{3}$$

where $\mathbf{u}_i$ are the support vectors (points lying closest to the decision boundary), N_{SV} is the number of support vectors, α_{im} are the Lagrange multipliers corresponding to support vector $\mathbf{u}_i$ for class m, c_i is the class label indicator for support vector $\mathbf{u}_i$, $k^{\mathrm{svm}}_c(\mathbf{u}, \mathbf{u}_i)$ is the kernel function, (Radial basis function (RBF) kernel function defined as $k^{\mathrm{svm}}_c(\mathbf{u}, \mathbf{u}_i) = \exp(-\gamma ||\mathbf{u} - \mathbf{u}_i||^2)$) and w_m is the bias term for class m. The hinge loss function was used as the loss function for training the SVM model (Chang and Lin, 2011; Bishop and Nasrabadi, 2006).

In the generative approach, GP models (one for each class) were used as latent functions. A softmax transformation (Bishop and Nasrabadi, 2006) was applied on the predictions of the GP models to compute the probability of each class. The decision function (Galy-Fajou et al., 2020) for class m can be defined as:

$$f^{\mathrm{gp}}_{c_m}(\mathbf{u}) = \frac{\exp(g_m(\mathbf{u}))}{\sum_{j=1}^{M} \exp(g_j(\mathbf{u}))} \tag{4}$$

where $g_m(\mathbf{u})$ denotes the latent function value for class m, which is modelled as a GP, and M is the number of classes, i.e., $M = |C|$. This means $g_m \sim \mathcal{N}(0, k^{\mathrm{gp}}_c)$, where k^{gp}_c is the corresponding kernel function. Due to the lack of prior information on the mean of the GP latent functions, a zero mean prior was assumed. This provides an unbiased starting point and allows the data to dictate the learning process. The Matérn 5/2 kernel was used as the kernel function. This kernel was chosen due to its flexibility in controlling the smoothness of the learned function. Benchmarking on several case studies, it is proven that Matérn kernels with standard GPs produce predictions closely aligned with the ground truth (Xu and Zhe, 2024). Due to the non-Gaussian (categorical) likelihood (Murphy, 2012) in the GP classification problem, exact inference is intractable. So, an approximate inference is carried out using variational inference (Blei et al., 2017). Evidence lower bound (ELBO) (Kingma and Welling, 2013) is the commonly used variational inference loss function, which was also used as the loss function in the multiclass GP classification.

In both classification approaches, confusion matrix and test classification accuracy computed over a number of random train-test datasets were used as metrics for evaluating the quality of the multi-class classifiers (Grandini et al., 2020). The multi-class SVM model was implemented using the 'SVC' class in the scikit-learn Python library (Pedregosa et al., 2011), while the GP classification model was developed in Pyro (Bingham et al., 2018).

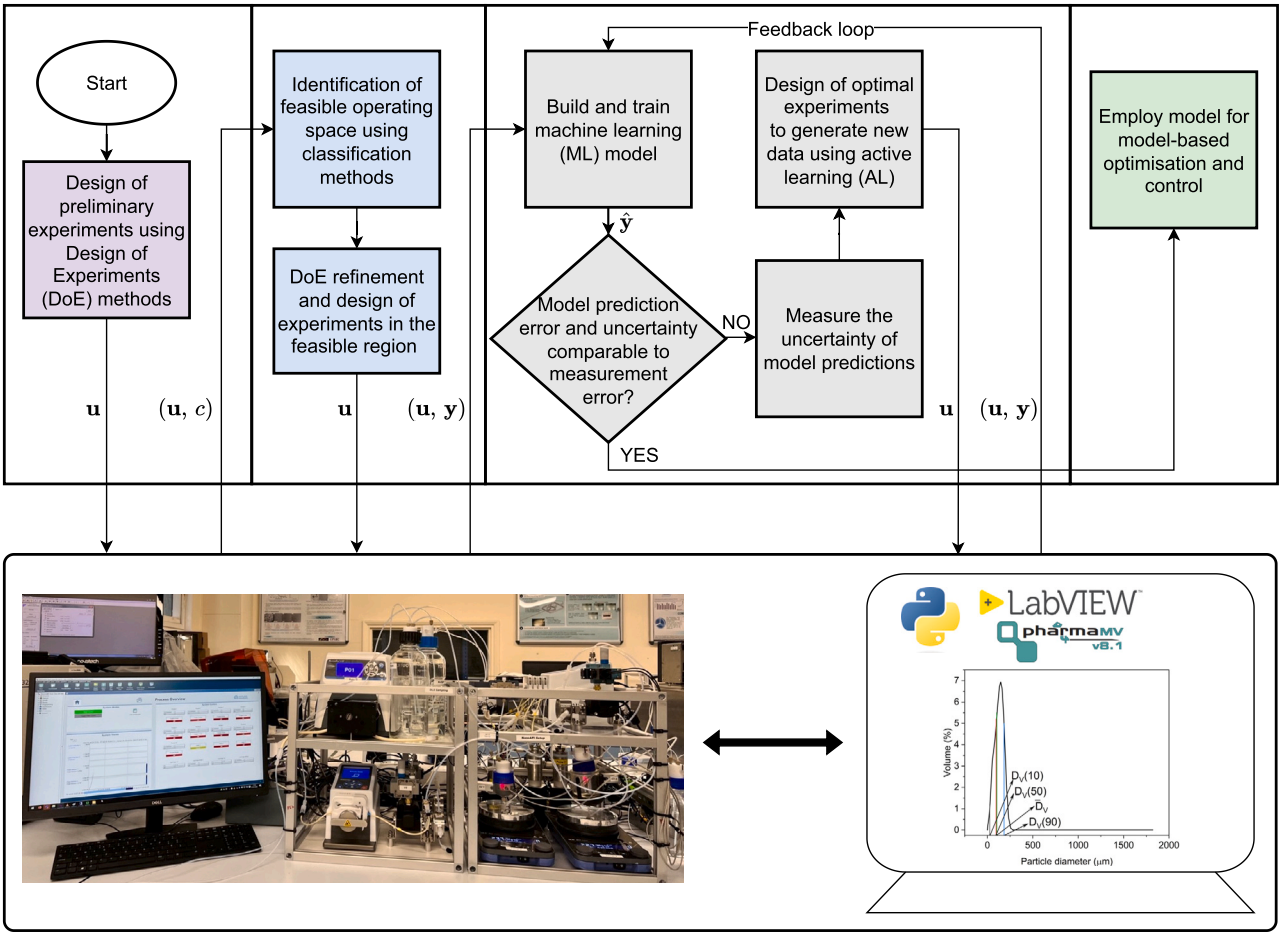


Fig. 3. Flowchart of the computational framework proposed in this work.

Experiment	Inputs			Outputs
	Antisolvent flow rate (ml/min)	Antisolvent to solvent flow ratio $(-)$	Additive conc. $(wt.\%)$	Class label 0 1 2
1	$u1_1$	$u2_1$	$u3_1$	✓
⋮	⋮	⋮	⋮	
i	$u1_i$	$u2_i$	$u3_i$	✓
⋮	⋮	⋮	⋮	
N	$u1_N$	$u2_N$	$u3_N$	✓

Fig. 4. An illustration of the classification dataset.

2.2.3. *Modelling of a precipitation process via GP regression*

The precipitation process was modelled using a multiple-output GP (a single GP model that gives predictions of multiple response variables by considering the cross-covariance between the response variables in the kernel function) regression model (Bonilla et al., 2007). In simple terms, GP regression is a non-parametric method that approximates the unknown function $f(\mathbf{u})$, which maps process inputs $\mathbf{u}$ into outputs $\mathbf{y}$. This approximation is achieved by defining probability distributions over the function values at each input $\mathbf{u}$. The means of these distributions at each input represent the corresponding function values. In GP regression, all these distributions are Gaussian, simplifying the computation for inference and learning (Rasmussen and Williams, 2006).

The regression dataset is formed of N data points, which include a $N \times N_u$ matrix $\mathbf{U}$ of process inputs and a $N \times N_y$ matrix $\mathbf{Y}$ of the measured process outputs. Here, N denotes the number of experiments, N_u denotes the number of process inputs and N_y denotes the number of measured process outputs or response variables. As in the case of classification, each row of the matrix $\mathbf{U}$ features a N_u dimensional vector $\mathbf{u}$ of inputs, while each row of the matrix $\mathbf{Y}$ features a N_y dimensional vector $\mathbf{y}$ of outputs. In the context of ibuprofen precipitation, the i-th row of the matrix $\mathbf{U}$ represents the conditions of the i-th experiment, encompassing antisolvent flow rate, antisolvent/solvent flow rate ratio and additive concentration. Similarly, the i-th row of the matrix $\mathbf{Y}$ represents the process outputs, that is, the PSD-related scalars $\bar{\mathrm{D}}_\mathrm{V}$, $\mathrm{D}_\mathrm{V}(10)$, $\mathrm{D}_\mathrm{V}(50)$, and $\mathrm{D}_\mathrm{V}(90)$, formed in the i-th experiment. A single scalar output variable from an experiment is denoted as $y(\mathbf{u})$ and the corresponding function value (i.e., the corresponding mean of the GP) as $f(\mathbf{u})$ or $\hat{y}(\mathbf{u})$.

The outputs in the regression problem are known to be correlated as they are the moments of the same PSD. Therefore, in order to use these correlations to improve predictions, we decided to employ multiple-output GPs. Multiple-output GPs can model these correlations as cross-covariance between the outputs in their kernel function. For any pair of latent functions $\left(f_r, f_s\right)$, the cross-covariance between the model outputs can be defined as

$$\mathrm{cov}\left(f_r(\mathbf{u}), f_s\left(\mathbf{u}'\right)\right) = k\left(\mathbf{u}, \mathbf{u}'\right) \times \mathbf{B}[r,s] \tag{5}$$

In Eq. (5), $\left(\mathbf{u}, \mathbf{u}'\right)$ represents any pair of input points. According to the equation, the covariance of the r-th function at $\mathbf{u}$ and the s-th function at $\mathbf{u}'$ is defined by a kernel applied at $\mathbf{u}$ and $\mathbf{u}'$, times the r,s-th entry of a positive definite matrix $\mathbf{B}$. This cross-covariance kernel function is known as the intrinsic coregionalization model (ICM) kernel (Bonilla et al., 2007). Using the multiple-output GP model with prior mean (assumed as zero, since prior to model training, the outputs are standardised to have zero mean and unit standard deviation) and with the prior ICM kernel, the process outputs can be modelled as

$$\underbrace{\begin{bmatrix} \mathbf{Y}[1,1] \\ \vdots \\ \mathbf{Y}[N,1] \\ \vdots \\ \mathbf{Y}[N,N_y] \\ \vdots \\ \mathbf{Y}[N,N_y] \end{bmatrix}}_{\mathbf{Y}_v} \sim \mathcal{N}\left(\begin{bmatrix} \mathbf{0} \\ \vdots \\ \mathbf{0} \end{bmatrix}, \mathbf{B} \otimes \mathbf{K} + \mathbf{\Sigma} \otimes \mathbf{I} \right) \tag{6}$$

$$\text{where, } \mathbf{\Sigma} = \begin{bmatrix} \sigma^2_{y_1} & \cdots & \mathbf{0} \\ \vdots & \ddots & \vdots \\ \mathbf{0} & \cdots & \sigma^2_{y_{N_y}} \end{bmatrix}$$

In Eq. (6), $\mathbf{Y}_v$ is the $N \cdot N_y$ dimensional vector obtained by vectorization of the matrix $\mathbf{Y}$, $\otimes$ denotes the Kronecker product, $\mathbf{K}$ is an $N \times N$ matrix, which has elements $k\left(\mathbf{u}, \mathbf{u}'\right)$, defined by the covariance function over inputs. As stated earlier, the Matérn 5/2 kernel was used as the covariance function over the inputs. The matrix $\mathbf{\Sigma}$ in Eq. (6) denotes an $N_y \times N_y$ diagonal matrix in which the (j,j)-th element is $\sigma^2_{y_j}$, the measurement error variance for the j-th output variable.

For a new input point, $\mathbf{u}_*$, the mean of the posterior predictive distribution for output j, $p\left(f_j\left(\mathbf{u}_*\right) \mid \mathbf{Y}\right)$, is given by

$$f_j(\mathbf{u}_*) = \hat{y}_j(\mathbf{u}_*) = (\mathbf{b}_j \otimes \mathbf{k}_*)^T \left(\mathbf{B} \otimes \mathbf{K} + \mathbf{\Sigma} \otimes \mathbf{I}\right)^{-1} \mathbf{Y}_v \tag{7}$$

In Eq. (7), $\mathbf{b}_j$ selects the j-th column of $\mathbf{B}$, $\mathbf{k}_*$ is the vector of covariances between the test point $\mathbf{u}_*$ and the training points, and $\mathbf{K}$ is the matrix of covariances between all pairs of training points.

Given a training dataset $\mathbf{Y}_0$, training of the GP model involves learning the parameters θ of $k(\cdot,\cdot)$ and the matrix $\mathbf{B}$ by maximizing the marginal log-likelihood $\log(p(\mathbf{Y}_0 \mid \theta, \mathbf{B}))$. Here, θ denotes the parameter vector of the Matérn 5/2 kernel. As mentioned in Section 2.1, experiments were performed in replicates in the automated platform to examine the reproducibility of the measurements. The standard deviations of the measured process outputs obtained from the repeated experiments were used to estimate the measurement error variances (diagonal elements of the matrix $\mathbf{\Sigma}$ in Eq. (6)). These data were then used to model the measurement error variance separately. In other words, during the GP model training, the measurement error variances (also known as noise parameters) were not treated as hyper-parameters. Repeated measurements are the reliable sources of estimating measurement error variance. Estimating these as hyperparameters during model training has critical implications on the predictions of GP models, particularly in cases of heteroscedastic noise variances (Binois et al., 2019; Ameli and Shadden, 2022; Le et al., 2005). The details of the variance models used to estimate the measurement error variances are provided in the supplementary material.

The class 'MultiTaskGP' with '`train_Yvar`' within the BoTorch Python library (Balandat et al., 2020) was used to implement the multiple-output GP model. During model training, the maximization of the marginal log-likelihood objective function was solved using the stochastic gradient descent (SGD) optimization method (Amari, 1993) implementing the Adam algorithm (Kingma and Ba, 2014). Within the proposed GP regression modelling framework, the process inputs were normalized to form the input features for the regression model. The process outputs were standardised (zero mean and unit variance) to form the labels for the GP regression model.

The validation of the trained GP regression model was carried out by evaluating the distribution of prediction errors or residuals standardised by measurement error standard deviations. The standard residuals e_{ij} can be obtained by

$$e_{ij} = \frac{y_{ij} - \hat{y}_{ij}}{\sigma_{y_{ij}}} \quad i = 1, \ldots, N;\ j = 1, \ldots, N_y \tag{8}$$

where y_{ij} is the measured value for experiment i and output j, $\hat{y}_{ij}$ is the predicted value for experiment i and output j, and $\sigma_{y_{ij}}$ is the standard deviation of measurement error for experiment i and output j. A statistically adequate model is defined by a standard residual distribution where the average is near zero ($\mu_e = 0$), suggesting the discrepancy between predictions and measured values is negligible, and the standard deviation is less than one ($\sigma_e < 1$) (Draper and Smith, 1998; Palmtag et al., 2023), suggesting that the majority of the prediction errors fall within one standard deviation of measurement error. Thus, the parameters of the standard residual distributions can be used to assess both model adequacy and model uncertainty.

2.2.4. Sequential experimental design using active learning

Once the multi-task GP regression model is trained with the initial dataset generated through the DoE method, the subsequent experiments required for further training and model validation were optimally designed using AL methods. As mentioned at the end of Section 1 and in Section 2.1, the objective of the study was developing a predictive regression model that relates process inputs and outputs. Hence, the goal was to increase the knowledge of the model in each new experiment. Thus, unlike directly employing Bayesian optimization, a technique for optimizing expensive-to-evaluate functions, by sampling and optimizing in regions where the expected value of the function is an optimum (Shahriari et al., 2015; Di Fiore et al., 2023), we employed an AL method to sample in the regions of high predictive posterior variance of the model. We selected this approach because querying an instance in the uncertain region yields more valuable information for the model than an expected or already learned point (Shannon, 1948; Seo et al., 2000). This strategy improves the understanding of the model about the data generating process. Application of the AL method to iteratively generate most informative data to train the GP model can be continued until a stopping rule is met. The stopping criterion can be defined either based on the uncertainty measure of the model or based on the experimental budget.

In each iteration of AL, a new experimental condition was designed by maximizing the total predictive posterior variance of the model. The total predictive posterior variance of the model is defined as the sum of the predictive posterior variances of all the outputs, given by the GP model trained on the full available dataset. An important distinction is that, in each iteration of the experiment after the initial DoE, the model training and validation are done using a train-test data split, whereas the experimental design uses a model trained on the entire dataset.

$$\arg\max_{\mathbf{u} \in \mathcal{U}} \psi(p(\mathbf{f}(\mathbf{u}) \mid \mathbf{Y})) \tag{9}$$

In Eq. (9), $p(\mathbf{f}(\mathbf{u}) \mid \mathbf{Y})$ represents the predictive posterior distribution of the GP and $\psi(\cdot)$ represents the total predictive posterior variance of the GP. The sequential experimental design procedure using the AL method was continued until the stopping rule was satisfied. In this work, the experimental budget, i.e., the maximum number of experiments ($N_{\max}$), was used as stopping rule, which was set at 15. The AL method was implemented using the Python packages GPyTorch (Gardner et al., 2018) and BoTorch (Balandat et al., 2020).

3. Results and discussion

3.1. Classification results

The initial set of precipitation experiments involved twenty-seven trials, designed using a three-factor, three-level full factorial DoE method, applied on the original parameter ranges reported in Table 1. Among these experiments, fourteen resulted in clogging of the precipitation process platform, preventing PSD measurements, and were assigned the class label '0'. Six experiments exhibited partial fouling of the precipitation process platform, but produced the PSD measurements. These experiments were assigned the class label '1'. The remaining seven experiments operated smoothly without causing fouling issues and enabled the PSD measurements. These were labelled as '2'. The entire data set is provided as supplementary material.

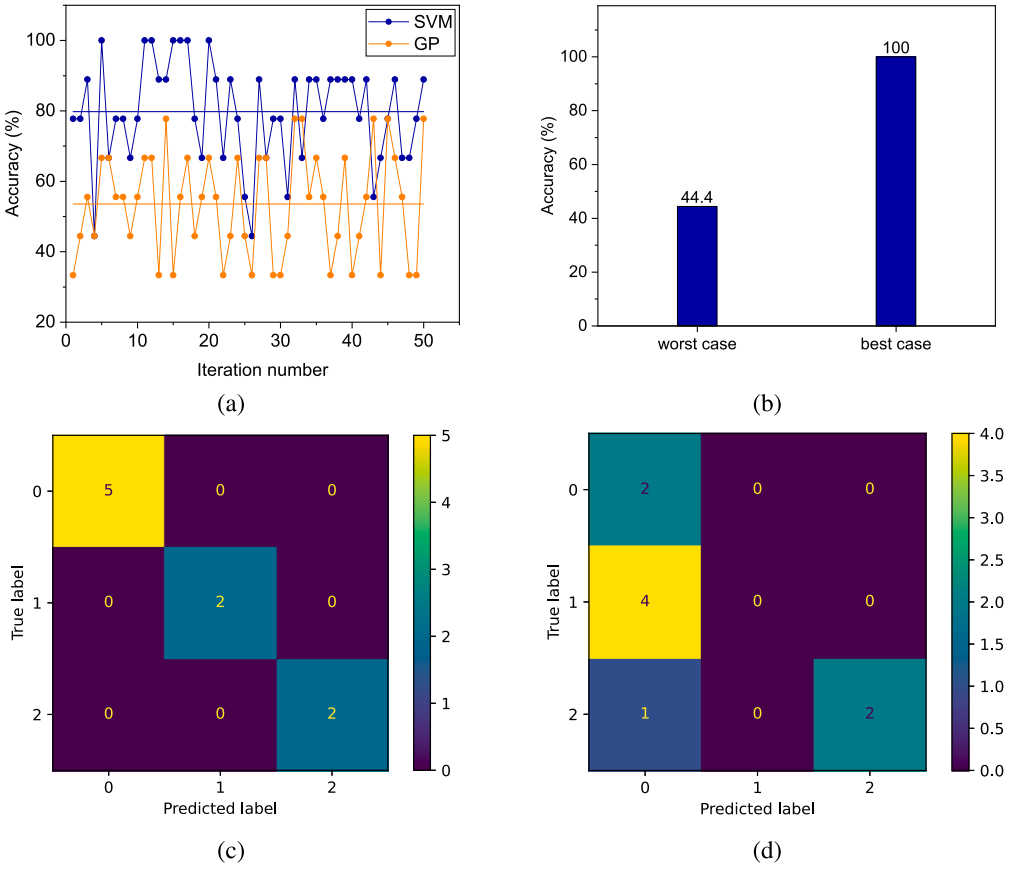


Fig. 5. Validation results of SVM and GP classifier models. Panel (a) shows the test accuracy for 50 random train and test subsets (70 % train size and 30 % test size in all the runs). Panel (b) shows the worst and best test accuracy of the SVM model out of the 50 runs. Panels (c) and (d) respectively show the confusion matrix for the best and worst case of the SVM model.

Table 1
Ranges of inputs.

Inupt	Original bounds		Feasible bounds	
	Lower	Upper	Lower	Upper
Antisolvent flow rate (ml/min)	1	4	2.5	4
Antisolvent / Solvent ratio (-)	1	9	5	9
Additive conc. (wt.%)	0	3	1.5	3

This labelled dataset was randomly split into training (70%) and test (30%) sets. Both SVM and GP classification models were trained on these data, and their performance was evaluated over 50 random train-test splits. This cross validation procedure (validations using different random train-test data) is an effective choice for validation of classifier models under data scarce situations. The SVM model demonstrated an average test accuracy of 80%, while the GP model achieved an average of 55% (Fig. 5a). The accuracy of the SVM model ranged from the worst value of 44.4% to the best value of 100%, as depicted in Fig. 5b. The corresponding confusion matrices indicated that even at the worst case of 44% test accuracy, the SVM model correctly predicted the troublesome class “0”. In the confusion matrices shown in Figs. 5c and 5d, the three classes appear in the same order both in the rows and columns. Therefore, the correctly classified elements align along the main diagonal, from the top-left to the bottom-right, indicating the number of times the predicted labels and actual labels agrees for each class.

Following this analysis, the SVM classification model was chosen as the better classifier for predicting feasibility classes. The SVM model was then trained on the full dataset, and its predictions were used to identify the feasible operating space across different conditions. Contour plots of the surface map of predicted class labels at varying antisolvent flow rates, antisolvent/solvent flow rate ratios, and additive concentrations were utilized to determine the feasible operating space. The contours of surface maps of the SVM model, trained on the full dataset, are illustrated in Figs. 6a - 6g.

These figures show that all three inputs, that is, antisolvent flow rate, antisolvent/solvent flow rate ratio, and additive concentration, affect the feasible operation of the precipitation process. Notably, fouling issues occurred at conditions of low antisolvent flow rate (<2.5 ml/min), low antisolvent/solvent flow rate ratio (<4) and low additive concentration (<1.5 wt.%). These results obtained from classification can be understood using precipitation intuition. At low antisolvent flow rate, the mixing in CIJR can be poor, resulting in undesirable nucleation and growth of particles on the flow channels (a phenomenon known as encrustation) downstream of the reactor, leading to fouling and clogging of the channels (Nandi et al., 2024). Similarly, a low antisolvent/solvent flow rate ratio generates supersaturation slowly, yielding large particles that can readily agglomerate in the absence of good stabilizers or additives. This can be inferred as the reason for fouling at low antisolvent/solvent flow rate ratio and zero to low additive concentrations. In general, high antisolvent flow rate (which provides enhanced mixing) and high antisolvent/solvent flow rate ratio (which generates

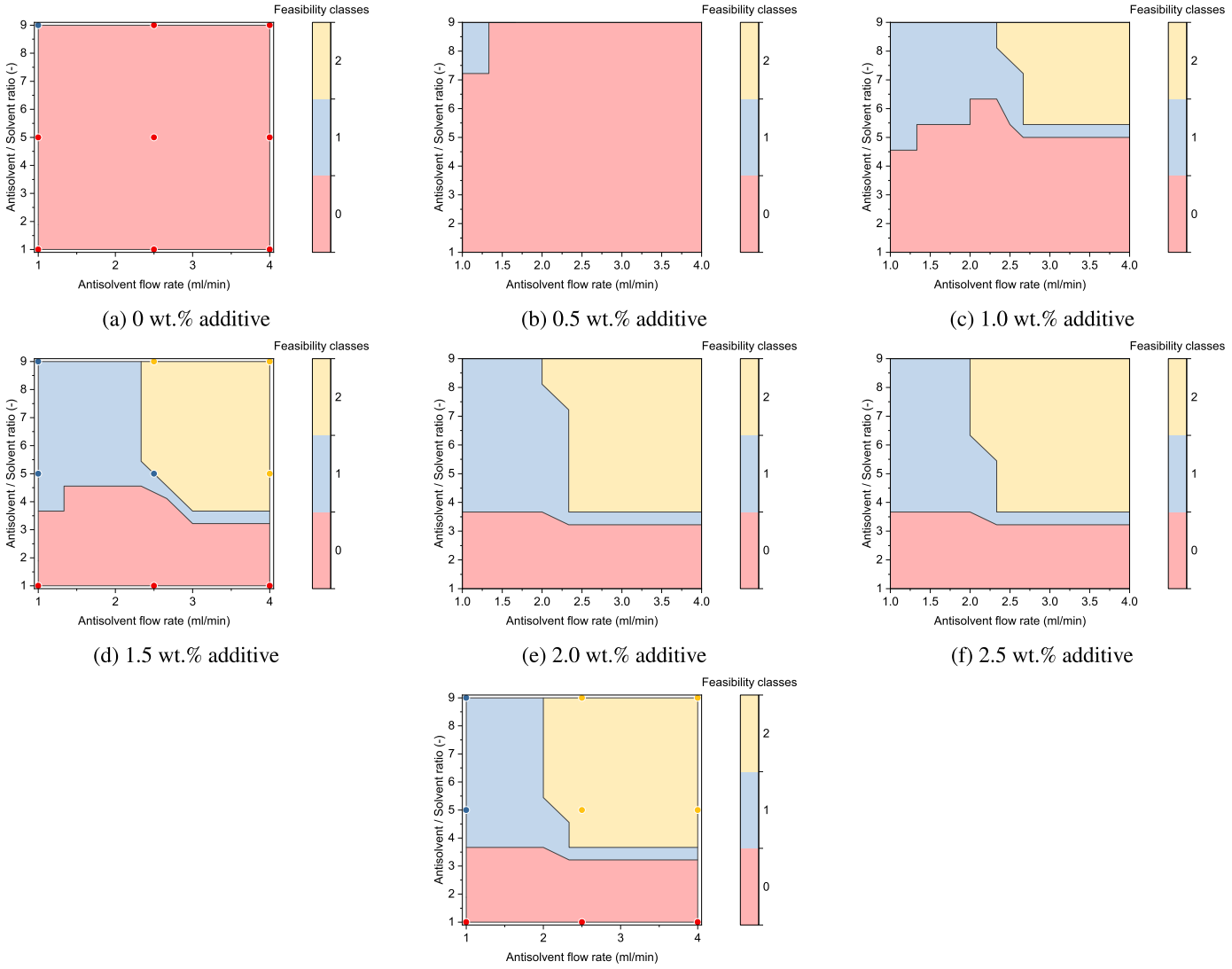


Fig. 6. Predictions of the SVM classifier model trained on the full data set. Feasibility classes are set such that class labels 0, 1, and 2 respectively represent infeasible, partially feasible and feasible experimental conditions. Wherever applicable, the training points are shown in coloured circles, where the colours represent the class labels.

supersaturation quickly, leading to high nucleation rate, resulting in the formation of large number of nuclei) favour the formation of small and fine particles. However, the absence of desirable additives to stabilise the particles formed can readily cause agglomeration and aggregation of particles leading to fouling and clogging of the system (Lonare and Patel, 2013). This can be understood as the reason for fouling in the absence of or at low concentrations of additives.

A similar analysis was conducted for the GP classification model. The predictions of the GP model, trained on the full dataset, closely aligned with those of the SVM model. Both classifier models indicated that all three inputs - antisolvent flow rate, antisolvent/solvent flow rate ratio, and additive concentration - influenced the feasible operations of the precipitation process and there is no specific factor driving the feasibility.

Finally, to define the feasible operating region post-classification, a conservative boundary was set to operate at fully feasible experimental conditions. For this, feasibility constraints defined as simple bounds were set based on the feasibility predictions of the SVM model trained on the full dataset. This region was determined by the following bounds: antisolvent flow rates of 2.5 - 4.0 ml/min, antisolvent/solvent flow rate ratios of 5 - 9, and additive concentrations of 1.5 - 3.0 wt.%, as reported in Table 1.

3.2. GP regression results

The experiments conducted within the feasible operating region identified by the SVM classification model (Section 3.1) supplied the dataset for the regression problem. The regression inputs consisted of the same process parameters: antisolvent flow rate, antisolvent/solvent flow rate ratio, and additive concentration. The regression outputs were based on PSD characteristics, specifically $\bar{D}_V$, $D_V(10)$, $D_V(50)$, and $D_V(90)$, measured by LD. The initial training of the multiple-output GP regression model took place after conducting a series of preliminary experiments within the feasible operating space. In this regard, seventeen experiments were designed using fractional factorial DoE methods. Later, one new experiment was optimally designed in each iteration of the AL method, and that was added to the dataset. The details of these experiments are provided in the supplementary information.

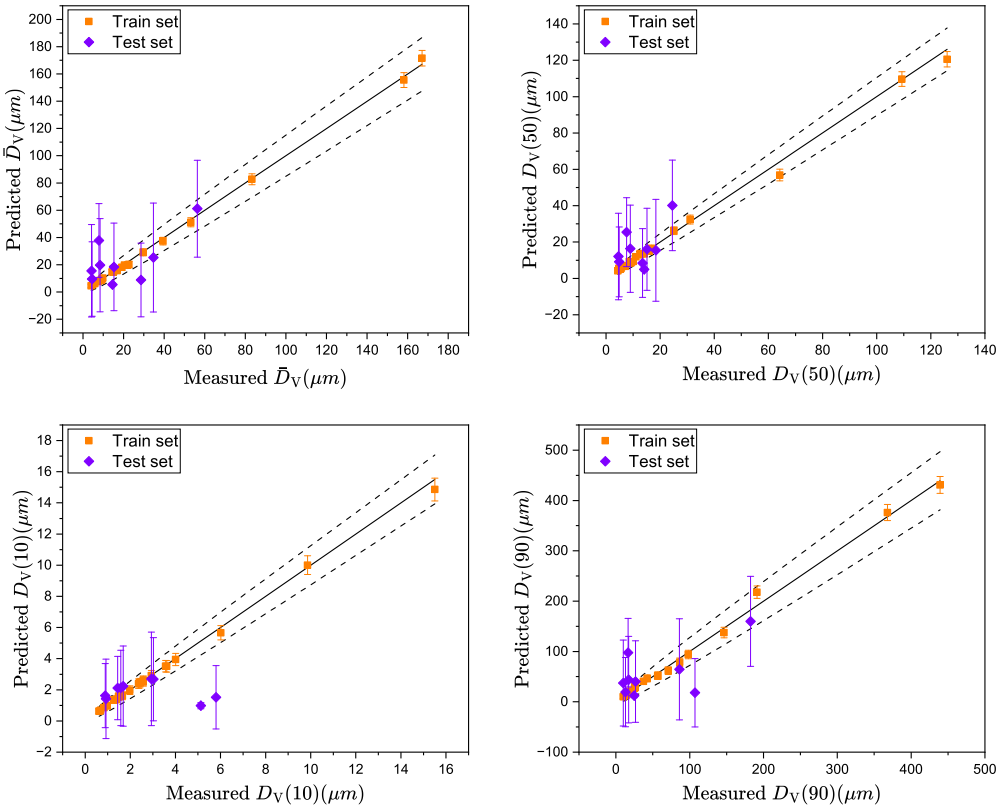


Fig. 7. Validation results of GP regression models based on the parity plots at the end of the full experimental campaign. The dashed line represents the measurement error confidence intervals (± twice the standard deviation of measurement error on measuring the four response variables). The error bars represent the error intervals (± twice the standard deviation of model predictions).

To statistically characterise the measurement errors, three conditions from the initial seventeen experiments were repeated three times each. Also, each new experiment designed using AL method was also repeated thrice. Measurements from these repeated experiments were used to calibrate measurement error variance models, revealing a constant relative variance model for the measurement error. The measurement noise, once obtained separately, was not considered a hyperparameter during the training of the multiple-output GP regression model.

It is noteworthy that replicating experiments posed challenges for random train-test splits, as replicates of the same experiment could end up in both training and test sets during the random selection of train and test data. This can cause an inappropriate validation. To address this, for the repeated experiments, the mean of replicates was used as the data point in the GP regression.

The test data in each train-test data split were employed for GP model validation. But the model trained on the entire historical dataset was iteratively used for selecting the next experimental conditions through the active learning method.

Fig. 7 displays the final parity plots obtained during model validation on the train-test data split after completing all fifteen new experiments designed using the AL method. The plot indicates a strong agreement between the values predicted by the model and the actual values of all four response variables in the test data, affirming the excellent calibration of the GP regression model. Further statistical analysis, illustrated in Fig. 8 by examining histograms of standardized residuals, confirms a good fit of the model, with means close to zero for all four response variables, i.e., $\bar{D}_V$, $D_V(10)$, $D_V(50)$, and $D_V(90)$. While Fig. 8 looks into the statistical validation of the GP regression model at the end of experimentation (after 32 experiments), Fig. 9 provides values of mean of standard residual distributions of the test data for four response variables at each iteration of the AL. A detailed analysis of Fig. 9 is carried out in Section 3.3.

3.3. Sequential design using active learning

After calibrating the GP model using the DoE experiments conducted within the feasible region, 15 new experiments were designed and executed iteratively using the AL method. These fifteen experiments were also repeated three times, and the repeated data were used to update the measurement error variance model in each iteration. The effectiveness of the implemented AL method can be assessed by examining the evolution of the contours representing the total predictive posterior variance of the model, as shown in Fig. 10. The first row of the figure illustrates the total predictive posterior variance of the model after 17 experiments, before initiating the AL procedure. The corresponding prediction quality of the model posterior (in terms of means of the standard residual distributions) on the then test dataset is illustrated in Fig. 9 (data points at experiment 17). The second row of Fig. 10 shows the predictive posterior model variance after 25 experiments, approximately midway through the AL process, while the last row presents the final predictive posterior variance of the model after 32 experiments. The corresponding model prediction performances on the test dataset at experiment 25 and 32 are shown in Fig. 9 (data points at experiment 25 and 32, respectively).

As shown in Fig. 10, the variance of the model initially increased from experiment 17 to 25 due to the exploration process, and subsequently started decreasing. The decreasing trend was observed from experiment 25 onward and resulted in a final total predictive posterior variance of the model smaller than the one computed after the 25th experiment, but larger than the one computed after the 17th experiment. This observation along with Fig. 9 suggests that throughout the AL process, the understanding of the GP model regarding the underlying

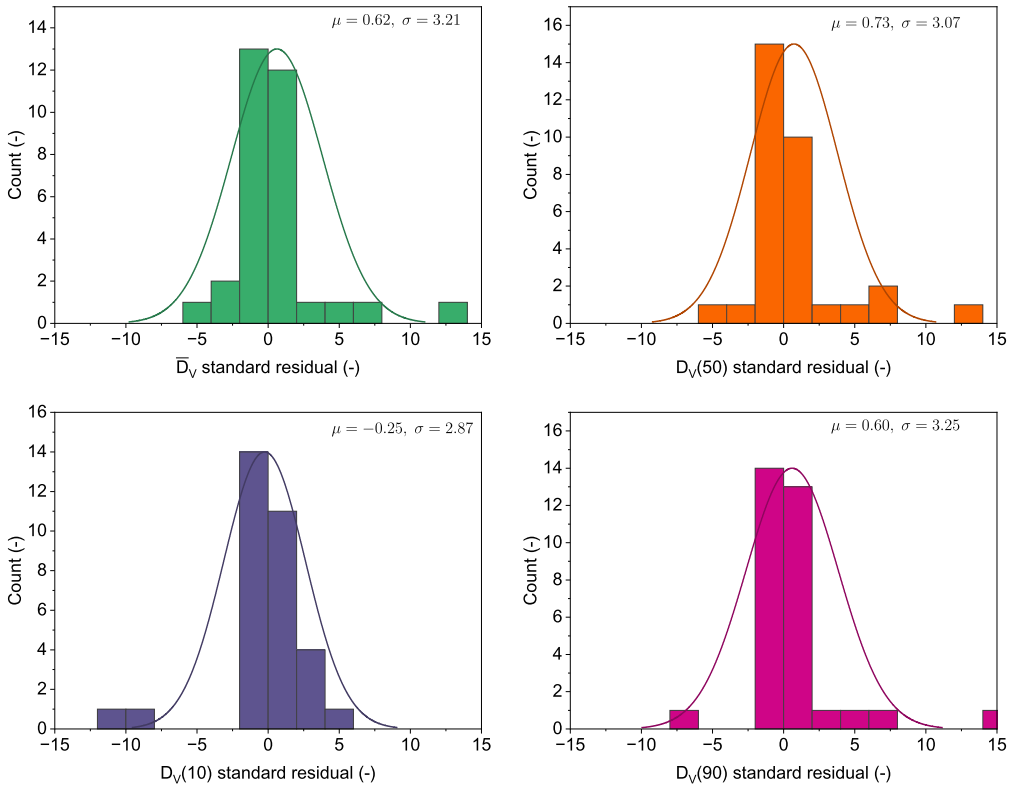


Fig. 8. Histograms of the standard residuals of the validated GP model on the test dataset in the final validation.

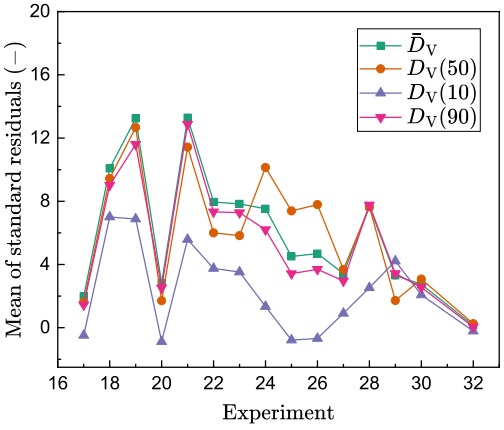


Fig. 9. Mean of the standard residual distribution of test dataset over the iterations of AL.

process changes significantly which has caused an increase in uncertainty after applying AL. As the AL procedure progressed, additional data generated in areas of high prediction uncertainty makes the GP model more confident in its prediction. This led to a gradual reduction in uncertainty from experiment 25 onward. Fig. 10 also shows that prior to applying AL, the posterior at experiment 17 had larger uncertainty at the inner regions of the design space and relatively smaller uncertainty at the corners and edges. This is because of the localised exploration achieved through the factorial DoE experiments (experiments 1 - 17). On applying AL, although there was an initial increase in uncertainty due to global exploration, the total predictive posterior variance of the model was distributed evenly across the feasible design space (see Fig. 10). This suggests that the AL method in each iteration systematically selected unexplored regions within the design space. In addition, Fig. 9 shows that over the iterations of the AL, the mean value of the standard residual distribution of the test dataset converges to values close to zero. This indicates that the global exploration phase achieved through AL also resulted in accurate predictions of the test dataset that grows in size in each iteration of AL. This observation also supports the hypothesis that successful adaptive learning of the GP model was achieved through the AL method. The predictive posterior distribution of the model was updated in each iteration of the AL to learn the true underlying function behind the data-generating process or the precipitation process herein. Finally, if the experimental budget is not a constraint, it is desirable to continue the AL procedure until the predictive posterior standard deviation of each response variable drops below the respective measurement error standard deviation. In this work, since each AL experiment was repeated thrice in order to accurately determine the measurement error, it was required to set a maximum budget on the number of AL experiments. Hence, as mentioned in Section 2.2.4, the AL procedure was stopped when the AL experiments reached the allowed maximum number, N_{max}, set at 15.

4. Conclusion

In this study, we have developed a ML-driven computational framework to assist in the synthesis of micro and nano-sized API particle formulations using an automated continuous flow precipitation process. The framework comprises four key modules: *i*) DoE module to generate an initial labelled dataset for training the ML models, *ii*) classification algorithms (both discriminative and generative) to identify feasible operating regions for the precipitation process, *iii*) multiple-output GP models for surrogate modelling to generate uncertainty-aware predictive models, and *iv*) AL for optimal experimental design to suggest the best conditions for subsequent experimentation for model training and validation.

The framework was effectively applied to steer an automated continuous flow precipitation platform for ibuprofen precipitation in continuous flow. For the identification of the feasible operating region, both SVM and GP models were utilized, with the SVM model demonstrating superior performance with an average test accuracy of 80%. The

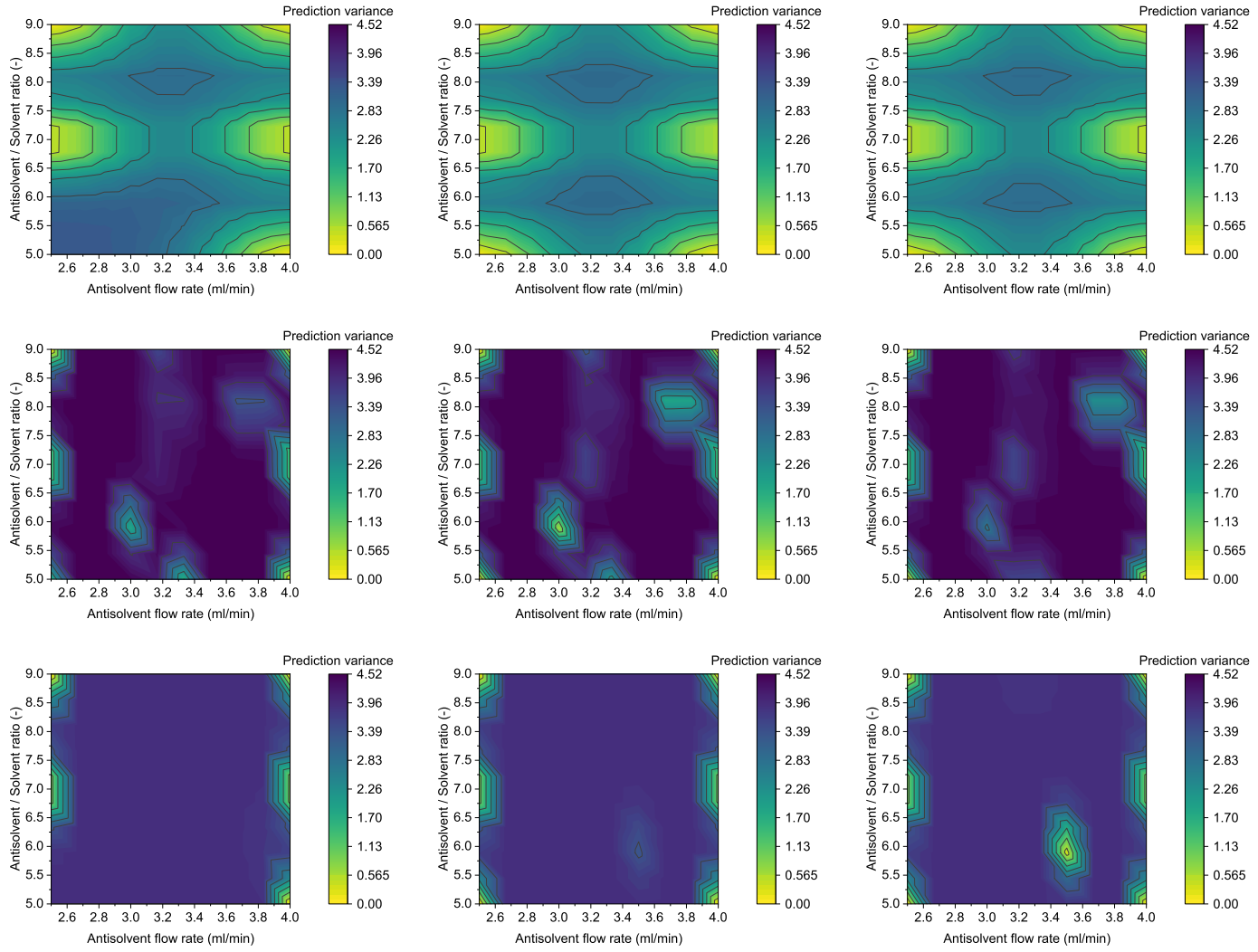


Fig. 10. Contour plots of the total predictive posterior variance of the GP regression model at varying antisolvent flow rates and antisolvent/solvent ratios. The first row corresponds to the variance after the first 17 experiments, while the second and third row respectively show the variances after 25 and 32 experiments. From left to right, the additive concentration varies from 1.5, 2.25, to 3.0 wt.%.

use of contour plots of predictions from the validated classifier model emerged as a visualization tool for determining feasibility regions, addressing reactor fouling issues. Moreover, this on-the-fly identification of the feasible operating region facilitates fast and efficient screening of stabilisers. This is critical, particularly in the first phases of drug development, where resource-intensive experimental trials need to be carried out at different process conditions and employing different stabilizers. Ruling out the infeasible operating space makes the particle size distribution measurement more accurate and reliable. It also streamlines the tuning of particle size distribution via feasibility constrained process control and optimization. Another advantage of developing cheap but accurate classifier models is to better understand the factors causing fouling. For a more accurate determination of the feasible operating space, one can also implement the classification problem in closed loop. AL methods can be employed to iteratively update the classifier model to learn the true feasibility boundary. In addition, one can also generate more accurate feasibility constraints by applying computational geometry methods on the feasibility predictions of the classifier model.

Regarding surrogate modelling, employing a multiple-output GP model holds promise not only because the technique is data-efficient and can handle data scarce situations while providing uncertainty-aware predictive models, but also because it can be seamlessly integrated with Bayesian inference and experimental design for process optimization. The strong agreement between the predictions of the calibrated GP model on the test dataset and the actual values indicates that this type of surrogate model can predict precisely the PSD moments, allowing for effective size control of precipitated particles. This is crucial for automation of API particle process development. As a next step, the calibrated GP model can be coupled with Bayesian optimization with classical acquisition functions such as upper confidence bound for process optimization.

Critically, we approached the surrogate modelling of multiple-output GP differently by calibrating the measurement error variance model separately from replications of experimental runs. This innovative method allowed us to use the calibrated measurement error variance model to infer the noise parameter of the GP model. This approach allows a precise integration of uncertainty of measurements during model identification tasks.

Glossary

Acronyms

AL	active learning
ANN	Artificial Neural Network
API	Active Pharmaceutical Ingredient
CNN	Convolutional Neural Network
CIJR	confined impinging jet reactor

DNN	Deep Neural Network
DoE	Design of Experiments
DRL	Deep Reinforcement Learning
FDA	Food and Drug Administration
GP	Gaussian process
GUI	Graphical User Interface
ICM	intrinsic coregionalization model
LD	laser diffraction
LHS	Latin Hypercube Sampling
ML	Machine learning
NPSV	network-published shared variable
OPC	Open Platform Communications
OPC DA	Open Platform Communications Data Access
PAT	Process Analytical Technology
PSD	particle size distribution
PSE	Process Systems Engineering
RBF	Radial basis function
RNN	Recurrent Neural Network
SGD	stochastic gradient descent
SVE	Shared Variable Engine
SVM	Support Vector Machine
VI	virtual instrument

Latin Symbols

c	Output of a classifier, which denotes a class label
$\mathbf{cov}(\cdot,\cdot)$	Covariance between two variables
e_{ij}	Standard residual (prediction error weighted by measurement error standard deviation) for the j -th output in the i - th experiment
f	Latent function of a regression model
f_c	Decision function of a classifier
g_m	Latent function of a classifier for predicting class m
k	Covariance or kernel function of a regression model
k_c	Covariance or kernel function of a classifier
L_c	Loss function used to train a classifier
M	Number of classes in classification problem
$N_{\mathbf{max}}$	Experimental budget for AL
N_u	Dimension of input space
N_y	Dimension of output space for regression model
$\mathcal{N}(\cdot,\cdot)$	Normal distribution with specified mean and variance
N	Total number of experiments
y	Response variable in regression
$\hat{y}$	Model predicted value of y

Greek Symbols

θ	Parameter vector of the kernel function of a regression model
θ_c	Parameter vector of the kernel function of a classification model
μ	Mean of a probability distribution
σ	Standard deviation of a probability distribution
σ_y	Standard deviation of measurement error
ψ	Objective function for AL

Vectors and Matrices

$\mathbf{u}$	input vector
$\mathbf{y}$	output vector
$\mathbf{U}$	$N \times N_u$ matrix formed by stacking all the input vectors
$\mathbf{Y}$	$N \times N_y$ matrix formed by stacking all the output vectors
$\mathbf{Y}_v$	$N \cdot N_y \times 1$ vector formed by vectorization of the matrix $\mathbf{Y}$

CRediT authorship contribution statement

Arun Pankajakshan: Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Methodology, Formal analysis, Data curation, Conceptualization. **Sayan Pal:** Writing – review & editing, Validation, Investigation, Data curation, Conceptualization. **Nicholas Snead:** Writing – review & editing, Software. **Juan Almeida:** Writing – review & editing, Software. **Maximilian O. Besenhard:** Writing – review & editing, Funding acquisition, Conceptualization. **Shorooq Abukhamees:** Writing – review & editing, Conceptualization. **Duncan Q.M. Craig:** Writing – review & editing, Funding acquisition, Conceptualization. **Asterios Gavriilidis:** Writing – review & editing, Supervision, Resources, Funding acquisition, Conceptualization. **Luca Mazzei:** Writing – review & editing, Supervision, Resources, Project administration, Funding acquisition, Conceptualization. **Federico Galvanin:** Writing – review & editing, Supervision, Resources, Methodology, Funding acquisition, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

This work was supported by the UK Engineering and Physical Sciences Research Council (EPSRC) [grant number EP/V050796/1].

Appendix A. Supplementary material

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.ces.2024.120780.

Data availability

I have provided the data as supplementary material

References

Amari, S., 1993. Backpropagation and stochastic gradient descent method. Neurocomputing 5, 185–196.
Ameli, S., Shadden, S.C., 2022. Noise estimation in gaussian process regression. preprint. arXiv:2206.09976.
Balandat, M., Karrer, B., Jiang, D.R., Daulton, S., Letham, B., Wilson, A.G., Bakshy, E., 2020. BoTorch: A Framework for Efficient Monte-Carlo Bayesian Optimization. Advances in Neural Information Processing Systems, vol. 33. http://arxiv.org/abs/1910.06403.
Benyahia, B., Lakerveld, R., Barton, P.I., 2012. A plant-wide dynamic model of a continuous pharmaceutical process. Ind. Eng. Chem. Res. 51, 15393–15412.
Besenhard, M.O., Pal, S., Gkogkos, G., Gavriilidis, A., 2023. Non-fouling flow reactors for nanomaterial synthesis. React. Chem. Eng.
Bingham, E., Chen, J.P., Jankowiak, M., Obermeyer, F., Pradhan, N., Karaletsos, T., Singh, R., Szerlip, P., Horsfall, P., Goodman, N.D., 2018. Pyro: deep universal probabilistic programming. J. Mach. Learn. Res.
Binois, M., Huang, J., Gramacy, R.B., Ludkovski, M., 2019. Replication or exploration? Sequential design for stochastic simulation experiments. Technometrics 61, 7–23.
Bishop, C.M., Nasrabadi, N.M., 2006. Pattern Recognition and Machine Learning, vol. 4. Springer.
Blei, D.M., Kucukelbir, A., McAuliffe, J.D., 2017. Variational inference: a review for statisticians. J. Am. Stat. Assoc. 112, 859–877.
Bonilla, E.V., Chai, K., Williams, C., 2007. Multi-task gaussian process prediction. Adv. Neural Inf. Process. Syst. 20.
Boobier, S., Hose, D.R., Blacker, A.J., Nguyen, B.N., 2020. Machine learning with physicochemical relationships: solubility prediction in organic solvents and water. Nat. Commun. 11, 5753.
Chang, C.C., Lin, C.J., 2011. Libsvm: a library for support vector machines. ACM Trans. Intell. Syst. Technol. 2, 1–27.
Coliaie, P., Prajapati, A., Ali, R., Korde, A., Kelkar, M.S., Nere, N.K., Singh, M.R., 2022. Machine learning-driven, sensor-integrated microfluidic device for monitoring and control of supersaturation for automated screening of crystalline materials. ACS Sensors 7, 797–805.

Di Fiore, F., Nardelli, M., Mainini, L., 2023. Active learning and bayesian optimization: a unified perspective to learn with a goal. preprint. arXiv:2303.01560.
Draper, N.R., Smith, H., 1998. Applied Regression Analysis, vol. 326. John Wiley & Sons.
Engineering, P., 2024. Perceptive engineering. https://www.perceptiveapc.com/. (Accessed 21 January 2024).
Fiordalis, A., Georgakis, C., 2013. Data-driven, using design of dynamic experiments, versus model-driven optimization of batch crystallization processes. J. Process Control 23, 179–188.
Fisher, R.A., 1935. The Design of Experiments. Oliver & Boyd, Edinburgh.
Foundation, O., 2024. Opc foundation: home page. https://opcfoundation.org/. (Accessed 21 January 2024).
Galy-Fajou, T., Wenzel, F., Donner, C., Opper, M., 2020. Multi-class gaussian process classification made conjugate: efficient inference via data augmentation. In: Uncertainty in Artificial Intelligence. PMLR, pp. 755–765.
Gardner, J., Pleiss, G., Weinberger, K.Q., Bindel, D., Wilson, A.G., 2018. Gpytorch: blackbox matrix-matrix gaussian process inference with gpu acceleration. Adv. Neural Inf. Process. Syst. 31.
Garg, M., Rathore, A.S., 2021. Process development in the qbd paradigm: implementing design of experiments (doe) in anti-solvent crystallization for production of pharmaceuticals. J. Cryst. Growth 571, 126263.
Grandini, M., Bagli, E., Visani, G., 2020. Metrics for multi-class classification: an overview. preprint. arXiv:2008.05756.
Griffin, D.J., Grover, M.A., Kawajiri, Y., Rousseau, R.W., 2016. Data-driven modeling and dynamic programming applied to batch cooling crystallization. Ind. Eng. Chem. Res. 55, 1361–1372.
Gutmann, B., Cantillo, D., Kappe, C.O., 2015. Continuous-flow technology—a tool for the safe manufacturing of active pharmaceutical ingredients. Angew. Chem., Int. Ed. 54, 6688–6728.
Hinz, D.C., 2006. Process analytical technologies in the pharmaceutical industry: the fda's pat initiative. Anal. Bioanal. Chem. 384, 1036–1042.
Jakkala, K., 2021. Deep gaussian processes: a survey. preprint. arXiv:2106.12135.
Jeffrey, T., Jim, K., 2006. Labview for Everyone: Graphical Programming Made Easy and Fun. Prentice Hall PTR.
Jia, S., Yang, P., Gao, Z., Li, Z., Fang, C., Gong, J., 2022. Recent progress in antisolvent crystallization. CrystEngComm 24, 3122–3135.
Kingma, D.P., Ba, J., 2014. Adam: a method for stochastic optimization. preprint. arXiv:1412.6980.
Kingma, D.P., Welling, M., 2013. Auto-encoding variational bayes. preprint. arXiv:1312.6114.
Lakerveld, R., Benyahia, B., Heider, P.L., Zhang, H., Wolfe, A., Testa, C.J., Ogden, S., Hersey, D.R., Mascia, S., Evans, J.M., et al., 2015. The application of an automated control strategy for an integrated continuous pharmaceutical pilot plant. Org. Process Res. Develop. 19, 1088–1100.
Lapkin, A., Loponov, K., Tomaiuolo, G., Guido, S., 2017. Solids in continuous flow reactors for specialty and pharmaceutical syntheses. Sustain. Flow Chem. Methods Appl., 277–308.
Le, Q.V., Smola, A.J., Canu, S., 2005. Heteroscedastic gaussian process regression. In: Proceedings of the 22nd International Conference on Machine Learning, pp. 489–496.
Lele, S.R., 2020. How should we quantify uncertainty in statistical inference? Front. Ecol. Evol. 8, 35.
Liu, J., Liu, T., Chen, J., Yue, H., Zhang, F., Sun, F., 2020. Data-driven modeling of product crystal size distribution and optimal input design for batch cooling crystallization processes. J. Process Control 96, 1–14.
Lonare, A.A., Patel, S.R., 2013. Antisolvent crystallization of poorly water soluble drugs. Int. J. Chem. Eng. Appl. 4, 337.
Mahmoud, M.S., Sabih, M., Elshafei, M., 2015. Using opc technology to support the study of advanced process control. ISA Trans. 55, 155–167.
Manee, V., Baratti, R., Romagnoli, J.A., 2022. Learning to navigate a crystallization model with deep reinforcement learning. Chem. Eng. Res. Des. 178, 111–123.
Mascia, S., Heider, P.L., Zhang, H., Lakerveld, R., Benyahia, B., Barton, P.I., Braatz, R.D., Cooney, C.L., Evans, J.M., Jamison, T.F., et al., 2013. End-to-end continuous manufacturing of pharmaceuticals: integrated synthesis, purification, and final dosage formation. Angew. Chem. 125, 12585–12589.
Meng, Q., Anandan, P.D., Rielly, C.D., Benyahia, B., 2023. Multi-agent reinforcement learning and rl-based adaptive pid control of crystallization processes. In: Computer Aided Chemical Engineering, vol. 52. Elsevier, pp. 1667–1672.
Mohammadi, H.S., Asl, A.H., Khajenoori, M., 2023. Production of pharmaceutical micro and nano particles by subcritical water based technologies: a review. J. Drug Deliv. Sci. Technol., 104621.
Morissette, S.L., Almarsson, Ö., Peterson, M.L., Remenar, J.F., Read, M.J., Lemmo, A.V., Ellis, S., Cima, M.J., Gardner, C.R., 2004. High-throughput crystallization: polymorphs, salts, co-crystals and solvates of pharmaceutical solids. Adv. Drug Deliv. Rev. 56, 275–300.
Murphy, K.P., 2012. Machine Learning: a Probabilistic Perspective. MIT Press.
Nagy, B., Galata, D.L., Farkas, A., Nagy, Z.K., 2022. Application of artificial neural networks in the process analytical technology of pharmaceutical manufacturing—a review. AAPS J. 24, 74.
Nagy, Z.K., Fevotte, G., Kramer, H., Simon, L.L., 2013. Recent advances in the monitoring, modelling and control of crystallization systems. Chem. Eng. Res. Des. 91, 1903–1922.
Nandi, S., Verstrepen, L., Hugo Silva, M., Padrela, L., Tajber, L., Collas, A., 2024. Continuous microfluidic antisolvent crystallization as a bottom-up solution for the development of long-acting injectable formulations. Pharmaceutics 16, 376.
Narayanan, H., Dingfelder, F., Condado Morales, I., Patel, B., Heding, K.E., Bjelke, J.R., Egebjerg, T., Butté, A., Sokolov, M., Lorenzen, N., et al., 2021. Design of biopharmaceutical formulations accelerated by machine learning. Mol. Pharm. 18, 3843–3853.
NI, 2024. NI LabVIEW. https://www.ni.com/en.html.software.
Öner, M., Montes, F.C., Ståhlberg, T., Stocks, S.M., Bajtner, J.E., Sin, G., 2020. Comprehensive evaluation of a data driven control strategy: experimental application to a pharmaceutical crystallization process. Chem. Eng. Res. Des. 163, 248–261.
Pal, S., Pankajakshan, A., Besenhard, M.O., Snead, N., Almeida, J., Abukhamees, S., Craig, D., Galvanin, F., Gavriilidis, A., Mazzei, L., 2024. Automated continuous crystallization platform with real-time particle size analysis via laser diffraction. Org. Process Res. Develop. 28, 2755–2764.
Palmtag, A., Rousselli, J., Gröschl, H., Jupke, A., 2023. Hybrid modeling of drop breakage in pulsed sieve tray extraction columns. Front. Chem. Eng. 5, 1274349.
Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E., 2011. Scikit-learn: machine learning in Python. J. Mach. Learn. Res. 12, 2825–2830.
Price, S.L., 2014. Predicting crystal structures of organic compounds. Chem. Soc. Rev. 43, 2098–2111.
Rasmussen, C.E., Williams, C.K., 2006. Gaussian Processes for Machine Learning, vol. 1. MIT Press, Cambridge, MA.
Rigollet, P., 2015. 18.657 Mathematics of Machine Learning. Lecture Notes. MIT OpenCourseWare, Cambridge, MA, USA.
Roberts, T.C., Langer, R., Wood, M.J., 2020. Advances in oligonucleotide drug delivery. Nat. Rev. Drug Discov. 19, 673–694.
Rohani, S., Haeri, M., Wood, H., 1999a. Modeling and control of a continuous crystallization process part 1. linear and non-linear modeling. Comput. Chem. Eng. 23, 263–277.
Rohani, S., Haeri, M., Wood, H., 1999b. Modeling and control of a continuous crystallization process part 2. model predictive control. Comput. Chem. Eng. 23, 279–286.
Salami, H., McDonald, M.A., Bommarius, A.S., Rousseau, R.W., Grover, M.A., 2021. In situ imaging combined with deep learning for crystallization process monitoring: application to cephalexin production. Org. Process Res. Develop. 25, 1670–1679.
Schmidt, J., Marques, M.R., Botti, S., Marques, M.A., 2019. Recent advances and applications of machine learning in solid-state materials science. npj Comput. Mater. 5, 83.
Seo, S., Wallat, M., Graepel, T., Obermayer, K., 2000. Gaussian process regression: active data selection and test point rejection. In: Mustererkennung 2000: 22. DAGM-Symposium, Kiel, 13.–15. September 2000. Springer, pp. 27–34.
Shahriari, B., Swersky, K., Wang, Z., Adams, R.P., De Freitas, N., 2015. Taking the human out of the loop: a review of bayesian optimization. Proc. IEEE 104, 148–175.
Shannon, C.E., 1948. A mathematical theory of communication. Bell Syst. Tech. J. 27, 379–423.
Sinha, B., Müller, R.H., Möschwitzer, J.P., 2013. Bottom-up approaches for preparing drug nanocrystals: formulations and factors affecting particle size. Int. J. Pharm. 453, 126–141.
Sjoegren, R., 2023. Design of experiments for python. https://pypi.org/project/pyDOE2/. (Accessed May 2023).
Smola, A.J., Schölkopf, B., 2004. A tutorial on support vector regression. Stat. Comput. 14, 199–222.
Talicska, C.N., O'Connell, E.C., Ward, H.W., Diaz, A.R., Hardink, M.A., Foley, D.A., Connolly, D., Girard, K.P., Ljubicic, T., 2022. Process analytical technology (pat): applications to flow processes for active pharmaceutical ingredient (api) development. React. Chem. Eng. 7, 1419–1428.
U.S. Food and Drug Administration, 2004. Guidance for industry: pat - a framework for innovative pharmaceutical development, manufacturing, and quality assurance. https://www.fda.gov/media/71012/download.
Vamathevan, J., Clark, D., Czodrowski, P., Dunham, I., Ferran, E., Lee, G., Li, B., Madabhushi, A., Shah, P., Spitzer, M., et al., 2019. Applications of machine learning in drug discovery and development. Nat. Rev. Drug Discov. 18, 463–477.
Woodley, S.M., Catlow, R., 2008. Crystal structure prediction from first principles. Nat. Mater. 7, 937–946.
Wu, H., Khan, M.A., Hussain, A.S., 2007. Process control perspective for process analytical technology: integration of chemical engineering practice into semiconductor and pharmaceutical industries. Chem. Eng. Commun. 194, 760–779.
Wyttenbach, N., Niederquell, A., Kuentz, M., 2020. Machine estimation of drug melting properties and influence on solubility prediction. Mol. Pharm. 17, 2660–2671.
Xiouras, C., Cameli, F., Quillo, G.L., Kavousanakis, M.E., Vlachos, D.G., Stefanidis, G.D., 2022. Applications of artificial intelligence and machine learning algorithms to crystallization. Chem. Rev. 122, 13006–13042.
Xu, Z., Zhe, S., 2024. Standard gaussian process is all you need for high-dimensional bayesian optimization. preprint. arXiv:2402.02746.
Zhang, P., Weeranoppanant, N., Thomas, D.A., Tahara, K., Stelzer, T., Russell, M.G., O'mahony, M., Myerson, A.S., Lin, H., Kelly, L.P., et al., 2018. Advanced continuous flow platform for on-demand pharmaceutical manufacturing. Chemistry 24, 2776–2784.
Zhou, G., Moment, A., Yaung, S., Cote, A., Hu, T.E., 2013. Evolution and application of an automated platform for the development of crystallization processes. Org. Process Res. Develop. 17, 1320–1329.