Safe Chance Constrained Reinforcement Learning for Batch Process Control

M. Mowbray[a], P. Petsagkourakis[b], E.A. del Rio-Chanona[c], R. Smith[a], D. Zhang[a,*]

[a] *Centre for Process Integration, School of Chemical Engineering and Analytical Science, The University of Manchester, Manchester, M13 9PL, United Kingdom*
[b] *Centre for Process Systems Engineering (CPSE), Department of Chemical Engineering, University College London, Torrington Place, London, WC1E 7JE, United Kingdom*
[c] *Centre for Process Systems Engineering (CPSE), Department of Chemical Engineering, Imperial College London, London, SW7 2AZ, United Kingdom*

Abstract

Reinforcement Learning (RL) controllers have generated excitement within the control community. The primary advantage of RL controllers relative to existing methods is their ability to optimize uncertain systems independently of explicit assumption of process uncertainty. Recent focus on engineering applications has been directed towards the development of safe RL controllers. Previous works have proposed approaches to account for constraint satisfaction through constraint tightening from the domain of stochastic model predictive control. Here, we extend these approaches to account for plant-model mismatch. Specifically, we propose a data-driven approach that utilizes Gaussian processes for the offline simulation model and use the associated posterior uncertainty prediction to account for joint chance constraints and plant-model mismatch. The method is benchmarked against nonlinear model predictive control via case studies. The results demonstrate the ability of the methodology to account for process uncertainty, enabling satisfaction of joint chance constraints even in the presence of plant-model mismatch.

Keywords: *Safe Reinforcement Learning, Optimal Control, Dynamic Optimization, Bioprocess Operation, Machine Learning*

1. Introduction

Recently, there has been growing interest amongst the research community and industry in the development of reinforcement learning (RL) based control schemes [1]. This is underpinned by the ability of RL to naturally account for process stochasticity and handle nonlinear dynamics, and reflected by a growing literature that demonstrates application empirically [2, 3, 4]. All of these works rely on offline simulation of a process model, with results often validated on the same model that the RL policy was trained. This implicitly considers that the model used offline is in fact a perfect description of the real process and provokes the question: *"if a model is available, why not use model predictive*

[*] Corresponding author
Email address: dongda.zhang@manchester.ac.uk (D. Zhang)

control (MPC)?". In practice, the real system is never perfectly described by the available model. In the presence of uncertainty, the predictions from a model may not have closed-form expression, e.g. propagation of uncertainty using Bayesian inference. Here lies the real attraction of RL controllers. The ability to find an optimal control policy [5, 6] independently of closed-form expressions of the uncertain process dynamics.

However, there is a dualism implicit to RL. RL is very data expensive because knowledge about the uncertain dynamics and the quality of a control policy is instead gained by sampling [7]. Offline learning (simulation) is absolutely required due to the cost of real world data and the operational and safety risk associated with conducting the RL process online. As a result, there remains a dependence on the availability of a description of the physical system for offline simulation, which provides means to conduct preliminary learning before deployment to the real system. Despite this, few works consider the transfer of the policy [8] to the real online system, which promotes concerns for operational safety[1]. For example, if model-process mismatch exists, constraints may be violated or the process driven to unsafe operating regimes. Given the acknowledgement that no model is a perfect description of the real process - the development of methods should consider that RL exploits the mathematical nature of the offline model. Similar concerns are addressed in [9].

Broadly, there are two approaches to synthesising the type of safe controller required: modifications could be made to the reinforcement learning process [10, 11, 12], or modifications made to the offline model [13, 14], which can then be integrated into the RL objective. Recent works are discussed in the following with consideration directed to both operational and safety concerns.

1.1. Safe Reinforcement Learning

One of the earliest works in process systems engineering (PSE), which considers the online operational safety of reinforcement learning is provided by [15]. Here, the authors present an action-value method, with integration of a Parzen probability density estimator [16] to bias the action-value function approximation based on the local data density. In this case, the data is used to construct the action-value function and hence the data density helps quantify epistemic uncertainty (i.e. the reducible part of model uncertainty arising from a lack of information - data or knowledge - about the underlying functional [17]). This concept is shared in more recent work [18], and enables the implementation to produce conservative controls and restricts optimization from exploiting the mathematical nature of the approximate action-value function. However, this approach does not consider operational constraints or the accuracy of the underlying model. For RL to be deployed to real process systems, operational constraints should be satisfied absolutely (if hard) or with high probability (if soft). One approach to achieve this is underpinned by modification of the control selected by the RL agent, in order to ensure the system remains within some safe set via direct optimal control (DOC) [19, 20]. However, the use of DOC retains explicit dependence upon a process model and imposes non-trivial learning rules that could affect the optimality of the policy produced.

Other methods directly leverage the Markov decision process (MDP) formulation, upon which the reinforcement learning problem is built. This approach tends to avoid DOC and promotes use of 'model-free' methods. A reasonably popular approach to address constraints in the RL setting is provided by the constrained MDP (CMDP) formulation. In [21], the authors approach the need for satisfaction of operational constraints via CMDP, but do so in expectation and simultaneously negate process-model mismatch. In [22], the authors propose the identification of a lyapunov function

[1]This is also placed in the scope of a wider concern regarding the *interpretability* of machine learning systems

(this time model-free and outside of the CMDP framework) to ensure the process stays within some safe set with a given probability. However, potential issues arising from plant-model mismatch are similarly ignored in offline simulations. In [23], an approach to robust control is presented (i.e. the method optimizes for the worst case event), and the presence of process-model mismatch is considered. However, the framework is limited to linear systems with additive uncertainty. Recently, in [24] the authors present an approach to address high probability constraint satisfaction based on the augmented lagrangian. However, the penalty term presented does not provide information about the quality of control selection (i.e. essentially ignoring the RL problem) and is likely to lead to conservative control policies. There have been two methods proposed recently, by [25, 26], which integrate a similar penalty method into the RL problem properly, and achieve high probability constraint satisfaction. This is achieved through deployment of the concept of constraint tightening, which is common to the stochastic MPC (sMPC) community [27, 28, 29] A further method has been proposed by [30] for the case of hard constraints, which constructs a slow non-stationary MDP to promote stability of learning via the implementation of a dynamic penalty method. However, the aforementioned works negate the presence of offline model-process mismatch.

Most of the previous works ignore issues arising from process-model mismatch. The domain of batch RL (otherwise known as offline RL) has drawn a lot of recent research interest [31]. The promise of this field lies in the synthesis of real-world control policies from existing datasets (offline). The key idea in batch RL is to learn with awareness of the limitations of the available data. Many of the works set in this domain focus on action-value methods and look to bias (or regularise) the action-value function approximation [11] by considering the data density [10] in a manner not dissimilar to [15]. More recently, attention has been directed towards considerate construction of an offline model, based on the available data and this directs attention in the following analysis.

1.2. Uncertainty Aware Modelling and Control

A key consideration in the development of model-based RL approaches is the relationship between model construction and policy learning. For example, in [32], the problem of learning under the limitations of a local model and improving policy performance on the real process is considered within a game theoretic framework (similar to model-based design of experiments). However, it is not clear as to whether this approach would ensure real-process safety unless modifications were made to the reward function. This problem is approached by the work presented in [33] and more recently in [13]. In [13], the epistemic uncertainty associated with offline prediction is quantified via the variance of a model ensemble. The epistemic uncertainty is used to modify the reward function of the MDP to synthesise a safe control policy without further interaction with the real system. A type of model, which achieves this more naturally than an ensemble, is the Gaussian process (GP). GPs are data-driven models and their use is well documented in PSE applications [34, 35, 36, 37, 38]. In part, this is due to their compatibility with small datasets, but primarily for their natural quantification of epistemic and aleatoric uncertainty. In a number of previous works, (realisations of) GPs have been used to inform control decisions. Most of these works lie in the domain of sMPC [36, 39], however, a few hail from the field of RL-based policy optimization [40, 41]. In [40], the authors compute gradients for policy improvement analytically, resulting in a highly efficient algorithm for unconstrained problems. In [41], the authors utilise GPs and the variance of the posterior distribution to produce a controller-directed exploration strategy, but negate propagation of model uncertainty and, again, process constraints. In the following, we draw from works closer to sMPC [39, 36], to synthesise a safe RL-based control policy, which considers both operational constraints and process-model mismatch.

1.3. Contribution

A number of RL-based methodologies have been proposed to ensure operational constraints are satisfied with high probability [26, 25, 24]. Other works have been proposed to consider the process-model mismatch that exists when learning an RL policy offline [10, 13, 12]. However, as far as the authors are aware, there are no methods, which achieve both. In this work, we propose a method that synchronously satisfies operational constraints with high probability, whilst respecting the limitations of a process model. In this work, we deploy the use of GPs to construct a data-driven state space model. The variance of the posterior predictive distribution of the GP is used in two different ways: firstly, it provides a constraint tightening mechanism to back the nominal (or expected) process away from the constraint boundary (to ensure constraint satisfaction with high probability); and, secondly, it is used to penalise exploration of regions of the GP model with high epistemic uncertainties. The full method as proposed also implements a Bayesian optimization strategy in order to tune the degree of constraint tightening - balancing operational risk with performance. Here, we draw analogue to reward shaping, except in this case, we identify a policy variant mechanism to ensure constraint satisfaction as desired [42]. Importantly, the dimensionality of the shaping problem is equivalent to the number of operational constraints imposed on the system, which provides means to scale the method to larger problems. Further, the approach inherently provides the added benefits of being completely data-driven and synchronously accounting for model uncertainty, removing demands for assumption of mechanistic process knowledge.

The following is structured as follows: in Section 2, we outline the problem statement and implicitly define the processes of interest; in Section 3, the methodology is presented; in Section 4 a fed-batch bioprocess case study is presented with a view to demonstrate the methodology; in Section 5 and 6 the results and discussion, and conclusion are presented, respectively.

2. Problem Statement

This work is concerned with the synthesis of an optimal control strategy for nonlinear, uncertain systems of the form:

$$\mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{u}_t, \mathbf{s}_t) \tag{1}$$

where $\mathbf{x} \in \mathbb{X} \subseteq \mathbb{R}^{n_x}$ denotes the system state; $\mathbf{u} \in \mathbb{U} \subseteq \mathbb{R}^{n_u}$ the control inputs to the system; $t = [1, \dots, T]$ denotes the discrete time index; $\mathbf{s} \in \mathbb{S} \subseteq \mathbb{R}^{n_s}$, where $\mathbb{S}$ represents a set of realisations of process stochasticity; and, $f : \mathbb{X} \times \mathbb{U} \times \mathbb{S} \rightarrow \mathbb{X}$. Here, no formal assumption is made regarding the source of stochasticity $\mathbb{S}$, but it could be introduced via parametric uncertainty or disturbances. In either case, given the presence of stochasticity within system description, Eq. 1 may be expressed equivalently via the following conditional probability density function:

$$\mathbf{x}_{t+1} \sim p(\mathbf{x}_{t+1} | \mathbf{x}_t, \mathbf{u}_t) \tag{2}$$

Specifically, it is assumed that the process dynamics adhere to description as a Markov process, and therefore that the associated decision-making problem may be formalized as a Markov decision process (MDP). MDPs provide a probabilistic value framework for decision making in uncertain systems, which display the Markov property. Under the MDP framework, the probability of observing a given process trajectory $p(\boldsymbol{\tau})$, under a control policy π is described:

$$p(\boldsymbol{\tau}) = p(\mathbf{x}_0) \prod_{t=0}^{T-1} \pi(\mathbf{u}_t|\mathbf{x}_t) p(\mathbf{x}_{t+1}|\mathbf{x}_t, \mathbf{u}_t) \tag{3}$$

where $\boldsymbol{\tau} = (\mathbf{x}_0, \mathbf{u}_0, \dots, \mathbf{x}_T)$ denotes the process trajectory; $p(\mathbf{x}_0)$ denotes the initial state distribution; $p(\mathbf{x}_{t+1}|\mathbf{x}_t, \mathbf{u}_t)$ the process dynamics; and the policy $\pi(\mathbf{u}_t|\mathbf{x}_t)$ is explicitly defined as a conditional probability function over control inputs. Provided process evolution is subject to a stochastic policy and process dynamics, the performance of a policy is evaluated via the expected discounted sum of rewards $R_{t+1} \in \mathbb{R}$ accumulated from the initial state:

$$\begin{aligned} G(\boldsymbol{\tau}) &= \sum_{t=0}^{T-1} \gamma^t R_{t+1} \\ J &= \int p(\boldsymbol{\tau}) G(\boldsymbol{\tau}) d\boldsymbol{\tau} \end{aligned} \tag{4}$$

where the reward is allocated by a reward function $R : \mathbb{X} \times \mathbb{U} \times \mathbb{X} \to R_{t+1}$ and $\gamma = [0, 1]$ is the discount factor. Therefore, the optimal policy π^*:

$$\pi^* = \arg\max_{\pi} J \tag{5}$$

One approach to learning such a controller is via Reinforcement Learning (RL). However, under the framework provided by MDPs, the optimal policy π^* (and, hence RL) implicitly neglects the satisfaction of both safety and operational constraints. In applications related to this work (i.e. industrial batch process systems), the satisfaction of both operational and safety constraints is of concern. As such, it is of interest to develop an RL-based methodology for the synthesis of an optimal control policy π_C^*, which respects constraints. The problem statement follows that common to works set in the domain of stochastic optimal control:

$$\mathcal{P}(\pi_C) := \begin{cases} \max\limits_{\pi} J \\ \text{s.t.} \\ \mathbf{x}_0 \sim p(\mathbf{x}_0) \\ \mathbf{x}_{t+1} \sim p(\mathbf{x}_{t+1}|\mathbf{x}_t, \mathbf{u}_t) \\ \mathbf{u}_t \sim \pi(\mathbf{u}_t|\mathbf{x}_t) \\ \mathbf{u}_t \in \hat{\mathbb{U}} \\ \mathbb{P}(\bigcap\limits_{i=0}^{T} \{\mathbf{x}_i \in \hat{\mathbb{X}}_i\}) \geq 1 - \alpha \\ \forall t \in \{0, ..., T-1\} \end{cases} \tag{6}$$

where $\hat{\mathbb{U}} \subset \mathbb{U}$ represents the set of control inputs, which satisfy hard constraints on the control space; and, $\hat{\mathbb{X}} \subset \mathbb{X}$ denotes the set of states, which satisfy operational and safety constraints imposed on the state space. Under the assumption that the problem definition may have n_g constraints, $\hat{\mathbb{X}}$ may be expanded more generally as the *joint* chance constraint set, such that:

$$\hat{\mathbb{X}}_t = \{\mathbf{x}_t \in \mathbb{G}_{j,t}, \forall j \in \{1, \dots, n_g\}\} \tag{7}$$

where $\mathbb{G}_{j,t} \subset \mathbb{R}^{n_x}$ defines the set of states, which ensure satisfaction of the j^{th} constraint at time step t. Specifically, in the following analysis, we assume that:

$$\mathbb{G}_{j,t} = \{\mathbf{x}_t \in \mathbb{R}^{n_x} : A_j^T \mathbf{x}_t - b_j \leq 0\} \tag{8}$$

where $A_j \in \mathbb{R}^{n_x}$ and $b_j \in \mathbb{R}$ define the j^{th} constraint. The general principles discussed subsequently extend to problems with nonlinear constraints. However, in that case, the constraints should be represented by lower order power series expansions of the nonlinear functions [43] i.e. the nonlinear expressions should be linearized. Given that the process is stochastic, the constraints are 'softened' such that satisfaction is guaranteed for all time $t = \{0, \ldots, T\}$ with a desired probability, denoted $1 - \alpha$.

Theoretically, solution to Eq. 6 may be realised via exact dynamic programming (DP), which requires exact descriptions of the probabilistic process dynamics. In process systems, these are typically unavailable. Further, DP is known to suffer from the *the curse of dimensionality*, which implies that high dimensional problems, or those that operate over continuous state and control spaces, are computationally intractable. In the domain of sMPC, works generally leverage reformulation of the problem via deterministic expressions for the joint chance constraints and modelling assumptions regarding the nature of process stochasticity [27, 44]. This work similarly forms a deterministic surrogate of Eq. 6 in combination with Gaussian process (GP) data-driven modelling, and identifies a reinforcement learning (RL) based control policy, which naturally accounts for process stochasticity in a closed-loop manner. These benefits are complementary to those noted in Section 1.3. In the following section, a methodology is proposed for synthesis of the controller π_C.

3. Methodology

3.1. Gaussian Processes for Data-Driven Dynamic Modelling

Model-free RL-based policies are learned through Monte Carlo (MC) sampling of the process dynamics and iteratively improved based on the collected data. This is otherwise known as policy iteration. For real world applications, the synthesis of RL-policies is dependent upon an accurate description (model) of the process dynamics. For nonlinear, uncertain processes, construction of mechanistic dynamical models can be problematic, even if understanding of the fundamental mechanisms driving process behaviour exists. Hence, the construction of a purely data-driven model is proposed to represent the discrete time, evolution of the nonlinear, uncertain dynamical system described by $\mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{u}_t, \mathbf{s}_t)$, i.e. Eq 1. In order to construct a representation of the system dynamics, it is assumed that: a) f is a smooth function and b) there is an available dataset $\mathcal{D}$, which is composed as follows:

$$\mathcal{D} = [\mathbf{\Upsilon}^T \ \mathbf{Y}^T], \qquad \mathbf{Y} = [\mathbf{y}_i, \ldots, \mathbf{y}_N], \qquad \mathbf{\Upsilon} = [\boldsymbol{\upsilon}_i, \ldots, \boldsymbol{\upsilon}_N], \qquad \boldsymbol{\upsilon}_i = \left[\mathbf{x}_i^T \ \mathbf{u}_i^T\right]^T, \tag{9}$$

where $\boldsymbol{\upsilon} \in \mathbb{R}^{n_\upsilon}$, $n_\upsilon = n_x + n_u$ are input measurements and $\mathbf{y} \in \mathbb{R}^{n_x}$ are output measurements of the system, which are gathered subject to some noisy process $\boldsymbol{\omega} \in \mathbb{W} \subseteq \mathbb{R}^{n_x}$ [45]. Here, $\mathbb{W}$ is assumed to be an infinite set representative of possible realisations of system noise, such that:

$$\begin{aligned} \mathbf{y}_i &= f(\boldsymbol{\upsilon}_i) + \boldsymbol{\omega}_i \\ \boldsymbol{\omega}_i &\sim \mathcal{N}(0, \Sigma_n) \end{aligned} \tag{10}$$

where $\Sigma_n = diag([\sigma_{n,1}^2, \ldots, \sigma_{n,n_x}^2]) \in \mathbb{R}^{n_x \times n_x}$ defines a diagonal matrix, where each element on the diagonal denotes a state dependent variance. Further, as usual, it is assumed that all datapoints

$d_i = [\boldsymbol{v}_i, \mathbf{y}_i]$ (equivalent to rows of $\mathcal{D}$) are independently and identically distributed (i.i.d.). Parallel can be drawn between Eq. 2, such that Eq. 10 is equivalently described as a conditional probability function $\mathbf{y} \sim p(\mathbf{y}|\boldsymbol{v})$. This description of data generation shares similarities to assumptions made in Section 2 and directs attention to a branch of probability theory known as stochastic processes (SPs), and in particular Gaussian processes (GPs).

3.1.1. Gaussian Processes

SPs define a probability model over an infinite collection of random variables, any finite subset of which have a joint distribution [46]. This definition leads to the interpretation of SPs as probability distributions over functions [45], such that one realisation of an SP can be thought of as obtaining a sample from a function space. When the distribution over the function space is assumed Gaussian, the resultant model is termed a GP. In the following, GPs are treated as multiple-input, single-output models. Hence a single GP provides a functional mapping descriptive of the future discrete time evolution of a single state, given observation of the full system state and control inputs at the current time index. It is of interest to this work to construct a multiple-input, multiple-output state space model. This is discussed subsequently in Section 3.1.2.

A GP is fully specified by a mean function $\mathbf{m}(\cdot)$ and covariance function $k(\cdot, \cdot)$, such that:

$$f_{GP}(\boldsymbol{v}) \sim \mathcal{GP}\big(\mathbf{m}(\boldsymbol{v}), k(\boldsymbol{v}, \boldsymbol{v}')\big) \tag{11}$$

A number of covariance functions exist within the GP toolbox with selection defining the properties of the GP in function space. As such, the decision constitutes a problem not dissimilar from architecture search in neural networks where information about the dataset and system concerned often aids model construction. Popular choices include the Matern 5/2 and radial basis function (RBF) covariance functions[45]. The RBF is detailed as follows:

$$k_{rbf}(\boldsymbol{v}, \boldsymbol{v}') = \sigma_f^2 \exp\big(-(\boldsymbol{v} - \boldsymbol{v}')^T \boldsymbol{H} (\boldsymbol{v} - \boldsymbol{v}')\big) \tag{12}$$

where $\boldsymbol{H} = diag(\boldsymbol{\eta}) \in \mathbb{R}^{n_v \times n_v}$, $\boldsymbol{\eta} = \big[1/l_1^2, \ldots, 1/l_{n_v}^2\big]^T \in \mathbb{R}^{n_v}$, and $\boldsymbol{\lambda} = [\boldsymbol{\eta}^T, \sigma_f^2]$ are hyperparameters of the kernel function, which dictate specific behaviour of the GP in function space and are to be tuned by the modeller. The approach to hyperparameter tuning, will be treated in the next section.

GP model inference takes place within the framework provided by Bayesian reasoning. The assertion of a modelling decision regarding the mean and covariance function therefore represents a prior belief about the possible properties of the hidden, functional relationship expressed in the dataset $\mathcal{D}$. Often, a zero mean ($\mathbf{m}(\boldsymbol{v}) = 0$) is assumed. When presented with a new test input $\boldsymbol{v}^* \in \mathbb{R}^{n_v}$, the construction of a single GP model for the j^{th} state leads to the generation of an associated prediction $y_j^* \in \mathbb{R}$ via the following joint prior distribution:

$$\begin{bmatrix} \mathbf{Y}_j^T \\ y_j^* \end{bmatrix} = \mathcal{N}\left(0, \begin{bmatrix} K + \sigma_n^2 I_N & K_* \\ K_*^T & k(\boldsymbol{v}^*, \boldsymbol{v}^*) \end{bmatrix}\right) \tag{13}$$

where $\mathbf{Y}_j \in \mathbb{R}^{1 \times N}$ denotes the j^{th} row of the output of the training dataset $\mathbf{Y}$; $K \in \mathbb{R}^{N \times N}$ denotes the Gram matrix, such that provided with training input measurements (see Eq. 9), element $k_{m,n} = k(\boldsymbol{v}_m, \boldsymbol{v}_n)$ (see Eq. 12), where $m = [1, \ldots, N]$ and $n = [1, \ldots, N]$; σ_n^2 denotes the variance of the noise associated with observation of state $y_j \in \mathbb{R}$ (see Eq. 10); $K_* \in \mathbb{R}^N$ denotes the covariance of the test datapoint $\boldsymbol{v}^*$ with the existing (training) input measurements; and, lastly, $k(\boldsymbol{v}^*, \boldsymbol{v}^*) \in \mathbb{R}$ represents the variance of the test datapoint.

Furthermore, as GPs operate through Bayesian reasoning, by *conditioning* the joint prior distribution (Eq. 13) upon the observed dataset $\mathcal{D}$ and the test point $\boldsymbol{\upsilon}^*$, we obtain a predictive posterior normal distribution, with mean μ_j and variance σ_j^2 as follows:

$$\begin{aligned}\mu_j(\boldsymbol{\upsilon}^*) &= K_*^T(K+\sigma_n^2 I_N)^{-1}\mathbf{Y}_j^T \\ \sigma_j^2(\boldsymbol{\upsilon}^*) &= k(\boldsymbol{\upsilon}^*,\boldsymbol{\upsilon}^*) - K_*^T(K+\sigma_n^2 I_N)^{-1}K_*\end{aligned} \tag{14}$$

In the context of dynamical systems modelling Eq. 14 represents a probability model over the next state of the dynamical system at the next discrete time index. The construction of a posterior probability function is particularly useful in engineering applications, given that it expresses elements of both aleotoric and epsitemic model uncertainty. Typically, the mean is taken as the model's prediction, however, prediction may also be directly sampled from posterior distribution [37]. This will be discussed further in section 3.1.3.

Until now, we have formalised the construction of the GP prior and the posterior predictive distributions, both of which are dependent upon the tuning of hyperparameters associated with the prior $\hat{\boldsymbol{\lambda}} = [\boldsymbol{\eta}^T, \sigma_f^2, \sigma_n^2] \in \mathbb{R}^{n_\upsilon+2}$. As usual, hyperparameters are estimated by maximising the marginal log-likelihood of the data $\log p(\mathbf{Y}_j^T|\boldsymbol{\Upsilon}, \hat{\lambda})$. This is discussed in Appendix A.1.1 and referred to as GP training.

Thus far, the methodology has formalised the construction of GPs, and defined them as multiple-input, single-output models. In the following section, the methodology details the construction of state space models with GPs to approximate Eq. 1. This has been presented previously by other related works [36, 39, 40] and we direct the interested reader for more information.

3.1.2. Gaussian Processes for State Space Modelling

In this study, state space models are constructed by training n_x GP models separately and combining them to simultaneously predict the state vector $\mathbf{x} \in \mathbb{R}^{n_x}$ at the next discrete time interval, $t+1$. Specifically, under the assumption that each of the n_x models has been constructed and trained according to Section 3.1.1 and Appendix A.1.1, this implies that the the posterior prediction from the GP state space model, when presented with $\boldsymbol{\upsilon}_t$ follows:

$$\begin{aligned}\boldsymbol{\mu}(\boldsymbol{\upsilon}_t;\mathcal{D}) &= \left[\mu_1(\boldsymbol{\upsilon}_t), \dots, \mu_{n_x}(\boldsymbol{\upsilon}_t)\right] \\ \boldsymbol{\Sigma}(\boldsymbol{\upsilon}_t;\mathcal{D}) &= diag(\sigma_1^2(\boldsymbol{\upsilon}_t), \dots, \sigma_{n_x}^2(\boldsymbol{\upsilon}_t)) \\ \mathbf{x}_{t+1} &\sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})\end{aligned} \tag{15}$$

where $\boldsymbol{\mu} \in \mathbb{R}^{n_x}$ and $\boldsymbol{\Sigma} \in \mathbb{R}^{n_x \times n_x}$ amd $\mathbf{x}_{t+1} \in \mathbb{R}^{n_x}$ is the next state . In the following section, we discuss how the GP state space model is used to generate realisations of underlying process stochasticity, and relate discussion directly to the decision making process.

3.1.3. Gaussian Process Realisations and Decision Making

For effective and safe control and optimization of process systems, a control policy must consider worst-case realisations of process stochasticity. In GP models, function realisations are sampled from the GP. Each function realisation represents a specific instance of model uncertainty across process evolution - including the worst case. In order to achieve this, model uncertainties must be propagated correctly. This work implements the method detailed in [39], which recursively updates the dataset $\mathcal{D}$ as the process evolves between discrete time indices. This process is detailed by Algorithm 1 and relies upon linear algebra to account for the effects of conditioning the GP models on the updated dataset. These operations are detailed by Appendix A.1.2 and [36, 47].

Algorithm 1: Function Realisations via GP State Space Model for Decision-making Under Uncertainty

```
Initialise: Experimental dataset $\mathcal{D}$; GP state space model $f_{GPSS} = [f_{GP}^1(\boldsymbol{\upsilon}), \dots, f_{GP}^{n_x}(\boldsymbol{\upsilon})]$ with hyperparameters $\hat{\Lambda} = [\hat{\boldsymbol{\lambda}}_1, \dots, \hat{\boldsymbol{\lambda}}_{n_x}]$ trained on $\mathcal{D}$; Control Policy $\pi(\mathbf{u}|\mathbf{x})$; Finite horizon length T; initial state distribution $p(\mathbf{x}_0)$; Memory for state $\mathcal{B}_x$ and control $\mathcal{B}_u$ trajectories, as well as for information related to decision making $\mathcal{B}_\pi$ for use in subsequent policy optimization.
1. Set $\mathcal{D}_+ = \mathcal{D}$
2. Draw $\mathbf{x}_0 \sim p(\mathbf{x}_0)$. Append $\mathbf{x}_0$ to $\mathcal{B}_x$
3. for $t = 1, \dots, T-1$ do
    a. Observe $\mathbf{x}_{t-1}$, sample $\mathbf{u}_{t-1} \sim \pi(\mathbf{u}|\mathbf{x})$ and concatenate, such that $\boldsymbol{\upsilon}_{t-1} = \left[\mathbf{x}_{t-1}^T \mathbf{u}_{t-1}^T\right]^T$;
    b. Condition the GP state space model on $(\mathcal{D}_+, \boldsymbol{\upsilon}_{t-1})$ to obtain the predictive posterior: $p(\mathbf{x}_t|\boldsymbol{\upsilon}_{t-1}, \mathcal{D}_+) = \mathcal{N}(\boldsymbol{\mu}(\boldsymbol{\upsilon}_{t-1}; \mathcal{D}_+), \boldsymbol{\Sigma}(\boldsymbol{\upsilon}_{t-1}; \mathcal{D}_+))$. See Appendix A.1.2 for explicit detail;
    c. Draw next state from posterior of the GP state space model, $\mathbf{x}_t \sim p(\mathbf{x}_t|\boldsymbol{\upsilon}_{t-1}, \mathcal{D}_+)$ ;
    d. Update $\mathcal{D}_+ = [\mathcal{D}_+^T \; d_{N+t}^T]^T$, where $d_{N+t} = [\boldsymbol{\upsilon}_{t-1}^T \; \mathbf{x}_t^T]$ and append $\mathbf{x}_t$ and $\mathbf{u}_{t-1}$ to $\mathcal{B}_x$ and $\mathcal{B}_u$, respectively;
Output: Function realisation stored in $\mathcal{B}_x$ and $\mathcal{B}_u$ and information related to decision making $\mathcal{B}_\pi$ (to be explained in Algorithm 2)
```

The use of Algorithm 1 allows for proper propagation of model uncertainty and sampling of functions from the GP. In essence, it is desired to obtain state sequences $\mathbf{x}_{0:T} = [\mathbf{x}_0, \dots, \mathbf{x}_T]$, which are expressive of Eq. 1 and an associated realisation of potential process uncertainty. As samples $\mathbf{x}_t$ are drawn from the posterior they are added, along with the respective input $\boldsymbol{\upsilon}_{t-1}$, to the dataset $\mathcal{D}_+$ upon which the GP is conditioned. This leads to a subsequent update of the GP posterior distribution (via Appendix A.1.2) considering previous samples $\mathbf{x}_{t-1}$ as noiseless observations, with retention of the original covariance function hyperparameters $\hat{\boldsymbol{\lambda}}$. This means that if the updated GP posterior were to be queried at the previous input $\boldsymbol{\upsilon}_{t-1}$, the exact realisation of $\mathbf{x}_{t-1}$ would be drawn again i.e. the GP would express $\mathbf{x}_{t-1}$ deterministically. Such an outcome highlights the Algorithm's utility in effective function space sampling and implies that future process evolution is explicitly dependent upon the past realisations of uncertainty. For further discussion the interested reader is directed to [48, 36, 49, 39]. In the following section, an approach that synchronously combines concepts from sMPC and RL to produce a self-optimizing, policy varying reward shaping mechanism is presented, which ensures probabilistic constraint satisfaction. Specifically, penalty function methods are combined with the concept of backoffs.

3.2. Safe Chance Constrained Policy Optimization with Gaussian Processes

In this section, we provide details of the methodology, which enables combination of GP state space models with RL-based policy optimization for high probability constraint satisfaction. To achieve this, the methodology is organised as follows and the full algorithm is detailed by Algorithm 3:

1. In Section 3.2.1, the general stochastic optimal control problem defined by Eq. 6 is modified to consider the nominal evolution of the states and obtain a deterministic expression for the

probabilistic joint constraints. To facilitate this, we implement an approach similar to [25] in combination with a GP state space model.

2. In Section 3.2.2 the deterministic surrogate constraints are incorporated into a reformulation of the RL objective (see Eq. 4) via an l_p penalty function[2] [50, 51] and detail of a general constrained policy optimization algorithm is provided [52].
3. Then, in Section 3.2.3, an 'efficient' global optimization strategy [53] is presented to iteratively tune the penalty function enabling satisfaction of the original joint chance constraints with the desired probability $1-\alpha$ establishing a strong connection between this work and reward shaping [42].

The methodology is formalized with view to the use of policy optimization methods in the following, however, the concepts discussed can also be integrated into actor-critic and action-value methods [7].

3.2.1. Probabilistic Joint Chance Constraints

In this section, reformulation of the probabilistic joint chance constraint detailed by Eq.6 is presented. The joint chance constraints are restated here for ease:

$$\mathbb{P}(\bigcap_{i=0}^{T}\{\mathbf{x}_i \in \hat{\mathbb{X}}_i\}) \geq 1-\alpha \tag{16}$$

The following analysis proceeds to obtain a set of deterministic surrogate constraints, which can then be integrated into a revised objective for RL-based policy optimization. In particular, we leverage Boole's inequality and the Cantelli-Chebyshev inequality to obtain a deterministic constraint for each of those that comprise the original joint constraint. The analysis follows [54, 55].

Lemma 1. ***Boole's Inequality*** *[56]: Consider a finite set of countable events $\{Z_1, Z_2, \dots, Z_{n_g}\}$, the probability that one of these events occurs is no greater than the sum of the probabilities of the individual events:*

$$\mathbb{P}\Big(\bigcup_{i=1}^{n_g} Z_i\Big) \leq \sum_{i=1}^{n_g}\mathbb{P}(Z_i) \tag{17}$$

Now, considering n_g constraints comprise the joint chance constraint, then applying this result enables decomposition of Eq. 16, into n_g individual chance constraints. As in [25], for ease of notation, we define the following:

$$X = \max_{(t,j)\in\{0,\dots,T\}\times\{1,\dots,n_g\}} A_j\mathbf{x}_t - b_j, \qquad g = \{\mathbf{x}\in\mathbb{R}^{n_x} : X\}, \qquad \mathbb{G}_j^{'} = \bigcap_{i=0}^{T}\{\mathbf{x}_i \notin \mathbb{G}_{j,i}\}$$

where $\mathbb{G}_j^{'}$ defines the set of states, which do not satisfy constraint j for all time indices and $X\in\mathbb{R}^{n_x}$ defines a random variable. From Lemma 1:

$$\mathbb{P}\Big(\bigcup_{j=1}^{n_g}\{g\subset\mathbb{G}_j^{'}\}\Big) \leq \sum_{i=1}^{n_g}\mathbb{P}(g\subset\mathbb{G}_j^{'}) \tag{18}$$

[2]The subscript p of l_p denotes the norm incorporated into the penalty function

Explicitly, Eq. 18, dictates that the probability of achieving joint constraint satisfaction under a given policy π is lower bounded by the probability of satisfying each of the respective constraints individually. Therefore, guaranteeing satisfaction of chance constraints individually can be considered a robust approximation to joint satisfaction:

$$\iota_j = \mathbb{P}(g \subset \mathbb{G}_j^{'}) \implies \alpha \leq \sum_{j=1}^{n_g} \iota_j$$

In theory, $\iota_j \in \mathbb{R}$, subject to satisfying Eq. 18. This enables approximation of Eq. 16 via the following:

$$\sum_{j=1}^{n_g} \mathbb{P}\big(\bigcap_{i=0}^{T} \{\mathbf{x}_i \in \mathbb{G}_{j,i}\}\big) = 1 - \sum_{j=1}^{n_g} \iota_j \tag{19}$$

In this work, we define $\iota_j = \alpha/n_g,\ j = [1, \ldots, n_g]$. Having decomposed the original joint chance constraint into a set of individual chance constraints, the methodology looks to express a set of deterministic surrogate expressions (of the original probabilistic chance constraints), which can then be incorporated into the method presented.

To proceed, we deploy the concept of *constraint tightening*, which is an approach commonly deployed within the domain of sMPC. The intuition behind constraint tightening is described as follows. The process of concern is subject to unbounded uncertainties. We consider that under a given policy π, the process will vary probabilistically within a given region of $\hat{\mathbb{X}}_t$. Specifically, one can assume that the process will vary within some euclidean distance from the nominal or expected behaviour with a given probability. If we *back* the nominal process *off* from the constraint boundary then we will be able to achieve chance constraint satisfaction with the desired probability. This is underpinned by the Cantelli-Chebyshev inequality, which is described by Lemma 2

Lemma 2. ***Cantelli-Chebyshev Inequality*** *[57]: Consider a random variable Z, with expected value $\mathbb{E}[Z]$ and finite variance $\Sigma[Z]$, then:*

$$\mathbb{P}\big(Z - \mathbb{E}[Z] \geq \delta\big) \leq \frac{\Sigma[Z]}{\Sigma[Z] + \delta^2}$$

The mechanism of constraint tightening takes the form of a set of *backoffs* $\boldsymbol{\varepsilon}_j = [\varepsilon_{j,0}, \ldots, \varepsilon_{j,T}]$, which can be conceptualised as the necessary euclidean distance from the expected or nominal state $\bar{\mathbf{x}}_t \in \mathbb{R}^{n_x}$ to the constraint boundary to guarantee chance satisfaction with a given probability (note, backoff values are specific to both the constraint and time index). As stated in Section 2, the analysis provided in this work assumes affine constraints. Therefore, the tightened constraint sets follow:

$$\begin{aligned} \bar{\mathbb{G}}_{j,t} &= \{\bar{\mathbf{x}}_t \in \mathbb{R}^{n_x} : A_j^T \bar{\mathbf{x}}_t + \varepsilon_{j,t} - b_j \leq 0\} \\ \bar{\mathbb{X}}_t &= \{\bar{\mathbf{x}}_t \in \bar{\mathbb{G}}_{j,t}, \forall j = \{1, \ldots, n_g\}\} \end{aligned} \tag{20}$$

The determination of the backoff values $\varepsilon_{j,t}$ is handled via the following analysis. Specifically, we work from the developments made in [55, 58], which (via Lemma 2) show that the Cantelli-Chebyshev approximation of the backoff set is equivalent to:

$$\varepsilon_{j,t} = \sqrt{\frac{1-\iota_j}{\iota_j}} \sqrt{A_j^T \Sigma[\mathbf{x}_t] A_j} \tag{21}$$

where $\boldsymbol{\varepsilon}_{j,t}$ represents a robust approximation of the backoff required to ensure individual chance constraint satisfaction with the desired probability ι_j. In this work, we deploy a GP state space model to estimate both the nominal state $\bar{\mathbf{x}} \in \mathbb{R}^{n_x}$ and the variance of the state $\Sigma[\mathbf{x}_t]$, as described by Eq. 15, enabling construction of a deterministic expression for each of the individual chance constraints. In practice, it is well documented that use of the Cantelli-Chebyshev approximation leads to overly-conservative control policies, which operate far from the constraint boundary. In order to balance the performance of the control trajectory, with constraint satisfaction, we propose to tune $\varepsilon_{j,t}$ via a multiplying factor $\xi_j = [0, 1]$ for each constraint. As such, the deterministic surrogate for each of the individual chance constraints, detailed by Eq.19, are described:

$$A_j^T \bar{\mathbf{x}}_t + \xi_j \sqrt{\frac{1-\iota_j}{\iota_j}} \sqrt{A_j^T \Sigma[\mathbf{x}_t] A_j} - b_j \leq 0 \tag{22}$$

The approach to tuning of the multiplying factors, $\boldsymbol{\xi} = [\xi_1, \ldots, \xi_{n_g}]$, could be handled via Bayesian optimization (BO) or bisection [25, 36]. This work employs a BO strategy, which is detailed by Section 3.2.3.

The use of a GP to parameterise the backoff values $\boldsymbol{\varepsilon}_j$ is more efficient than the set of methods proposed previously by [25, 26]. In those works, initial *backoff* values were estimated via MC sampling and then tuned. Here, we provide a method to analytically express the backoff values via the posterior predictive distribution of the GP state space model, removing the requirement for sampling and the potential inaccuracies it brings in initialisation.

The methodology has now obtained a set of deterministic surrogate constraints to ensure joint chance constraint satisfaction. Identification of these expressions enables reformulation of the original problem statement $\mathcal{P}(\cdot)$ described by Eq.6 as follows:

$$\hat{\mathcal{P}}(\pi_C) := \begin{cases} \max\limits_{\pi} J \\ \text{s.t.} \\ \mathbf{x}_0 \sim p(\mathbf{x}_0) \\ \mathbf{x}_{t+1} \sim \mathcal{N}(\boldsymbol{\mu}(\boldsymbol{v}_t), \Sigma(\boldsymbol{v}_t)) \\ \mathbf{u}_t \sim \pi(\mathbf{u}_t|\mathbf{x}_t) \\ \mathbf{u}_t \in \hat{\mathbb{U}} \\ \mathbf{x}_t \in \bar{\mathbb{X}}_t \\ \forall t \in \{0, ..., T-1\} \end{cases} \tag{23}$$

where $\boldsymbol{v}_t = \left[\mathbf{x}_t^T \; \mathbf{u}_t^T\right]^T$ and solution to $\hat{\mathcal{P}}(\cdot)$ is equivalent to that of the original $\mathcal{P}(\cdot)$. Due to the presence of a GP state space model within the problem description, $\hat{\mathcal{P}}$ is a function space optimization problem. Previous works have solved this problem via nonlinear MPC with precalculation of the backoff values, and description of the discrete time state evolution according to the mean of the GP (equivalent to the nominal process) [36]. In this work, we use RL to solve $\hat{\mathcal{P}}$ (hence the use of function realisations and Algorithm 1) with incorporation of the deterministic surrogate constraints into a modified RL objective. This is achieved via an l_p penalty function, under a given value of the backoff multipliers, $\boldsymbol{\xi}$. Solution to this problem under the optimal backoff multipliers $\boldsymbol{\xi}^*$ is deemed equivalent to finding solution to Eq. 23 as discussed subsequently.

3.2.2. Safe Constrained Policy Optimization with Fixed Backoffs

As GPs express process uncertainties, they present an avenue to synthesise policies, which only exploit regions of the state space in which the model is confident of the true process behaviour i.e. where epistemic uncertainties are low. By incorporating the variance prediction, $\Sigma(\boldsymbol{\upsilon}_t)$, of the GP state space model posterior directly in the RL performance index, we force the ultimate RL policy to avoid the areas that the GP is uncertain and provide explicit mechanism to mitigate exploitation of the mathematical nature of the GP model. This way the policy implicitly *accounts for the limitations of the data-driven model* when deployed to the real process.

Use of the l_1 or l_2 penalty functions is particularly appealing because of the exactness (under certain conditions) to the solution of $\hat{\mathcal{P}}$ [50]. This would further preserve the approximation provided by Eq. 23 to Eq. 6. The general penalty function, $\boldsymbol{\varphi}_p : \mathbb{X} \times \mathbb{U} \times \mathbb{X} \to \mathbb{R}$ is detailed as follows:

$$\boldsymbol{\varphi}_p(\mathbf{x}, \mathbf{u}, t) = R_{t+1} - \mathbf{tr}\big(\zeta \Sigma[\boldsymbol{\upsilon}_t]\big) - \kappa \left\| [A^T \mathbf{x}_{t+1} + \boldsymbol{\varepsilon}_t - b]^- \right\|_p \tag{24}$$

where $A \in \mathbb{R}^{n_x \times n_g}$ and $b \in \mathbb{R}^{n_g}$ define the set of inequality constraints; $\boldsymbol{\varepsilon}_t \in \mathbb{R}^{n_g}$ the set of backoff values relevant to the set of constraints at a given time index (see Eq. 22); $[\mathbf{z}]^- = \max(0, \mathbf{z})$ defines an element wise operation over $\mathbf{z} \in \mathbb{R}^{n_g}$; $\|\cdot\|_p$ the general p-norm; $R_{t+1} \in \mathbb{R}$ the rewards accumulated under the original process objective e.g. productivity maximisation in a (bio)chemical process; and, $\kappa \in \mathbb{R}$ and $\zeta \in \mathbb{R}^{n_x \times n_x}$ (a diagonal matrix) weight the penalty for constraint violation and model uncertainty, respectively - relative to R_{t+1}. The incorporation of model uncertainty, therefore, is represented by the term $\mathbf{tr}\big(\zeta \Sigma[\boldsymbol{\upsilon}_t]\big)$. Expression of the penalty function, enables redefinition of the RL objective $J(\boldsymbol{\tau})$ via $\bar{J}_C(\boldsymbol{\tau})$:

$$\begin{aligned} \bar{G}_C(\boldsymbol{\tau}) &= \sum_{t=0}^{T-1} \gamma^t \boldsymbol{\varphi}_p(\mathbf{x}, \mathbf{u}, t) \\ \bar{J}_C &= \int p(\boldsymbol{\tau}) \bar{G}_C(\boldsymbol{\tau}) d\boldsymbol{\tau} \end{aligned} \tag{25}$$

It is hypothesised that RL-based optimization of this new objective will synthesise a policy, which ensures chance constraint satisfaction and exploits regions of the state space well characterised by the model - encouraging the learning of inherently safe control policies. Further, because the modifications are made directly to the reward function itself, the approach is compatible with any RL method. Given the GP state space model is constructed over continuous state and control variables, as usual, the RL can learn a parameterisation of the optimal constrained policy:

$$\begin{aligned} \pi_C^*(\mathbf{u}|\mathbf{x}; \theta, \cdot) &\approx \pi_C^*(\mathbf{u}|\mathbf{x}) \\ \pi_C^*(\cdot, \theta) &= \arg\max_\theta \bar{J}_C \end{aligned} \tag{26}$$

where $\theta \in \mathbb{R}^{n_\theta}$ denotes a vector representation of the policy parameters (typically the weights and bias of a neural network). A general algorithm for constrained policy optimization under a fixed set of backoff values is provided by Algorithm 2. These backoffs are adjusted via BO, details are presented later in the manuscript in Section 3.2.3 and Algorithm 3.

```
Algorithm 2: Safe Policy Optimization for Fixed Backoffs
  Initialise: Experimental dataset $\mathcal{D}$; GP state space model $f_{GPSS} = [f_{GP}^1(\boldsymbol{\upsilon}), \dots, f_{GP}^{n_x}(\boldsymbol{\upsilon})]$ with hyperparameters $\hat{\Lambda} = [\hat{\boldsymbol{\lambda}}_1, \dots, \hat{\boldsymbol{\lambda}}_{n_x}]$ trained on $\mathcal{D}$; Initial control policy $\pi(\mathbf{u}|\mathbf{x};\theta_0)$; Policy optimization algorithm $f_{PO}(\cdot)$; backoff multipliers $\boldsymbol{\xi}$; Finite horizon length T; initial state distribution $p(\mathbf{x}_0)$; Memory $\mathcal{B}_{info}$ for information required for $f_{PO}(\cdot)$; $K$ episodes; tolerance criterion
  1. $i = 0$
  2. while not converged do
        a. Obtain a batch of $K$ rollouts over horizon of $T$ discrete intervals according to Algorithm 1, via $\pi(\mathbf{u}|\mathbf{x};\theta_i)$, $f_{GPSS}$, and $p(\mathbf{x}_0)$. Return trajectory information[3] corresponding to each rollout and store in $\mathcal{B}_{info}$. Return any further necessary information for $f_{PO}(\cdot)$ via $\mathcal{B}_x, \mathcal{B}_u$ and $\mathcal{B}_\pi$, and add to $\mathcal{B}_{info}$;
        b. Perform policy optimization $\theta_{i+1} = f_{PO}(\mathcal{B}_{info}, \theta_i)$;
        c. Reset memory $\mathcal{B}_{info}$;
        d. i += 1;
        e. Assess tolerance criterion;
     end
  3. Assess final policy performance $J(\theta_i)$ under the unconstrained reward function $R$ and Eq. 4 and approximate the probability of joint constraint violation, Eq. 16, denoted $F_{LB}(0)$ via the method discussed in Section 3.2.3 and detailed in Appendix A.3
  Output: Optimal constrained policy $\pi_C^*(\mathbf{u}|\mathbf{x};\theta_i)$ under backoff multipliers $\boldsymbol{\xi}$ and associated performance indices $J(\theta_i)$ and $F_{LB}(0)$
```

The description provided by Algorithm 2 considers all *on-policy* policy optimization approaches, denoted generally as $f_{PO}(\cdot)$, although there is no reason the approach could not utilise an off-policy method too [26]. The detail provided formalises the process of obtaining function space realisations from the GP state space model f_{GPSS}, each of which represents a potential instance of the uncertain process detailed by Eq. 1. Every process trajectory is ranked according to Eq. 24 and the current iterate backoff multiplier $\boldsymbol{\xi}$ values. Using the collected experience (including relevant information that describes decision making), stored in the memory $\mathcal{B}_{info}$, the weights of the policy are updated by $f_{PO}(\cdot)$. This is repeated until a convergence criterion is satisfied. In the following computational experiments detailed by this work, *the methodology was integrated with the proximal policy optimization (PPO) algorithm* [52]. This is an attractive option given: a) the ability to directly parameterise a policy as a conditional probability distribution over a *continuous* control input space; b) compatibility with recurrent neural networks [52]; c) sample efficiency relative to conventional policy optimization methods i.e. *reinforce*; and, d) ease of implementation. Full detail of the PPO algorithm is provided by Appendix A.2.

In this section, the methodology has provided mechanism to incorporate information about the

[3]This includes the rewards $\varphi_{0:T-1}^{(k)} = [\varphi_1^{(k)}, \dots, \varphi_{T-1}^{(k)}]$ under Eq. 24 and the current backoff multipliers $\boldsymbol{\xi}$, for the sequence of controls $\mathbf{u}_{0:T-1}^{(k)} = [\mathbf{u}_1^{(k)}, \dots, \mathbf{u}_{T-1}^{(k)}]$ and states $\mathbf{x}_{0:T}^{(k)} = [\mathbf{x}_1^{(k)}, \dots, \mathbf{x}_T^{(k)}]$

constrained problem into the reward signal characteristic of the MDP. In doing so, a strong connection to reward shaping is established. In reward shaping, policy invariant modifications of the reward function are identified to aid learning of the optimal policy π^* [42, 59]. In this work, we construct a policy varying modification of the reward function in order to satisfy operational constraints. The resultant penalty function (Eq. 24) contains a number of free parameters. In the subsequent section, it is proposed to tune the backoffs, $\boldsymbol{\varepsilon}$, via the multipliers, $\boldsymbol{\xi}$, (see Eq. 22) and a BO scheme. This leaves decision as to the parameters $\kappa \in \mathbb{R}$ and $\zeta \in \mathbb{R}$ open to the implementation, although it is recommended that they are large [50]. Ultimately this provides mechanism for the implementation to balance operational risk and performance manually.

3.2.3. Optimization of Backoff Multipliers

The primary objective of this section is to identify a mechanism which facilitates synthesis of a policy that:

(a) achieves high probability constraint satisfaction as desired, and

(b) performs with respect to the original process objective as specified by $R : \mathbb{X} \times \mathbb{U} \times \mathbb{X} \rightarrow \mathbb{R}$.

'Efficient' global optimization of the backoff multipliers $\boldsymbol{\xi}$ is proposed, and so the methodology explores definition of an objective to evaluate candidate values of $\boldsymbol{\xi}$ as follows.

Firstly, discussion is directed in how best to evaluate a) from the policy generated by Algorithm 2. Specifically, with reference to Eq. 18 and the following works [54, 25]:

$$F_X(0) = \mathbb{P}(X \leq 0) = \mathbb{P}(\bigcap_{i=0}^{T} \{\mathbf{x}_i \in \hat{\mathbb{X}}_i\}) \tag{27}$$

where $F_X(\cdot)$ indicates the cumulative distribution function (cdf), which in this case is analytically intractable. In order to assess a), it is proposed to validate the probability of constraint satisfaction empirically via MC sampling under the GP state space model i.e. via the sample approximation of $F_X(0)$, denoted $F_{SA}(0)$. The specific approach is detailed in [25] and repeated in Appendix A.3 for completeness. Ultimately, through this sampling-based method, a lower bound for Eq. 27 and $F_{SA}(0)$ is obtained and denoted $F_{LB}(0)$. This accounts for potential inexactness introduced through finite samples. The evaluation of b) is more simple and directed via the definition of the process objective in the form of the reward function R. As such, the investigation may evaluate the performance of the policy under the original, unconstrained objective provided by Eq. 4. This work therefore proposes the use of the following objective function in evaluation of candidate multiplier values $\boldsymbol{\xi} \in \mathbb{R}^{n_g}$:

$$\begin{aligned} \boldsymbol{U} &= (F_{LB}(0) - (1-\alpha))^2 \\ J_{BO} &= -(J(\boldsymbol{\tau}) - \beta\sigma_J)\exp(-c\boldsymbol{U}) \end{aligned} \tag{28}$$

where $\beta = [0, 1]$, $c \in \mathbb{R}^+$ and σ_J denotes the standard deviation of the policy with respect to the unconstrained process objective. This is a modification to the objective function proposed previously in [25], which equated J_{BO} to $\boldsymbol{U}$. Here, Eq. 28, provides a smoother latent function and naturally balances the objectives a) and b). The inclusion of the term σ_J also incentivises those policies, which exploit regions of the state space well characterised by the model. The factor c provides a shape parameter for the RBF part of the objective, with higher values providing greater incentive to obtain joint constraint satisfaction as desired. Care should be taken in selection as the higher the value, the sparser the mapping provided by the objective. This is likely to have consequences for the efficacy of optimization.

In the following case studies, $\beta = 0.1, c = 1$ and a BO scheme was deployed via GP surrogate models with RBF covariance functions and zero mean priors to optimize the backoff multipliers $\boldsymbol{\xi}$:

$$\boldsymbol{\xi}^* = \arg\min_{\boldsymbol{\xi}} J_{BO} \tag{29}$$

Due to the expensive black box optimization proposed BO is deemed the most appropriate approach. BO proceeds to construct and exploit a GP surrogate model to sample new candidate points. Construction of the GP surrogate demands a small initial dataset, describing a set of inputs, $\boldsymbol{\Xi}$, and their corresponding fulfilment of the objective function, $J_{\boldsymbol{\Xi}}$. New sampling points (or in this case, candidate backoff multipliers) are sampled to maximise an acquisition function (AF), which is a function of the posterior distribution of the GP surrogate. The AF, denoted $f_{AF}(\cdot)$, used in this work was the expected improvement (EI) function [60, 53]. It was found that the EI AF, $f_{AF}^{EI}(\cdot)$, was most efficient in this case, balancing exploration and exploitation of the GP surrogate model to find the optimal solution. The expected improvement function is detailed as follows [53, 61]:

$$\varrho = \frac{\mu(\boldsymbol{\xi}) - J_{BO}^{+}}{\sigma(\boldsymbol{\xi})}$$

$$f_{AF}^{EI}(\boldsymbol{\xi}) = \begin{cases} \sigma(\boldsymbol{\xi})\psi(\varrho) + (\mu(\boldsymbol{\xi}) - J_{BO}^{+})\phi(\varrho), & \text{if } \sigma(\boldsymbol{\xi}) > 0 \\ 0, & \text{otherwise} \end{cases} \tag{30}$$

where $\phi(\cdot)$ is the Gaussian cumulative distribution function, $\psi(\cdot)$ is the Gaussian probability density function, J_{BO}^{+} is the objective value of the current best backoff multiplier values $\boldsymbol{\xi}^{+}$ [53, 60], and $\mu(\cdot)$ and $\sigma(\cdot)$ are detailed by Eq. 14. Dissecting Eq. 30, the first term on the right hand side incentivises exploring regions of the input space associated with high uncertainty in the posterior distribution, and the second term provides basis to exploit regions of the input space corresponding to high mean predictions in the posterior [53]. As such, Eq. 30 provides explicit mechanism to balance exploration and exploitation of the GP surrogate model, in a fashion not dissimilar to the exploration-exploitation paradigm in RL. For more detail on BO in this context, we direct the interested reader to previous work [25, 38] and a comprehensive review [60].

Algorithm 3 formalises the approach to reward shaping detailed by this Section. In **Step 1**, an optimal policy parameterisation is learned for the unconstrained problem. This is used as an initialisation for learning of the optimal constrained policy thereafter. In **Step 2.a**, a number of policies are learned for the constrained problem each utilising different values of the backoff multipliers. In **2.b**, each of the policies is assessed with respect to J_{BO}, providing an input-output dataset, where the inputs are backoff multipliers and the outputs are corresponding performances under the objective (J_{BO}). In **Step 3**, a surrogate GP model is built via this input-output dataset for subsequent BO. Pseudocode for BO is provided by **Step 4**, with **Step 5** documenting the rollout of the utlimate constrained policy on the real process. As data is collected from the process, the GP state space model can be further improved and the policy retrained from batch to batch. This is encapsulated by **Step 6**.

```
Algorithm 3: Safe Chance Constrained Policy Optimization

  Initialisation: Desired probability of joint chance constraint satisfaction $\alpha$; GP prior for Bayesian Optimization $f_{BO}$; Acquisition function $f_{AF}$; Objective function $J_{BO}$; maximum number of acquisitions for BO $M$; Initial set of $B$ backoff multiplier values $\boldsymbol{\Xi} = [\boldsymbol{\xi}_1, \dots, \boldsymbol{\xi}_B]$ generated via sobol sequence [62]; Trained GP state space model $f_{GPSS}$ and corresponding dataset $\mathcal{D}$; Initial state distribution $p(\mathbf{x}_0)$; Initial policy parameters $\pi(\mathbf{u}|\mathbf{x}, \theta_0)$; Policy optimization algorithm $f_{PO}$; $K$ episodes (function realisations) for each iteration of policy optimization; finite control horizon $T$;
  1. Perform policy optimization for unconstrained problem to learn the optimal unconstrained policy $\pi^*(\cdot, \theta)$ to maximise Eq. 4 via modification to Algorithm 2 and $f_{PO}$, $f_{GPSS}$, $\mathcal{D}$, and $p(\mathbf{x}_0)$. Return policy $\pi^*(\cdot, \theta)$.
  2.a. Train a set of $B$ constrained policies $\pi^*_{init} = [\pi^*_C(\cdot, \theta_1), \dots, \pi^*_C(\cdot, \theta_B)]$ via Algorithm 2 with $f_{PO}$, $f_{GPSS}$, $\mathcal{D}$, $p(\mathbf{x}_0)$, and $\pi^*(\cdot, \theta)$ for initialisation under the backoff values $\boldsymbol{\Xi}$.
  2.b. Return performance indices $F_{LB}(0)$ and $J(\boldsymbol{\tau}, \theta)$ $\forall$ $\pi^*_C(\cdot, \theta) \in \pi^*_{init}$ and assess $J_{BO}$, such that $J_\Xi = [J_{BO}(\boldsymbol{\xi}_1), \dots, J_{BO}(\boldsymbol{\xi}_B)]$.
  3. Train GP model given input-output mappings (of backoffs to constraint violation) provided by $\boldsymbol{\Xi}$ and $J_{\boldsymbol{\Xi}}$ according to Appendix A.1.1 and condition $f_{GP}$ to obtain updated predictive posterior distribution $p(J_{BO}|\boldsymbol{\xi}, \boldsymbol{\Xi})$.
  4. for $m$ = 1, ..., $M$ do
      a. According to $p(J_{BO}|\boldsymbol{\xi}, \boldsymbol{\Xi})$ find $\boldsymbol{\xi}_{B+m} = \arg\max_{\boldsymbol{\xi}} f_{AF}(\cdot)$ and update $\boldsymbol{\Xi} = [\boldsymbol{\xi}_1, \dots, \boldsymbol{\xi}_{B+m}]$
      b. Train constrained policy $\pi^*_C(\cdot, \theta_{B+m})$ via Algorithm 2 as proposed with $f_{PO}$, $f_{GPSS}$, $\mathcal{D}$, $p(\mathbf{x}_0)$, and $\pi^*(\cdot, \theta)$ for initialisation under the backoff values $\boldsymbol{\xi}_{B+m}$. Return performance indices $F_{LB}(0)$ and $J(\boldsymbol{\tau}, \theta)$ for $\pi^*_C(\cdot, \theta_{B+m})$, assess $J_{BO}(\boldsymbol{\xi}_{B+m})$ and append to dataset $J_\Xi = [J_{BO}(\boldsymbol{\xi}_1), \dots, J_{BO}(\boldsymbol{\xi}_{B+m})]$
      c. if $m < M$: repeat step 3.
  5. Return $\pi^*_C(\theta)$ corresponding to $\max J_{\boldsymbol{\Xi}}$ and one rollout of $\pi^*_C(\theta)$ on the real process (see Eq. 1). Append collected data to $\mathcal{D}$ and update $f_{GPSS}$ via Appendix A.1.1.
  6. Repeat Steps 2 - 4 and return final $\pi^*_C(\theta)$.
  Output: Optimal Constrained Policy $\pi^*_C(\theta)$ trained under $\boldsymbol{\xi}^*$
```

In the following section, the method is demonstrated on a microalgal lutein photo-production dynamic process and benchmarked against dynamic optimization and MPC strategies.

4. Case Study

To demonstrate the methodology, a case study was selected from previous work conducted by [63, 64], which is underpinned by a set of ordinary differential equations (ODEs). The problem and standard benchmarks are detailed via the following subsections.

4.1. A Microalgal Lutein Photo-Production Dynamic Process

Fed-batch fermentation processes are thought to be ideal systems for RL-based controllers and particularly suited to data-driven approaches to control and optimization. This is due to characteristics of predominantly batch mode operation and complex physical phenomena driven by the metabolic reaction network. The complexity of the process physics often provides impediment to structural and practical model identification, with large parametric uncertainties common across bioprocess systems. To demonstrate the method proposed here, we consider an *in-silico* microalgal lutein photo-production process described as follows:

$$\begin{aligned}\dot{c}_X &= u_0 \frac{c_N}{c_N + K_N} c_X - u_d c_X \\ \dot{c}_N &= -Y_{N/X} u_0 \frac{c_N}{c_N + K_N} c_X + F_{N,_{in}} \\ \dot{c}_L &= k_0 \frac{c_N}{c_N + K_{NL}} c_X - k_d c_L c_X\end{aligned} \tag{31}$$

where c_X $(g\ L^{-1})$ defines the biomass concentration; c_N $(mg\ L^{-1})$ defines the nitrate concentration; c_L $(mg\ L^{-1})$ defines the lutein (product) concentration; $F_{N,_{in}}(mg\ h^{-1})$ is the nitrate inflow to the system (a control input); $u_0 \in \mathbb{R}$ (h^{-1}) is the specific biomass growth rate, which is a function of the incident light intensity $I_0 \in \mathbb{R}$ $(\mu mol\ m^{-2}\ s^{-1})$ to the reactor and the maximum theoretical growth rate $u_m \in \mathbb{R}$ (h^{-1}); $k_0 \in \mathbb{R}$ $(mg\ g^{-1}\ h^{-1})$ is the specific lutein production rate, which is a function of I_0 and the maximum theoretical production rate $k_m \in \mathbb{R}$ $(mg\ g^{-1}\ h^{-1})$; $k_d \in \mathbb{R}$ $(L\ g^{-1}\ h^{-1})$ is the lutein consumption rate; $u_d \in \mathbb{R}$ (h^{-1}) is the biomass specific decay rate; $Y_{N/X} \in \mathbb{R}$ $(mg\ g^{-1})$ is the nitrate yield coefficient; and, $K_N \in \mathbb{R}$ $(mg\ L^{-1})$ and $K_{NL} \in \mathbb{R}$ $(mg\ L^{-1})$ are the nitrate half-velocity constant for cell growth and lutein synthesis, respectively. The growth rates of biomass and lutein are constituted by the terms u_0 and k_0. These are both functions of the incident light intensity to the reactor and are detailed as follows:

$$\begin{aligned}u_0 &= \frac{u_m}{20} \sum_{n=1}^{9} \left(\frac{I_0}{I_0 + k_s + \frac{I_0^2}{k_i}} + 2 \frac{I_{\frac{nL}{10}}}{I_{\frac{nL}{10}} + k_s + \frac{I_{\frac{nL}{10}}^2}{k_i}} + \frac{I_L}{I_L + k_s + \frac{I_L^2}{k_i}} \right) \\ k_0 &= \frac{k_m}{20} \sum_{n=1}^{9} \left(\frac{I_0}{I_0 + k_{sL} + \frac{I_0^2}{k_{iL}}} + 2 \frac{I_{\frac{nL}{10}}}{I_{\frac{nL}{10}} + k_{sL} + \frac{I_{\frac{nL}{10}}^2}{k_{iL}}} + \frac{I_L}{I_L + k_{sL}} + \frac{I_L^2}{k_{iL}} \right)\end{aligned} \tag{32}$$

where $k_i \in \mathbb{R}$ and $k_{iL} \in \mathbb{R}$ are light inhibition terms for biomass growth and lutein synthesis, respectively. Similarly, $k_s \in \mathbb{R}$ and $k_{sL} \in \mathbb{R}$ are light saturation terms for biomass growth and lutein synthesis, respectively. More information on parameter definitions and values is provided by [64, 63]. The states $\mathbf{x} = [c_X, c_N, c_L]$ of the system are absolute and hence $c_i \geq 0\ \forall\ i \in \{X, N, L\}$. Further to $F_{N,_{in}}$, the control input is also constituted by I_0, such that $n_u = 2$ and $\mathbf{u}(t) = [F_{N,_{in}}, I_0]^T$, with bounds $0.1 \leq F_{N,_{in}} \leq 100$ $mg\ h^{-1}$ and $100 \leq I_0 \leq 1000$ $\mu mol\ m^{-2}\ s^{-1}$. It is assumed that the process is subject to stochasticity in the form of 5 % parametric uncertainty with $u_m \sim \mathcal{N}(0.152, 0.0038)$, $K_N \sim \mathcal{N}(30, 0.75)$, $u_d \sim \mathcal{N}(5.93 \times 10^{-3}, 1.483 \times 10^{-4})$, $Y_{N/X} \sim \mathcal{N}(305, 7.625)$, $k_m \sim \mathcal{N}(0.35, 0.00875)$ and $K_d \sim \mathcal{N}(3.71 \times 10^{-3}, 9.275 \times 10^{-5})$. Other values are assumed constant following the original works [64, 63]. The initial state distribution follows the work of [63] and is specified as $\mathbf{x}_0 \sim [\mathcal{N}(0.27, 3.125 \times 10^{-3}), \mathcal{N}(765.0, 9.5625), \mathcal{N}(0.0, 0.0)]$. The following constraints are specific to this work and not the previous. Here, affine constraints are defined using notation from Section 3.2.2:

$$A = \begin{bmatrix} 1 & 0 & -1.67 \\ 0 & -1 \times 10^{-3} & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad b = \begin{bmatrix} 2.6 \\ 0.15 \\ 0 \end{bmatrix} \tag{33}$$

The constraints were constructed to represent common operational concerns in bioprocessing. The first column of A considers the potential raw material to product conversion via constraint of the maximum biomass concentration (as biomass is a 'by-product'). The second column considers the protection of cell growth (via a minimum nitrate constraint) and the third ensures continued productivity (via constraint of the maximum ratio of secondary metabolite to biomass). The process objective reward function $R : \mathbb{X} \times \mathbb{U} \times \mathbb{X} \rightarrow R_{t+1}$ is as follows:

$$R_{t+1} = \begin{cases} \mathbf{d}^T \mathbf{x}_{t+1} - \Delta \mathbf{u}_t^T C \Delta \mathbf{u}_t & \text{if } t = T - 1 \\ -\Delta \mathbf{u}_t^T C \Delta \mathbf{u}_t, & \text{otherwise} \end{cases} \tag{34}$$

where $t = [0, \ldots, T]$ and the length of the finite horizon is defined $T = 6$; $\Delta \mathbf{u}_t = \mathbf{u}_t - \mathbf{u}_{t-1} \in \mathbb{R}^{n_u}$ defines the change of controls between discrete time steps; $C = diag([0.16, 8.1 \times 10^{-5}]) \in \mathbb{R}^{n_u \times n_u}$ provides a penalty for changing the controls and promotes the learning of 'stable' control profiles; and, $d = [0, -0.001, 4]^T \in \mathbb{R}^{n_x}$ provides an overall objective for process operation i.e. to maximise the production of lutein and minimise waste of nitrate. The problem definition is common to both Section 4.2 and the benchmark described in Section 4.3, except the benchmark does not consider any form of parametric uncertainty.

4.2. Safe Chance Constrained Policy Optimization

To demonstrate the methodology, this work deploys the PPO algorithm with both actor and critic recurrent long-short term memory (LSTM) neural network parameterisations. The actor network expresses a mapping between observed states and controls (i.e. a control policy) and the critic provides a mapping between a state and the value of that state under the policy (this is known as the value function). The use of a critic provides means to deploy the general advantage estimate (GAE) form of the policy gradient (PG) within the PPO framework. The GAE enables the implementation to manually balance the bias and variance of the advantage PG. This provides means to synchronously ensure stable learning, improve the sample efficiency of the algorithm and find a clear direction (in weight space) for policy improvement. Further, PPO acts to efficiently learn a policy by performing conservative policy updates. The conservative updates are performed by constraining the distance the policy is updated (in policy space) via a distance measure (e.g. the KL divergence). This shares strong conceptual similarities to *trust-region* optimization. In PPO, this trust-region constraint is implemented approximately either via a penalty method or by appropriate clipping of the update gradients. For more information, the reader is directed to Appendix A.2 and [65, 52]. The implementation utilised Pytorch 1.7.1. Information about the structure of the actor, critic and all hyperparameters defining the PPO algorithm as used in this work, may be found in Appendix A.2.3.

The factors in Eq. 24 are $\kappa = 34$ and $\zeta = 300 \times diag([1/\sigma^2_{1\Upsilon}, \ldots, 1/\sigma^2_{n_x\Upsilon}])$, where $\sigma^2_{n_x\Upsilon}$ represents the variance of the distribution of state x_{n_x} in the dataset $\mathcal{D}$. These factors were determined empirically (by analysing the relative magnitudes of each of the constituent features of Eq. 24) and provide balance between the objectives of safety and performance with respect to the unconstrained objective. The tolerance criterion for Algorithm 2 was $|\bar{J}_C(\boldsymbol{\tau}, \theta_i) - \bar{J}_C(\boldsymbol{\tau}, \theta_{i-1})| \leq 10^{-3}$ and the number of episodes, $K = 200$. The desired probability of joint chance constraint satisfaction was $1 - \alpha$, with $\alpha = 0.001$. The decomposition of the joint chance constraints to individual chance constraints dictates split of the probability of violation such that $\iota_j = \alpha / n_g,\ j = [1, \ldots, n_g]$.

In order to train the desired policy $\pi^*_C(\cdot, \theta)$ via Algorithm 3, a GP state space model is required. In this work, the model was built using an initial dataset $\mathcal{D}$, generated by simulation of the uncertain process' response to 32 different control sequences, $\mathbf{u}^{(j)}_{0:T},\ j = [1, \ldots, 32]$ (hence the dataset contains information from 32 separate batch experiments). Each control sequence was generated via transformation of a Sobol sequence (of length T) to the bounded controls space (as detailed in Section 4.1). In practice, this dataset could be generated via an initial design of experiments [33]. Having generated $\mathcal{D}$, ($n_x = 3$) individual GP models were constructed to form a state space model (for the prediction of each state) via the methodology outlined in Section 3.1. A prior distribution with mean function $\mathbf{m}(\upsilon) = 0$ and a matern 5/2 covariance function was specified for each of the constituent models. The covariance function was selected according to preliminary experiments, which examined the model's predictive accuracy. All GPs were constructed with the GPy 1.9.9 python package and subsequent BO utilised GPyOpt 1.2.6.

In the presentation of results for this work, the investigation is concerned with two main questions. Firstly, does Algorithm 3 enable identification of a reward function, which ensures policy performance with respect to the process objective and constraint satisfaction? And, secondly, does the incorporation of the posterior variance prediction of the GP state space model (into the reward function (Eq. 24)) provide means to minimise the risk of policy deployment (to the real uncertain process), by ensuring the policy exploits regions of the model with small model-process mismatch? These two questions will direct discussion in Section 5. All results were generated under view of the policy as deterministic i.e. $\mathbf{u}_t = \pi(\mathbf{x}_t)$. This was achieved through selection of the control corresponding to the mode of the conditional distribution $\pi(\mathbf{u}|\mathbf{x})$.

4.3. Benchmark for Process Optimization

The results from the proposed methodology were benchmarked relative to the control profiles generated from a) dynamic optimization (DO) strategies, and b) nonlinear model predictive control (NMPC). Both a) and b) use the process model detailed by Eq. 31. The deterministic form of this model (i.e. with no parametric uncertainty) represents the most accurate deterministic model, which may be built for process prediction and optimization. Therefore, the controls generated from a) and b) assume that the underlying process is deterministic, and are subsequently validated on the stochastic analogue of the process concerned. As both a) and b) neglect the existence of uncertainty over the parameter values assumed from [64], validation of the strategies on the stochastic variant of the process (detailed by Section 4.1) directly investigates the effects of uncertainty on performance with respect to the objective and constraint satisfaction. The control strategy for a) was generated offline through optimization of the control inputs to the model detailed in Eq. 31. Hence the control policy generated is deterministic and unconditional to online state observation. Conversely. the control strategy for b) was generated online through perfect state observation as in the RL case. Both benchmarks utilised the orthogonal collocation method and one finite element per control interval [66, 67] and the IPOPT solver [68]. This was facilitated by the Casadi 3.5.1 Python package [69]. In the case that a feasible solution could not be found online, the MPC scheme was tuned further with an approximate problem solved to minimise constraint violation. From empirical analysis, this tuning increased the performance of the NMPC scheme.

4.4. Key Performance Indicators

In the following section, this work will investigate the utility of the algorithm, and presentation of the results will focus on the ability of the proposed method to find a safe constrained policy $\pi^*_C(\cdot, \theta)$. Explicitly, the policy should exploit regions of the real process state space (i.e. Eq. 31), well characterised by the approximating process model (i.e. Eq. 15), therefore minimising mismatch between the state distributions simulated under the offline process model and observed under the real uncertain process. This will be demonstrated in two ways. First, via visual comparison as presented figuratively, and secondly via the quantitative metrics (key performance indicators) available to the investigation. Primarily, these metrics are the performance of the policy with respect to the unconstrained process objective $J(\tau)$ (see Eqs. 4 and 34) and the probability of joint chance satisfaction as evaluated by $F_{SA}(0)$ and $F_{LB}(0)$. The same metrics will be used to evaluate the performance of the benchmarks of DO and NMPC.

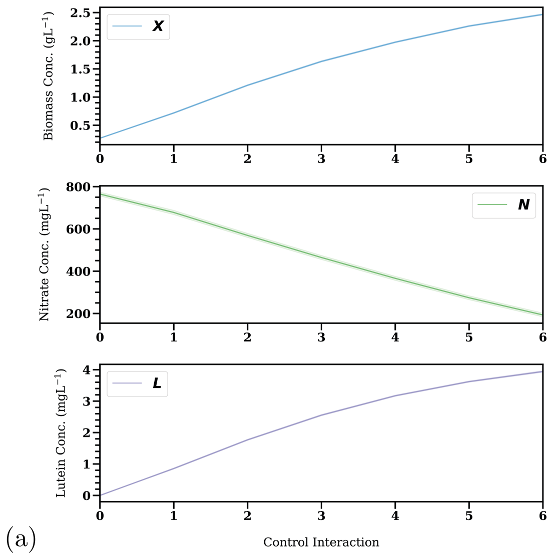
5. Results and Discussion

5.1. Results of Safe Chance Constrained Policy Optimization

Firstly, the results of Algorithm 3 with respect to the approximate offline state space model are displayed by Fig. 1. Explicitly, here, we demonstrate the performance of the final policy $\pi_C^*(\cdot, \theta)$ on the *GP state space model.* The results were obtained according to 500 function realisations of $\pi_C^*(\cdot, \theta)$ via Algorithm 1. Fig. 1 a) expresses a representation of the state evolution $\mathbf{x}_{0:T}$ and Fig. 1 b) provides a visualisation of the performance of the policy with respect to the constraints. In Fig. 1 a), the average state evolution and an associated confidence interval of one standard deviation for the validation trajectories is represented by a solid line and a shaded region, respectively. It can be seen that the agent learns to maximise the productivity objective - balancing maximisation of the lutein product at the end of the batch with a decrease in the concentration of nitrate left in the system. This is achieved in a manner that accounts for worst case process stochasticity by backing the nominal or expected state trajectory away from the constraint boundary. This is highlighted by Fig. 1 b). In particular, the shaded regions indicate 99% confidence intervals for process deviation and the dark blue solid line plot indicates the nominal process. Further, the utility of tuning the backoff multipliers via Algorithm 3 is highlighted given that the worst case realisations of process stochasticity do not violate, but approach the constraint boundary very closely. The performance of the policy $\pi_C^*(\cdot, \theta)$ with respect to both process objective and constraint satisfaction on the GP process model is detailed by Table 1.

The performance of the policy on the process model is however, not the primary contribution of this work. Rather, it is of interest to validate the safety of the policy on deployment to the real stochastic process and highlight the particular use of the training approach detailed. To achieve this, results were obtained by sampling the real process described by Eq. 31, with the parametric uncertainty detailed in Section 4. The results of this are expressed by Fig. 2.

Similar to Fig. 1 b), Fig. 2 a) details the performance of the policy with respect to the constraints, but upon deployment to the real uncertain process. Again, the shaded regions indicate 99% confidence intervals of process deviation and the dark blue solid line indicates the nominal process. Fig. 2 b) provides comparative detail of the distribution of the trajectories with respect to the constraints when the policy is deployed on the GP state space model (blue) and when deployed to the real uncertain process (red). As previously, the shaded regions indicate 99% confidence intervals for process deviation and the solid line indicates the nominal process. It is observed that there is very little mismatch between the model and real process in this region of the state space and as a result the distributions of the first and second constraint (g_1 and g_2) are almost indistinguishable. Notably, however, there is indeed clear, but small amounts of mismatch between the nominal process trajectories on the GP state space model and the real process as demonstrated via the third constraint plot of g_3. Interestingly, this plot shows that 99% of the real process trajectories (the red region) are contained within the (blue) region described by the samples from the GP state space model. This indicates the potential that the offline GP model (epistemic) uncertainty, expressed via the variance of the posterior, could be able to guarantee constraint satisfaction and ensure safe RL policies.



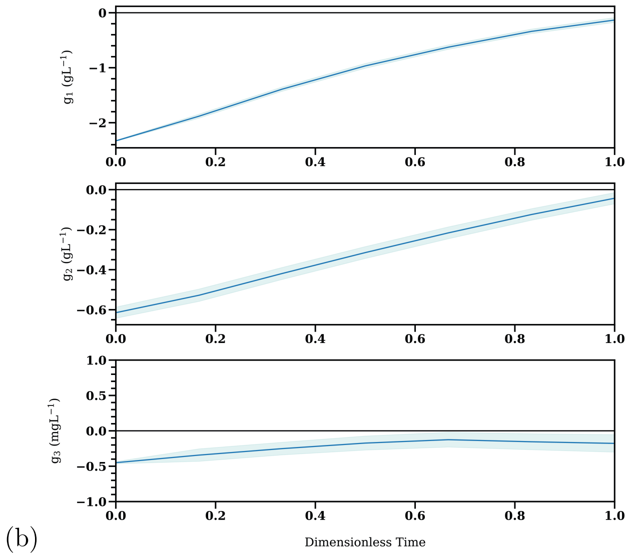
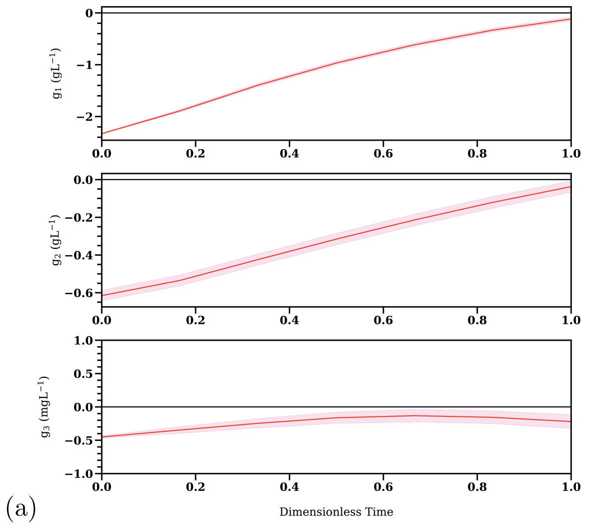


Figure 1: Results from Case Study. (a) The state profile produced from the final policy learned on the Gaussian Process model. (b) The corresponding distribution of trajectories with respect to the operational constraints. The i^{th} constraint is denoted $g_i := A_i^T\mathbf{x} - b_i$. The light blue shaded areas represent the 99th to 1st percentiles and solid blue line represents the expected trajectory. The black line plot represents the threshold of constraint violation i.e. when $g_i = 0$



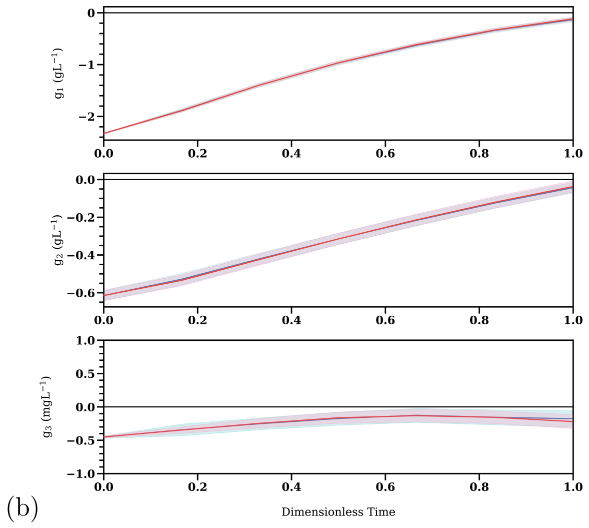


Figure 2: Results from Case Study. (a) The distribution of trajectories with respect to the operational constraints as sampled from the real uncertain process. (b) An overlay of the distributions observed when the policy is deployed on the real uncertain process (red) and the GP state space model (blue) as plotted in Fig. 1. The i^{th} constraint is denoted $g_i := A_i^T\mathbf{x} - b_i$. The shaded areas represent the 99th to 1st percentiles and solid line represents the expected trajectory. The black line plot represents the threshold of constraint violation i.e. when $g_i = 0$

Table 1: Case Study: Comparison of probabilities of joint constraint satisfaction $F_{LB}(0)$ and $F_{SA}(0)$ and objective values of $\pi^*_C(\cdot, \theta)$ as learned via the methodology on the real process and GP state space model. The objective performance is quantified via the mean and variance due to process stochasticity. See Eq. 34 for detail of the process objective.

Process	$F_{LB}(0)$	$F_{SA}(0)$	Process Objective (Eq. 34)
Offline Gaussian process model	1.0	1.0	15.29 +/- 0.11
Online real uncertain process	1.0	1.0	15.23 +/- 0.096

Table 1 demonstrates the utility of the algorithm in achieving constraint satisfaction as desired in both the offline model and real uncertain process. There is a small discrepancy between the performances of the two validations. This could be explained either due to the number of finite samples (500) used in assessment of policy performance, or via small amounts of nominal process mismatch between the GP model and the real process. If the latter view is taken and it is assumed the biomass and nitrate states are perfectly predicted (biomass is not included in the objective directly and nitrate is, but weakly), then this difference corresponds to a 0.375% prediction error of the nominal lutein trajectory. In the following sections, the work detailed here is benchmarked against results observed from implementing control policies on the uncertain process determined via a) offline DO and b) NMPC. The approach to generation of these results is discussed in Section 4.3.

5.2. Comparison to Benchmark Methods

The benchmark for this case study is provided by DO and NMPC, both of which are common approaches to process control. In the following sections the investigation provides comparative analysis to demonstrate the utility and limitations of the methodology.

5.2.1. Comparison to Dynamic Optimization

In order to demonstrate the effects of process stochasticity for dynamic optimization (DO), control profiles were generated for the system (Eq. 31) from four different initial conditions, all of which are probable to be drawn from the initial state distribution detailed in Section 4. As previously, all results are derived from 500 realisations of the real uncertain process model. The comparative performance of the DO benchmark is detailed by Table 2.

Table 2: Case Study: Comparison of probabilities of joint constraint satisfaction $F_{LB}(0)$ and $F_{SA}(0)$ and objective values of $\pi^*_C(\cdot, \theta)$ under the proposed dynamic optimization (DO) benchmark. Four different results are reported for DO, corresponding to the four different initial conditions used to generate the control profile offline. The objective performance is quantified via the mean and variance due to process stochasticity. See Eq. 34 for detail of the process objective.

Algorithm	Initial Conditions $\mathbf{x}_0$	$F_{LB}(0)$	$F_{SA}(0)$	Process Objective $J(\tau)$
DO I	[0.276, 784, 0.0]	0.036	0.056	16.68 +/- 0.24
DO II	[0.273, 774, 0.0]	0.046	0.068	16.68 +/- 0.25
DO III	[0.270, 765, 0.0]	0.030	0.048	16.65 +/- 0.25
DO IV	[0.267, 755, 0.0]	0.043	0.064	16.61 +/- 0.25
Current	$\mathbf{x}_0 \sim p(\mathbf{x}_0)$	1.0	1.0	15.23 +/- 0.096

From Table 2 it is clear that the effects of small amounts of stochasticity have dramatic implications for the probability of joint chance constraint satisfaction for DO. Both the statistically robust $F_{LB}(0)$

and the sample approximate $F_{SA}(0)$ are less than 0.06 for all DO control profiles. This highlights the utility of the method proposed in accounting for process stochasticity. It is also necessary to comment on the standard deviation of the performance with respect to the process objective as reported. The RL policy trained by the method achieves a lower variance in performance than that of the DO scheme. This is worth discussion as it highlights the ability of RL policies to naturally account for process stochasicity in a closed loop feedback control manner. Whereas, the variance of performance reported for the DO strategies is similar across all results and expresses the effects of process stochasticity on an open loop nominal (and deterministic) control policy. However, it is also important to note that although the RL method proposed performs with respect to constraint satisfaction, it does not achieve as well as DO with respect to the expected unconstrained process objective $J(\tau)$. This is mainly because backing the nominal process away from the constraint boundaries in order to account for variability, will naturally incur a decrease in the nominal performance of the policy. However, as the process objective function is only reduced by 8% and the constraints are satisfied with high probability, the current approach is still advantageous.

5.3. Comparison to Nonlinear Model Predictive Control

The generation of the NMPC trajectory similarly assumes use of the deterministic variant of Eq. 31, as the process model. Here, however, the control policy is updated online via complete observation of the real uncertain process state (as is typical). The initial state is drawn from the initial state distribution detailed in Section 4.1, which was also used to train and validate the RL policy $\pi_C^*(\cdot, \theta)$. Table 3 reports the respective KPIs for the method proposed and the NMPC scheme.

Table 3: Case Study: Comparison of probabilities of joint constraint satisfaction $F_{LB}(0)$ and $F_{SA}(0)$ and objective values of $\pi_C^*(\cdot, \theta)$ under the proposed benchmark of nonlinear model predictive control (NMPC). The objective performance is quantified via the mean and variance due to process stochasticity. See Eq. 34 for detail of the process objective.

Algorithm	$F_{LB}(0)$	$F_{SA}(0)$	Process Objective $J(\tau)$
NMPC	0.12	0.148	11.58 +/- 4.07
Ours	1.0	1.0	15.23 +/- 0.096

Interestingly, with reference to Table 3, the method proposed performs better than NMPC with respect to the process objective. In this case, this is primarily due to the destabilisation of NMPC by process stochasitity, which was evidenced by the frequent inability to find control solutions online. This is common when stochastic systems are driven close to constraint boundaries with deterministic methods. The inability of NMPC to find control solutions online is the primary reason for the difference in objective performance as detailed by Table 3 (Note: if solution could not be found, an approximate problem was solved to minimise constraint violation, and this was found to considerably improve performance - see Section 4.3 for information). In combination with worst cases of process stochasticity, this provides a skewing of the nominal process performance as reported. Demonstration of the sensitivity of the NMPC control scheme to process stochasticity is best expressed in analysis of the control trajectories generated in validation on the real uncertain process. This is reported by Fig. 3.

From Fig. 3 the relative effect of stochasticity on the NMPC scheme is apparent. Fig. 3 a) displays the distribution of controls selected under the RL policy, $\pi_C^*(\cdot, \theta)$, on the real uncertain process. The red solid line represents the average control trajectory and the red shaded region, which is essentially indistinguishable, represents a confidence interval of one standard deviation. Fig. 3 b) represents the distribution of controls selected by the benchmark NMPC control policy when validated under the real

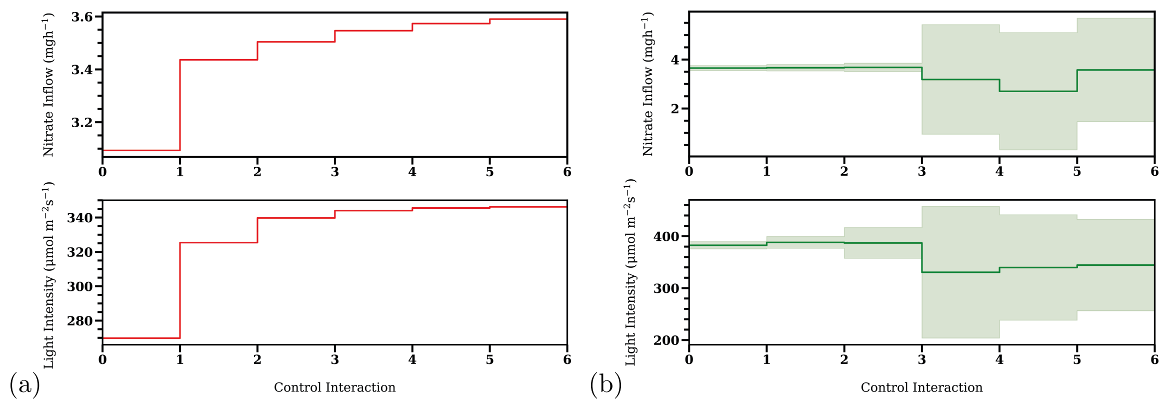


Figure 3: Results from Case Study. (a) The distribution of controls selected by the RL policy, $\pi^*_C(\cdot, \theta)$, upon validation on the real uncertain process. Red solid line represents the average control trajectory and the light red shaded region represents a 1 standard deviation confidence interval (which is essentially non-existent), (b) The distribution of controls selected by the NMPC policy upon validation on the real uncertain process. Green solid line represents the average control trajectory and the light red shaded region represents a 1 standard deviation confidence interval

uncertain process. Here, the green solid line represents the average control trajectory and the green shaded region, which is relatively large, represents a confidence interval of one standard deviation. It is likely that the average control trajectory plotted is not representative of actual control behaviour, i.e. the distribution of controls at each time interval is not best described by a unimodal Gaussian. However, the figure plotted well expresses the relative variance of controls selected.

From comparison of 3 a) and b), it is clear that the RL method proposed naturally accounts for process stochasticity in a closed loop manner, with little variance in the distribution of controls shown. This is characteristic of a control strategy, which is robust to process uncertainty. This is especially beneficial in the context of cell cultivation or fermentation processes, where cell metabolism is sensitive to variation in the environmental conditions. As a result, the oscillatory control behaviour demonstrated by the NMPC control scheme would likely have a detrimental effect on the efficacy of operation/cell metabolism. It should be noted that the detrimental effects of process stochasticity on deterministic control strategies are demonstrated here, with small amounts of uncertainty. In the types of processes of concern to this work, (parametric) uncertainties can be much larger. This further contextualises the benefits provided by the strategy propsoed.

6. Conclusion

In this work, an efficient, purely data-driven method has been proposed, which considers the safe deployment of RL policies from the offline training environment (process model) to the real uncertain process. The method also provides approach to ensuring joint chance constraint satisfaction with a set probability. This is facilitated through use of the aleatoric and epistemic uncertainties expressed naturally by Gaussian process models, as well as the concept of constraint tightening. The method was analysed empirically and benchmarked against two commonly used, deterministic approaches to control and optimization of fed-batch process systems. It was demonstrated that the presence of even small amounts of process stochasticity may have a destabilising effect on the performance of deterministic methods and their relative probabilities of achieving joint constraint satisfaction. It should be highlighted that the level of parametric uncertainty (5%) expressed in this case study is a common lower-bound to that typically observed in the processes of concern to this work. It is likely that the benefits of this method would be even more apparent in cases where higher uncertainties were present. Further, the formalisation of this approach and the link drawn to reward shaping, enables combination of the method with any RL algorithm - policy optimization, action-value methods and all that lies inbetween. We hypothesise that once deployed, the policy could be continuously improved offline as the local model is iteratively improved and updated between batches [32]. Further, it is possible the method could be adapted to the multi-agent setting for distributed control of fed-batch processes or into the domain of continuous processing [70]. We do however, assume the availability of an existing dataset, which provides information about the operational region of interest, however this could be developed using available mechanistic models and [33] in a model-based design of experiments. Future work should consider the quantification of uncertainties in the parameterisation of the control function.

References

[1] J. Shin, T. A. Badgwell, K.-H. Liu, J. H. Lee, Reinforcement learning–overview of recent progress and implications for process control, Computers & Chemical Engineering 127 (2019) 282–294.

[2] J. W. Kim, B. J. Park, H. Yoo, T. H. Oh, J. H. Lee, J. M. Lee, A model-based deep reinforcement learning method applied to finite-horizon optimal control of nonlinear control-affine system, Journal of Process Control 87 (2020) 166–178.

[3] T. Joshi, S. Makker, H. Kodamana, H. Kandath, Application of twin delayed deep deterministic policy gradient learning for the control of transesterification process (2021). `arXiv:2102.13012`.

[4] S. Spielberg, A. Tulsyan, N. P. Lawrence, P. D. Loewen, R. Bhushan Gopaluni, Toward self-driving processes: A deep reinforcement learning approach to control, AIChE Journal 65 (10) (Jun 2019). `doi:10.1002/aic.16689`.
URL `http://dx.doi.org/10.1002/aic.16689`

[5] D. E. Kirk, Optimal control theory: an introduction, Courier Corporation, 2004.

[6] D. P. Bertsekas, D. P. Bertsekas, D. P. Bertsekas, D. P. Bertsekas, Dynamic programming and optimal control, Vol. 1, Athena scientific Belmont, MA, 1995.

[7] R. S. Sutton, A. G. Barto, Reinforcement learning: An introduction, MIT press, 2018.

[8] P. Petsagkourakis, I. O. Sandoval, E. Bradford, D. Zhang, E. A. del Rio-Chanona, Reinforcement learning for batch-to-batch bioprocess optimisation, in: Computer Aided Chemical Engineering, Vol. 46, Elsevier, 2019, pp. 919–924.

[9] G. Hüllen, J. Zhai, S. H. Kim, A. Sinha, M. J. Realff, F. Boukouvala, Managing uncertainty in data-driven simulation-based optimization, Computers & Chemical Engineering 136 (2020) 106519.

[10] A. Kumar, A. Zhou, G. Tucker, S. Levine, Conservative q-learning for offline reinforcement learning (2020). `arXiv:2006.04779`.

[11] R. Agarwal, D. Schuurmans, M. Norouzi, An optimistic perspective on offline reinforcement learning (2020). `arXiv:1907.04543`.

[12] T. Yu, A. Kumar, R. Rafailov, A. Rajeswaran, S. Levine, C. Finn, Combo: Conservative offline model-based policy optimization (2021). `arXiv:2102.08363`.

[13] R. Kidambi, A. Rajeswaran, P. Netrapalli, T. Joachims, Morel : Model-based offline reinforcement learning (2021). `arXiv:2005.05951`.

[14] T. Yu, G. Thomas, L. Yu, S. Ermon, J. Zou, S. Levine, C. Finn, T. Ma, Mopo: Model-based offline policy optimization (2020). `arXiv:2005.13239`.

[15] J. M. Lee, J. H. Lee, Approximate dynamic programming-based approaches for input–output data-driven control of nonlinear processes, Automatica 41 (7) (2005) 1281–1288.

[16] E. Parzen, On estimation of a probability density function and mode, The annals of mathematical statistics 33 (3) (1962) 1065–1076.

[17] E. Hüllermeier, W. Waegeman, Aleatoric and epistemic uncertainty in machine learning: An introduction to concepts and methods, arXiv preprint arXiv:1910.09457 (2019).

[18] W. R. Clements, B. V. Delft, B.-M. Robaglia, R. B. Slaoui, S. Toth, Estimating risk and uncertainty in deep reinforcement learning (2020). `arXiv:1905.09638`.

[19] Y. Li, N. Li, H. E. Tseng, A. Girard, D. Filev, I. Kolmanovsky, Safe reinforcement learning using robust action governor (2021). `arXiv:2102.10643`.

[20] K. P. Wabersich, M. N. Zeilinger, A predictive safety filter for learning-based control of constrained nonlinear dynamical systems (2021). `arXiv:1812.05506`.

[21] J. Achiam, D. Held, A. Tamar, P. Abbeel, Constrained policy optimization (2017). `arXiv:1705.10528`.

[22] S. Huh, I. Yang, Safe reinforcement learning for probabilistic reachability and safety specifications: A lyapunov-based approach, arXiv preprint arXiv:2002.10126 (2020).

[23] E. Leurent, D. Efimov, O.-A. Maillard, Robust-adaptive control of linear systems: beyond quadratic costs (2020). `arXiv:2002.10816`.

[24] B. Peng, Y. Mu, J. Duan, Y. Guan, S. E. Li, J. Chen, Separated proportional-integral lagrangian for chance constrained reinforcement learning (2021). `arXiv:2102.08539`.

[25] P. Petsagkourakis, I. O. Sandoval, E. Bradford, F. Galvanin, D. Zhang, E. A. del Rio-Chanona, Chance constrained policy optimization for process control and optimization, arXiv preprint arXiv:2008.00030 (2020).

[26] E. Pan, P. Petsagkourakis, M. Mowbray, D. Zhang, A. del Rio-Chanona, Constrained model-free reinforcement learning for process optimization (2020).

[27] A. Mesbah, I. V. Kolmanovsky, S. Di Cairano, Stochastic model predictive control, in: Handbook of Model Predictive Control, Springer, 2019, pp. 75–97.

[28] Y. I. Valdez-Navarro, L. A. Ricardez-Sandoval, A novel back-off algorithm for integration of scheduling and control of batch processes under uncertainty, Industrial & Engineering Chemistry Research 58 (48) (2019) 22064–22083.

[29] M. Rafiei, L. A. Ricardez-Sandoval, Integration of design and control for industrial-scale applications under uncertainty: a trust region approach, Computers & Chemical Engineering 141 (2020) 107006.

[30] H. Yoo, V. M. Zavala, J. H. Lee, A dynamic penalty function approach for constraints-handling in reinforcement learning (2021). `arXiv:2012.11790`.

[31] A. Kumar, R. Agarwal, G. Tucker, L. Li, D. Precup, A. Kumar, Workshop: Offline reinforcement learning, neural Information Processing Systems Online Conference 2020 (2020).

[32] A. Rajeswaran, I. Mordatch, V. Kumar, A game theoretic framework for model based reinforcement learning (2020). `arXiv:2004.07804`.

[33] P. Petsagkourakis, F. Galvanin, Safe model-based design of experiments using gaussian processes (2020). `arXiv:2011.10009`.

[34] W. Sternberg, M. P. Deisenroth, Identification of gaussian process state-space models (2017).

[35] R. Frigola, Bayesian time series learning with gaussian processes, Ph.D. thesis, University of Cambridge (2015).

[36] E. Bradford, L. Imsland, D. Zhang, E. A. del Rio Chanona, Stochastic data-driven model predictive control using gaussian processes, Computers & Chemical Engineering 139 (2020) 106844.

[37] E. Bradford, A. M. Schweidtmann, D. Zhang, K. Jing, E. A. del Rio-Chanona, Dynamic modeling and optimization of sustainable algal production with uncertainty using multivariate gaussian processes, Computers & Chemical Engineering 118 (2018) 143–158.

[38] E. del Rio Chanona, P. Petsagkourakis, E. Bradford, J. A. Graciano, B. Chachuat, Real-time optimization meets bayesian optimization and derivative-free optimization: A tale of modifier adaptation, Computers & Chemical Engineering (2021) 107249.

[39] J. Umlauft, T. Beckers, S. Hirche, Scenario-based optimal control for gaussian process state space models, in: 2018 European Control Conference (ECC), IEEE, 2018, pp. 1386–1392.

[40] M. Deisenroth, C. E. Rasmussen, Pilco: A model-based and data-efficient approach to policy search, in: Proceedings of the 28th International Conference on machine learning (ICML-11), Citeseer, 2011, pp. 465–472.

[41] S. Curi, F. Berkenkamp, A. Krause, Efficient model-based reinforcement learning through optimistic policy search and planning, arXiv preprint arXiv:2006.08684 (2020).

[42] A. Y. Ng, D. Harada, S. Russell, Policy invariance under reward transformations: Theory and application to reward shaping, in: Icml, Vol. 99, 1999, pp. 278–287.

[43] M. Rafiei, L. A. Ricardez-Sandoval, Stochastic back-off approach for integration of design and control under uncertainty, Industrial & Engineering Chemistry Research 57 (12) (2018) 4351–4365.

[44] S. Subramanian, S. Lucia, R. Paulen, S. Engell, Tube-enhanced multi-stage model predictive control for flexible robust control of constrained linear systems with additive and parametric uncertainties, International Journal of Robust and Nonlinear Control (Mar 2021). `doi:10.1002/rnc.5486`.
URL `http://dx.doi.org/10.1002/rnc.5486`

[45] C. E. Rasmussen, Gaussian processes for machine learning, MIT Press, 2006.

[46] G. Lindgren, Stationary stochastic processes: theory and applications, CRC Press, 2012.

[47] V. Strassen, Gaussian elimination is not optimal, Numerische mathematik 13 (4) (1969) 354–356.

[48] E. Bradford, A. M. Schweidtmann, A. Lapkin, Efficient multiobjective optimization employing gaussian processes, spectral sampling and a genetic algorithm, Journal of global optimization 71 (2) (2018) 407–438.

[49] L. S. Bastos, A. O'Hagan, Diagnostics for gaussian process emulators, Technometrics 51 (4) (2009) 425–438.

[50] J. Nocedal, S. Wright, Numerical optimization, Springer Science & Business Media, 2006.

[51] J. Larson, M. Menickelly, S. M. Wild, Derivative-free optimization methods, arXiv preprint arXiv:1904.11585 (2019).

[52] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, O. Klimov, Proximal policy optimization algorithms, arXiv preprint arXiv:1707.06347 (2017).

[53] D. R. Jones, M. Schonlau, W. J. Welch, Efficient global optimization of expensive black-box functions, Journal of Global optimization 13 (4) (1998) 455–492.

[54] J. A. Paulson, E. A. Buehler, R. D. Braatz, A. Mesbah, Stochastic model predictive control with joint chance constraints, International Journal of Control 93 (1) (2020) 126–139.

[55] M. Farina, L. Giulioni, L. Magni, R. Scattolini, An mpc approach to output-feedback control of stochastic linear discrete-time systems (2014).

[56] G. Boole, The mathematical analysis of logic, Philosophical Library, 1847.

[57] H. Ogasawara, The multiple cantelli inequalities, Statistical Methods & Applications 28 (3) (2019) 495–506.

[58] L. Magni, D. Pala, R. Scattolini, Stochastic model predictive control of constrained linear systems with additive uncertainty, in: 2009 European Control Conference (ECC), IEEE, 2009, pp. 2235–2240.

[59] Y. Dong, X. Tang, Y. Yuan, Principled reward shaping for reinforcement learning via lyapunov stability theory, Neurocomputing 393 (2020) 83–90.

[60] P. I. Frazier, A tutorial on bayesian optimization (2018).

[61] E. Brochu, V. M. Cora, N. De Freitas, A tutorial on bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning, arXiv preprint arXiv:1012.2599 (2010).

[62] I. M. Sobol', On the distribution of points in a cube and the approximate evaluation of integrals, Zhurnal Vychislitel'noi Matematiki i Matematicheskoi Fiziki 7 (4) (1967) 784–802.

[63] D. Zhang, E. A. Del Rio-Chanona, P. Petsagkourakis, J. Wagner, Hybrid physics-based and data-driven modeling for bioprocess online simulation and optimization, Biotechnology and bio-engineering 116 (11) (2019) 2919–2930.

[64] E. A. del Rio-Chanona, N. r. Ahmed, D. Zhang, Y. Lu, K. Jing, Kinetic modeling and process analysis for desmodesmus sp. lutein photo-production, AIChE Journal 63 (7) (2017) 2546–2554.

[65] J. Schulman, S. Levine, P. Moritz, M. I. Jordan, P. Abbeel, Trust region policy optimization (2017). `arXiv:1502.05477`.

[66] L. T. Biegler, An overview of simultaneous strategies for dynamic optimization, Chemical Engineering and Processing: Process Intensification 46 (11) (2007) 1043–1053.

[67] M. Kelly, An introduction to trajectory optimization: How to do your own direct collocation, SIAM Review 59 (4) (2017) 849–904.

[68] A. Wächter, L. T. Biegler, On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming, Mathematical programming 106 (1) (2006) 25–57.

[69] J. Andersson, J. Åkesson, M. Diehl, Casadi: A symbolic package for automatic differentiation and optimal control, in: Recent advances in algorithmic differentiation, Springer, 2012, pp. 297–307.

[70] D. G. McClement, N. P. Lawrence, P. D. Loewen, M. G. Forbes, J. U. Backström, R. B. Gopaluni, A meta-reinforcement learning approach to process control (2021). `arXiv:2103.14060`.

[71] R. S. Sutton, D. A. McAllester, S. P. Singh, Y. Mansour, et al., Policy gradient methods for reinforcement learning with function approximation., in: NIPs, Vol. 99, Citeseer, 1999, pp. 1057–1063.

[72] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, K. Kavukcuoglu, Asynchronous methods for deep reinforcement learning, in: International conference on machine learning, PMLR, 2016, pp. 1928–1937.

[73] G. Neu, A. Jonsson, V. Gómez, A unified view of entropy-regularized markov decision processes (2017).

[74] Z. Ahmed, N. Le Roux, M. Norouzi, D. Schuurmans, Understanding the impact of entropy on policy optimization, in: K. Chaudhuri, R. Salakhutdinov (Eds.), Proceedings of the 36th International Conference on Machine Learning, Vol. 97 of Proceedings of Machine Learning Research, PMLR, 2019, pp. 151–160.

[75] J. Schulman, P. Moritz, S. Levine, M. Jordan, P. Abbeel, High-dimensional continuous control using generalized advantage estimation (2018). `arXiv:1506.02438`.

[76] Z. Ahmed, N. L. Roux, M. Norouzi, D. Schuurmans, Understanding the impact of entropy on policy optimization (2019). `arXiv:1811.11214`.

[77] T. Haarnoja, A. Zhou, P. Abbeel, S. Levine, Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor, in: International Conference on Machine Learning, PMLR, 2018, pp. 1861–1870.

[78] B. D. Ziebart, Modeling purposeful adaptive behavior with the principle of maximum causal entropy (2010).

[79] S. M. Kakade, A natural policy gradient, Advances in neural information processing systems 14 (2001).

[80] C. J. Clopper, E. S. Pearson, The use of confidence or fiducial limits illustrated in the case of the binomial, Biometrika 26 (4) (1934) 404–413. `doi:10.2307/2331986`.
URL `http://www.jstor.org/stable/2331986`

[81] L. D. Brown, T. T. Cai, A. DasGupta, Interval estimation for a binomial proportion, Statistical Science 16 (2) (2001) 101–117. `doi:10.2307/2676784`.
URL `http://www.jstor.org/stable/2676784`

Appendix A. Appendices

Appendix A.1. Gaussian Process State Space Modelling

Appendix A.1.1. Training of Gaussian Process Models

Selection of the appropriate hyperparameters for a covariance function provides considerable improvement in the predictive abilities of GPs and can be viewed as a parallel to parameter estimation for mechanistic process models. The tuning procedure acts to maximise the marginal log-likelihood $p(\mathbf{Y}_j^T|\boldsymbol{\Upsilon}, \hat{\lambda})$ of the state specific, noisy output data points $\mathbf{Y}_j$, provided with the respective input measurements $\boldsymbol{\Upsilon}$ and hyperparameters $\hat{\lambda}$:

$$\log p(\mathbf{Y}_j^T|\boldsymbol{\Upsilon}, \hat{\lambda}) = -\frac{1}{2}(\mathbf{Y}_j(K + \sigma_n^2 I_N)^{-1}\mathbf{Y}_j^T + \log|K + \sigma_n^2 I_N| + N \log 2\pi) \tag{A.1}$$

Gradient-based optimization may then be deployed to find $\hat{\lambda}$, which maximise the likelihood of observing our output data, given the covariance function chosen and the input data. This problem is non-convex and so typically multi-start schemes are deployed to find the best solution.

Appendix A.1.2. Obtaining Function Realisations from GP State Space Models

In this work, we are concerned with obtaining function realisations from a Gaussian process state space model. The state space model is composed of n_x individual Gaussian process models of each state. Here, we use the method proposed by [47, 36, 39]. The method aims to update the posterior distribution of the Gaussian process model according to the initial dataset $\mathcal{D}$ used for model construction, as well as the states and control inputs observed during each trajectory evolution. This combined dataset is denoted $\mathcal{D}_+$ as discussed in Section 3.1.3. We can express the updated posterior distribution of the j^{th} GP after transition from one discrete time index at $t = t_0$ to $t = t_1$ as follows:

$$\begin{aligned} \mu_j(\boldsymbol{v}^*; \mathcal{D}_+) &= {K_*^+}^T {\Sigma^+}^{-1} {\mathbf{Y}_j^+}^T \\ \sigma_j^2(\boldsymbol{v}^*; \mathcal{D}_+) &= k(\boldsymbol{v}^*, \boldsymbol{v}^*) - {K_*^+}^T {\Sigma^+}^{-1} K_*^+ \end{aligned} \tag{A.2}$$

where $\mathbf{Y}_j^+ = [\mathbf{Y}_j, y_j^+] \in \mathbb{R}^{1\times(N+1)}$, where $y_j^+ \in \mathbb{R}$ is state $x_j \in \mathbb{R}$ observed at time index $t = t_1$; The updated covariance matrices are expressed as follows:

$${K_*^+}^T = \left[K_*^T, k(v_*, v^+)\right] \qquad \Sigma^{+-1} = \begin{bmatrix} K + \sigma_n I_N & K_+ \\ K_+^T & k(\boldsymbol{v}^+, \boldsymbol{v}^+) \end{bmatrix}^{-1} \tag{A.3}$$

where $K_{+}^{T} = [k(\boldsymbol{v}_{+}, \boldsymbol{v}_i), \ldots, k(\boldsymbol{v}_{+}, \boldsymbol{v}_N)] \in \mathbb{R}^{1\times N}$, and $\boldsymbol{v}_{+} \in \mathbb{R}^{n_v}$ is the state and control input pair at time index $t = t_0$. This process is repeated iteratively for state transitions thereafter, which means that the memory and computation requirements will grow quadratically and cubically, respectively, with the time horizon T. Updating K_{*}^{+}, Y_{j}^{+} is relatively easy, however, updating ${\Sigma^{+}}^{-1}$, is slightly more involved due to inversion. In order to do this we use the method from [47] as proposed in [36, 39]. We refer the reader to these works for more information.

Appendix A.2. Proximal Policy Optimization, The Advantage Function and Entropy Regularisation

PPO is at its core a policy gradient (PG) method. PG methods have previously been discussed, and so this work directs the interested reader to the original paper [71] and other recent work [8]. PPO utilises a specific instance of the PG, known as the advantage policy gradient (APG). The APG [72] is a powerful, low variance form of the policy gradient, which utilises the generalised advantage function estimate A_{φ} (GAE), rather than the action-value estimate, as in vanilla policy gradient methods [71]. Further detail on the GAE and PPO is provided by Appendix A.2.1 and Appendix A.2.3, respectively. In practice, the investigation found the addition of an entropy regularisation term useful in RL training. Entropy regularisation is widely studied in the RL literature, and at a high level provides mechanism to ensure the policy does not converge deterministically to a poor local optimum. This is particularly important in view of RL as a set of sampling-based algorithms [73, 74] and is discussed further in Appendix A.2.2.

Appendix A.2.1. The Advantage Function

The advantage function [75] is formalised:

$$\begin{gathered} V^{\pi}(\mathbf{x}_t) = \mathbb{E}_{\pi}\left[\sum_{t'=t}^{T-1} R_{t'+1} | \mathbf{x} = \mathbf{x}_t\right] \qquad Q^{\pi}(\mathbf{x}_t, \mathbf{u}_t) = \mathbb{E}_{\pi}\left[R_{t+1} + \gamma V^{\pi}(\mathbf{x}') | \mathbf{x} = \mathbf{x}_t, \mathbf{u} = \mathbf{u}_t\right] \\ A^{\pi}(\mathbf{x}_t, \mathbf{u}_t) = Q^{\pi}(\mathbf{x}_t, \mathbf{u}_t) - V^{\pi}(\mathbf{x}_t) \end{gathered} \tag{A.4}$$

and, represents the difference between the expected returns under a policy in the current state, V^{π}, and the returns accumulated from selecting a given control in the current state and the current policy thereafter, Q^{π}. In RL practice, parameterisation of the value function V_{ψ} is required in order to approximate the true value function V^{π}, such that $V_{\psi} \approx V^{\pi}$. Decision as to the model structure and initialisation of the parameters asserts bias into estimation of the advantage function. This is reduced through use of the generalised advantage function estimate $\hat{A}^{\pi}$ (GAE). The GAE provides a mechanism to explicitly *trade off* variance and bias, by maximising the information provided by the reward signal. Explicitly, the GAE is formalised as:

$$\begin{aligned} \hat{A}_t^{\pi} &= \delta_{t+1} + (\rho\gamma)\delta_{t+2} + \ldots + (\rho\gamma)^{T-t+1}\delta_T \\ \delta_{t+1} &= R_{t+1} + \gamma V_{\psi}(\mathbf{x}_{t+1}) - V_{\psi}(\mathbf{x}_t) \end{aligned} \tag{A.5}$$

The parameter $\rho = [0, 1]$ provides the mechanism to balance the bias and variance. Values closer to 1 reduce bias by utilising more information from the reward signal, but at the compromise of increasing the variance of the estimate. The opposite applies as values tend to 0.

Appendix A.2.2. Entropy Regularisation

There is a rich literature on maximum entropy (Max.Ent.) RL [76, 77, 78]. Instead of simply optimizing for the process objective and accumulated reward, $G(\boldsymbol{\tau})$, Max.Ent. RL also optimizes for the expected entropy of the stochastic policy learned. As a result, we can formulate the Max.Ent. RL objective, J_H as follows:

$$J_H = \mathbb{E}\big[G(\boldsymbol{\tau}) + H_\pi\big] \tag{A.6}$$

where $H_\pi = -\mathbb{E}_\pi\big[\log \pi(\mathbf{u}|\mathbf{x})\big]$ is the entropy of the policy. Typically, in practice, this objective is maximised via a regularisation term i.e. not as an extrinsic addition of entropy to the reward signal, and therefore not optimized via the PG. It is thought that entropy regularisation provides two main benefits: 1) it modifies the optimization landscape 'for the better', and in some cases provides a smoother landscape than the vanilla objective, and 2) the use of entropy plays some role in tackling the exploration-exploitation paradigm, i.e. by regularising entropy, exploration is encouraged, preventing convergence to a suboptimal deterministic policy. The use of entropy was found to be particularly helpful in this study, aiding the learning dynamics. It could be perceived that the constraint boundary provides a discontinuity in the reward landscape, and the promotion of exploration via entropy provides mechanism to 'escape' local optima.

Appendix A.2.3. Entropy Regularised Proximal Policy Optimization

PPO aims to provide conservative policy updates, by utilising the concept of trust region optimization. The idea of trust region optimization in the RL sense, is to constrain the update of an initial policy, such that the ultimate policy remains within a given distance of the initial in policy space. This distance could e.g. be quantified by the Kullback-Liebler divergence. One algorithm known as trust-region policy optimization (TRPO) necessitates estimate of the Hessian of the approximate KL divergence with respect to the policy parameters [65] (this also shares similarities with the natural policy gradient [79]). PPO sidesteps this complexity through approximation of the 2nd order TRPO update with a first order update - instead of explicitly enforcing this as a hard constraint, PPO enforces this via a penalty method [52]. This means that PPO is more computationally efficient than TRPO and provides flexible use of different function approximators (policy parameterisations).

The objective function L^{CLIP} formalised within the PPO framework follows:

$$\begin{aligned} r_t(\theta) &= \frac{\pi_\theta(\mathbf{u}_t|\mathbf{x}_t)}{\pi_{\theta_{old}}(\mathbf{u}_t|\mathbf{x}_t)} \\ L^{CLIP}(\theta) &= \hat{\mathbb{E}}_t\Big[\min(r_t(\theta)\hat{A}_t^\pi, clip(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t^\pi)\Big] \end{aligned} \tag{A.7}$$

where, $\epsilon = [0, 1]$ and $\hat{A}_t^\pi$ is the advantage function, as discussed previously. By clipping the ratio r, updates corresponding to negative advantages are clipped with a ratio of $r = 1 + \epsilon$, whereas updates with positive advantages are clipped at $r = 1 - \epsilon$. The minimum is taken in order to provide a pessimistic update and enforce what could be interpreted as a trust-region. A full entropy regularised

PPO algorithm is presented by Algorithm 4.

Algorithm 4: Entropy Regularised Proximal Policy Optimization

```
Initialise: Approximate state space model or process dynamics $f_{SS}(\cdot)$; Initial control policy
  $\pi(\mathbf{u}|\mathbf{x};\theta_0)$; Initial critic $V(\mathbf{x}_t,\psi_0)$; Reward function $R_{xx'}$; Finite horizon length $T$; initial state
  distribution $p(\mathbf{x}_0)$; entropy penalty term $\beta \in \mathbb{R}^+$; Learning rate $w^\pi \in \mathbb{R}^+$; Learning rate
  $w^V \in \mathbb{R}^+$; Strategies for updating the learning rates (schedules) $f_w^\pi(\cdot)$ and $f_w^V(\cdot)$; Memory
  $\mathcal{B}_{info}$ for information required for policy optimization; $K$ episodes; Learning updates per batch
  $J$; batchsize of $M$ trajectories; tolerance criterion;
1. $i = 0$;
2. while not converged do
    a. Obtain a batch of $K$ rollouts over horizon of $T$ discrete intervals, via $\pi(\mathbf{u}|\mathbf{x};\theta_i)$, $f_{SS}$,
       and $p(\mathbf{x}_0)$;
    b. Return trajectory information i.e. rewards $R^{(k)}_{0:T-1} = [R^{(k)}_1, \dots, R^{(k)}_T]$ under $R_{xx'}$
       for the sequence of controls $\mathbf{u}^{(k)}_{0:T-1} = [\mathbf{u}^{(k)}_0, \dots, \mathbf{u}^{(k)}_{T-1}]$ and states $\mathbf{x}^{(k)}_{0:T} = [\mathbf{x}^{(k)}_0, \dots, \mathbf{x}^{(k)}_T]$,
       corresponding to each rollout and store in $\mathcal{B}_{info}$;
    c. $j = 0$;
    while $j < J$ do
        i. Perform policy optimization by sampling the information of $M$ trajectories from
           $\mathcal{B}_{info}$, calculating the respective importance ratios $r_t$ via Eq. A.7 and GAEs via
           Eq. A.5:
           $\theta_{i+1} = \theta_i + w_i \nabla_\theta \big[\frac{1}{MT}\sum_{m=1}^{M}\sum_{t=0}^{T-1} L^{CLIP}(\mathbf{x}^{(m)}_t, \mathbf{u}^{(m)}_t, \mathbf{x}^{(m)}_{t+1}, \theta_i) + \beta H_\pi(\pi(\mathbf{u}^{(m)}_t|\mathbf{x}^{(m)}_t))\big]$;
        ii. Update the critic $V(\mathbf{x},\psi_i)$ on the same data sampled in c.i. and the respective
            returns, $G_t$:
            $\psi_{i+1} = \psi_i - \frac{1}{MT}\sum_{m=1}^{M}\sum_{t=0}^{T-1} \nabla_{\psi_i} V(\mathbf{x},\psi_i)(V(\mathbf{x}^{(m)}_t,\psi_i) - G^{(m)}_t)$ ;
        iii. Update the learning rate : $w^\pi_{i+1} = f^\pi_w(w_i)$ ;
        iv. Update the learning rate : $w^V_{i+1} = f^V_w(w_i)$ ;
        v. $i+=1$, $j+=1$ ;
    end
    d. Reset memory $\mathcal{B}_{info}$;
    e. Assess tolerance criterion ;
end
Output: Optimal policy $\pi(\theta^*)$ and critic $V(\psi^*)$;
```

Appendix A.3. Evaluating Joint Constraint Satisfaction Empirically

In this work, we are concerned with the satisfaction of the joint chance constraints expressed by:

$$F_X(0) = \mathbb{P}(X \le 0) = \mathbb{P}\Big(\bigcap_{i=0}^{T}\{\mathbf{x}_i \in \hat{\mathbb{X}}_i\}\Big) \tag{A.8}$$

where $\hat{\mathbb{X}}_i$ is the tightened joint constraint set and

$$X = \max_{(t,j)\in\{0,\dots,T\}\times\{1,\dots,n_g\}} A_j \mathbf{x}_t - b_j,$$

defines the maximum constraint violation during process evolution. As analytical expression of Eq. A.8 is not available, it is proposed to instead estimate it via Monte Carlo sampling. Hence we can define the empirical cumulative distribution function (ecdf) via S Monte Carlo samples:

$$F_X(0) \approx F_{SA}(0) = \frac{1}{S}\sum_{s=1}^{S} \mathbb{1}(X^{(s)} \leq 0) \tag{A.9}$$

where $\mathbb{1}$ is the indicator function. However, due to the limits imposed by finite samples, the approximation is likely to include error. Therefore, in order to account for this we deploy a concept from the *binomial proportion confidence interval* literature. Specifically, the Clopper–Pearson interval [80], which enables us to ensure the probability of joint satisfaction with a given confidence level, $1-\upsilon$, on the basis of empirical observation. This is expressed by Lemma 3, which is recycled from [25].

Lemma 3. ***Joint chance constraint satisfaction via the Clopper-Pearson confidence interval*** *[80, 81]: Consider the realisation of $F_{SA}(0)$ based on S independently and identically distributed samples. The lower bound of the true value $F_{LB}(0)$ may be defined with a given confidence $1-\upsilon$, such that:*

$$\begin{aligned} &\mathbb{P}(F_X(0) \geq F_{LB}(0)) \geq 1-\upsilon \\ &F_{LB}(0) = 1 - \mathit{betainv}(\upsilon, S+1-SF_{SA}(0), SF_{SA}(0)) \end{aligned} \tag{A.10}$$

where betainv$(\cdot)$ is the inverse of the Beta cumulative distribution function with parameters $\{S+1-SF_{SA}(0)\}$ and $\{SF_{SA}(0)\}$.

Appendix A.4. Parameters for Learning in Case Study

Table A.4: Miscellaneous hyperparameters specific to Proximal policy optimization algorithm used in this work.

Parameter	Value
Episodes, K	200
Nodes per LSTM layer of Policy Net.	30
LSTM Layers in Policy Net.	4
Activation function in output layer of Policy Net.	ReLU6
Nodes per LSTM layer in Value Net.	30
LSTM layers in Value Net.	2
Activation function in output layer of Value Net.	Leaky ReLU
Policy learning rate, w^{π}	5×10^{-3}
Value learning rate, w^{V}	5×10^{-3}
GAE weight, ρ	0.99
Batch size, M	100
Weight updates, J	2
Clipping factor, ϵ	0.2
Discount factor, γ	0.99
Entropy regularisation weights, β	5×10^{-2}