Satisfaction of path chance constraints in dynamic optimization problems

Eduardo S. Schultz[a], Simon Olofsson[b], Adel Mhamdi[a], Alexander Mitsos[c,a,d,*]

[a] *Process Systems Engineering (AVT.SVT), RWTH Aachen University, 52074 Aachen, Germany*
[b] *Department of Computing, Imperial College London, SW7 2AZ, London, United Kingdom*
[c] *JARA-CSD, 52056 Aachen, Germany*
[d] *Institute of Energy and Climate Research, Energy Systems Engineering (IEK-10), Forschungszentrum Jülich GmbH, 52425 Jülich, Germany*

Abstract

We propose an algorithm that calculates heuristically optimal solutions for dynamic optimization problems with path chance constraints. The solution is a feasible point in the chance constraint sense and an optimal point of an approximated problem. Uncertainty in parameters and initial conditions is modelled as Gaussian distributions. The method solves nonlinear programs (NLP) generated by replacing the probability constraint by a set of approximated deterministic pointwise constraints with a right-hand side restriction. For each NLP solution, the probability of constraint violation is calculated by Monte Carlo integration. When the NLP solution does not respect the chance constraint, new pointwise constraints are added, and we update the approximation and the restriction with the results from Monte Carlo integration. These steps are repeated until a feasible solution is found. The algorithm terminates after a finite number of iterations under mild assumptions. We demonstrate the algorithm in a fed-batch bioreactor case study, showing that it provides a solution in a shorter CPU time and fewer iterations when compared to using a fixed set of pointwise constraints where only the restriction is updated.

Keywords: Robust optimal control, Parametric uncertainty, Nonlinear systems, Chance constraint.

*Corresponding author
Email addresses: `eduardo.schultz@rwth-aachen.de` (Eduardo S. Schultz), `simon.olofsson15@alumni.imperial.ac.uk` (Simon Olofsson), `adel.mhamdi@avt.rwth-aachen.de` (Adel Mhamdi), `amitsos@alum.mit.edu` (Alexander Mitsos)

1. Introduction

Satisfaction of path constraints in industrial processes is fundamental to guarantee safe operation, product specification, and fulfilment of environmental regulations. In industrial processes, uncertainty has different sources, e.g., process model mismatch or error in the model parameters, cf. the review articles (Pistikopoulos, 1995; Bemporad and Morari, 1999; Geletu and Li, 2014). Thus, dynamic optimization algorithms that guarantee satisfaction of path constraints under uncertainty are desired (Srinivasan et al., 2003). Herein, we are interested in computing feasible solutions to offline dynamic optimization problems with uncertain parameters and initial conditions, assuming correct model structure. Different approaches exist to deal with uncertain variables, such as worst-case scenario, multi-scenario, and probabilistic optimization that we briefly describe in the next section.

We employ the probabilistic approach and we focus on optimization problems with single chance constraints, i.e., with one probability constraint for each state constraint of the system. Additionally, each state constraint under uncertainty does not need to be enforced over the whole domain. Instead, we calculate the probability of the constraint to be respected over the joint distribution of the Gaussian distributed parameters and the independent variable domain, i.e., if we sample from the parameter normal distribution and the independent variable domain, the probability of constraint violation must be below a user-defined maximum. As a simple example, we can think about a reactor with an uncertain kinetic parameter k_0, described by a normal distribution, that operates from time t_0 to t_f. The state constraints are the reactor temperature and pressure. For each, a given upper bound needs to be respected with a probability of 90%. This probability is calculated over the uncertain kinetic parameter and the time interval $[t_0, t_f]$.

The methods available to solve dynamic optimization problems with chance constraints (e.g., Srinivasan et al. (2003); Telen et al. (2015); Shi et al. (2016)) usually employ a procedure to approximate the probability constraint, assuming the state can be described by a Gaussian distribution at each point of the domain (which is not valid for nonlinear systems (Ricardez-Sandoval, 2012)). The approximated constraints are then enforced only on a finite number of points over the domain of the independent variable. Additionally, a restriction on the resulting constraints is often imposed. In the following we refer to this as "pointwise" constraints. The satisfaction of path constraints under uncertanty, over the whole domain, is only obtained by enforcing many pointwise constraints. There are no available methods in the literature that solve dynamic optimization problems with guaranteed satisfaction of the state chance constraints over the whole domain, or as a probability of the joint distribution of the parameters and the independent variable domain.

In this work we deal with these issues and propose an algorithm, where the optimization problem with path constraint enforces the state constraint over a joint distribution of the uncertain parameters and independent variable domain. We use ideas from our previous works that guarantee satisfaction of path constraint in dynamic systems without uncertainty (Fu et al., 2015; Faust et al., 2016; Schultz et al., 2020a,b). We extend the methodology for the case where the parameters and initial conditions of ODEs systems are uncertain. The method consists of propagating the uncertainty through the dynamic system approximately using the method of Srinivasan et al. (2003). Since the propagation is exact only for linear systems, we correct the resulting distributions with a Monte Carlo integration step. For a user-defined threshold on the constraint satisfaction probability, an explicit path chance constraint can be calculated using the quantile function of normal distributions. The approximated path constraint is then discretized as pointwise constraints. A restriction is imposed on each pointwise constraint, resulting in a nonlinear program (NLP) approximation of optimization problem.

In each iteration, we solve the approximated problem with an NLP local solver. For each solution, the algorithm calculates the probability of violation of the path chance constraint using Monte Carlo integration for a fixed user-defined confidence. If the solution does not respect the chance constraint of the original problem, we calculate the probability of violation over each interval of the domain between the points where the pointwise constraints are enforced. We employ as an upper bound for the probability of violation on each interval, the fraction of the overall probability proportional to the size of the interval compared to the size of the domain. For example, if the probability of violation must be below 10% from time 0 to 10, the probability of violation from time 0 to 5 must be below 5%. The intervals with probability of violation greater than the upper bound are halved with the addition of new pointwise constraints. Then, each approximated constraint is updated. Next, the restriction is updated based on the empirical quantile function at each point where the constraint is enforced. The restriction is only updated if the approximation gives a pointwise probability of violation smaller than the one calculated by Monte Carlo sampling. We show that the algorithm finds a solution of the original problem that respects the chance constraint in a finite number of iterations. We do not verify the optimality condition of the calculated point for the original problem since such conditions are not available for general nonlinear systems with probability constraints. However, we assume that the result is a heuristically optimal point, since it is an optimal point of the approximated problem. Our main assumption is that there exists a solution where the probability constraint is not active, i.e., the problem has a feasible point that is not the exact user-defined upper bound for probability of violation.

We demonstrate the algorithm in a simple bioreactor case study adapted from literature (Visser et al., 2000; Fu et al., 2015; Schultz et al., 2020b). The results show that the algorithm finds a feasible solution after a small number of iterations. We also show that using existing approaches that enforce the approximated constraint on a finite number of points and continuously update the restriction may fail, if the number of points is small. Additionally, if we use the same number of points as initial setup of the proposed algorithm, a feasible solution is found. The solution is found in a shorter CPU time and after solving fewer NLPs when compared to using a fixed number of pointwise constraints where only the restriction is updated along the iterations.

In the next section we discuss different approaches to solving optimization problems with uncertain variable. Thereafter, we state the dynamic optimization problem with path chance constraints. Section 4 presents the development of the algorithm to solve the dynamic optimization problem, and a case study is presented in Section 5. Section 6 presents the conclusions and possible future work.

2. Background information

Uncertain variables are often described by continuous probability density functions (PDFs) or bounded distributions, e.g., Latin hypercube, ellipsoid. Different methods are available in the literature to handle uncertainty during optimization depending on how the uncertainty is characterized. The main approaches are (Ruppen et al., 1995; Puschke et al., 2017):

- Worst-case optimization, also known as robust optimization: this method enforces the constraint for the worst-case realization of the uncertain variable (Ruppen et al., 1995), resulting in a min-max problem (Dimitriadis and Pistikopoulos, 1995). This approach has the main advantage that the solution is guaranteed to be feasible. However, the solution is conservative (Kadam et al., 2007) in a statistical sense, i.e., for most realizations of uncertain variable(s) (Ruppen et al., 1995). Robust optimization is applied in many studies regarding optimal control (Dimitriadis and Pistikopoulos, 1995; Mohideen et al., 1996; Ma et al., 1999; Ma and Braatz, 2001; Nagy and Braatz, 2003; Diehl and Bjornberg, 2004; Nagy and Braatz, 2004; Diehl et al., 2008; Houska et al., 2012; Shi et al., 2016; Houska et al., 2018; Puschke et al., 2018), and the main difficulty is the identification of the worst-case scenario, as this depends on the optimization variables. For nonlinear dynamic systems, this involves solving a global dynamic optimization problem, e.g., Mohideen et al. (1996). Although methods exist for solving dynamic problems globally (Chachuat et al., 2006; Lin and Stadtherr, 2007; Zhao and Stadtherr, 2011; Scott et al., 2013; Houska and Chachuat, 2014, 2019), no solvers are currently available (Puschke et al., 2018) and the computational time is usually prohibitive even for small problems.

- Multi-scenario optimization (Huang et al., 2009; Puschke and Mitsos, 2016; Puschke et al., 2017): the uncertain variable sets are discretized and the constraints are enforced for finite low-cardinality subsets. This approach is advantageous because it allows the selection of subsets that are more likely to occur, instead of worst-case values that may never happen, resulting in less conservative solutions. However, selecting a good subset is non-trivial, so heuristics are usually employed (Puschke and Mitsos, 2016; Puschke et al., 2017). The main drawback of this approach is that it needs to simulate the dynamic model for each subset, which increases the solution time.

- Probabilistic approach (Terwiesch et al., 1994, 1998; Wendt et al., 2002; Srinivasan et al., 2003; Li et al., 2004; Nagy and Braatz, 2007; Li et al., 2008; Ricardez-Sandoval, 2012; Telen et al., 2015; Nimmegeers et al., 2016; Jiang et al., 2017; Maußner and Freund, 2018a,b): the uncertain variables are approximated using known PDFs and constraints are defined as chance constraints. The optimization method enforces a given threshold probability of satisfying the constraint (Wendt et al., 2002). The choice of probability thresholds provides a trade-off between reliability and performance. The main problem is to predict the probability of the output constraint for nonlinear models (Wendt et al., 2002). For example, a nonlinear transformation of a normal distribution may result in a non-normal distribution (Terwiesch et al., 1994;

Ricardez-Sandoval, 2012). Additionally, estimating the PDFs of the parameters may be a difficult task, because of ill-posed inverse problems, scarce data and expensive experimentation setups. However, different methods exist to perform parameter estimation as shown in Tarantola (2005).

Additionally, hybrid approaches can also be employed. For example, some recent works have addressed optimization problems where the uncertain variables are partially stochastic and described by a PDF, and partially are described by a closed interval (van Ackooij et al., 2016; González Grandón et al., 2017; Adelhütte et al., 2020; Berthold et al., 2021). For example, see the following chance constraint:

$$P(g(u, p, t) \leq 0 \, \forall \, t \in [0, t_{\mathrm{f}}]) > \xi$$

where u is the decision variable, p is an uncertain variable described by a PDF and t is an uncertain variable described by a closed interval. In this case, the probability constraint is related to probability of g to be respected over the whole interval $[0, t_{\mathrm{f}}]$ for a defined u. An optimization problem with this type of constraint can be viewed as a generalised SIP, that is often more difficult to solve.

In this work, we consider probabilistic approach, where the uncertain variables are described by Gaussian distributions. This choice is motivated by the fact that characterization of uncertain variables by Gaussian distribution is a common approach when solving optimization problems under uncertainty (e.g., Jazwinski (1970); Ierapetritou and Pistikopoulos (1996); Schwarm and Nikolaou (1999); Li et al. (2004). Two key considerations in solving this problem are constraint formulation and constraint enforcement. Optimization problems may have joint probability constraints, where different state constraints must be satisfied with a desired probability, or single chance constraints where the probability of satisfaction for each state constraint is enforced. The probability may also be related to the value of the constraint at fixed point of the domain (e.g., the concentration of a product at the final time), a finite subset of points (e.g., the temperature every one minute), or over a continuous part of the domain (e.g., the probability that the maximum pressure of a reactor is over the maximum allowed in the equipment must be below 1% during the whole operation). The main difficulty in enforcing a probability constraint over a continuous part of the domain is that it results in a semi-infinite optimization problem, i.e., a problem with infinitely many probabilistic constraints.

3. Problem statement

We are interested in solving dynamic optimization problems with finite degrees of freedom $u \in U = \{u \in \mathbb{R}^{n_u} : u_{l,i} \leq u_i \leq u_{u,i}, i = 1, \ldots, n_u\}$, i.e., after control vector parametrization. A system of ordinary differential equations describes the model dynamics and the states are obtained by integration of the system along the independent variable $t \in T = [t_0, t_f]$, for given initial conditions, parameters and controls. The model has time-invariant uncertain parameters $\theta \in \Theta \subset \mathbb{R}^{n_\theta}$. We make the following assumption regarding the parameters:

Assumption 1. *Each component of θ is an independent Gaussian distributed random variable, θ_i, with known mean value, $\bar{\theta}_i$, and standard variation, σ_{θ_i}*

We employ an approach similar to Puschke et al. (2018), where the objective function is optimized for the nominal value of the parameters $\bar{\theta}$ and robustness is regarding constraint satisfaction, resulting in the following optimization problem:

$$\min_{u \in U} \varphi\left(x(t_f, u, \bar{\theta})\right) \tag{1a}$$

$$\text{s.t.} \quad \dot{x}(t, u, \theta) = f(x(t, u, \theta), t, u, \theta)), \quad \forall \quad t \in T, \tag{1b}$$

$$x(t_0, \theta) = x_0(\theta), \tag{1c}$$

$$\Pr(g_i(x(t_r, u, \theta_r), t_r, u, \theta_r) > 0) \le \xi_i, \quad i = 1, ..., n_g \tag{1d}$$

where $x(t, u, \theta) \in X \subset \mathbb{R}^{n_x}$ represents the differential variables with initial conditions $x_0(\theta) \in X \subset \mathbb{R}^{n_x}$. The objective function is given by $\varphi : X \to \mathbb{R}$ and $f : X \times T \times U \times \Theta \to \mathbb{R}^{n_x}$ describes the differential equations. The problem has n_g single chance path constraints, with each constraint defined by $\Pr(\cdot) - \xi \le 0$ where ξ is a given tolerance level. The constraint is related to a probability that g_i, $g_i : X \times T \times U \times \Theta \to \mathbb{R}$, is greater than 0, i.e., probability of violating g_i. Note that the probability constraint is not enforced for all $t \in T$, instead Eq. (1d) is the probability that g_i is greater than 0 for a random point $\{t_r, \theta_r\}$, with t_r and θ_r sampled from T and Θ, respectively. This is different from previous work (Srinivasan et al., 2003; Telen et al., 2015; Shi et al., 2016), where pointwise constraints are imposed for a finite number of points on T, resulting in a chance constraint for each point. The objective function and each path constraint g_i are assumed to be sufficiently smooth.

The rationale behind this probabilistic approach is to cover general dynamic models, without many assumptions about the dynamic model. Enforcing the constraint for all $t \in T$ is equivalent to imposing the constraint $\max_{t \in T} g_i(x(t, u, \theta, t, u, \theta) \le 0$ for fixed u and with a defined probability over Θ. Without further assumptions on g_i and the dynamic model (e.g., linearity, convexity, monotonicity), the max function may be non-smooth or discontinuous (see for example Berthold et al. (2021)). With the current formulation, we cover a broader class of dynamic models, for which we make the follow less restricted assumption:

Assumption 2. *g_i is a Carathéodory function, i.e., $g_i(\cdot, t, \cdot, \theta)$ is continuous on $X \times U$ for a.e. $t \in T$ and $\theta \in \Theta$, and $g_i(x(\cdot, u, \cdot), \cdot, u, \cdot)$ is measurable on $T \times \Theta$ for every $u \in U$.*

Assumption 2 is necessary to guarantee that the probability constraint can be calculated with a finite number of samples by Monte Carlo integration (Proposition 5.29, Shapiro et al. (2014)). If this assumption is not valid, we cannot calculate the probability of constraint violation based on Monte Carlo integration and the algorithm cannot guarantee the solution is a true feasible point of the problem.

Assumption 3. *Problem* (1) *has a feasible solution u^* such that for any $\gamma > 0$, there exists $u \in U$ with $\|u - u^*\| \le \gamma$ and* $\Pr(g_i(x(t, u, \theta), t, u, \theta) > 0) < \xi_i, \quad i = 1, ..., n_g$.

Assumption 3 refers to the existence of a feasible solution of problem (1) where the probability constraint is not active. More details are given by Shapiro et al. (2014). The proposed algorithm will fail to find a solution if the only possible solution to problem (1) only exists when $\Pr(g_i(x(t,u,\theta),t,u,\theta) > 0) = \xi_i$.

4. Solution Algorithm

In this section, we first give a general overview of the algorithm, followed by a detailed description of each step. The algorithm solves problem (1) by solving a sequence of subproblems. The first subproblem consists of an approximated dynamic optimization problem (ADOP) where we transform problem (1) into an NLP, as follows:

1. Propagate the parameter θ probability density function, $pdf(\theta)$, through the dynamic system following the method of Srinivasan et al. (2003) and generate an approximate posterior distribution $pdf(g_i)$ for each $t \in T$. Remember that θ is Gaussian distributed with known mean and variance. We briefly describe the method in section 4.1.

2. Generate a path constraint based on the mean value, $\bar{g}_i$, and standard deviation, σ_{g_i}, of the approximated $pdf(g_i)$, $c_i(t,u) = \bar{g}_i(t,u) + \beta_i(t)\sigma_{g_i}(t,u) \leq 0$, where $\beta_i(t)$ is a backoff function. Note that c_i is based on the cumulative distribution function of g_i. We use a function instead of a constant value to be able to correct the approximation along the domain.

3. Impose a restriction $c_i(t,u) \leq -\varepsilon_i(t)$ on the path constraint. Here the restriction is also a function. The reason is that the path constraint will be transformed later in a set of pointwise constraints over the domain, and the restriction of each pointwise constraint is different along the domain.

4. Replace the system of differential equations and the probabilistic constraint in (1) by the propagation equations and deterministic path constraint, respectively.

5. Impose the constraint $c_i(t,u)$ on a finite number of points $T^{\mathrm{d}} = \{t_1, \ldots, t_j, t_{j+1}, \ldots, t_n\} \in T$.

The ADOP is only an NLP approximation of the original problem. Thus, for each ADOP solution, the algorithm computes the probability of constraint violation, $p_i(u) = \Pr(g_i(x(t,u,\theta),t,u,\theta) > 0)$, by Monte Carlo integration. The probability is computed for a user-specified confidence level $\delta_i \in (0,1)$. Note that δ_i is different from ξ_i (ξ_i is the probability of constraint violation and δ_i is the confidence level used to compute (1d) by Monte Carlo integration). Additionally, the algorithm also calculates the probability of constraint violation on each point $t_j \in T^{\mathrm{d}}$ and for each interval $(t_j, t_{j+1}]\ \forall\ t_j \in T^{\mathrm{d},*} = \{t_0\} \cup T^{\mathrm{d}} \backslash \{t_{\mathrm{f}}\}$.

The algorithm proceeds by calculating a local optimal point u^k for the approximated problem, where the superscript k denotes the iteration number. Then, it computes $p_i(u^k) \pm \epsilon_{\mathrm{pi}}$ by Monte Carlo integration, where ϵ_{pi} is the error of Monte Carlo integration based on the variance of the expected value of the integration, for a confidence level δ_i. If $p_i(u^k)+\epsilon_{\text{pi}} > \xi_i$, then u^k is an infeasible point of problem (1), i.e., the chance constraint is not respected. The algorithm updates T^{d}, $\beta_i(t)$ and $\varepsilon_i(t)$, and solves the approximated problem again. If $p_i(u^k) + \epsilon_{\text{pi}} \leq \xi_i$, then the solution is a feasible solution of the problem (1) and the algorithm terminates and returns u^k as a heuristically optimal point of problem (1), since it is a local optimal point of the approximated problem and it is guaranteed feasible solution of problem (1) with a confidence level δ_i.

In the next section, we describe how to propagate the uncertainty through the dynamic system and generate $c_i(t,u)$. Then, the approximated problem is stated, followed by the Monte Carlo algorithm used in this work. Next, we introduce the procedure to update $\beta_i(t)$, $\varepsilon_i(t)$ and T^{d}. Finally, we present the algorithm to solve (1) and prove its finite termination.

4.1. Uncertainty propagation

The propagation follows the procedure employed by Srinivasan et al. (2003). We assume that $x(t,u,\theta)$ can be described by a Gaussian distribution given u and for each $t \in T$. We heuristically approximate the expected value $\bar{x}(t,u,\bar{\theta})$ of $x(t,u,\theta)$ by integration of the differential equations (1b) for the mean value of the parameters, $\bar{\theta}$. To calculate the variance of the states at each point t, we first need to integrate the first-order derivative $S_{lin}(t,u,\bar{\theta}) = \frac{\partial x}{\partial \theta}|_{t,u,\bar{\theta}}$. Then, we use a first-order Taylor expansion to calculate the variance of $x(t,u,\theta)$, $V_x(t,u,\bar{\theta},\sigma_\theta)$, based on the known variance of θ, V_θ, that is assumed to be a square diagonal matrix, resulting in $V_x(t,u,\bar{\theta},\sigma_\theta) = S_{lin}(t,u,\bar{\theta})V_\theta S_{lin}^\top(t,u,\bar{\theta})$. Note that this propagation is only exact for linear systems. For nonlinear systems, the distribution describing the state at each point t is most likely not Gaussian and the mean value and standard deviation are not the values calculated by the propagation equations (see for example the graphs of Ricardez-Sandoval (2012)). The procedure results in the following system of equations:

$$\dot{\bar{x}}(t,u,\bar{\theta}) = f(\bar{x}(t,u,\bar{\theta}),t,u,\bar{\theta}), \quad \bar{x}(t_0) = \bar{x}_0 \tag{2a}$$

$$\dot{S}_{lin}(t,u,\bar{\theta}) = \left.\frac{\partial f}{\partial x}\right|_{t,u,\bar{\theta}} S_{lin}(t,u,\bar{\theta}) + \left.\frac{\partial f}{\partial \theta}\right|_{t,u,\bar{\theta}}, \quad S_{lin}(t_0) = S_{lin,0} \tag{2b}$$

$$V_x(t,u,\bar{\theta},\sigma_\theta) = S_{lin}(t,u,\bar{\theta})V_\theta S_{lin}^\top(t,u,\bar{\theta}). \tag{2c}$$

Analogously, the approximated expected value of a constraint $\bar{g}_i$, that may be a single state or a nonlinear function of the states, is calculated by the mean value of the parameters, i.e., $\bar{g}_i(x(t,u,\bar{\theta}),t,u,\bar{\theta}) = g_i(x(t,u,\bar{\theta}),t,u,\bar{\theta})$. And the variance of the constraint, $V_{g_i}(t,u,\bar{\theta},\sigma_\theta)$, is calculated using the chain rule:

$$V_{g_i}(t,u,\bar{\theta},\sigma_\theta) = \left(S_{lin}(t,u,\bar{\theta})\left.\frac{\partial g_i}{\partial x}\right|_{t,u,\bar{\theta}}\right) V_\theta \left(S_{lin}(t,u,\bar{\theta})\left.\frac{\partial g_i}{\partial x}\right|_{t,u,\bar{\theta}}\right)^\top. \tag{3}$$

The standard deviations of the states $\sigma_x(t,u,\bar{\theta},\sigma_\theta)$ and the constraints $\sigma_{g_i}(t,u,\bar{\theta},\sigma_\theta)$ are the square root of the diagonal elements of $V_x(t,u,\bar{\theta},\sigma_\theta)$ and $V_{g_i}(t,u,\bar{\theta},\sigma_\theta)$, respectively.

4.2. Approximated optimization problem

In this section, we define the approximated problem that the algorithm solves in each iteration. We first replace the original system of differential equations Eq. (1b) and (1c) by Eqs. (2a) and we add the propagation equations of the mean value and standard deviation of the states Eqs. (2b) and (2c) to the system. Next, using the cumulative distribution function of g_i, we generate the constraint:

$$c_i(\bar{x}, t, u, \bar{\theta}, \sigma_\theta) = \bar{g}_i(\bar{x}, t, u, \bar{\theta}) + \beta_i(t)\sigma_{g_i}(t, u, \bar{\theta}, \sigma_\theta) \tag{4}$$

where $\beta_i(t)$ is a backoff described by a piecewise constant function, i.e., $\beta_i(t) = \tilde{\beta}_{i,j}$, $\forall\, t \in (t_j, t_{j+1}], j = \{0, \dots, N_{\beta,i} - 1\}$, where $N_{\beta,i}$ is the number of line segments of β_i, and $\tilde{\beta}_{i,j} \in \mathbb{R}_{\geq 0}$. We omit the arguments of $\bar{x}$ to simplify the notation. Different from previous works that use a constant β, we use a piecewise function to have a better approximation of the real probability distribution along the domain. This way, each coefficient $\tilde{\beta}_{i,j}$ is used to correct c_i in order to have the probability of violation between t_j and t_{j+1} close to the probability calculated by Monte Carlo integration.

Although it is not an algorithm requirement, the initial discretization of $\beta_i(t)$ is usually the same discretization applied to the controls (via control vector parametrization). As we discuss in a previous work (Schultz et al., 2020b), this is a common practice since the algorithm needs to compute the model states and sensitivities on those points. Thus the length of line segments is not necessarily uniform, but, without loss of generality, we employ the same discretization applied to the path constraint, which usually coincide with the control discretization in the first iteration. We do not investigate the impact of this initial setting on the algorithm performance because the impact depends on the model structure and the initial guess, u^0, what could result in misleading conclusions if analyzed for only simple case studies.

Observe that setting $\beta_i(t) = \sqrt{2}\text{erf}^{-1}(1 - 2\xi_i)$ and imposing $c_i(\bar{x}, t, u, \bar{\theta}, \sigma_\theta) \leq 0$, $\forall\, t \in T$ guarantees that Eq. (1d) is respected when f and g_i are linear functions. However, dynamic models describing industrial processes are usually nonlinear, and the propagation of the uncertainty from θ to x and g_i does not necessarily result in a Gaussian distribution (Ricardez-Sandoval, 2012). For nonlinear systems, enforcing $c_i(\bar{x}, t, u, \bar{\theta}, \sigma_\theta) \leq 0$, $\forall\, t \in T$, only approximates the constraint Eq. (1d). Moreover, the approximation can over-estimate or underestimate the probability. Therefore, we impose a restriction on c_i to guarantee that after a finite number of iterations the new constraint overestimates the probabilistic constraint. Then, we have the path constraint:

$$c_i(\bar{x}, t, u, \bar{\theta}, \sigma_\theta) = \bar{g}_i(\bar{x}, t, u, \bar{\theta}) + \beta_i(t)\sigma_{g_i}(t, u, \bar{\theta}, \sigma_\theta) \leq -\varepsilon_i(t),\ \forall\, t \in T \tag{5}$$

where $\varepsilon_i(t) = \tilde{\varepsilon}_{i,j}$, $\forall\, t \in (t_j, t_{j+1}], j = \{0, \dots, N_{\beta,i} - 1\}$ is a piecewise constant function with $\tilde{\varepsilon}_{i,j} \in \mathbb{R}_{\geq 0}$. We use a piecewise constant function to impose different restrictions over the domain based on the error of the distribution function on each point t_j. Note that the error of c_i is reduced by adding more piecewise line segments to $\varepsilon(t)$ and $\beta(t)$, for a fixed u. The algorithm iteratively improves the approximation by increasing the number of line segments of both functions along the iterations.

Additionally, we impose the path constraint in a finite number of points $T^{\mathrm{d}} = \{t_1, \dots, t_j, t_{j+1}, \dots, t_n\} \in T$. Although it is not a requirement, we employ the same number of segments for $\beta_i(t)$ and $\varepsilon_i(t)$. Moreover, T^{d} defines the points where the piecewise constant functions $\beta_i(t)$ and $\varepsilon_i(t)$ change their constant values. Substituting the probability constraint, Eq. (1d), by Eq. (5) and imposing the constraint only on the finite subset T^{d}, results in the following optimization problem:

$$\min_{u \in U} \varphi\left(\bar{x}(t_f, u, \bar{\theta})\right)) \tag{6a}$$

$$\text{s.t.} \quad c_i(\bar{x}, t, u, \bar{\theta}, \sigma_\theta) \leq -\varepsilon_i(t), \; \forall\, t \in T^{\mathrm{d}}, \quad i = 1, ..., n_g, \tag{6b}$$

where $c_i(u, t)$ is given by Eq. (4). $\bar{x}(t, u, \bar{\theta})$ is calculated by Eq. (2a), $\sigma_x(t, u, \bar{\theta}, \sigma_\theta)$ is computed by the propagation equations (2b) and (2c), where the standard deviation is the square root of the diagonal elements of V_x. $\bar{g_i}(\bar{x}, t, u, \theta)$ is calculated using the mean value of θ and Eq. (3) computes the variance $V_{g_i}(t, u, \bar{\theta}, \sigma_\theta)$, where the square root of the diagonal elements gives $\sigma_{g_i}(t, u, \bar{\theta}, \sigma_\theta)$.

4.3. Monte Carlo Integration

Each solution $\bar{u}$ of problem (6) may be either a feasible or infeasible solution point of the DOP. We employ Monte Carlo integration, for a user-specified confidence $\delta_i \in (0, 1)$, to compute the probability of constraint violation $\bar{u}$, $p_i(\bar{u}) = Pr(g_i(x(t, \bar{u}, \theta), t, \bar{u}, \theta) > 0)$ for the candidate solution. Before introducing the algorithm to calculate $p_i(\bar{u})$, let us first define the probability of constraint violation over T and Θ. Note that $t \in T$ can be viewed as an additional parameter with a probability density function equal to $1/(t_f - t_0)$, i.e., the probability of any point $t \in [t_0, t_f]$ to occur is the same.

$$p_i(\bar{u}) = \int_{-\infty}^{\infty} \cdots \int_{-\infty}^{\infty} \int_{t_0}^{t_f} pdf(\theta_1) \dots pdf(\theta_{n_\theta}) pdf(t) \; \mathbb{1}_{(0,\infty)}(g_i(x(t, \bar{u}, \theta), t, \bar{u}, \theta)) \, dt \, d\theta_1 \dots d\theta_{n_\theta} \tag{7}$$

where the indicator function $\mathbb{1}_{(0,\infty)}$ is:

$$\mathbb{1}_{(0,\infty)}(g_i(x(t, \bar{u}, \theta), t, u, \theta)) = \begin{cases} 0, \text{ if } g_i(x(t, \bar{u}, \theta), t, \bar{u}, \theta) \leq 0 \\ 1, \text{ if } g_i(x(t, \bar{u}, \theta), t, \bar{u}, \theta) > 0 \end{cases}$$

Note that the inner integral of Eq. (7) describes the probability of violation over T. Since T is uniformly distributed, we have

$$p_{i,t}(\bar{u}, \theta) = \frac{\int_{t_0}^{t_f} \mathbb{1}_{(0,\infty)}(g_i(x(t, \bar{u}, \theta), t, \bar{u}, \theta)) \, dt}{t_f - t_0}$$

that is easily calculated by integration of the dynamic system over T. On the other hand, the integrals over Θ are expensive to evaluate, motivating the use of Monte Carlo method. Therefore, we compute $p_i(\bar{u})$ with N realizations of Θ by:

$$p_{i,N}(\bar{u}) = N^{-1} \sum_{m=1}^{N} p_{i,t}(\bar{u}, \hat{\theta}^{(m)}) \tag{8}$$

where N is the number of realizations of the normal distributed vector θ, resulting in $\hat{\theta} \in \mathbb{R}^{n_\theta \times N}$. The superscript (m) denotes the value of θ in each realization.

Since $g_i(x(t, \bar{u}, \theta), t, \bar{u}, \theta)$ is a Caratheodory function, $p_{i,N}(\bar{u}) \to p_i(\bar{u})$ when $N \to \infty$, by the law of large numbers (Shapiro et al., 2014). If N is large, then $p_{i,N}(\bar{u}, \theta)$ behaves as Gaussian-distributed and we can define a confidence interval for $p_i(\bar{u})$ as $[p_{i,N}(\bar{u}) - \epsilon_{pi}, p_{i,N}(\bar{u}) + \epsilon_{pi}]$. The error of the integration is given by $\epsilon_{pi} = z_i s_{p_{i,t}}$, where $s_{p_{i,t}}$ denotes the sample standard deviation of $p_{i,t}$ and $z_i = \sqrt{2}\mathrm{erf}^{-1}(\delta_i)$. The solution $\bar{u}$ may have three different outcomes:

(1) $p_{i,N}(\bar{u}) + \epsilon_{pi} \leq \xi_i, i = 1, \dots, n_g$: $\bar{u}$ respects the chance constraint of problem (1) with confidence level δ;

(2) $p_{i,N}(\bar{u}) - \epsilon_{pi} > \xi_i$: $\bar{u}$ is infeasible;

(3) $p_{i,N}(\bar{u}) - \epsilon_{pi} \leq \xi_i < p_{N,i}(\bar{u}) + \epsilon_{pi}$: nothing can be stated about the feasibility of $\bar{u}$. It is necessary to increase N to reduce ϵ_{pi}.

An initial number of samples N^k results in $p^k_{i,N} \pm z_i s^k_{p_{i,t}}$. Since the error of Monte Carlo integration is proportional to $\frac{1}{\sqrt{N}}$, we update the required number of samples as:

$$N^{k+1} = N^k \left(\max_{i \in \{1,\dots,n_g\}} \left(\frac{\epsilon^k_{pi}}{|\tilde{p}^k_{i,N} - \xi_i|} \right) \right)^2 . \tag{9}$$

Algorithm 1 proceeds by drawing an initial number of samples N^0, specified by the user. Then, it calculates $p^0_{i,N} \pm \epsilon^0_{pi}$. If $p_{i,N}(\bar{u})^0 - \epsilon^0_{pi} > \xi_i$ for any $i \in 1, \dots, n_g$, then $\bar{u}$ is infeasible . If $p^0_{i,N}(\bar{u}) + \epsilon^0_{pi} \leq \xi_i, i = 1, \dots, n_g$, then the algorithm returns $\bar{u}$ is a feasible point. Otherwise, the required number of samples is updated using Eq. (9) and we draw $N^{k+1} - N^k$ additional samples and calculate the new $p^{k+1}_{i,N} \pm \epsilon^{k+1}_{pi}$. The algorithm repeats the procedure until $\bar{u}$ is either a feasible or unfeasible point for a δ_i confidence.

Algorithm 1 also calculates the probability of constraint violation $p^{(t_j,t_{j+1})}_{i,N}(\bar{u})$ on each interval $(t_j, t_{j+1}]$, for each $t_j \in T^{\mathrm{d}^*} = t_0 \cup T^{\mathrm{d}} \backslash t_{\mathrm{f}}$, and the empirical cumulative probability distribution function of g_i, $\hat{G}^{t_j}_{i,N}(\bar{u})$, for each point $t_j \in T^{\mathrm{d}}$. Both are used to iteratively update $\beta_i(t)$ and $\epsilon_i(t)$ as described in the next section. Algorithm 1 computes $p^{(t_j,t_{j+1})}_{i,N}(\bar{u})$ using Eq. (8), replacing $p_{i,t}(\bar{u}, \theta)$ by:

$$p^{(t_j,t_{j+1})}_{i,g}(u, \theta) = \lim_{t^*_j \to t_j} \frac{\int_{t^*_j}^{t_{j+1}} \mathbb{1}_{(0,\infty)}(g_i(x(t, u, \theta), t, u, \theta))\, dt}{t^*_j - t_{j+1}} \tag{10}$$

where $p^{(t_j,t_{j+1})}_{i,g}$ denotes the probability of constraint violation in the interval $(t_j, t_{j+1}]$.

Algorithm 1 Monte Carlo Integration

```
 1: inputs : Initial number of samples, $N^0$; desired confidence, $\delta_i$; candidate optimal solution, $\bar{u}$; maximum approximation error, $\varepsilon_{\text{app}}$; iteration counter $k = 0$;
 2: repeat
 3:     $k \leftarrow k + 1$
 4:     Compute $p^k_{N,i}(\bar{u})$ and $\epsilon^k_{pi}(\bar{u}),\ i = 1, \dots, n_g$ by Monte Carlo integration
 5:     Compute $p^{(t_j,t_{j+1}),k}_{i,N}(\bar{u})$, for each $t_j \in T^{\text{d}*},\ i = 1, \dots, n_g$ using Eqs. (8) and (10)
 6:     Compute $\hat{G}^{t_j,k}_{i,N}(\bar{u}),\ t_j \in T^{\text{d}},\ i = 1, \dots, n_g$ using the value of $g_i$ at each point $t_j$ for each element of $\hat{\theta}$
 7:     if $p^k_{i,N}(\bar{u}) - \epsilon^k_{pi}(\bar{u}) > \xi_i$ for any $i \in 1, \dots, n_g$ then
 8:         Return $p^k_{i,N}(\bar{u})$, $\epsilon^k_{pi}(\bar{u})$, $p^{(t_j,t_{j+1}),k}_{i,N}(\bar{u})$, $\hat{G}^{t_j,k}_{i,N}(\bar{u})$,
 9:         Terminate.
10:     else if $p^k_{i,N}(\bar{u}) + \epsilon^k_{pi}(\bar{u}) \leq \xi_i, i = 1, \dots, n_g$ then
11:         Return $p^k_{i,N}(\bar{u})$, $\epsilon^k_{pi}(\bar{u})$, $p^{(t_j,t_{j+1}),k}_{i,N}(\bar{u})$, $\hat{G}^{t_j,k}_{i,N}(\bar{u})$
12:         Terminate.
13:     else
14:         $N^{k+1} \leftarrow N^k$, using Eq. (9)
15:         Draw more $N^{k+1} - N^k$ samples from $\theta$
16:     end if
17: until $p^k_{i,N}(\bar{u}) - \epsilon^k_{pi}(\bar{u}) > \xi_i$ for any $i \in 1, \dots, n_g$ or $p^k_{i,N}(\bar{u}) + \epsilon^k_{pi}(\bar{u}) \leq \xi_i, i = 1, \dots, n_g$
```

4.4. Update of back-off, restriction, and pointwise constraints

Along the iterations, $\beta_i(t)$, $\epsilon_i(t)$, and T^{d} are updated. Let the superscript k denote the iteration index. The update policy consists of halving the intervals $(t_j, t_{j+1}]$ where $p^{(t_j,t_{j+1})}_{i,N}(u) > \frac{\xi_i(t_j - t_{j+1})}{(t_0 - t_{\text{f}})}$:

$$T^{\text{d},k+1} = T^{\text{d},k} \cup \left\{ \frac{t_{j+1} + t_j}{2} \middle| \, t_j \in T^{\text{d}*,k} \wedge \tilde{p}^{(t_j,t_{j+1})}_{i,N}(u^k) > \frac{\xi_i(t_j - t_{j+1})}{(t_0 - t_{\text{f}})} \right\}. \tag{11}$$

The algorithm updates $\beta(t)$ using a linear correlation between the ξ_i and $p^{(t_j,t_{j+1}),k}_{i,N}(u^k)$:

$$\tilde{\beta}^{k+1}_{i,t_j} = \tilde{\beta}^k_{i,t_j} \frac{1 - \xi_i \frac{t_{j+1} - t_j}{t_{\text{f}} - t_0}}{1 - p^{(t_j,t_{j+1}),k}_{i,N}(u^k)},\ \ t_j \in T^{\text{d}*,k} \tag{12}$$

$$\beta^{k+1}_i(t) = \tilde{\beta}^{k+1}_{i,t_j},\ \forall\, t \in (t_j, t_{j+1}],\ \ j = 0, \dots, |T^{\text{d}*,k}| - 1. \tag{13}$$

Alternative update strategies could also be applied to update $\beta_i(t)$, e.g., adding more points where the system dynamics are fast. Since the dynamics may change with the parameters and control values, it may be difficult to identify where we have fast and slow dynamics over the domain. Thus, we chose to use this simple and well-known strategy (to halve the intervals) because it works for situations where the dynamic of the system is substantially affected by the inputs and parameters values, leaving the investigation of the optimal update policies based on the system dynamics for future work.

To update the restriction, the algorithm uses the empirical cumulative distribution function of g_i for each point $t_j \in T^{\mathrm{d},k}$, $\hat{G}^{t_j,k}_{i,N}(u^k)$, and the quantile function

$$Q(t, \xi_i, u^k) = \inf\{\hat{G}^{t,k}_{i,N}(u^k) \in \mathrm{I\!R} : \hat{G}^{t,k}_{i,N}(u^k) \geq 1 - \xi_i\}$$

where $Q(t, u^k, \xi_i)$ returns the threshold value of g_i below which random samples from the empirical probability distribution $\hat{G}$ fall with a probability $1 - \xi_i$.

The pointwise error of $c_i(\bar{x}, t, u^k, \bar{\theta}, \sigma_\theta)$ is given by

$$\Delta c^k_i(\bar{x}, t, u^k, \bar{\theta}, \sigma_\theta, \xi_i) = \min(0, c^{k+1}_i(0, \bar{x}, t, u^k, \bar{\theta}, \sigma_\theta) - Q(t, u^k, \xi_i)).$$

The evaluation of c_i uses the updated $\beta^{k+1}(t)$ and the restriction update policy is:

$$\tilde{\varepsilon}^{k+1}_{i,t_j} = \Delta c^k_i(\bar{x}_j, t_j, u^k, \bar{\theta}, \sigma_\theta, \xi_i),\ t_j \in T^{\mathrm{d}*,k} \tag{14}$$

where we use the simplified notation $\bar{x}_j = \bar{x}(t_j, u^k, \bar{\theta})$. Then, we have:

$$\varepsilon^{k+1}_i(t) = \tilde{\varepsilon}^{k+1}_{i,t_j},\ \forall\, t \in (t_j, t_{j+1}],\ j = 0, \ldots, |T^{\mathrm{d}*,k}| - 1 \tag{15}$$

4.5. Robust dynamic optimization algorithm

We introduce Algorithm 2 below and present the algorithm workflow in Appendix A. The inputs of the algorithm are:

- initial guess u_0;
- initial set of constraint enforcement points $T^{\mathrm{d},0}$;
- initial backoff $\beta^0_i(t) \geq 0$. We employ $\beta^0_i(t) = \sqrt{2}\mathrm{erf}^{-1}(1 - 2\xi_i)\ \forall\, t \in (t_j, t_{j+1}]$, $j = 0, \ldots, |T^{\mathrm{d}*,0}| - 1$, that is the optimal choice for the linear case, but this is not a requirement;
- initial restriction $\varepsilon^0_i(t) \geq 0$. We set $\varepsilon^0_i(t) = 0\ \forall\, t \in (t_j, t_{j+1}]$, $j = 0, \ldots, |T^{\mathrm{d}*,0}| - 1$ to avoid having a Monte Carlo integration step for the initial guess to calculate the actual error of the approximation at each point t_j;
- desired confidence level of the solution δ_i;
- initial number of samples employed by Algorithm 1;

Algorithm 2 solves problem (1) by solving a sequence of approximated problems. In each iteration, Algorithm 2 solves Eq. (6) resulting in a candidate solution u^k. The probability of constraint violation for the candidate solution u^k is calculated using Monte Carlo integration (Algorithm 1) for a confidence δ_i. Algorithm 2 updates $\beta^k_i(t)$, $\varepsilon^k_i(t)$, and $T^{\mathrm{d},k}$ when u^k is an infeasible point of problem (1). After each update the algorithm again solves the

Algorithm 2 Dynamic optimization with path chance constraint

```
1: inputs : initial guess $u^0$, initial backoff $\beta_i^0(t)$, initial restriction $\varepsilon_i(t)$, initial set of points $T^{d,0}$, confidence level $\delta$, initial number of samples $N^0$, iteration counter $k = 0$.
2: repeat
3:     $k \leftarrow k + 1$
4:     Calculate $u^k$ by solving problem (6)
5:     Compute $p_{i,N}^k(u^k)$, $\epsilon_{pi}^k(u^k)$, $p_{i,N}^{(t_j,t_{j+1}),k}(u^k)\,\forall\, t_j \in T^{\mathrm{d}*}$, $\hat{G}_{i,N}^{t_j,k}(u^k)$, $\forall\, t_j \in T^{\mathrm{d}*}$ using Algorithm 1
6:     if $p_{i,N}^k(u^k) + \epsilon_{pi}^k(u^k) > \xi$ then
7:         Update $T^{\mathrm{d},k}$ using (11)
8:         Update $\beta_i(t)^k$ using (12) and (13)
9:         Update $\varepsilon_i(t)^k$ using (14) and (15)
10:    else
11:        Return $u^k$
12:        Terminate.
13:    end if
14: until $p_{i,N}^k(u^k) + \epsilon_{pi}^k(u^k) \leq \xi$
```

approximated problem and repeats the previous steps. The algorithm terminates when u^k is a feasible point of (1) and is also an optimal point of the approximated problem.

We claim no guarantee that Algorithm 2 converges monotonically towards an optimal point u^* over the iterations. The main reason is that the models' equations are usually nonlinear and problems (1) and (6) are nonconvex. Since the subproblems are solved with local solvers, the algorithm may find different local minima in each iteration. However, we can prove that the algorithm finds an optimal point after a finite number of iterations.

Theorem 1. *Under Assumptions 1 to 3, Algorithm 2 terminates after a finite number of iterations.*

Proof. For each solution u^k of Problem (6), the algorithm corrects $\beta_i(t)$ to better approximate the probability of violation on each interval, and imposes a restriction $\varepsilon_i(t) \geq 0$, based on the error of the approximated constraint. Algorithm 2 calculates the probability of violation using Monte Carlo integration (Algorithm 1) with a finite number of samples by assumption (1) to (3). Additionally, it halves the intervals $(t_j, t_{j+1}]$, where $p_{i,N}^{(t_j,t_{j+1})}(u) > \frac{\xi_i(t_j - t_{j+1})}{(t_0 - t_\mathrm{f})}$. In the limit, after infinite iteration, $|T^\mathrm{d}| \to \infty$, $\varepsilon_i(t) \to 0$ and the error of the approximated constraint tends to zero. By assumption (3), there exists an $\bar{\epsilon} > 0$ where $\Pr(g_i(x(t,u,\theta),t,u,\theta) > 0) + \bar{\epsilon} < \xi_i$ and a sequence $\{u^k\} \subset U$ that converges to a feasible solution u^* with $\Pr(g_i(x(t,u^*,\theta),t,u,\theta) > 0) + \bar{\epsilon} < \xi_i$, $i = 1, ..., n_g$ (Shapiro et al., 2014). Therefore, by compactness of U and sufficient smoothness of f and g, there exists a finite number of points $T^\mathrm{d} \subset T$ where $\varepsilon_i(t) < \bar{\epsilon}$, $\forall\, t \in T$. Since the algorithm continuously increases the number of points in T^d, after a finite number of iteration, the solution to the approximated problem is a feasible point of problem (1) and the optimal solution to the approximated problem. □

4.6. Scalability

If the problem has more than one constraint, the algorithm scales, in principle, as the underlying single shooting algorithm. The main impact of additional constraints is the computation additional sensitivities in the optimization step. This step may increase significantly the computational time depending on the method employed. The Monte Carlo computation will have almost no additional burden, since the more expensive step during Monte Carlo computation is the integration of the dynamic model, and this step does not change with additional constraints. Only additional fast arithmetical computations will be required. This drives our conclusion that the algorithm scales as the sensitivity method employed in the single shooting method.

5. Case study

We demonstrate Algorithm 2 in a case study often used in the optimization under uncertainty literature (Srinivasan et al., 2003; Puschke et al., 2017, 2018). The case study considers a semi-batch penicillin fermentation process adapted from previous works (Visser et al., 2000; Fu et al., 2015; Schultz et al., 2020b). The adaptation consists of transforming the path constraint into a path chance constraint and assuming that the parameters are Gaussian distributed. The optimization problem is described by:

$$\begin{aligned}
&\min_{x,u} -x_3\left(t_f = 40\right)\\
\text{s.t.}\quad &\dot{x}_1(t,u,\theta) = \frac{\mu x_1(t)x_2(t)}{K_l x_1(t)+x_2(t)} - \frac{u x_1(t)}{x_4(t)},\\
&\dot{x}_2(t,u,\theta) = \frac{-\mu x_1(t)x_2(t)}{Y_{xs}(K_l x_1(t)+x_2(t))} - M_x x_1(t) - \frac{\theta_m x_1(t)x_2(t)}{Y_p(x_2(t)+K_p+\frac{x_2(t)^2}{K_i})} +\\
&\qquad u\frac{S_0 - x_2(t)}{x_4(t)},\\
&\dot{x}_3(t,u,\theta) = \frac{\theta_m x_1(t)x_2(t)}{x_2(t)+K_p+\frac{x_2(t)^2}{K_i}} - K_{xp}x_3(t) - u\frac{x_3(t)}{x_4(t)},\\
&\dot{x}_4(t,u,\theta) = u,\\
&x(t=0) = \left[1, 0.2, 0.0001, 250\right],\\
&\Pr(x_2 > 0.5) \leq 0.1,\\
&0 \leq u \leq 10,
\end{aligned}$$

where $\theta = \{\mu, \theta_m, K_i, K_l, K_p, K_{xp}, M_x, Y_p, Y_{xs}\}$ denotes the model parameters. The model parameter means and standard deviations are listed in Table 1.

The original optimization problem has one control that is a function of time, but here we use control vector parametrization to discretize the control. We use piecewise constant

Table 1: Parameters for the penicillin fermentation model.

Parameter	Mean Value	Variance	Standard deviation
μ	1.10×10^{-1}	1.10×10^{-4}	1.05×10^{-2}
θ_m	4.00×10^{-3}	4.00×10^{-6}	2.00×10^{-3}
K_i	1.00×10^{-1}	1.00×10^{-4}	1.00×10^{-2}
K_l	6.00×10^{-3}	6.00×10^{-6}	2.40×10^{-3}
K_p	1.00×10^{-4}	1.00×10^{-7}	3.16×10^{-4}
K_{xp}	1.00×10^{-2}	1.00×10^{-5}	3.16×10^{-3}
M_x	2.90×10^{-2}	2.90×10^{-5}	5.38×10^{-3}
Y_p	1.20	1.20×10^{-3}	3.46×10^{-2}
Y_{xs}	4.70×10^{-1}	4.70×10^{-4}	2.17×10^{-2}

functions with 40 equidistant intervals, resulting in 40 degrees of freedom. We test Algorithm 2 using an initial set of points $T^{\mathrm{d},0} = \{2, 4, \ldots, 40\}$, i.e., 20 equally spaced points over the domain T where the constraint is enforced. The problem is solved for a confidence $\delta = 90\%$. The remaining parameters used to solve the approximated problems at each iteration are $\varepsilon^0(t) = 0$. The algorithm is implemented in the open-source Python package *doepy*[1](Olofsson et al., 2020).

We compare the solution computed by Algorithm 2 to three alternative strategies:

1. Fixed number of time points where the constraint is enforced without updating $\beta(t)$ and T^{d}, and without restriction. We test 20 and 40 equally spaces discrete points, that we call 1NLP20 and 1NLP40 respectively.
2. 1SIP: Enforcing the approximated constraint over the whole domain, i.e., the approximated path constraint (Eq. (5)) is guaranteed to be respected for all $t \in T$. However, β is not updated and the restriction is 0. The algorithm of Schultz et al. (2020b) is employed to enforce the constraint over the whole domain.
3. Fixed number of points along the time where the constraint is enforced with adaptive restriction. We solve a sequence of NLP where the constraint is enforced on a fixed number of discrete points of the domain and a restriction is imposed on each constraint. We use 20 and 40 equally spaced discrete points, that we call nNLP20 and nNLP40, respectively. In the first run, the restriction is zero. For each solution of the NLP the probability of violation is calculated using Algorithm 1, and we increase the restriction when the solution is not a feasible point of the DOP.

We compare our proposed algorithm to the above strategies to demonstrate the importance of using different update strategies to find a feasible solution of the DOP. The first two alternatives show that increasing the number of points where the approximated constraint is enforced, or even enforcing the approximated constraint over the whole domain, is not sufficient to enforce the path chance constraint for nonlinear systems. The third strategy

[1]https://github.com/scwolof/doepy

has been employed by different authors (e.g. Srinivasan et al. (2003); Telen et al. (2015); Shi et al. (2016)) to enforce pointwise constraints. Our comparison aims to show that: (1) using only a few pointwise constraints with an adaptive restriction may not converge to a feasible point of the DOP, (2) using many pointwise constraints is not efficient.

Table 2 presents the case study comparison results with the objective function for the mean value of the parameters and the probability of violation with a 90% confidence. Firstly, we observe that solving only one SIP can significantly reduce the probability of constraint violation when compared to solving an NLP with 20 points where the constraint is enforced. Even when we increase the number of points from 20 to 40, the SIP provides lower probability of violation. Additionally, enforcing 20 pointwise constrains and imposing an adaptive restriction on each constraint still lead to path chance constraint violation. In this case, we increased the restriction up to 0.5 (maximum value allowed in order for x_2 to be positive) and the minimum probability of violation is greater than the desired value 10%. We would like to highlight that the goal of previous work (Srinivasan et al., 2003; Telen et al., 2015; Shi et al., 2016) was to guarantee the satisfaction of the pointwise chance constraint and not a path chance constraint.

Table 2: Performance (number of NLPs solved and CPU time) of the different strategies compared to Algorithm 2, and the objective function and probability of constraint violation for each calculated optimal point (u^{opt}).

Method	1NLP20	1NLP40	1SIP	nNLP20	nNLP40	Alg. 2
Obj. Fun.	-0.785	-0.758	-0.746	-0.700	-0.683	-0.675
$p_{i,N}(u^{\text{opt}})(\%)$	51.6	27.6	23.9	32.8	8.96	9.54
$\epsilon_{pi}(u^{\text{opt}})(\%)$	1.63	2.24	2.75	0.60	0.36	0.38
NLPs	1	1	2	6	21	6
CPU time(s)	1960	1220	2403	7491	8969	5103

The results show that using 40 pointwise constraints with a restriction, we can find a solution that respects the path chance constraint. Note, however, that we do not know *a priori* the number of points we need. The solution provided by Algorithm 2 satisfies the path chance constraint with 90% confidence and it does not depend on the initial number of points, since the algorithm iteratively adds more points. However, the solution seems to be more conservative than the solution using the adaptive restriction and 40 equally spaced pointwise constraints. We see that the objective function is worse than the one obtained by nNLP40. The possible reasons for this are the convergence to different local minima since we employed local NLP solvers, or too conservative tuning parameters of Algorithm 2. On the other hand, Algorithm 2 is more efficient for this case study when compared to nNLP20 and nNLP40, both approaches that also change the restriction along the iterations. It needs to solve only a few NLPs with smaller CPU time.

We also compute the path constraint for all 6 solutions for 100 random parameter sets (the same for all control strategies). The constraint is given by $x_2 - 0.5 < 0$, thus we show the curves of x_2 in Fig. 1 that must be below 0.5. We highlight that the graphs have the same limits in order to make easier to compare the results for the different controls. However, it may give the impression that most of the curves are above the 0.5 threshold. To demonstrate that this is not actually the case, we present in Figure 2 a zoomed-in graph for the controls nNLP40 and Algorithm 2 (the controls with lower probability of constraint violation), where the reader can clearly see that most of the curves do not violate the constraint. Note that the nNLP40 and Algorithm 2 solutions provide similar results regarding constraint violation. Based on the probability of constraint violation the reader can observe that Algorithm 2 provides a more conservative solution. If a less conservative policy is desired, the user can increase ξ. The optimal controls obtained for each approach are shown in Fig. 3, where we see that using 20 discrete constraints (1NLP20 and nNLP20) results in an oscillatory behavior. The root cause for this behavior is the violation of the constraint in-between the points where the optimizer enforces the constraint. The controls for the other four approaches have a similar shape and do not present oscillation.

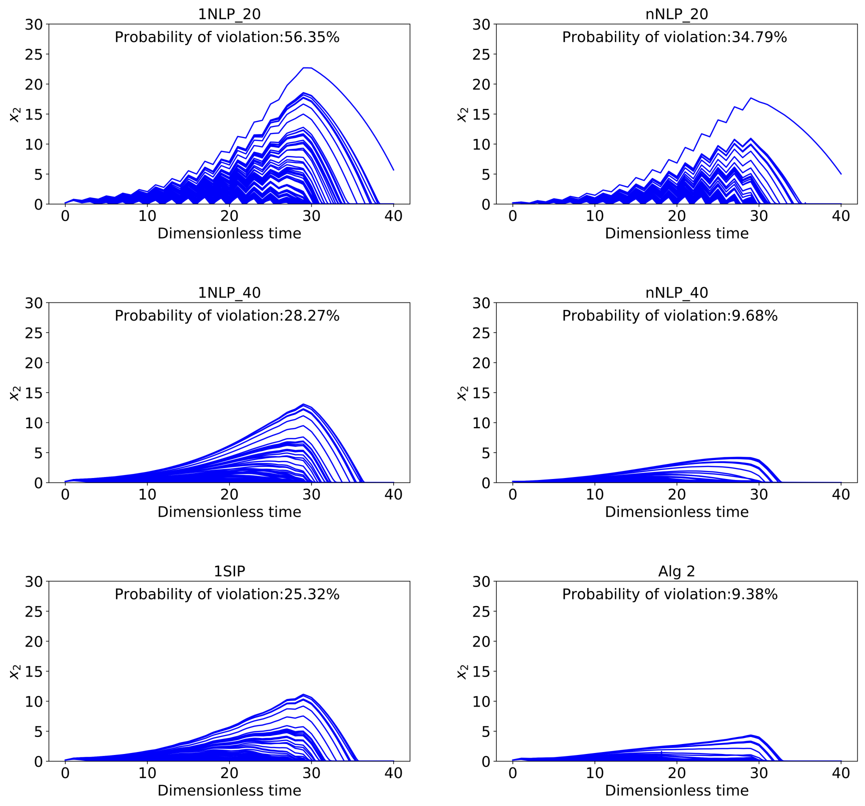

Figure 1: x_2 for 100 random parameter values. Path constraint is $x_2 < 0.5$.

Note that the robustness of a control strategy is not only evaluated based on how often a constraint is violated, i.e., by the probability of violation, but also by the magnitude of possible violations. Therefore, we calculate two different metrics to evaluate the criticality of

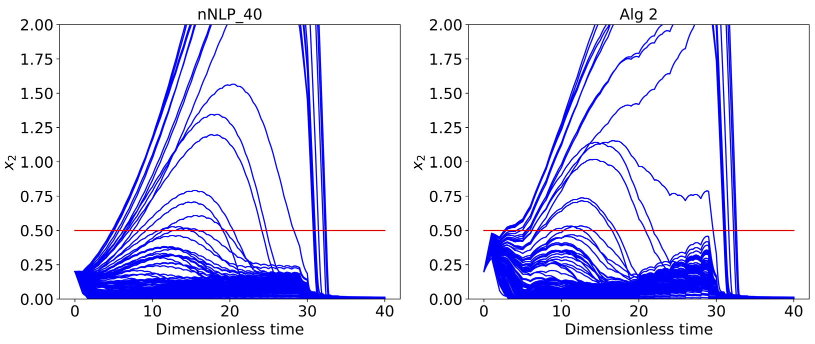

Figure 2: Zoomed-in of x_2 for nNLP40 and Alg 2 controls. Red line represents the threshold of x_2.

the violations for each solution based on the curves presented in Fig. 1: the average value of x_2 when it is above its threshold 0.5, and the maximum value of x_2. We present the results in Table 3. Comparing the two strategies that provide the lowest probability of violation (nNLP40 and Alg. 2), we observe that the maximum value of x_2 is slightly higher for the solution computed by the proposed algorithm, although the average value is lower. Based on these results, further investigation, not covered in this work, is required to understand how to change the objective function of the optimization problem to compute a solution that is a good trade-off between the probability of violation and the magnitude of possible violations.

Table 3: Magnitude of constraint violation for each solution.

Method	1NLP20	1NLP40	1SIP	nNLP20	nNLP40	Alg. 2
Average value of x_2 when $x_2 > 0.5$	2.61	2.75	2.44	1.90	1.78	1.61
Maximum value of x_2	19.1	13.6	11.7	11.3	4.46	4.64

6. Conclusions

We propose an algorithm to solve dynamic optimization problems with path chance constraints. Unlike previous work, our proposed algorithm enforces the path chance constraint over the whole domain and not only on pointwise constraints. In order to guarantee the satisfaction of the chance constraints, the algorithm solves an approximated NLP and uses Monte Carlo integration to calculate the probability of constraint violation for each NLP solution. The approximated constraint is adapted along the iterations to better approximate the original problem and new pointwise contraints are added. Thus, the algorithm does not depend on the initial number of points where constraint is enforced. The algorithm terminates after a finite number of iterations under mild assumptions. The main drawback of the algorithm is the computationally expensive procedure to propagate the uncertainty through

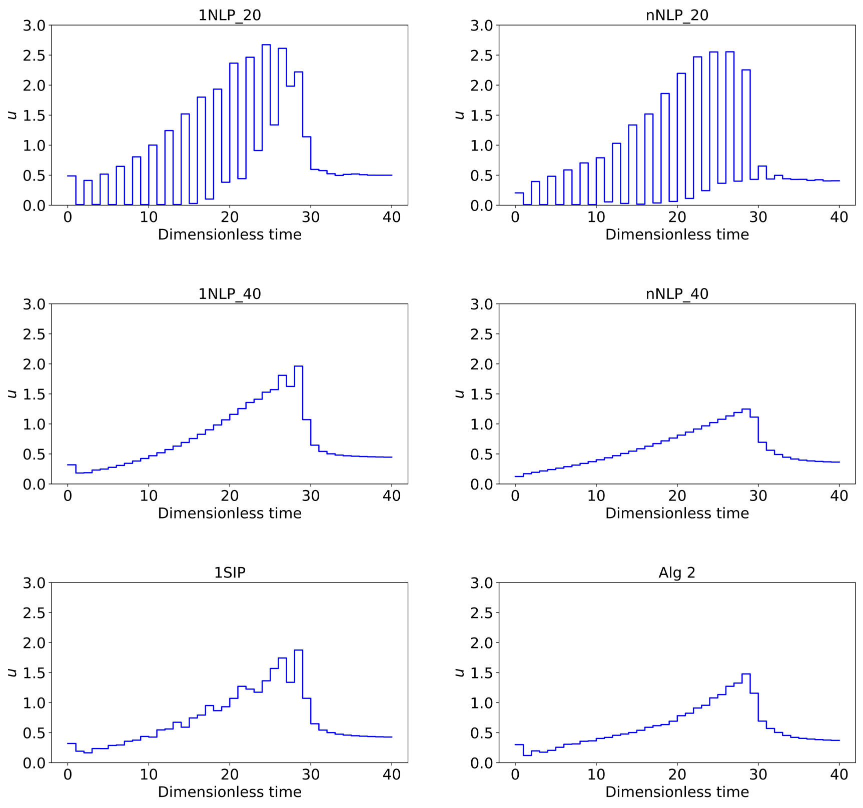

Figure 3: Calculated optimal controls for each solution strategy.

the dynamic system, which significantly increases the number of differential equations to integrate over.

We demonstrate the algorithm in a simple case study, and show that the result respects the chance constraint, but may produce in conservative solutions. Conservativeness may be reduced, though, by using different values for the tolerance level ε_i and the confidence δ_i. We also demonstrate that the discretization of the path constraint in pointwise constraint with an adaptive restriction may not work if the number of discrete points is small and fixed. Using a large number of pointwise constraints results in larger CPU times and may require the solution of many NLPs until a DOP feasible point is found. The application of the algorithm to more complex problems and the adaption to solve the problem quickly and online are the subjects of potential future work.

Authors Contributions

Eduardo S. Schultz and Alexander Mitsos developed the algorithm. Eduardo S. Schultz and Simon Olofsson developed the python code. Adel Mhamdi and Alexander Mitsos supervised the work at RWTH Aachen University.

Eduardo S. Schultz: Writing - Original draft preparation, Conceptualization, Methodology, Software; Simon Olofsson: Software, Writing - Review and Editing; Adel Mhamdi: Writing - Review and Editing, Supervision, Funding acquisition; Alexander Mitsos: Conceptualization, Methodology, Writing - Review and Editing, Supervision, Funding acquisition

Acknowledgements

This work has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement no.675251. We would like to thank Dr. Ruth Misener for supervising the work at Imperial College London.

References

van Ackooij, W., Frangioni, A., de Oliveira, W., 2016. Inexact stabilized benders' decomposition approaches with application to chance-constrained problems with finite support. Computational Optimization and Applications 65, 637–669.

Adelhütte, D., Aßmann, D., Grandón, T.G., Gugat, M., Heitsch, H., Henrion, R., Liers, F., Nitsche, S., Schultz, R., Stingl, M., Wintergerst, D., 2020. Joint model of probabilistic-robust (probust) constraints applied to gas network optimization. Vietnam Journal of Mathematics , 2308–2228.

Bemporad, A., Morari, M., 1999. Robust model predictive control: A survey, in: Robustness in identification and control. Springer, pp. 207–226.

Berthold, H., Heitsch, H., Henrion, R., Schwientek, J., 2021. On the algorithmic solution of optimization problems subject to probabilistic/robust (probust) constraints. Mathematical Methods of Operations Research , 1–37.

Chachuat, B., Singer, A.B., Barton, P.I., 2006. Global methods for dynamic optimization and mixed-integer dynamic optimization. Industrial & Engineering Chemistry Research 45, 8373–8392.

Diehl, M., Bjornberg, J., 2004. Robust dynamic programming for min-max model predictive control of constrained uncertain systems. IEEE Transactions on Automatic Control 49, 2253–2257.

Diehl, M., Gerhard, J., Marquardt, W., Mönnigmann, M., 2008. Numerical solution approaches for robust nonlinear optimal control problems. Computers & Chemical Engineering 32, 1279–1292.

Dimitriadis, V.D., Pistikopoulos, E.N., 1995. Flexibility analysis of dynamic systems. Industrial & Engineering Chemistry Research 34, 4451–4462.

Faust, J.M.M., Fu, J., Chachuat, B., Mitsos, A., 2016. Optimization of dynamic systems with rigorous path constraint satisfaction, in: 26th European Symposium on Computer Aided Process Engineering, Elsevier. pp. 643–648.

Fu, J., Faust, J.M.M., Chachuat, B., Mitsos, A., 2015. Local optimization of dynamic programs with guaranteed satisfaction of path constraints. Automatica 62, 184–192.

Geletu, A., Li, P., 2014. Recent developments in computational approaches to optimization under uncertainty and application in process systems engineering. ChemBioEng Reviews 1, 170–190.

González Grandón, T., Heitsch, H., Henrion, R., 2017. A joint model of probabilistic/robust constraints for gas transport management in stationary networks. Computational Management Science 14, 443–460.

Houska, B., Chachuat, B., 2014. Branch-and-lift algorithm for deterministic global optimization in nonlinear optimal control. Journal of Optimization Theory and Applications 162, 208–248.

Houska, B., Chachuat, B., 2019. Global optimization in hilbert space. Mathematical programming 173, 221–249.

Houska, B., Li, J.C., Chachuat, B., 2018. Towards rigorous robust optimal control via generalized high-order moment expansion. Optimal Control Applications and Methods 39, 489–502.

Houska, B., Logist, F., Impe, J.V., Diehl, M., 2012. Robust optimization of nonlinear dynamic systems with application to a jacketed tubular reactor. Journal of Process Control 22, 1152–1160.

Huang, R., Patwardhan, S.C., Biegler, L.T., 2009. Multi-scenario-based robust nonlinear model predictive control with first principle models, in: Computer Aided Chemical Engineering. Elsevier. volume 27, pp. 1293–1298.

Ierapetritou, M.G., Pistikopoulos, E.N., 1996. Batch plant design and operations under uncertainty. Industrial & Engineering Chemistry Research 35, 772–787.

Jazwinski, A.H., 1970. Stochastic Processes and Filtering Theory. Academic Press.

Jiang, Y., Nimmegeers, P., Telen, D., Impe, J.V., Houska, B., 2017. A distributed optimization algorithm for stochastic optimal control. IFAC-PapersOnLine 50, 11263–11268.

Kadam, J.V., Schlegel, M., Srinivasan, B., Bonvin, D., Marquardt, W., 2007. Dynamic optimization in the presence of uncertainty: From off-line nominal solution to measurement-based implementation. Journal of Process Control 17, 389–398.

Li, P., Arellano-Garcia, H., Wozny, G., 2008. Chance constrained programming approach to process optimization under uncertainty. Computers & chemical engineering 32, 25–45.

Li, P., Wendt, M., Arellano-Garcia, H., Wozny, G., 2004. Process optimization and control under chance constraints, in: International Symposium on Advanced Control of Chemical Processes, pp. 962–967.

Lin, Y.D., Stadtherr, M.A., 2007. Deterministic global optimization of nonlinear dynamic systems. AICHE Journal 53, 866–875.

Ma, D.L., Braatz, R.D., 2001. Worst-case analysis of finite-time control policies. IEEE Transactions on Control Systems Technology 9, 766–774.

Ma, D.L., Chung, S.H., Braatz, R.D., 1999. Worst-case performance analysis of optimal batch control trajectories. AIChE journal 45, 1469–1476.

Maußner, J., Freund, H., 2018a. Efficient calculation of constraint back-offs for optimization under uncertainty: A case study on maleic anhydride synthesis. Chemical Engineering Science 192, 306–317.

Maußner, J., Freund, H., 2018b. Optimization under uncertainty in chemical engineering: Comparative evaluation of unscented transformation methods and cubature rules. Chemical Engineering Science 183, 329–345.

Mohideen, M.J., Perkins, J.D., Pistikopoulos, E.N., 1996. Optimal design of dynamic systems under uncertainty. AIChE Journal 42, 2251–2272.

Nagy, Z.K., Braatz, R.D., 2003. Worst-case and distributional robustness analysis of finite-time control trajectories for nonlinear distributed parameter systems. IEEE Transactions on Control Systems Technology 11, 694–704.

Nagy, Z.K., Braatz, R.D., 2004. Open-loop and closed-loop robust optimal control of batch processes using distributional and worst-case analysis. Journal of process control 14, 411–422.

Nagy, Z.K., Braatz, R.D., 2007. Distributional uncertainty analysis using power series and polynomial chaos expansions. Journal of Process Control 17, 229–240.

Nimmegeers, P., Telen, D., Logist, F., Impe, J.V., 2016. Dynamic optimization of biological networks under parametric uncertainty. BMC systems biology 10, 86.

Olofsson, S., Schultz, E.S., Mhamdi, A., Mitsos, A., Deisenroth, M.P., Misener, R., 2020. Design of dynamic experiments for black-box model discrimination. Manuscript submitted for publication.

Pistikopoulos, E.N., 1995. Uncertainty in process design and operations. Computers & Chemical Engineering 19, 553–563.

Puschke, J., Djelassi, H., Kleinekorte, J., Hannemann-Tamás, R., Mitsos, A., 2018. Robust dynamic optimization of batch processes under parametric uncertainty: Utilizing approaches from semi-infinite programs. Computers & Chemical Engineering 116, 253–267.

Puschke, J., Mitsos, A., 2016. Robust dynamic optimization of a semi-batch emulsion polymerization process with parametric uncertainties-a heuristic approach. IFAC-PapersOnLine 49, 907–912.

Puschke, J., Zubov, A., Kosek, J., Mitsos, A., 2017. Multi-model approach based on parametric sensitivities–a heuristic approximation for dynamic optimization of semi-batch processes with parametric uncertainties. Computers & chemical engineering 98, 161–179.

Ricardez-Sandoval, L.A., 2012. Optimal design and control of dynamic systems under uncertainty: A probabilistic approach. Computers & Chemical Engineering 43, 91–107.

Ruppen, D., Benthack, C., Bonvin, D., 1995. Optimization of batch reactor operation under parametric uncertainty—computational aspects. Journal of Process Control 5, 235–240.

Schultz, E.S., Hannemann-Tamás, R., Mitsos, A., 2020a. Guaranteed satisfaction of inequality state constraints in pde-constrained optimization. Automatica 111, 108653.

Schultz, E.S., Hannemann-Tamás, R., Mitsos, A., 2020b. Polynomial approximation of inequality path constraints in dynamic optimization. Computers & Chemical Engineering 135, 106732.

Schwarm, A.T., Nikolaou, M., 1999. Chance-constrained model predictive control. AIChE Journal 45, 1743–1752.

Scott, J.K., Chachuat, B., Barton, P.I., 2013. Nonlinear convex and concave relaxations for the solutions of parametric odes. Optimal Control Applications and Methods 34, 145–163.

Shapiro, A., Dentcheva, D., Ruszczyński, A., 2014. Lectures on stochastic programming: modeling and theory. SIAM.

Shi, J., Biegler, L.T., Hamdan, I., Wassick, J., 2016. Optimization of grade transitions in polyethylene solution polymerization process under uncertainty. Computers & Chemical Engineering 95, 260–279.

Srinivasan, B., Bonvin, D., Visser, E., Palanki, S., 2003. Dynamic optimization of batch processes: Ii. role of measurements in handling uncertainty. Computers & Chemical Engineering 27, 27–44.

Tarantola, A., 2005. Inverse problem theory and methods for model parameter estimation. SIAM.

Telen, D., Vallerio, M., Cabianca, L., Houska, B., Impe, J.V., Logist, F., 2015. Approximate robust optimization of nonlinear systems under parametric uncertainty and process noise. Journal of Process Control 33, 140–154.

Terwiesch, P., Agarwal, M., Rippin, D.W.T., 1994. Batch unit optimization with imperfect modelling: a survey. Journal of Process Control 4, 238–258.

Terwiesch, P., Ravemark, D., Schenker, B., Rippin, D.W.T., 1998. Semi-batch process optimization under uncertainty: Theory and experiments. Computers & Chemical Engineering 22, 201–213.

Visser, E., Srinivasan, B., Palanki, S., Bonvin, D., 2000. A feedback-based implementation scheme for batch process optimization. Journal of Process Control 10, 399–410.

Wendt, M., Li, P., Wozny, G., 2002. Nonlinear chance-constrained process optimization under uncertainty. Industrial & engineering chemistry research 41, 3621–3629.

Zhao, Y., Stadtherr, M.A., 2011. Rigorous global optimization for dynamic systems subject to inequality path constraints. Industrial & Engineering Chemistry Research 50, 12678–12693.

Appendix A. Algorithm 2 workflow

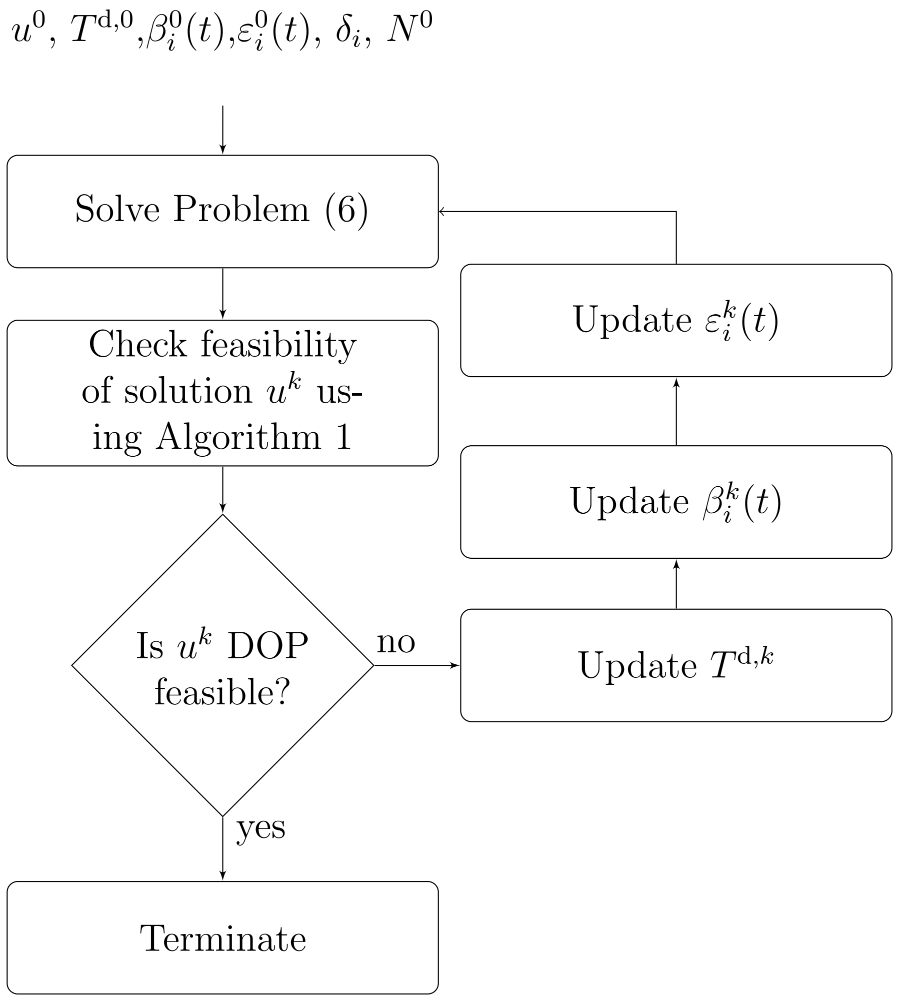
u^0, $T^{\mathrm{d},0}$,$\beta_i^0(t)$,$\varepsilon_i^0(t)$, δ_i, N^0
Solve Problem (6)
Check feasibility of solution u^k using Algorithm 1
Is u^k DOP feasible?
no
yes
Terminate
Update $T^{\mathrm{d},k}$
Update $\beta_i^k(t)$
Update $\varepsilon_i^k(t)$

Figure A.4: Algorithm 2 workflow.