A Design Centering Methodology for Probabilistic Design Space ⋆

Kennedy P. Kusumo * **James Morrissey** * **Hamish Mitchell** *
Nilay Shah * **Benoît Chachuat** *

* *The Sargent Centre for Process Systems Engineering, Department of Chemical Engineering, Imperial College London, London, UK.*

Abstract: The use of mathematical models for design space characterization has become commonplace in pharmaceutical quality-by-design, providing a systematic risk-based approach to assurance of quality. This paper presents a methodology to complement sampling algorithms by computing the largest box inscribed within a given probabilistic design space at a desired reliability level. Such an encoding of the samples yields an operational envelope that can be conveniently communicated to process operators as independent ranges in process parameters. The first step involves training a feed-forward multi-layer perceptron as a surrogate of the sampled probability map. This surrogate is then embedded into a design centering problem, formulated as a semi-infinite program and solved using a cutting-plane algorithm. Effectiveness and computational tractability are demonstrated on the case study of a batch reactor with two critical process parameters.

Keywords: quality-by-design, probabilistic design space, sampling algorithm, design centering, multi-layer perceptron

1. INTRODUCTION

The increasing pressure to continually improve pharmaceutical manufacturing practices has promoted the use of process systems engineering techniques to support pharmaceutical process development. One example is the use of model-based techniques for characterizing a design space in pharmaceutical quality-by-design (QbD) (García-Muñoz et al., 2015). Such mathematical models carry uncertainty, typically in the form of uncertain parameters. Accounting for this uncertainty leads to the concept of probabilistic design spaces (Peterson, 2008), in agreement with the ICH quality guideline Q8 (R2) (Holm et al., 2017).

As a consequence of these developments, interest in numerical tools capable of characterising a probabilistic design space has grown significantly. Notice that the concept of a probabilistic design space is akin to stochastic flexibility analysis (Straub and Grossmann, 1993; Pulsipher and Zavala, 2019) in the process systems engineering literature. Existing algorithms either approximate the design space with a set of samples (Bano et al., 2018; Kusumo et al., 2020) or try to inscribe a given shape inside the design space of interest (Laky et al., 2019). This work concentrates on bridging the two.

An important aspect of sampling-based techniques is the ability to encode the resulting samples into an operational envelope (Samsatli et al., 1999) that can be conveniently communicated to process operators and easily exploited. Our approach entails the construction of a surrogate of the probabilistic design space using machine learning techniques, which provides an inexpensive way of predicting the feasibility probability of any process operating points. In a second step, a design centering problem that embeds this surrogate is solved to determine the largest box inscribed within the probabilistic design space at a desired reliability level. The main advantage of communicating a box as the operational envelope is that each process parameter can be varied independently inside their feasible ranges.

Design centering problems can be formulated as semi-infinite programs (SIPs) (Hettich and Kortanek, 1993). Effective solution techniques for SIPs include cutting plane algorithms (Blankenship and Falk, 1976), which iteratively solve a discrete relaxation of the SIP and a feasibility subproblem for checking where the semi-infinite constraints are violated. This idea was recently extended by Mitsos (2011), where a restriction of the SIP relaxation is solved alongside the SIP relaxation and the feasibility problem to ensure finite termination of the algorithm.

The rest of the paper is organized as follows: Sec. 2 reviews the mathematical formulation of probabilistic design spaces; the design centering methodology is detailed in Sec. 3; computational results are showcased in Sec. 4 for a batch reactor with two critical process parameters; and final conclusions are drawn in Sec. 5.

2. BACKGROUND

Consider a mathematical model of a pharmaceutical manufacturing process, built to predict critical quality attributes (CQAs) of the product, $\boldsymbol{s} \in \mathbb{R}^{n_s}$ for a given set of critical process parameters (CPPs), $\boldsymbol{d} \in \mathcal{K}$ within the knowledge space, $\mathcal{K} \subset \mathbb{R}^{n_d}$:

$$\boldsymbol{s} = \boldsymbol{f}(\boldsymbol{d}, \boldsymbol{\theta}) \tag{1}$$

⋆ Corresponding author: b.chachuat@imperial.ac.uk

The model $\boldsymbol{f} : \mathbb{R}^{n_d} \times \mathbb{R}^{n_\theta} \to \mathbb{R}^{n_s}$ depends on uncertain parameters $\boldsymbol{\theta} \in \mathbb{R}^{n_\theta}$, which could represent model parameter estimates, such as physical constants, regression coefficients; or process uncertainties, such as disturbances that affects the CQAs. We do not assume that $\boldsymbol{f}$ is given in closed form, that is, it could be implicitly defined via differential-algebraic equations or any black-box functions. The process is subject to process constraints and CQA limits, represented as a set of inequalities $\boldsymbol{g}(\boldsymbol{s}, \boldsymbol{d}, \boldsymbol{\theta}) \leq \boldsymbol{0}$. Without loss of generality, $\boldsymbol{s}$ can be expressed in terms of $\boldsymbol{d}$ and $\boldsymbol{\theta}$ by inverting the model (Eqn. 1), thus leading to the *reduced* inequalities:

$$\boldsymbol{G}(\boldsymbol{d}, \boldsymbol{\theta}) \leq \boldsymbol{0}$$

A *nominal* design space (of the process) is defined as

$$\mathcal{D}_{\text{nom}} = \{\boldsymbol{d} \in \mathcal{K} : \boldsymbol{G}(\boldsymbol{d}, \boldsymbol{\theta}_{\text{nom}}) \leq \boldsymbol{0}\}$$

for some nominal value of the uncertain parameters $\boldsymbol{\theta}_{\text{nom}}$. Such a design space ignores the uncertain nature of $\boldsymbol{\theta}$ by assuming a deterministic value $\boldsymbol{\theta}_{\text{nom}}$. Characterisation of a nominal design space is likely to cause false negatives or positives about the feasibility of a process, making it unsuitable for regulated processes.

The focus herein is on *probabilistic* (or Bayesian) design spaces. This framework considers $\boldsymbol{\theta}$ as a random variable, described by a probability density function $p(\boldsymbol{\theta})$, so the *feasibility probability* of a CPP $\boldsymbol{d}$ is given by:

$$\mathbb{P}\left[\boldsymbol{G}(\boldsymbol{d}, \cdot) \leq \boldsymbol{0} \mid p(\boldsymbol{\theta})\right] = \int_{\{\boldsymbol{\theta}: \boldsymbol{G}(\boldsymbol{d}, \boldsymbol{\theta}) \leq \boldsymbol{0}\}} p(\boldsymbol{\theta}) d\boldsymbol{\theta}$$

In turn, the Bayesian design space at a desired reliability level $\alpha \in (0, 1]$ is defined as:

$$\mathcal{D}_\alpha := \{\boldsymbol{d} \in \mathcal{K} : \mathbb{P}\left[\boldsymbol{G}(\boldsymbol{d}, \cdot) \leq \boldsymbol{0} \mid p(\boldsymbol{\theta})\right] \geq \alpha\}$$

Observe that $\mathcal{D}_\alpha \supseteq \mathcal{D}_{\alpha'}$ whenever $\alpha < \alpha'$. Consequently, a higher reliability value α corresponds to higher conservatism and thus a smaller design space. The trade-off is that a higher α implies lower chance of false positives, but higher chance of false negatives. A suitable choice of α is largely process-specific in practice, but since false positives pose a larger threat to quality assurance, most practitioners tend to prefer higher α.

Sampling algorithms such as nested sampling (Kusumo et al., 2020) approximate a Bayesian design space with a set of N_d CPP samples, where each sample $\boldsymbol{d}_s$ is associated with an (estimated) feasibility probability $p_s \approx \mathbb{P}\left[\boldsymbol{G}(\boldsymbol{d}_i, \cdot) \leq \boldsymbol{0} \mid p(\boldsymbol{\theta})\right]$. These samples and their corresponding feasibility probabilities are denoted collectively by $\mathbf{D} := [\boldsymbol{d}_1 \cdots \boldsymbol{d}_{N_d}] \in \mathbb{R}^{n_d \times N_d}$ and $\mathbf{P} := [p_1 \cdots p_{N_d}] \in \mathbb{R}^{N_d}$ subsequently. A Bayesian design space $\mathcal{D}_\alpha$ at a given reliability level α comprises those CPP samples $\boldsymbol{d}_s$ such that $p_s \geq \alpha$.

3. METHODOLOGY

In principle, the methodology described below may be used in combination with any sampling algorithm. Our focus throughout this paper is on an adaptation of the nested sampling algorithm for design space characterization (NS-DS) that was recently proposed by Kusumo et al. (2020).

NS-DS starts with a pre-specified number of *live points* N_L, uniformly sampled within $\mathcal{K}$. The feasibility probability of each live point is estimated via Monte Carlo simulation, by drawing a sample $\mathcal{S}_\theta$ from the probability distribution $p(\boldsymbol{\theta})$:

$$p_s = \sum_{(\boldsymbol{\theta}, \omega) \in \mathcal{S}_\theta} \mathbb{I}\left[\boldsymbol{G}(\boldsymbol{d}_s, \boldsymbol{\theta})\right] \omega \tag{2}$$

where the indicator function $\mathbb{I}[\cdot]$ evaluates to 1 if $\boldsymbol{G} \leq \boldsymbol{0}$ and 0 otherwise; and ω denotes the weight associated to each model parameter scenario $\boldsymbol{\theta}$ in $\mathcal{S}_\theta$.

At each iteration, NS-DS then generates N_P *replacement proposals*. For instance, the proposals may be sampled from an enlarged ellipsoid enclosing the current live points (Mukherjee et al., 2006). Every time a proposal point $\boldsymbol{d}_s$ is generated, its feasibility probability p_s is estimated using Eqn. (2). If p_s is greater than the lowest feasibility probability among all live points, $p_{\min}$, the corresponding live point $\boldsymbol{d}_{\min}$ is replaced with $\boldsymbol{d}_s$. The replaced point becomes a *dead point* and is recorded with its estimated feasibility probability in separate set, then $\boldsymbol{d}_{\min}$ and $p_{\min}$ are updated accordingly. A termination criterion is checked after each iteration, for instance if $p_{\min}$ is above the target reliability α. Upon termination, NS-DS returns all current live points and all recorded dead points as $\mathbf{D}$, and their estimated feasibility probabilities as $\mathbf{P}$.

3.1 Surrogate Regression of Probability Map

The first step of our methodology entails the construction and training of a feed-forward multi-layer perceptron (MLP) as a surrogate of the probability map $\varrho : \mathcal{K} \to [0, 1]$.

The MLP is trained using the CPP samples in $\mathbf{D}$ as inputs, and their estimated feasibility probabilities in $\mathbf{P}$ as outputs. Therefore, the input layer comprises n_d perceptrons, while the output layer contains a unique perceptron whose activation state represents the predicted feasibility probability. The MLP furthermore comprises one hidden layer with n_h perceptrons. All input and output perceptrons have linear activation functions, and all hidden perceptrons have sigmoidal activation functions.

An important pre-processing step before training is feature scaling, whereby the range of each CPP is normalized. This eliminates the MLP's bias towards inputs with larger magnitude, which could adversely affect the accuracy of the MLP's predictions. This unwanted bias may arise from different units of measurement used for the CPPs.

In summary, the MLP surrogate presents the following structure:

$$\begin{aligned}
\varrho(\boldsymbol{d}) &= a^{(3)}(\boldsymbol{d}, \boldsymbol{w}^{(1)}, \boldsymbol{w}^{(2)}, \boldsymbol{b}^{(2)}, b^{(3)}) \\
a^{(3)} &= \sum_{j=1}^{n_h} \left(a_j^{(2)} w_j^{(2)}\right) + b^{(3)} \\
a_j^{(2)} &= \sigma\left(\sum_{i=1}^{n_d} \left(a_i^{(1)} w_{i,j}^{(1)}\right) + b_j^{(2)}\right), \quad j = 1 \ldots n_h \\
a_i^{(1)} &= \frac{d_i - \bar{d}_i}{\sigma_i}, \quad i = 1 \ldots n_d
\end{aligned}$$

where $a^{(3)}$ is the activation state of the output perceptron, $a_j^{(2)}$ the activation state of the jth hidden perceptron, $a_i^{(1)}$ the activation state of the ith input perceptron, $w_{i,j}^{(1)}$ the weight between the ith input perceptron and jth

hidden perceptron, $w_j^{(2)}$ the weight between the jth hidden perceptron and the output perceptron, $b_j^{(2)}$ the bias of the jth hidden perceptron, $b^{(3)}$ the bias of the output perceptron, $\bar{d}_i$ the mean over all samples in $\mathbf{D}$ of the ith CPP, and similarly σ_i the standard deviation of the ith CPP. Lastly, the activation function $\sigma : \mathbb{R} \to [0,1]$ is chosen as the following sigmoid function:

$$\sigma(x) = \frac{1}{1+\exp(-x)}$$

Feed-forward MLPs were chosen over other regression techniques due to their ability to describe highly nonlinear relationships, making them a suitable choice for a wide range of design space problems. In particular, feed-forward MLPs with at least one hidden layers and sigmoid activation functions are known to be universal approximators of continuous functions (Cybenko, 1989). A caveat is that it may require a large number of perceptrons in that hidden layer to meet a desired accuracy. We use $n_h = 64$ perceptrons in the case study of Sec. 4, but this could be increased as necessary or multi-layer MLPs could be considered instead. It is also worth mentioning that MLPs typically require large training data sets in order to be effective. However, this is not a limitation here since the training data is provided by the probabilistic design space characterisation activity without further costs—with NS-DS, this is easily controlled by increasing the number of live points n_L.

3.2 Design Centering Problem

The second step of our methodology entails the computation of the largest box inscribed within the desired probabilistic design space $\mathcal{D}_\alpha$.

The n_d-dimensional box is parameterized by $2n_d$ shape parameters, e.g. grouped as lower range values $\boldsymbol{d}^L \in \mathbb{R}^{n_d}$ and upper range values $\boldsymbol{d}^U \in \mathbb{R}^{n_d}$ on the CPPs. The design centering problem problem can be stated as:

$$\max_{\boldsymbol{d}^L, \boldsymbol{d}^U} \quad \prod_{i=1}^{n_d} \left(d_i^U - d_i^L\right) \tag{3}$$

$$\text{s.t.} \quad \varrho(\boldsymbol{d}) \geq \alpha, \ \forall \boldsymbol{d} \in [\boldsymbol{d}^L, \boldsymbol{d}^U] \tag{4}$$

We consider the box volume as the cost function (3), but other measures such as the perimeter could be readily used instead. Notice also that this problem falls into the class of generalized SIP (GSIP) since the uncertainty set in the semi-infinite constraint (4) is dependent on the decision variables $\boldsymbol{d}^L$, $\boldsymbol{d}^U$. However, it is readily reformulated as an SIP by normalizing each CPP as $d_i(x_i) := d_i^L + x_i(d_i^U - d_i^L)$ with $x_i \in [0,1]$:

$$\max_{\boldsymbol{d}^L, \boldsymbol{d}^U} \quad \prod_{i=1}^{n_d} \left(d_i^U - d_i^L\right) \tag{5}$$

$$\text{s.t.} \quad \varrho(\boldsymbol{d}(\boldsymbol{x})) \geq \alpha, \ \forall \boldsymbol{x} \in [0,1]^{n_d} \tag{6}$$

We consider the cutting-plane (CP) algorithm by Blankenship and Falk (1976) to solve the SIP (5,6). A pseudo-code of this algorithm tailored to design centering in a design space is presented below.

(1) Initialize $\mathcal{S}_x$ with the centroid $x_i = 0.5, i = 1 \dots n_d$.

(2) Solve the discretised SIP to global optimality:

$$\max_{\boldsymbol{d}^L, \boldsymbol{d}^U} \quad \prod_{i=1}^{n_d} \left(d_i^U - d_i^L\right)$$
$$\text{s.t.} \quad \varrho\left(\boldsymbol{d}\left(\boldsymbol{x}\right)\right) \geq \alpha, \ \forall \boldsymbol{x} \in \mathcal{S}_x$$

If the problem is infeasible, terminate; else assign the optimal point to $\boldsymbol{d}_{\text{opt}}^L, \boldsymbol{d}_{\text{opt}}^U$.

(3) Solve a feasibility problem for the semi-infinite constraint to global optimality:

$$\min_{\boldsymbol{x}, \boldsymbol{\delta}} \ \varrho\left(\boldsymbol{\delta}\right)$$
$$\text{s.t. } \delta_i = d_{\text{opt},i}^L + x_i\left(d_{\text{opt},i}^U - d_{\text{opt},i}^L\right), \ \forall i$$
$$\boldsymbol{x} \in [0,1]^{n_d}$$

Append the optimal point to $\mathcal{S}_x$, and assign the optimal solution value to p_{opt}.

(4) If $p_{\text{opt}} \geq \alpha$, terminate with $\boldsymbol{d}_{\text{opt}}^L$, $\boldsymbol{d}_{\text{opt}}^U$ as the final solution; else return to Step 2.

In practice, it is of paramount importance to solve the feasibility subproblems to guaranteed global optimality, otherwise the cutting-plane algorithm may violate the semi-infinite constraint (6) upon termination. Solving the discretized SIP subproblems to global optimality is also desirable in order to ensure that $\boldsymbol{d}_{\text{opt}}^L, \boldsymbol{d}_{\text{opt}}^U$ is a global optimum of the design centering problem. A caveat with the CP algorithm is it may not terminate finitely. This could be circumvented by applying the improved SIP algorithm by Mitsos (2011), where a restriction of the SIP relaxation is solved alongside the SIP relaxation and the feasibility problem to ensure finite termination. Another approach, which is used herein, consists in adjusting the termination condition in Step 4 to $p_{\text{opt}} \geq \alpha - \epsilon^\circ$, where ϵ° can be the same as the convergence tolerance used in the global solver for instance.

4. CASE STUDY: SEQUENTIAL REACTION

This case study investigates an isothermal batch reactor, hosting two reactions in series:

$$2\text{A} \longrightarrow \text{B} \longrightarrow \text{C}$$

The two CPPs are the reactor temperature, T and the batch time, τ. The differential equations (ODEs) governing the concentrations c_A, c_B and c_C during the batch are given by:

$$\dot{c}_\text{A}(t) = -2k_1 \exp\left(-\frac{E_1}{RT}\right) c_\text{A}(t)^2$$
$$\dot{c}_\text{B}(t) = k_1 \exp\left(-\frac{E_1}{RT}\right) c_\text{A}(t)^2 - k_2 \exp\left(-\frac{E_2}{RT}\right) c_\text{B}(t)$$
$$\dot{c}_\text{C}(t) = k_2 \exp\left(-\frac{E_2}{RT}\right) c_\text{B}(t)$$

where the kinetic parameters (E_1, E_2, k_1, k_2) are normally distributed with mean values $(2.5 \times 10^3, 5 \times 10^3, 6.409 \times 10^{-2}, 9.938 \times 10^3)$ and 1% standard deviations; and the batch is initial loaded with A at a concentration of $C_\text{A}(0) = 2$ (mol L^{-1}). Two design space constraints are imposed on the purity (mole fraction) of the desired product B at the end of the batch and on the profit per unit time:

$$\frac{c_\text{B}(\tau)}{c_\text{A}(\tau) + c_\text{B}(\tau) + c_\text{C}(\tau)} \geq 0.8, \quad \frac{100c_\text{B}(\tau) - 20c_\text{A}(0)}{\tau + 30} \geq 128$$

The problem is to compute the largest box inscribed within $\mathcal{D}_{0.55}$, the probabilistic design space at reliability level $\alpha = 0.55$. This reliability level was chosen because the maximum predicted feasibility probability in this design space is only 0.75. In many practical pharmaceutical processes, the target reliability level will be higher. This will mean that the practitioner needs to reduce the uncertainty in the model parameters before being able to characterise a design space suitable for a validated process, typically via gathering more (informative) experimental data.

As a starting point, we assume that the practitioner has characterised a probabilistic design space at reliability levels up to $\alpha = 0.65$, using NS-DS through the open-source Python software DEUS (Kusumo et al., 2020). The ODEs in the batch process model are integrated using the open-source optimisation modelling language Pyomo (Hart et al., 2011, 2017) and Pyomo.DAE (Nicholson et al., 2018). The obtained point samples $\mathbf{D}$ and their estimated feasibility probabilities $\mathbf{P}$ are illustrated in Fig. 1.

In the first step of the methodology, the MLP (1 hidden layer with 64 perceptrons) is constructed using the open-source Python library Keras (Chollet et al., 2015), the high-level API to TensorFlow (Abadi et al., 2015). MLP training is set to a maximum of 4000 epochs, with an early termination condition of non-improving fit over the last 500 epochs. In the second step, the design centering problem is solved numerically using the global solver BARON (Tawarmalani and Sahinidis, 2005) in GAMS, using default parameters and subsolvers apart from setting the relative convergence tolerance to 10^{-3}.

Table 1. Computational performance metrics

Metric	Value
MLP Training	
Number of Epochs	4000
Training Time (min) [1]	5.8
Design Centering	
Number of Iterations	3
Solution Time (min) [2]	5.5

[1] on a single core of AMD Ryzen 5 2600x.
[2] on a single core of AMD Opteron 6164.

4.1 Multilayer Perceptron Training

Fig. 2 presents the parity plot of the trained MLP for the point samples shown in Fig. 1, where the scattered points are translucent to portray sample density. It can be seen that the sample density is higher towards the target $\alpha = 0.65$. This is a feature of NS-DS which is designed to draw denser samples from reliability levels close to the target α. Although there are a few point samples relatively far from the parity line $y = x$ in regions of low probability feasibility, the simple MLP performs well on average, as indicated by a mean absolute error (MAE) of 0.01. For this simple case study, the MLP training used 4,000 epochs and took just a few minutes to complete (Table 1).

An important consideration in choosing the MLP structure is the interplay between the MLP accuracy, and the approximation error in estimating the feasibility probability of each sample. The latter arises because the feasibility probability is estimated using Monte Carlo sampling,

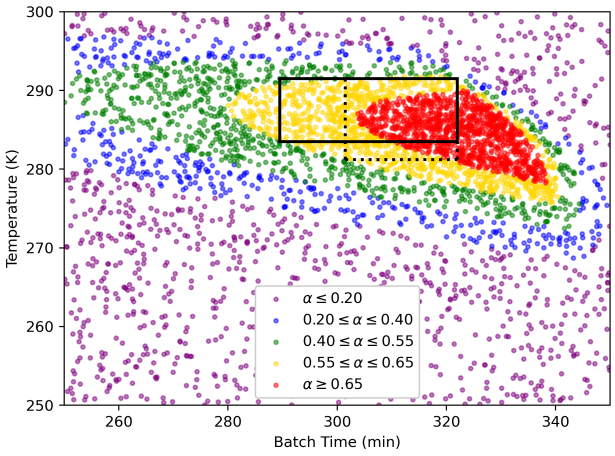

Fig. 1. Scattered point samples $\mathbf{D}$ generated by NS-DS with $N_L = 1000$ live points and $N_P = 500$ replacement proposals. Colours denote the corresponding feasibility probabilities $\mathbf{P}$, estimated using $N_\theta = 1000$ random samples. The computed largest hyperrectangle inscribed within $\mathcal{D}_{0.55}$ is shown as solid black lines. Dotted lines show the modified solution that ensures a minimum temperature window of 10 K.

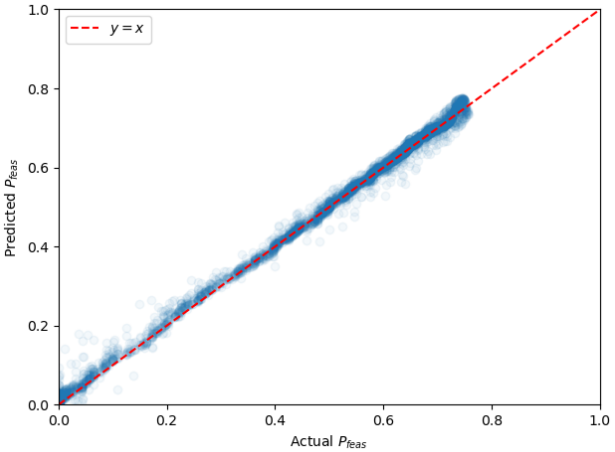

Fig. 2. Parity plot of the trained MLP. Red dotted line is the parity line $y = x$, and scattered translucent blue circles represent individual training samples.

making it highly sensitive to the number of uncertainty scenarios N_θ. For comparison, Fig. 3 shows the same design space computed using $N_\theta = 100$ uncertainty scenarios only and fewer live points and replacement proposals. It appears that both probabilistic design spaces are comparable at lower reliability levels, yet become significantly different at higher reliability levels. The predicted $\mathcal{D}_{0.65}$ is significantly smaller with $N_\theta = 100$ than with $N_\theta = 1000$. Even for the same number of scenarios N_θ, there could be significant differences between two random samples. In general, the approximation error will decrease and the consistency between different random samples will improve as N_θ increases. The MLP may also be useful in filtering out some of this error, which highlights the importance of limiting the complexity of the MLP to avoid overfitting. Without careful consideration of the chosen uncertainty

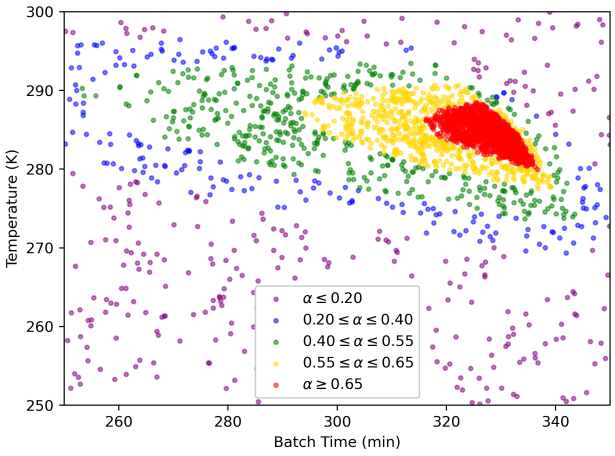

Fig. 3. Scattered point samples $\mathbf{D}$ generated by NS-DS with $N_L = 300$ live points and $N_P = 200$ replacement proposals. Colors denote the corresponding feasibility probabilities $\mathbf{P}$, estimated using $N_\theta = 100$ random samples.

scenarios, arbitrarily increasing the number of perceptrons or hidden layers in the MLP may backfire as the MLP will start fitting the noise of the probability feasibility estimates.

Another important aspect of MLP training is the density and uniformity of samples in the CPP space. All sampling algorithms produce an inner-approximation of the design space, so there is always a chance that part of the feasible region may be missed. As an example, Fig. 3 shows a relatively sparse region in the middle. Here, the MLP plays a role as an interpolant and makes predictions to "fill in the blanks". For a given sample density, uniformity of the samples minimizes the likelihood for such regions to exist. The NS-DS algorithm has a feature whereby sample density increases towards the given target reliability level whilst ensuring that the samples are also uniformly spread, as illustrated in Figs. 1 & 3. Such a feature gives the MLP a larger number of samples from a given target reliability level, thus minimising the MLP error around it (for a given computational budget).

4.2 Design Centering

Table 1 shows the overall time and number of iterations taken to solve the design centering problem, for the MLP trained on the point samples in Fig. 1. The feasibility problems could be solved to global optimality within 10 seconds in each iteration of the cutting-plane algorithm. However, a test global run of the relaxed optimisation problem in the first iteration with BARON on default parameters took over one day to complete. Upon inspection of the optimizer's log files, we found that BARON could identify a global optimum early on (within 20 seconds), then most of the solution time was spent on branch and bound to confirm the global optimality of the solution. As a remedy, we implemented a time limit of 100 seconds and took the best solution at termination. A consequence of this restriction is that the computed box could be suboptimal, that is, smaller. The alternative is to use a local NLP solver, implying a much faster solution time.

Table 2. Computed box for the sequential reaction case study

Critical Process Parameter	d_i^L	d_i^U
No Additional Constraints		
Batch Time (min)	289.5	322.0
Temperature (K)	283.5	291.2
Minimum 10 (K) Window		
Batch Time (min)	301.5	322.0
Temperature (K)	281.2	291.2

We found that IPOPT takes a few seconds to solve the relaxed problems in all iterations, but it did not lead to the same solution in this case. Because of this, the time limit alternative was our preferred remedy as it reduces the likelihood of erroneous results, essentially using BARON as a multistart heuristic within the time limit.

Overlaid with the point samples $\mathbf{D}$ generated with NS-DS and their feasibility probabilities $\mathbf{P}$, Fig. 1 shows the largest box inscribed within $\mathcal{D}_{0.55}$ as the region enclosed within the solid lines. Notice that there are infeasible points in the top left corner, i.e. sample points with a feasibility probability below the target α. A likely cause for this is the prediction accuracy of the MLP, which could be precented by refining the MLP. Except for this discrepancy, Fig. 1 shows that the methodology was successful in computing the largest box inscribed within the a probabilistic design space at a given reliability level. The computed coordinates of this box are reported in Table 2. The overlay in Fig. 1 also confirms that the locally optimal solutions of the relaxed problems did not lead to an overly conservative solution.

A practical concern that may come up is that three of the four corners of the inscribed box lie on the boundary of the chosen probabilistic design space $\mathcal{D}_{0.55}$. Some may argue that considering these process parameters as safe may be dangerous because there are errors introduced in the various steps of the methodology. One example is the MLP surrogate error being an imperfect encoding of the real model. There is also uncertainty in the estimated feasibility probability arising from the nature of Monte Carlo simulations, as well as unavoidable modelling uncertainties present in any model-based techniques. The practical remedy is to be more conservative, i.e. to solve the design centering problem for a probabilistic design space with a reliability level α slightly higher than the desired target.

Another important aspect is how sensitive the CPPs are to disturbances. This issue is related to the formulation of the most appropriate design centering problem. For instance, assume that the CPPs T and τ are subject to random variations equal to $\Delta T = \pm 5$ K and $\Delta\tau = \pm 5$ min. The optimal box on Fig. 1 may be deemed unsuitable since it provides a 32.5 min window for τ, but only a 7.7 K window for T. In such cases, it is possible to impose additional constraints to achieve a more practical operational box. For instance, the dotted lines in Fig. 1 show a box computed with an additional constraint to ensure a minimum temperature window of 10 K. Notice how this extra constraint alters both the volume and position of the operating region.

5. CONCLUSIONS

Sampling algorithms constitute an important toolset to support the model-based characterisation of a design space in pharmaceutical quality-by-design. This paper has introduced a two-step methodology to encode and communicate the resulting sampling points into a set of independent CPP ranges. The first step entails the construction of an MLP as a surrogate of the probability map. This surrogate is then embedded into a design centering problem that seeks to inscribe the largest box within a design space of a given reliability level, via the numerical solution of a semi-infinite program using a cutting-plane algorithm.

The effectiveness and tractability of the methodology has been demonstrated on a case study with two CPPs. The MLP was found to play a central role in filtering the approximation error inherent to sampling-based techniques, and care should be taken to avoid overfitting when constructing and training the MLP. Although the cutting-plane algorithm proved effective in this low-dimensional case study, there is no guarantee that it will terminate finitely in general. For higher-dimensional design problems or more complex design space geometries, it might become advantageous to implement the cutting-plane algorithm by Mitsos (2011), whereby restricted right-hand side subproblems are solved to enforce finite termination. Another key challenge for the methodology is the difficulty in performing global optimization on the trained MLP using complete search methods; see, e.g., Schweidtmann and Mitsos (2019) for a recent contribution in this area. A possible alternative would be to explore the suitability of other surrogate model structures, such as poynomial functions and radial basis functions.

There may be advantages in inscribing more versatile shapes into a design space, such as ellipsoids or polytopes which have the potential of being a lot less conservative than boxes. Conducting an in-depth analysis on various measures of the inscribed shapes and possible scaling strategies constitutes another interesting avenue for future research. Finally, it would be interesting to investigate design centering methodologies for problems presenting non-connected design spaces.

ACKNOWLEDGEMENTS

This work is supported by Eli Lilly & Company through the Pharmaceutical Systems Engineering Lab (PharmaSEL) program and by an EPSRC Prosperity Partnership under grant EP/T005556/1.

REFERENCES

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G.S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X. (2015). TensorFlow: Large-scale machine learning on heterogeneous systems. URL `https://www.tensorflow.org/`.

Bano, G., Facco, P., Bezzo, F., and Barolo, M. (2018). Probabilistic design space determination in pharmaceutical product development: A Bayesian/latent variable approach. *AIChE Journal*, 64, 2438–2449. doi:10.1002/aic.16133.

Blankenship, J.W. and Falk, J.E. (1976). Infinitely constrained optimization problems. *Journal of Optimization Theory & Applications*, 19(2), 261–281. doi:10.1007/BF00934096.

Chollet, F. et al. (2015). Keras. `https://keras.io`.

Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals & Systems*, 2(4), 303–314. doi:10.1007/BF02551274.

García-Muñoz, S., Luciani, C.V., Vaidyaraman, S., and Seibert, K.D. (2015). Definition of design spaces using mechanistic models and geometric projections of probability maps. *Organic Process Research & Development*, 19(8), 1012–1023. doi:10.1021/acs.oprd.5b00158.

Hart, W.E., Laird, C.D., Watson, J.P., Woodruff, D.L., Hackebeil, G.A., Nicholson, B.L., and Siirola, J.D. (2017). *Pyomo–optimization modeling in python*, volume 67. Springer Science & Business Media, second edition.

Hart, W.E., Watson, J.P., and Woodruff, D.L. (2011). Pyomo: modeling and solving mathematical programs in python. *Mathematical Programming Computation*, 3(3), 219–260.

Hettich, R. and Kortanek, K.O. (1993). Semi-infinite programming: Theory, methods, and applications. *SIAM Review*, 35(3), 380–429. doi:10.1137/1035089.

Holm, P., Allesø, M., Bryder, M.C., and Holm, R. (2017). Q8(R2). In *ICH Quality Guidelines*, 535–577. John Wiley & Sons, Inc., Hoboken, NJ, USA. doi:10.1002/9781118971147.ch20.

Kusumo, K.P., Gomoescu, L., Paulen, R., Garciá Munõz, S., Pantelides, C.C., Shah, N., and Chachuat, B. (2020). Bayesian approach to probabilistic design space characterization: A nested sampling strategy. *Industrial & Engineering Chemistry Research*, 59(6), 2396–2408. doi:10.1021/acs.iecr.9b05006.

Laky, D., Xu, S., Rodriguez, J.S., Vaidyaraman, S., García Muñoz, S., and Laird, C. (2019). An optimization-based framework to define the probabilistic design space of pharmaceutical processes with model uncertainty. *Processes*, 7(2), 96. doi:10.3390/pr7020096.

Mitsos, A. (2011). Global optimization of semi-infinite programs via restriction of the right-hand side. *Optimization*, 60(10-11), 1291–1308. doi:10.1080/02331934.2010.527970.

Mukherjee, P., Parkinson, D., and Liddle, A.R. (2006). A nested sampling algorithm for cosmological model selection. *The Astrophysical Journal*, 638(2), L51–L54. doi:10.1086/501068.

Nicholson, B., Siirola, J.D., Watson, J.P., Zavala, V.M., and Biegler, L.T. (2018). Pyomo.Dae: a modeling and automatic discretization framework for optimization with differential and algebraic equations. *Mathematical Programming Computation*, 10(2), 187–223. doi:10.1007/s12532-017-0127-0.

Peterson, J.J. (2008). A Bayesian approach to the ICH Q8 definition of design space. *Journal of Biopharmaceutical Statistics*, 18(5), 959–975. doi:10.1080/10543400802278197.

Pulsipher, J.L. and Zavala, V.M. (2019). A scalable stochastic programming approach for the design of flexible systems. *Computers & Chemical Engineering*, 128, 69–76. doi:10.1016/j.compchemeng.2019.05.033.

Samsatli, N., Papageorgiou, L., and Shah, N. (1999). Batch process design and operation using operational envelopes. *Computers & Chemical Engineering*, 23, S887–S890. doi:10.1016/S0098-1354(99)80218-X.

Schweidtmann, A.M. and Mitsos, A. (2019). Deterministic global optimization with artificial neural networks embedded. *Journal of Optimization Theory & Applications*. doi:10.1007/s10957-018-1396-0.

Straub, D.A. and Grossmann, I.E. (1993). Design optimization of stochastic flexibility. *Computers & Chemical Engineering*, 17(4), 339–354. doi:10.1016/0098-1354(93)80025-I.

Tawarmalani, M. and Sahinidis, N.V. (2005). A polyhedral branch-and-cut approach to global optimization. *Mathematical Programming*, 103, 225–249. doi:10.1007/s10107-005-0581-8.