Noname manuscript No.
(will be inserted by the editor)

Two-objective robust optimisation with quantification and propagation of uncertainties from surrogate model and manufacturing process

Yongxing Wang · Hazim Hamad · Jochen Voss · Harvey M. Thompson

Received: date / Accepted: date

Abstract A robust optimisation formulates the uncertainties of a physical system into the objective functions and/or its constraints, and uncertainties are consequently quantified at the same time as minimising the objectives, which therefore provides a robust solution to the corresponding physical system. In this paper, we present a general mathematical framework for data-driven robust optimisation problems: first, the Gaussian Process Regression (GPR) method is used to create a surrogate model, which allows us to analyse and quantify uncontrollable uncertainties such as noise from the training dataset or finite/limited discrete data points; then, the Polynomial Chaos Expansion (PCE) method is used to propagate controllable uncertainties, such as a manufacturing tolerance, from the input to the output and create a probabilistic model; finally, three robust optimisation problems are formulated and solved with a detailed analysis of the results. The proposed optimisation framework is implemented in open-source Python libraries and assessed by validated dataset from a Polymerase Chain Reaction (PCR) thermal flow system with consideration of uncertainties from manufacturing process.

Keywords Robust optimisation · uncertainty quantification and propagation · Gaussian process regression · Polynomial chaos expansion · Polymerase chain reaction

Yongxing Wang
School of Computing, University of Leeds, Leeds, UK
E-mail: scsywan@leeds.ac.uk

Hazim Hamad
School of Mechanical Engineering, University of Leeds, Leeds, UK
E-mail: mnhsh@leeds.ac.uk

Jochen Voss
School of Mathematics, University of Leeds, Leeds, UK
E-mail: J.Voss@leeds.ac.uk

Harvey M. Thompson
School of Mechanical Engineering, University of Leeds, Leeds, UK
E-mail: H.M.Thompson@leeds.ac.uk

1 Introduction

Robust design approaches have been developed to make the product and manufacturing process robust and insensitive to external noise or tolerance [42,40,24,63]. Since the pioneering work by [53], robust design approaches have been developed to include different statistical methods [55,60] and robust optimisations [3,7,6], and applied to solve a variety of engineering problems [9,49,57]. In this article, we investigate a two-objective robust optimisation approach, through Gaussian Process Regression (GPR) as a surrogate model and Polynomial Chaos Expansion (PCE) as a tool of uncertainty propagation, which can be directly extended to multi-objective cases as well.

Robust optimisation is based on well-developed mathematical optimisation techniques. We consider a robust optimisation based upon the information of mean and standard deviation of data-driven surrogate models $f_i\mathbf{x})$ $(i = 1, 2, \ldots, m)$, which can be formulated, with consideration of general inequality constraints, as follows:

$$\begin{aligned} \underset{\mathbf{x}\in\mathbb{R}^d}{\text{minimize}} \quad & \sum_{i=1}^{m} \left[a_i \mu_{f_i(\mathbf{x})} + b_i \sigma_{f_i(\mathbf{x})} \right], \\ \text{subject to} \quad & g_j\left(\mathbf{x}, \sigma_{f_i(\mathbf{x})}\right) n \leq 0, \\ & i = 1, \ldots, m,\ j = 1, \ldots, n, \end{aligned} \tag{1}$$

where $\mathbf{x}$ is a d-dimensional design variable and $f_i(\mathbf{x})$ are the objectives with $a_i, b_i \geq 0$ being the weights. In the above formulation (1), robustness of the objective is considered as a weighting combination of μ_{f_i} and σ_{f_i}, and robustness of the constraints is considered in function g_j. Objective $f_i(\mathbf{x})$ usually represents a physical system, which will be approximated by a data-driven surrogate model in this article, through the use of a combination of the GPR and PCE methods.

Gaussian process regression is a principled, practical and probabilistic machine learning approach [59], which can learn the mean and standard deviation from data at the same time, and thus convenient to be applied to robust optimisations [27, 33]. Compared with other machine learning methods [38], GPR can estimate all the hyperparameters by maximising the so-called marginal likelihood using the training dataset [59], and consequently provides the user a non-parametric model. The mean of the GPR may be directly used for prediction if the user has a noise-free training data. In this paper, we use the standard deviation predicted by the GPR method to analyse the noise from our training dataset and compare it with an input manufacturing error. The error from surrogate model is measured at the output of the GPR model, while the error from manufacturing process is measured at the input. Therefore, we propagate the manufacturing error to the output using the PCE method, quantify these two types of errors in the same physical domain, and then formulate a new surrogate model with consideration of these two types of uncertainties.

Polynomial chaos expansion is a rigorous approach for representing a random variable (such as an output variable) in terms of a polynomial function of other random variables (such as input variables), which was first introduced by [58] and widely applied in engineering community [19,20,12,54]. Its generalization to different polynomial families and proof of existence and convergence was introduced in [61,14]. The PCE method can be used to propagate input uncertainties to the

output variables of interest, which can be analysed and assessed correspondingly [62,15]. The PCE method has a strong mathematical basis and represents a collection of polynomial approximations, which was particularly designed for uncertainty quantification [16]. There are intrusive and non-intrusive PCE methods [39,29,52]: the former is a complete non-sampling-based method to determine the uncertainty propagation by modification of the control equations, Ordinary/Partial Differential Equations (ODE/PDE), as a stochastic ODE/PDE considering a probabilistic assumption of the input parameters; while the latter computes the uncertainties without knowing the analytical expression of the control equations, which however is a less-sampling-based method compared to the Monte Carlo method [15,52]. We shall adopt the non-intrusive PCE method to propagate and analyse uncertainties in this paper.

Many publications in the domain of robust optimisations are based on linear or quadratic programming, or convex optimisations [4,5,7]. For engineering applications, most robust optimisations focus on a straightforward use of Monte Carlo method to create input data points and a surrogate model to evaluate the outputs [43,63,50,34] assuming that the data are reliable or noise-free; or use of the PCE method with black-box simulation models to create probabilistic surrogates and optimising objectives [2,18,26]. Both the GPR and PCE methods have been adopted in a recent paper [56] for a probabilistic surrogate model and uncertainty propagation respectively. A combined study of surrogate model, error quantification and robust optimisation is limited. Most of existing methods usually provide a black-box flow chart of the overall algorithm [26,56], without study of the intermediate result where errors may be introduced. There is a lack of a general framework which enables us to distinguish and analyse different types of uncertainties, for example, uncontrollable noise from the training dataset, noise due to limited/finite discrete data points, or controllable errors from the inputs. We shall consider these types of uncertainties in this paper and consequently formulate rigorous optimisation problems.

There are specific questions we try to answer in this article: (i) when using a surrogate model to predict outcomes, are there any uncertainties from the surrogate itself and how to quantify these? (ii) when propagating errors from the input to the output variables, has the prediction (mean or standard deviation) converged in terms of, such as sampling points when using Monte Carlo method or the polynomial order when using PCE method? (iii) after quantification of different sources of uncertainties, how to formulate the robust optimisation problems?

In order to address the above questions, we shall use the GPR method to create a surrogate model with consideration of uncertainties and the PCE method to quantify the error propagation. Both the GPR and PCE methods rely on a training dataset (including inputs and outputs), and the model parameters have to be trained/optimised before applying these methods to predict output values on any new input data. A subtle difference between these two methods is that the PCE model requires its input data satisfies a specified probabilistic distribution, while the GPR model does not (although it requires the error satisfies Gaussian distribution when training the model). The GPR method is designed to predict a mean as well as quantify uncontrolled uncertainties such as noise from the training dataset, while the PCE method is convenient to quantify uncertainty (both controlled and uncontrolled) propagation from input variables to output variables so that one can analyse and optimise corresponding variables of interest. To the

best of our knowledge, the combination of these two methods has not been fully studied. A PC-Kriging method is introduced in [48] to achieve more accuracy than two distinct techniques GPR and PCE; another surrogate modelling technique is developed based on GPR but using the least-angle-regression selection algorithm of PCE method in [31]; [32] combines the GPR and PCE methods and formulates a multi-fidelity design optimisation algorithm, and they focus on the study of the use of low-fidelity simulation data to obtain more information about the design space. Our study is focused on creating a rigorous robust optimisation framework, which allows us to optimise multi-objective functions while considering different types of uncertainties.

Geometry of the serpentine channels of PCR devices is very important and a number of studies have attempted to improve the PCR performance based upon different objectives. For example, spiral microchannels are used in [23, 37] to reduce the reaction time, while radial [46] and straight channels [11, 17] have also been explored to achieve the same goal. It is demonstrated in [13] that the wall temperature is more uniform by adopting a diverging fluidic channel, which was a first step to improve the overall temperature uniformity within the PCR zones. In addition, an electro-kinetic flow is used to create a plug-like velocity profile in [21], which reduces sample dispersion and increases flow-rate control within PCR channels and further improves the flow uniformity. Other objectives and control variables have also been explored, for example, minimising the heating power by optimising the thermal properties which are biologically compatible with the PCR liquid has been studied in [36]. In our previous study [22], we performed the Computational Fluid Dynamics (CFD) analysis for a PCR device, and focused on single objective and deterministic optimisation. This paper applies a novel robust optimisation approach to the geometrical design and pattern arrangement of the serpentine channel of a PCR device, with the aim of minimising competing objectives: pressure drop and temperature uniformity.

The original contributions of this paper are summarised as follows: (1) we propose a general two-objective robust optimisation framework based on a combination of Gaussion process regression and polynomial chaos expansion, by asking rigorous mathematical questions in every step, to the best of our knowledge, which have not been addressed in the engineering community. For example, how to evaluate the error of the surrogate model when two groups of sampling points provide different outputs? whether the order of the PCE polynomials is appropriate (too low or too high)? (2) we analyse and quantify different types of uncertainties, such as uncontrollable noise from the training dataset, noise due to limited/finite discrete data points, or controllable errors from the inputs; (3) three new robust optimisation problems are formulated and solved using open-source libraries GPy, ChaosPy and SciPy; (4) the proposed approach is applied to robust design optimisation of a polymerase chain reaction flow system, which provides engineers a reliable guidance for manufacturing the PCR device.

The paper is organised as follows. Section 2 introduces the GPR and PCE methods generally, which are implemented in Section 3 to produce surrogate models with uncertainty. Three robust optimisation problems are introduced and solved in Section 4, with conclusions drawn in section 5. To avoid diverging from the main context of this paper, we provide the source of the training dataset in Appendix 6.1, and Python code to test the PCE method in Appendix 6.3, to predict on uni-

form grids in Appendix 6.4 and to optimise the objective functions in Appendix 6.5.

2 Gaussian process regression and polynomial chaos expansions

In this section, we briefly review and introduce the main tools used through this paper: Gaussian process regression and polynomial chaos expansion.

2.1 Gaussian process regression

Without losing generality, let us consider a dataset with two input and two output variables: $(x_1, x_2, y_1, y_2)_j$ $(j = 1, 2, \ldots, n)$ denote the n data points with input $\mathbf{x} = (x_1, x_2)$ and corresponding output (y_1, y_2). The GPR can learn the relation $f_i(\cdot)$ between $\mathbf{x}$ and y_i $(i = 1, 2)$ from these n training data points, i.e. it computes a surrogate model $f_i(\mathbf{x}) \sim \mathcal{N}\left(\mu_{f_i}^{gpr}, (\sigma_{f_i}^{gpr})^2\right)$ with the superscript "gpr" denoting the mean or standard deviation from the GPR model:

$$\mu_{f_i(\mathbf{x})}^{gpr} = \mathbf{k}^T \left(\mathbf{K} + \alpha \mathbf{I}\right)^{-1} \mathbf{y}_i, \tag{2}$$

and

$$\sigma_{f_i(\mathbf{x})}^{gpr} = k(\mathbf{x}, \mathbf{x}) - \mathbf{k}^T \left(\mathbf{K} + \alpha \mathbf{I}\right)^{-1} \mathbf{k}. \tag{3}$$

In the above, $\mathbf{y}_i = (y_{i1}, y_{i2}, \ldots, y_{in})^T$ $(i = 1, 2)$ is a column vector from the observations of the i^{th} component of the output variables, and α is a specified noise in this observation $\mathbf{y}_i$. $k = k(\mathbf{x}_p, \mathbf{x}_q) = \sigma_f^2 exp\left(-\frac{1}{2l^2}|\mathbf{x}_p - \mathbf{x}_q|^2\right)$ is the squared-exponential covariance function, with σ_f and l being hyperparameters, which can be determined by cross validation [59,35] or maximising the marginal likelihood [59], **I** is the identify matrix, and

$$\mathbf{k} = \left[k(\mathbf{x}, \mathbf{x}_1), k(\mathbf{x}, \mathbf{x}_2), \ldots, k(\mathbf{x}, \mathbf{x}_n)\right]^T, \tag{4}$$

$$\mathbf{K} = \begin{bmatrix} k(\mathbf{x}_1, \mathbf{x}_1) & k(\mathbf{x}_1, \mathbf{x}_2) & \cdots \\ \vdots & \ddots & \\ k(\mathbf{x}_n, \mathbf{x}_1) & & k(\mathbf{x}_n, \mathbf{x}_n) \end{bmatrix}. \tag{5}$$

The standard deviation in (3) of the GPR model can be interpreted as uncertainties from two different sources: one is the noise α of the training data point (y_1, y_2), and the other one is the discrete error of the training dataset itself, due to the finite/limited data points. It is reasonable to assume that data points from CFD simulations are clean and noise free [45,51], so that we will be able to specify a very small α when training our GPR model in Section 3.1. However, we do not take this for granted and we also test the convergence of $\mu_{f_i(\mathbf{x})}^{gpr}$ and $\sigma_{f_i(\mathbf{x})}^{gpr}$ in terms of this noise parameter α, and choose the converged α for other analyses. For the second type of uncertainty, we shall analyse and quantify this uncontrollable noise in Section 3.2 and distinguish it from the manufacturing errors.

2.2 Polynomial chaos expansions

Real manufacturing processes create inevitably a certain level of error in the input geometrical parameters. Our approach is to assume a reasonable control error in the input parameters and use the PCE method to propagate this error to the output variables. These resultant probabilistic surrogate models are then used to solve optimisation problems in Section (4).

Considering an error e_x around an input point $\mathbf{x} = (x_1, x_2)$, we need to know the corresponding error e_y in the outputs $y_i = f_i(\mathbf{x})$ $(i = 1, 2)$. A fast analysis of this problem is to use the Taylor expansion to express y_i around $\mathbf{x}$ if the derivative of f_i is available. Alternatively, a convenient approach might be to create random points (based upon a distribution assumption such as $e_x \sim \mathcal{N}(0, \sigma_n^2)$) around the input $\mathbf{x}$, and calculate the corresponding statistics around the output $y_i = f_i(\mathbf{x})$. The former is an efficient approach, but unfortunately one cannot easily access the derivative of f_i for many engineering problems. The latter is the so-called Monte Carlo method which can be very slow – millions of data points can be needed in order to achieve a converged result (see Appendix 6.3 for two tests using this method).

The PCE method is much more efficient than the Monte Carlo approach (see 6.3), and only needs a few data points around $\mathbf{x}$ in order to calculate the error at the output. We assume x_1 and x_2 are independent random variables, and the PCE method approximates the output y_i $(i = 1, 2)$ as a linear combination of orthogonal polynomial basis $\varphi_k(\mathbf{x}) = \varphi_{k_1}(x_1)\varphi_{k_2}(x_2)$,

$$y_i \approx \sum_{k=0}^{m} \beta_k^i \varphi_k(\mathbf{x}). \tag{6}$$

Both x_i and y_i $(i = 1, 2)$ are regarded as random variables. We assume our input variables satisfy Gaussian distribution, which requires Hermite PCE basis functions. Other distributions require different basis functions [1, 15].

There are intrusive and non-intrusive PCE methods to compute the PCE coefficients β_k^i $(k = 0, \ldots, m)$. The former requires modification of the governing equations of the system under study, while the latter are sampling-based methods requiring solutions of the governing equations for specific values of the random variables considered [52]. Non-intrusive methods provide data-driven models based on experimental or simulation data. Here, we use a non-intrusive method, for which there are pseudo-spectral projection and linear regression methods to compute the coefficients based on design variable sampling. Both of these methods are implemented in ChaosPy [15], which is used in this paper.

3 Surrogate model

In this section, we demonstrate the process of creating a probabilistic surrogate model step by step, with quantification of different sources of uncertainties. Considering simulation data from a PCR thermal flow system (Appendix 6.1): one simulation provides one data point: $(x_1, x_2, y_1, y_2) = (W_c, H_c, \Delta p, T_{dev})$, with W_c and H_c being the channel width and height of the PCR device respectively, and Δp and T_{dev} being the pressure drop and temperature standard deviation of the

PCR thermal flow system respectively. We generate simulation data at $n = 100$ input points $(W_c, H_c) \in [0.015, 0.5] \times [0.05, 0.15]$, including 20 evenly spaced points at boundaries and 80 random, uniformly distributed points inside the domain as shown in Figure 1. These generate the corresponding output variables $(\Delta p, T_{dev})$ which are found to lie within $[50.87, 1437] \times [12.87, 16.29]$. We deliberately use data at the boundaries to improve the accuracy of extrapolation outside of the domain, which is required when applying the PCE method, as will be discussed in Section 3.2.

For presentational convenience, all the data points $(x_1, x_2, y_1, y_2)_j$ $(j = 1, 2, \ldots, n)$ are normalised to the range $[0, 1]$ by

$$\frac{x_i - x_i^{min}}{x_i^{max} - x_i^{min}}, \quad \frac{y_i - y_i^{min}}{y_i^{max} - y_i^{min}}, \quad i = 1, 2 \tag{7}$$

before feeding the training data to the GPR or PCE codes. The results can be easily transformed from the normalised space to the physical space using (7), and the corresponding scales in the physical domain are given when necessary in the following sections. We also generate a test dataset of $N_{test} = 100 \times 100$

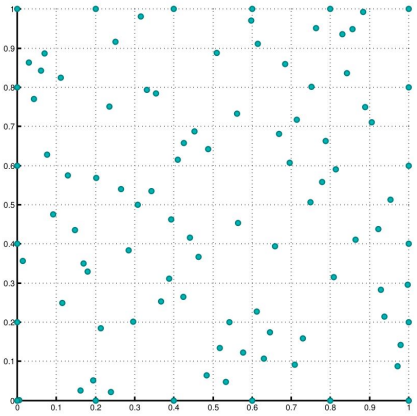


Fig. 1: Training data points: 20 evenly spaced points on boundaries and 80 random, uniformly distributed points inside the domain.

evenly spaced points in the domain $\Omega = [-0.1, 1.1] \times [-0.1, 1.1]$ in order to test the surrogate model. The domain is extended by 0.1 around the boundary of the original domain $\Omega_0 = [0, 1] \times [0, 1]$, so that we can extrapolate values to accurately compute the error propagation at the boundaries.

3.1 GPR model

As discussed in Section 2.1, parameter α is a specified noise in the GPR model; it is also a regularisation parameter to inverse the matrix $\mathbf{K}$ as indicated in formula (2) and (3). Although it is reasonable to specify a small noise parameter α for data from the CFD simulations [45,51], the question is how small should α be and

whether too small α would lead to a singular matrix $\mathbf{K} + \alpha\mathbf{I}$. A proper method to address this question is to test convergence of the outputs (mean and standard deviation here) in terms of the parameter α. Here, we investigate the effect of varying α from 10^{-12} to 10^{-8} on the accuracy and stability of the GPR algorithm by comparing the predicted mean and standard deviation. We plot the predicted norm of the mean and standard deviation as a function α in Figure 2, from which it can be seen that the mean is effectively constant, and there is no instability issue when using such small values of α. Therefore, we will use a converged value $\alpha = 10^{-11}$ in the following sections. The mean and standard deviation response surfaces are plotted in Figure 3, noting from (3) that the standard deviation only depends on the input data, so we have $\sigma^{gpr}_{f_1} = \sigma^{gpr}_{f_2}$.

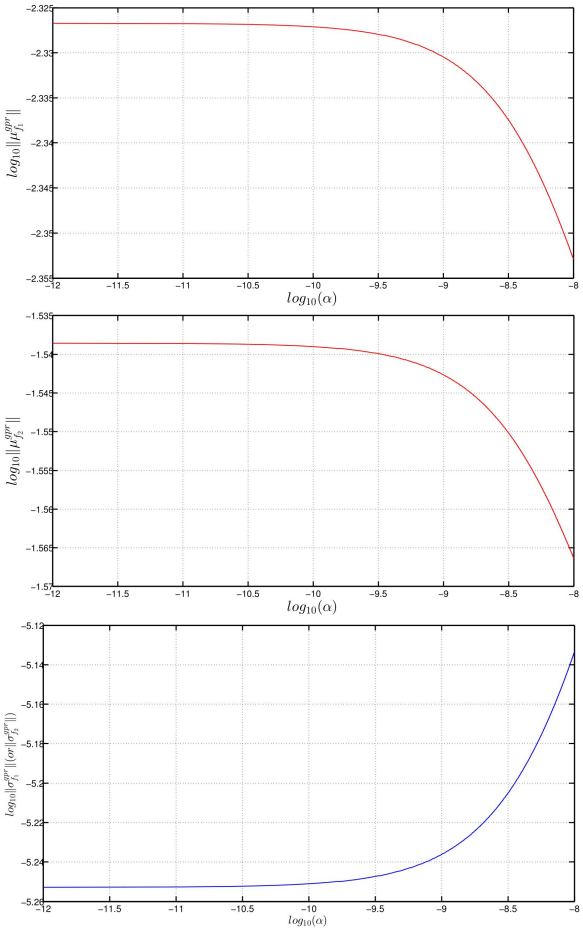


Fig. 2: Mean and standard deviation for f_1 and f_2 as a function α, where $\|\cdot\| = \frac{1}{N_{test}}\sqrt{\sum_{k=1}^{N_{test}}(\cdot)^2}$.

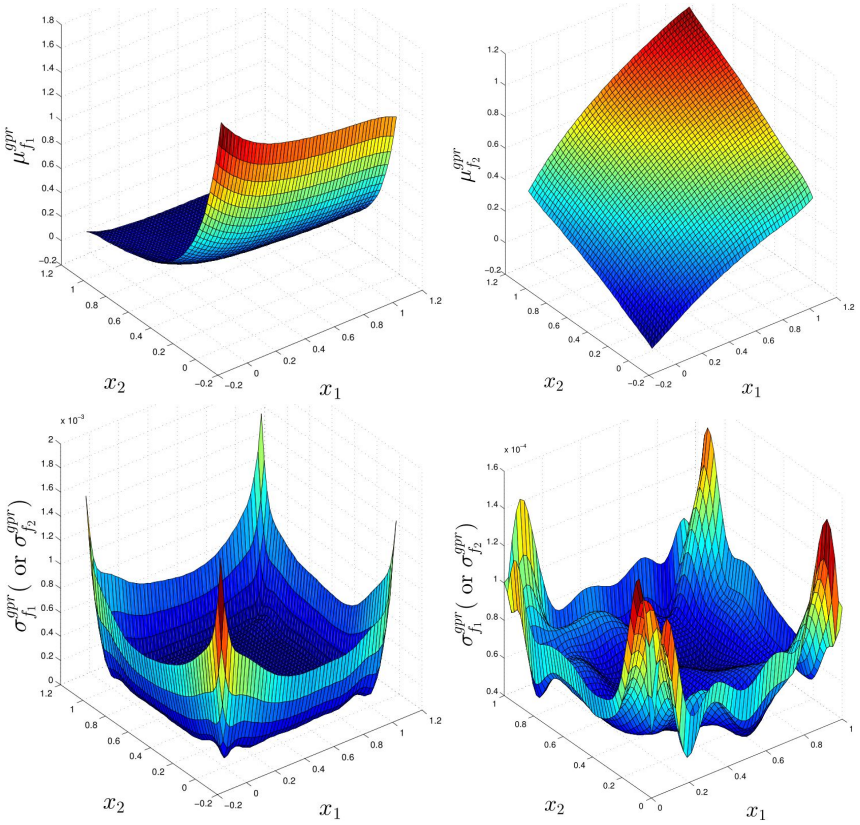


Fig. 3: The mean of $f_1(\mathbf{x})$ and $f_2(\mathbf{x})$, and standard deviation plotted on $\Omega = [-0.1, 1.1] \times [-0.1, 1.1]$ and $\Omega_0 = [0, 1] \times [0, 1]$.

3.2 Probabilistic surrogate model

As a starting point, we assume a manufacturing error $e_x = 0.05$ in the input training dataset [43,47], so that we have 95% confidence that an input data point x_i $(i = 1, 2)$ lies in $[x_i - e_x, x_i + e_x]$. Under the assumption of $e_x \sim \mathcal{N}(0, \sigma_n^2)$, we have $\sigma_n = e_x/2 = 0.025$. We then use the PCE method to propagate this error to the outputs and create a probabilistic surrogate model, based upon the GPR surrogate model for calculating function values around point $\mathbf{x} = (x_1, x_2)$ (see Appendix 6.3 for a comparison of this PCE method to the Monte Carlo method widely used in literature). However, before doing this we ask two questions as follows: (i) what is the appropriate order for the polynomial basis used in the PCE method? (ii) How to quantify and distinguish the uncertainties from the surrogate model and the manufacturing error?

3.2.1 Convergence of the PCE method:

in order to compute the PCE coefficients at a point $\mathbf{x} \in \Omega_0$, we need several quadrature points $\tilde{\mathbf{x}}$ around $\mathbf{x}$ (pseudo-spectral projection method [15]). The input

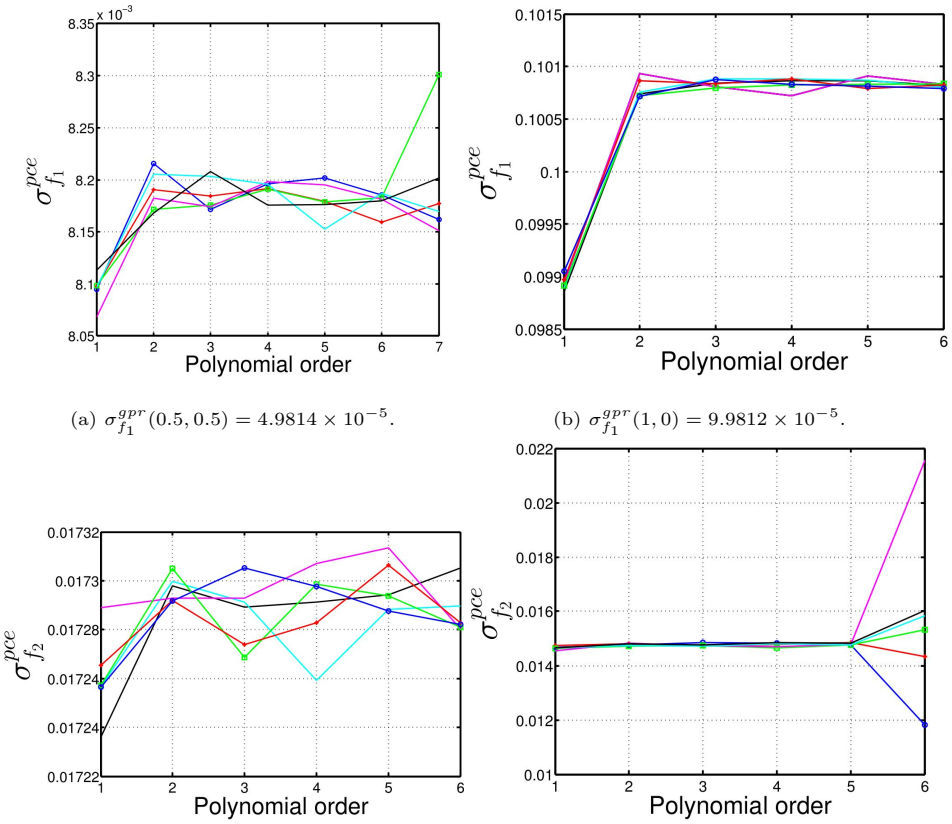
of these quadrature points are determined by $\mathbf{x}$ and the order of the polynomial basis of the PCE method, and the outputs are computed by our GPR surrogate model. However, instead of using $\mu^{gpr}_{f_i(\tilde{\mathbf{x}})}$ to directly compute the ouptut values, we should consider the uncertainty $\sigma^{gpr}_{f_i(\tilde{\mathbf{x}})}$ as well; we can *not* neglect $\sigma^{gpr}_{f_i(\tilde{\mathbf{x}})}$ by directly comparing its magnitude with the manufacturing error $\sigma_n = e_x/2$, because σ_n is from the input space while $\sigma^{gpr}_{f_i(\tilde{\mathbf{x}})}$ is in the output space.

We test the PCE method at several different points inside the domain Ω_0 as well as on its boundaries. In order to test the influence of $\sigma^{gpr}_{f_i}$ on the error propagation, we add Gaussian noise, generated by $\mathcal{N}\left(0, \left(\sigma^{gpr}_{f_i(\tilde{\mathbf{x}})}\right)^2\right)$, to $\mu^{gpr}_{f_i(\tilde{\mathbf{x}})}$ for every point $\tilde{\mathbf{x}}$ to compute the PCE coefficients as discussed in Section 2.2 – the Python implementation is given in Appendix 6.3.

We test our code based on several cases of random noise at different random points, and we report in Figure 4 the error at three particular points of interest for six cases of random noise, from which it can be seen that a 3^{rd} order polynomial basis would be sufficiently accurate to approximate the error propagation. Notice that in Figure 4 the superscript "*pce*" denotes the mean or standard deviation computed using the PCE method, corresponding to the superscript "*gpr*" (used through this paper) for the GPR model. In order to compute the coefficients of this 3^{rd} order polynomial (using the pseudo-spectral projection method [15]), we need to evaluate $\mu^{gpr}_{f_i}$ at points 0.05836 away from the boundary of Ω_0, which can be achieved by the extrapolation of our GPR model. Also notice that higher order PCE polynomial basis is needed to evaluate $\mu^{gpr}_{f_i}$ at points further away from the boundary. For example, a 6^{th} order polynomial chaos expansion at point $(1, 1)$ needs to compute $\mu^{gpr}_{f_i}$ at point $(1.09376, 1.09376)$ which has large variation as shown in Figure 3. This introduces extrapolation errors as well, which can be observed from Figure 4 (a) and (d) using a 7^{th} and 6^{th} order of polynomials respectively. We also present the corresponding standard deviation of the GPR model in the captions in Figure 4, from which we can see this standard deviation of the GPR model is negligible compared with the standard deviation induced by the manufacturing errors. This observation is further validated in Figure 5 where the ratio of the standard deviation of the GPR model to the standard deviation of the PCE model (where the random noise is considered as described above) is less than 0.1 for f_1 and 0.001 for f_2.

3.2.2 Noise propagation and response surface with confidence region:

we can now use the 3^{rd} order polynomial and the GPR model to propagate the input errors to the outputs and create a probabilistic surrogate model, which incorporates the uncertainty from manufacturing process. The response surfaces with a 95% (two standard deviation) confidence region are plotted in Figure 6, from which it can be seen that the input error is amplified where the response surface is steep. This is consistent with analysis using Taylor expansions. We also notice that the second design variable x_2 has less influence on objective f_1 as shown in Figure 6: $f_1(x_1, x_2)$ is almost constant for the same x_2, and the two design variables have almost equal influences on objective f_2: $f_2(x_1, x_2)$ is visually symmetric along $x_1 = x_2$. We plot in Figure 7 the projection of this mean surface of $f_1(x_1, x_2)$ to $x_1 = 0$ with a confidence interval for increased clarity.



(a) $\sigma^{gpr}_{f_1}(0.5, 0.5) = 4.9814 \times 10^{-5}$.

(b) $\sigma^{gpr}_{f_1}(1, 0) = 9.9812 \times 10^{-5}$.

(c) $\sigma^{gpr}_{f_2}(0.5, 0.5) = 4.9814 \times 10^{-5}$.

(d) $\sigma^{gpr}_{f_2}(1, 1) = 9.9805 \times 10^{-5}$.

Fig. 4: Error propagation using the PCE method at different points for six cases of random noise in each figures.

4 Robust optimisation

The probabilistic surrogate model is now used to solve the following three robust optimisation problems. The optimisation problems considered in this section all involve non-linear objective functions and constraints, and we shall use the trust region algorithm provided in the Python library SciPy to solve these challenging optimisation problems. We find that the convergence of the algorithm highly relies on how stringent the constraints are: the more stringent the constraints are, the more difficult the algorithm converges. However, we would not focus on the study of the performance of the optimisation algorithm in the paper. Instead, we focus on the use of open optimisation libraries and we publish a typical implementation of one of the following optimisation problems in Appendix 6.5.

Problem 1: Given the probabilistic models $f_1(\mathbf{x})$ and $f_2(\mathbf{x})$,

$$\begin{aligned} \underset{\mathbf{x}\in\Omega_0}{\text{minimize}} \quad & \mu_{\omega(\mathbf{x})} =: \omega\mu^{pce}_{f_1(\mathbf{x})} + (1-\omega)\mu^{pce}_{f_2(\mathbf{x})}, \\ \text{subject to} \quad & \sigma^{pce}_{f_1(\mathbf{x})} < \sigma_1, \quad \sigma^{pce}_{f_2(\mathbf{x})} < \sigma_2. \end{aligned} \tag{8}$$

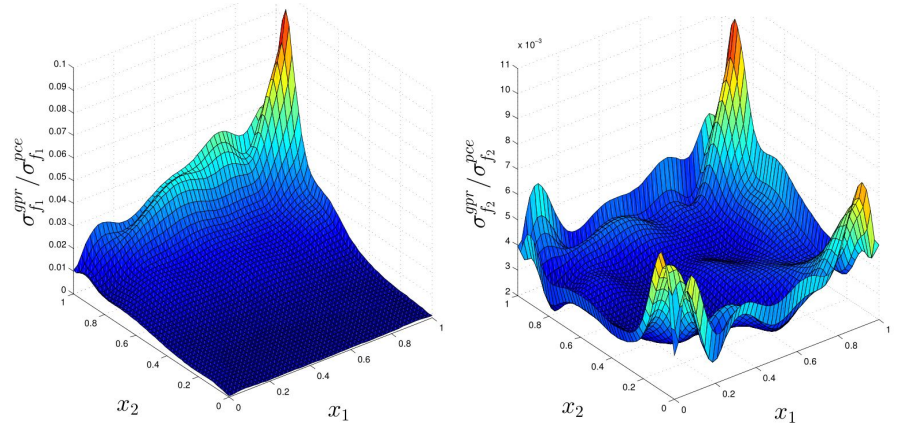


Fig. 5: The ratio of the standard deviation of the GPR model to the standard deviation of the PCE model.

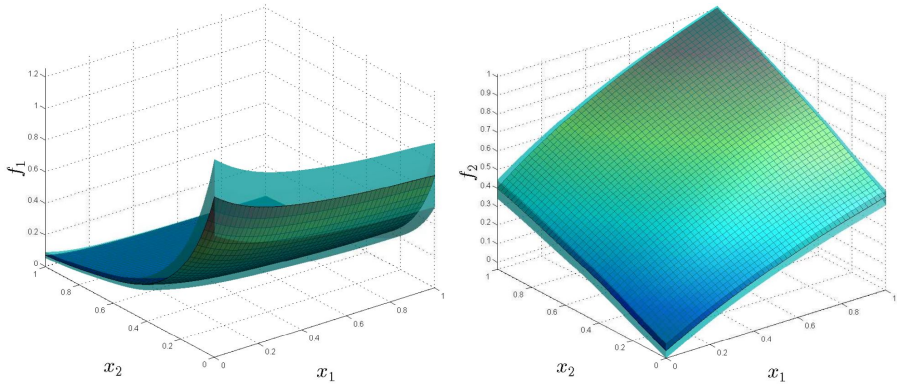


Fig. 6: Response surfaces with 95% confidence region using the PCE method to propagate an error of 5% (of the maximum: 1 in the normalised space, and 0.5 for W_c and 0.15 for H_c in the physical space) from the input to the output.

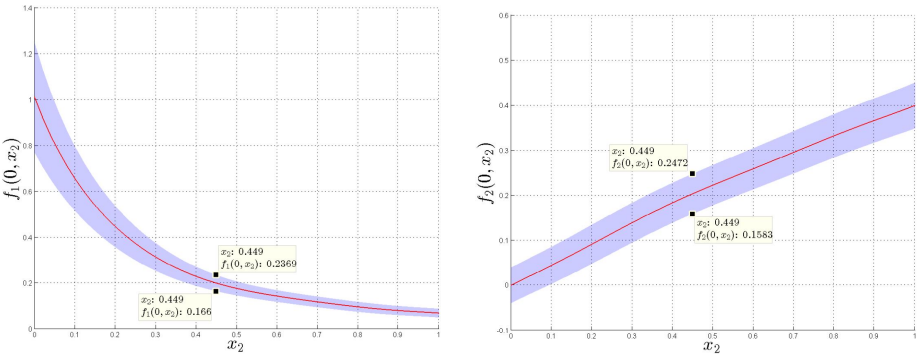


Fig. 7: Response surfaces projected to $x_1 = 0$ with 95% confidence interval.

with $\omega \in [0, 1]$.

We first study the Pareto curve which is defined by

$$\mathbf{p}(\omega) = \left(\mu^{pce}_{f_1(\mathbf{x}^*(\omega))}, \mu^{pce}_{f_2(\mathbf{x}^*(\omega))}\right) \text{ with } \mathbf{x}^*(\omega) = \underset{\mathbf{x}\in\Omega_0}{\text{argmin}}\mu_{\omega(\mathbf{x})}. \tag{9}$$

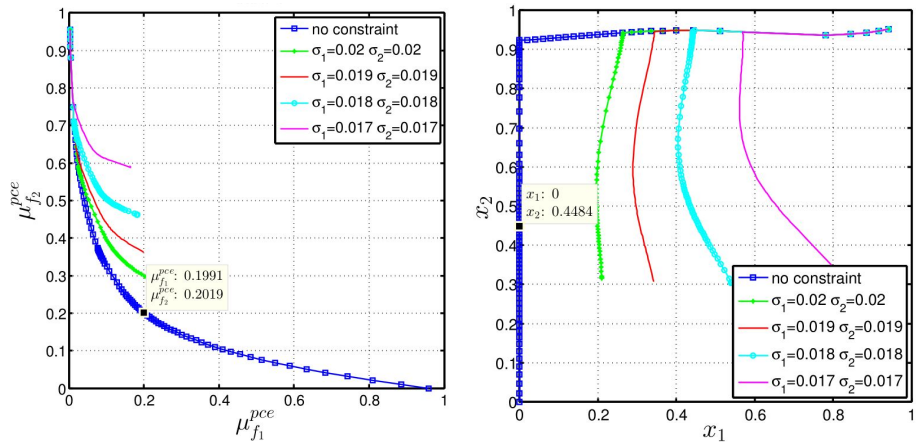
The Pareto curve is a function of parameter ω, which gives a set of points $\left(\mu^{pce}_{f_1(\mathbf{x}^*(\omega))}, \mu^{pce}_{f_2(\mathbf{x}^*(\omega))}\right)$ or curve on a two dimensional plane. In Figure 8 (a), the Pareto curve are plotted for four different constrained cases and one unconstrained case. First, we observe that the Pareto curve is pushed away from the origin as the constraints equally become more stringent, noting also this has a greater influence on the second objective $\mu^{pce}_{f_2}$. Consequently, the corresponding optimal design space is pushed away form the boundaries of the original design space Ω_0 as shown in Figure 8(b). Secondly, the marked points in Figure 8 indicate a compromise minimisation between the two objectives $\mu^{pce}_{f_1}$ and $\mu^{pce}_{f_2}$, where both objectives achieve a minimum of around 0.2 ($328.096Pa$ for pressure drop and $13.554°C$ for the temperature deviation in the physical space) with corresponding $\omega = 0.45$ in (8). The optimal design point $(0, 0.4484)$ as shown in Figure 8(b) has a 95% confidence interval $(0.166, 0.237) = 0.2015 \pm 0.071$ for f_1 and $(0.158, 0.247) = 0.2025 \pm 0.089$ for f_2 as indicated by the marked points in Figure 7. Thirdly, if we want to shrink the confidence intervals so that our prediction is more robust, such as with a 95% confidence interval of ± 0.04 (standard deviation of 0.02) for both objectives, we can move from point $(0.1991, 0.2019)$ on the blue curve in Figure 8(a) to the green curve with the same $\mu^{pce}_{f_1}$. In this case, we are more confident that we would be able to achieve that objective, which is now around 0.2 for $\mu^{pce}_{f_1}$ (unchanged) and 0.3 for $\mu^{pce}_{f_2}$. We briefly conclude as follows: the more we can reduce $\mu^{pce}_{f_1}$ ($\mu^{pce}_{f_2}$), the less we would reduce $\mu^{pce}_{f_2}$ ($\mu^{pce}_{f_1}$) – moving from one end to other on each curve in Figure 8 (a); the more confident we are in the reduction of $\mu^{pce}_{f_1}$ (moving from the blue curve across the green one, and towards the purple curve at a fixed $\mu^{pce}_{f_1}$ in Figure 8(a)), the less confident we are in the reduction in $\mu^{pce}_{f_2}$. We can achieve this by keeping the second design variable constant and increasing only the first design variable as shown in 8 (b). The reason we can do this is because the response surface for f_1 is almost constant for a fixed x_2 as shown in Figure 6.

We test another case of constraints in Figure 9, where the upper bound of $\sigma^{pce}_{f_1}$ varies while the upper bound of $\sigma^{pce}_{f_2}$ stays the same. It can be seen from Figure 9 that the first parameter x_1 of the optimal designs is almost unchanged while the second one x_2 varies rapidly as the upper bound of σ_1 varies. This is different in the first test case shown in Figure 8 where the main variation of the optimal designs lies in the first parameter x_1.

Problem 2: Given the probabilistic models $f_1(\mathbf{x})$ and $f_2(\mathbf{x})$,

$$\begin{aligned} \underset{\mathbf{x}\in\Omega_0}{\text{minimize}} \quad & f_w(\mathbf{x}) =: \mu^{pce}_{f_2(\mathbf{x})} + 2\sigma^{pce}_{f_2(\mathbf{x})}, \\ \text{subject to} \quad & \mu^{pce}_{f_1(\mathbf{x})} + 2\sigma^{pce}_{f_1(\mathbf{x})} < \bar{f}_1. \end{aligned} \tag{10}$$

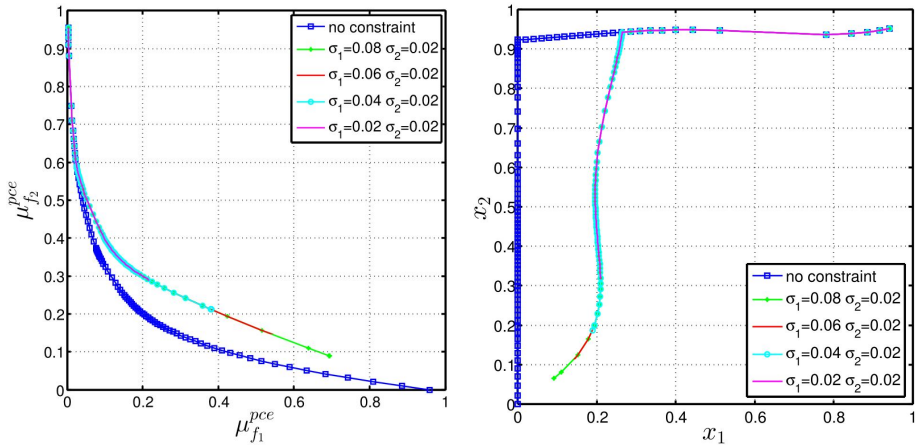
The objective function $f_w(\mathbf{x})$ in Problem 2 defines the "worst case" for f_2 with 95% confidence, and a minimisation of $f_w(\mathbf{x})$ given a safe (95% confidence) upper bound $\bar{f}_1$ for f_1. We plot, in Figure 10, $f^*_w = \min_{\mathbf{x}\in\Omega_0} f_\omega(x_1, x_2)$ and $(x^*_1, x^*_2) =$



(a) Pareto curve for the first test case of Problem 1.

(b) Opimal design space for the first test case of Problem 1.

Fig. 8: Pareto curve $\mathbf{p}(\omega)$ and the corresponding designs for different cases of constraints.



(c) Pareto curve for the second test case of Problem 1.

(d) Opimal design space for the second test case of Problem 1.

Fig. 9: Pareto curve $\mathbf{p}(\omega)$ and the corresponding designs for different cases of constraints.

$\underset{\mathbf{x}\in\Omega_0}{\text{argmin}} f_\omega(x_1, x_2)$ as functions of $\bar{f}_1$, i.e.: $f_w^*(\bar{f}_1)$, $x_1^*(\bar{f}_1)$ and $x_2^*(\bar{f}_1)$. It can be seen from Figure 10 that minimising f_w is always accompanied with an increasing upper bound $\bar{f}_1$. The meaning of the marked points (which correspond to the same points in the design space) shown in Figure 10 is: we can reduce f_w to $f_w^* = 0.2418$ by choosing $x_1 = x_1^* = 0$ and $x_2 = x_2^* = 0.4354$, at the same time we also have 95% confidence that $f_1 < 0.2424$.

Problem 3: Given the probabilistic models $f_1(\mathbf{x})$ and $f_2(\mathbf{x})$,

$$\begin{aligned} \underset{\mathbf{x}\in\Omega_0}{\text{minimize}} \quad & f_w(\mathbf{x}) =: \mu^{pce}_{f_1(\mathbf{x})} + 2\sigma^{pce}_{f_1(\mathbf{x})}, \\ \text{subject to} \quad & \mu^{pce}_{f_2(\mathbf{x})} + 2\sigma^{pce}_{f_2(\mathbf{x})} < \bar{f}_2. \end{aligned} \tag{11}$$

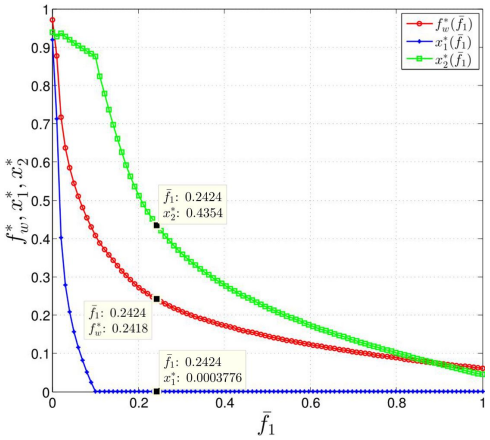


Fig. 10: Solution of Problem 2 as a function of upper bound $\bar{f}_1$.

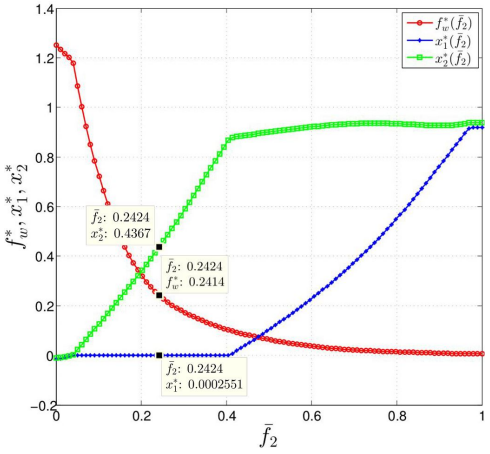


Fig. 11: Solution of Problem 3 as a function of upper bound $\bar{f}_2$.

Similar to Problem 2, Problem 3 can be interpreted as minimising the "worst case" of f_1 given a stringent (95% guaranteed) constraint for f_2. It can be seen from Figure 11 that we can reduce f_1, with 95% confidence, to $f_w^* = 0.2414$ by choosing $x_1 = x_1^* = 0$ and $x_2 = x_2^* = 0.4367$, at the same time we also have 95%

confidence that $f_2 < 0.2424$. This is consistent with the result obtained by solving Problem 2.

5 Conclusion

In this paper, we present a robust optimisation framework which can be applied to general data-driven two-objective optimisation problems. We provide a rigorous and detailed analysis of uncertainties from both the input and output variables based upon Gaussian Process Regression (GPR) and Polynomial Chaos Expansion (PCE), and quantify the uncertainties of surrogate model by analysing statistics of different types of noise from the dataset. Our surrogate model is not deterministic due to white noise and limited training dataset, and it provides us a quantified error. Therefore, we quantitatively distinguished and compared this error with the manufacturing error when propagating it from the input to output. Before evaluating the error at the output, we also tested the convergence of error in terms of number of sampling points and polynomial order, and ensured the output error was not varying. Based upon these rigorous evaluation, we formulated and solved rigorous optimisation problems based upon our surrogate model, which provides us a way to optimise the objectives with control of input errors. We assess the proposed methodology using validated data from CFD simulation of a thermal fluid process on a micro Polymerase Chain Reaction (PCR) device, and conclude that the approach is efficient, robust, general and easy to implement using open-source libraries: GPy [25] to implement the GPR method, ChaosPy [15] to implement the PCE method and SciPy [8] to solve the optimisation problems. Our optimisation results provide engineers a reliable guidance for manufacturing the PCR device. Although the method is presented based upon a two-objective optimisation problem and data from CFD simulations of a PCR device, it is straightforward to be extended to multi-objective case using laboratory data, which is our ongoing research.

6 Appendix

6.1 Validated data from CFD analysis

The thermal flow model is based on a prototypical PCR flow analysed in our recent paper [22], which has been validated both numerically and experimentally. We briefly describe the model in this appendix.

PCR systems are typically based on a serpentine fluidic channel arrangement, see Figure 12, in order to create a thermal cycling procedure which amplifies DNA segments, allowing detection and identification of gene sequences. Within each straight channel component there are three thermal stages of denaturation (95^oC), annealing (56^oC) and extension (72^oC). The geometrical parameters include W_c, H_c, W_w, H_b and L, which are the channel width, height, the spacing between the channels, the bottom height and total length respectively.

A complex heat transfer model coupled with incompressible Navier-Stokes equations is adopted to describe this thermal flow system, and the governing equations are solved using COMSOL Multiphysics 5.4, see [22] for more details. Nu-

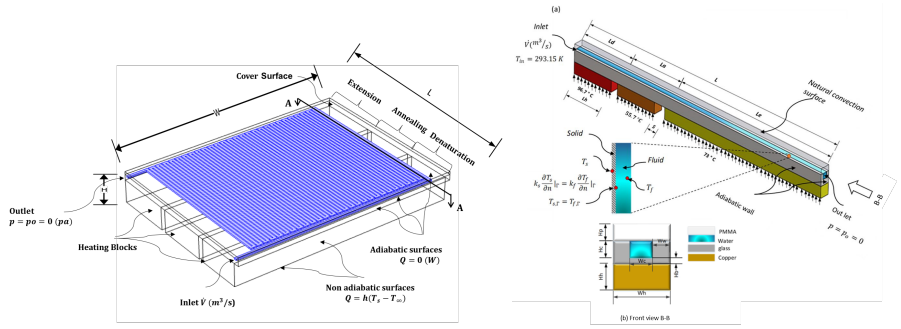


Fig. 12: A schematic diagram of the whole (left) and a section (right) of the serpentine microfluidic channel.

merical validations include first, mesh convergence on a series of structured finite element grids, and then comparison with both experimental [41] and published numerical results [10]: Figure 13 shows a good agreement between the experimental and our numerical results, and Figure 14 shows that the numerical predictions of the temperature profile along the three temperature zones are in very good agreement with the published results.

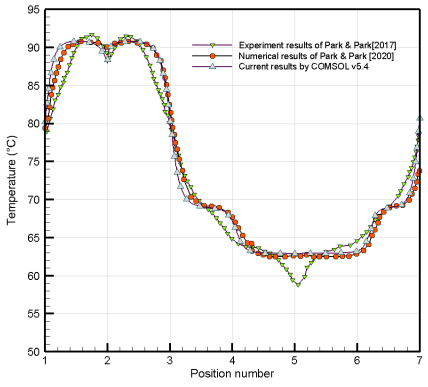


Fig. 13: Surface temperature variation along flow direction in a diverging microchannel.

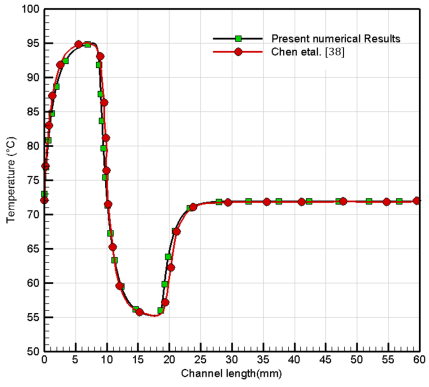


Fig. 14: Average temperature profile along the centreline.

6.2 The GPR-assisted optimisation

In this section, we test the GPR-assisted optimisation algorithm and compare it with a two-dimensional and a three-dimensional analytical benchmarks: the Rosenbrock function [44] and the Ishigami function [28], which have been widely used to validate surrogate model and optimisation algorithms in the literature [48,

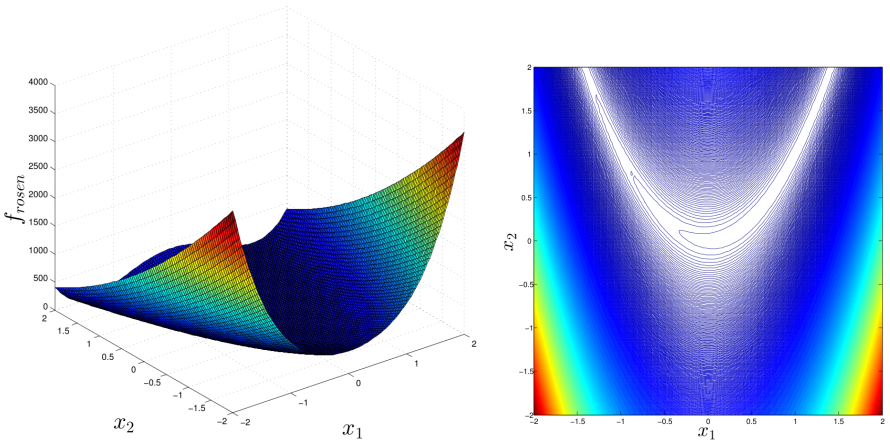
30]. The Rosenbrock function is a polynomial function with a two-dimensional input space:

$$f_{rosen} = 100\left(x_2 - x_1^2\right)^2 + (1 - x_1)^2, \tag{12}$$

with $x_1, x_2 \in [-2, 2]$. The Ishigami function is a smooth function with three input parameters:

$$f_{ishigami} = sin(x_1) + 7sin^2(x_2) + 0.1x_3^4 sin(x_1), \tag{13}$$

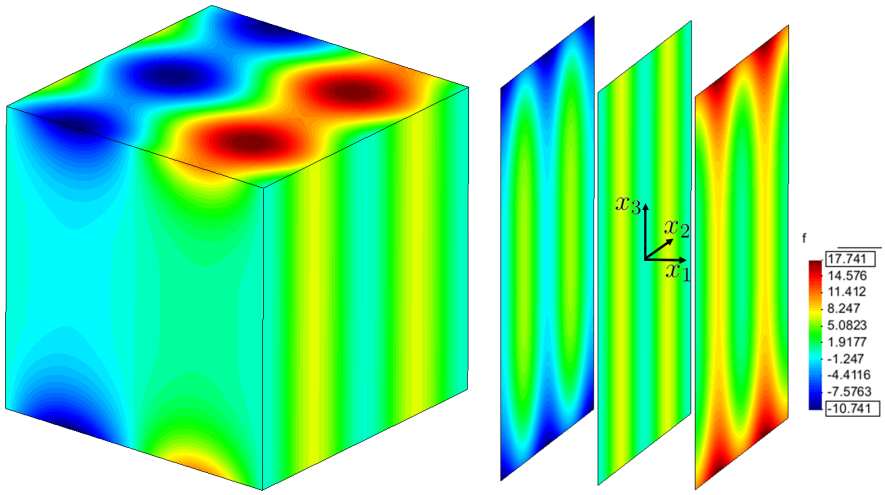
with $x_1, x_2, x_3 \in [-\pi, \pi]$. Visualizations of these two functions are shown in Figure 15 and 16, from which it can be seen that they are highly non-linear and present different types of numerical challenges: the Rosenbrock function is slightly unsymmetrical (see Figure 15 (b)) and its minimum sits at point $(1, 1)$ with $f_{rosen}(1, 1) = 0$, which is numerically sensitive to a surrogate model; the Ishigami function is symmetric with respect to x_1-x_2 plane and x_1-x_3 plane, which has six minimal points (see Figure 16 (b)), $(-\pi/2, \pi, \pi)$, $(-\pi/2, -\pi, \pi)$, $(-\pi/2, 0, \pi)$, $(-\pi/2, \pi, -\pi)$, $(-\pi/2, -\pi, -\pi)$, $(-\pi/2, 0, -\pi)$, with all their function values being $f_{ishigami} = -10.7409091$. It is understandable that a numerical minimal point depends on the initial guess of the numerical algorithm. We shall demonstrate that our surrogate model and surrogate-assisted optimisation algorithm can compute the minimums to a sufficient accuracy.



(a) 3D plot of the Rosenbrock function. (b) Contour plot of the Rosenbrock function.

Fig. 15: Visualization of the Rosenbrock function.

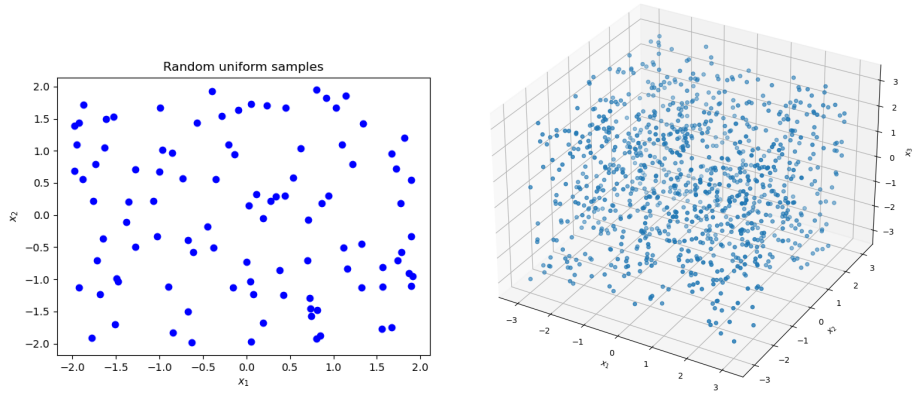
For these two test functions, although a uniform sampling method would be better due to the prior knowledge of the smooth functions (we actually tested the case of using uniform training dataset, which produced as accurate results as the analytical functions. We do not present the results here due to limited space, but please refer the our GitHub repository for the implementations and more results: https://github.com/yongxingwang/), we purposely use random sampling data strategies: 10 random points in one dimension, which gives us 100 random training points for the two-dimensional Rosenbrock function and 1000 random



(a) 3D heat map of the Ishigami function. (b) Contour plot of the Ishigami function.

Fig. 16: Viusalization of the Ishigami function.

training points for the three-dimensional Ishigami function as shown in Figure 17.

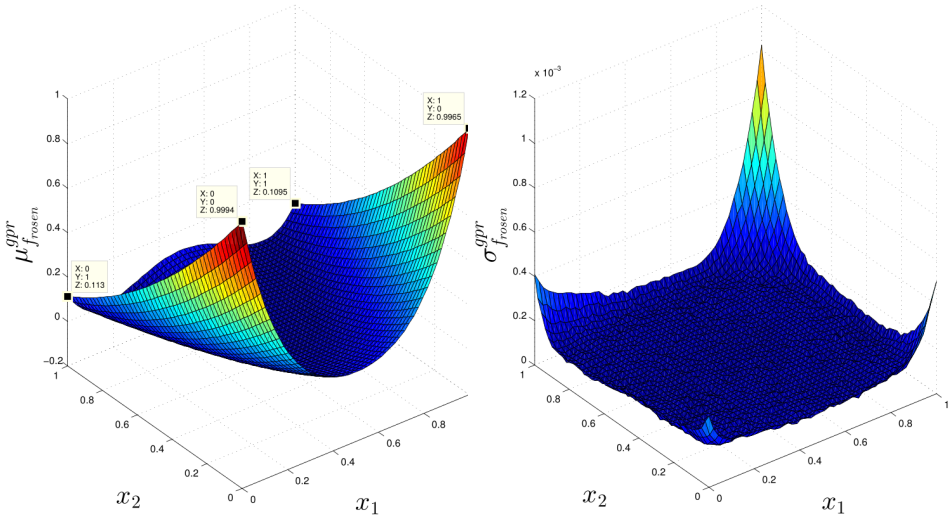


(a) Training dataset for the Rosenbrock function. (b) Training dataset for the Ishigami function.

Fig. 17: Random traning dataset for the Rosenbrock function and the Ishigami function.

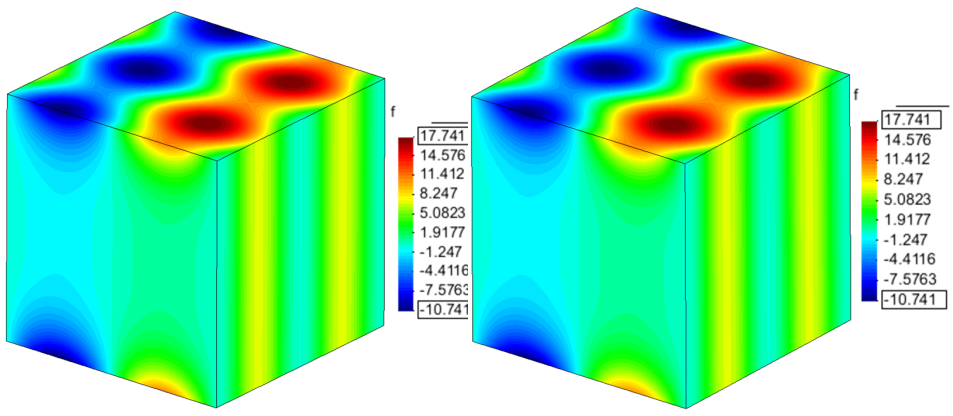
The GPR prediction for the Rosenbrock function is shown in Figure 18 in the normalized space. By comparing with the analytical function, the L^2-error (in the normalized space) is very small: 6.758355×10^{-3}. A comparison between the analytical and GPR prediction of the Ishigami function is shown in Figure 19, from

which we can not see a difference by naked eyes – the L^2-error in the normalized space is 9.297995×10^{-3}.



(a) Mean. (b) Standard deviation

Fig. 18: GPR prediction of the Rosenbrock function.



(a) Heat map of the Ishigami function. (b) Heat map of the mean of GPR prediction of the Ishigami function

Fig. 19: Comparison between the analytical and GPR prediction of the Ishigami function.

We finally compare the optimisation results between the analytical function and the surrogate-assisted one. We implemented the optimisation algorithms for

both the GPR surrogate and the analytical function. Although the gradient is available for this analytical function, we do not use the gradient information in order to make a fair comparison. We use the non-gradient based Nelder-Mead algorithm and set the same convergence criterion in scipy.optimize package. Using the analytical function expression, the Nelder-Mead algorithm can accurately find the minimums for both the two test cases: $(0.999527, 0.999502)$ with $f_{rosen}\,(0.999527, 0.999502) = 2.24154 \times 10^{-11}$ for the Rosenbrock function after 49 iterations starting from initial guess $(0, 0)$; $\left(-1.5707964, -1.34840126 \times 10^{-9}, -3.14159265\right)$ with $f_{ishigami}\left(-1.5707964, -1.34840126 \times 10^{-9}, -3.14159265\right) = -10.7409091 \times 10^{-11}$ after 15 iterations starting form initial guess $(0, 0)$. It is possible to find other minimums for the Ishigami function if starting from different initial guesses, but it may be not necessary to present all the results here. To compare with the surrogate-assisted optimisation method, the corresponding minimum is $(0.949747, 0.985490)$ with $f_{rosen}\,(0.949747, 0.985490) = -0.004411$ for the Rosenbrock function after 66 iterations; and the corresponding minimum $\left(-1.570884, -6.786592 \times 10^{-5}, -3.141593\right)$ with $f_{ishigami}\left(-1.570884, -6.786592 \times 10^{-5}, -3.141593\right) = -10.737585 \times 10^{-11}$ after 45 iterations. All these results demonstrate the accuracy of the GPR surrogate model and the GPR-assisted optimisation algorithm. We also point our that the accuracy can be further improved if one uses a larger training dataset, and please refer to our GitHub repository (https://github.com/yongxingwang/) for more cases.

6.3 Python code for convergence test and validation of the PCE method

In this appendix, we present the Python code for testing the PCE method's convergence in terms of the order of the orthogonal polynomial basis. We also validate the PCE by combination of two normal distributions, and compare its efficiency against the Monte Carlo method.

```
import numpy as np
import chaospy as cp
import pandas as pd
import GPy
####################
dim_in=2
dim_out=2
dim=dim_in+dim_out
####################
pd.set_option('precision',16)
data = pd.read_csv('train.txt',header=None)

x = data.iloc[:, 0:dim_in].values
y = data.iloc[:, dim_in:dim].values

rbf = GPy.kern.RBF(input_dim=dim_in,
variance=1, lengthscale=1)
gp = GPy.models.GPRegression(x, y, rbf)

gp.optimize()
gp = GPy.models.GPRegression(x, y,
rbf,noise_var=1.e-10)

c0 = cp.Normal(0.5, 0.025)
```

```
c1 = cp.Normal(0.6, 0.025)

distribution = cp.J(c0,c1)

def pce(poly_order):
    nodes,weights
    = cp.generate_quadrature(poly_order,
    distribution,rule="Gaussian")

    x=np.transpose(nodes)
    y_pred, var = gp.predict(x)

    f1=y_pred[:,0]
    f2=y_pred[:,1]
    '''
    sd=np.sqrt(var)
    f1=y_pred[:,0]
    f2=y_pred[:,1]
    for i in np.arange(np.size(f1)):
        noise = np.random.normal(0, sd[i,0], 1)
        f1[i] += noise
        f2[i] += noise
    '''
    polynomials = cp.orth_ttr(order=poly_order,
    dist=distribution)

    model_approx = cp.fit_quadrature(polynomials,
    nodes,weights,f1)
   #model_approx = cp.fit_quadrature(polynomials,
   nodes,weights,f2)

    mean = cp.E(model_approx, distribution)
    deviation = cp.Std(model_approx, distribution)

    print("mean, deviation=",mean,deviation)

    file=open("pce_convergence.txt","a")
    file.write("%.16f  %.16f\n" % (mean,deviation))
    file.close()

for i in np.arange(6):
    poly_order=i+1
    pce(poly_order)
```

6.3.1 Linear combination of two normal distributions:

we first test the PCE code using $aX_1 + bX_2 \sim \mathcal{N}(a\mu_1 + b\mu_2, (a\sigma_1)^2 + (b\sigma_2)^2)$, given $X_1 \sim \mathcal{N}(\mu_1, \sigma_1^2)$ and $X_2 \sim \mathcal{N}(\mu_2, \sigma_2^2)$. Since this is a linear relation, the PCE code exactly computes the mean and standard deviation, using a first order basis. For example, $a = b = 1$, $\mu_1 = 0.5$, $\mu_2 = 0.7$, $\sigma_1 = 3$ and $\sigma_2 = 4$, the PCE model gives exact $E(X_1 + X_2) = 1.2$ and $\sigma(X_1 + X_2) = 5$ using a first order (or higher) basis, or $a = 1$, $b = 2$, $\mu_1 = 0.3$, $\mu_2 = 0.5$, $\sigma_1 = 0.3$ and $\sigma_2 = 0.2$, the PCE model gives exact $E(X_1 + 2X_2) = 1.3$ and $\sigma(X_1 + 2X_2) = 0.5$.

If we use the Monte Carlo method to compute the statistics of $X_1 + 2X_2$, a random test gives: $\mu = 1.2993$ and $\sigma = 0.5013$ using 10^5 samples, $\mu = 1.3007$ and $\sigma = 0.4965$ using 10^6 samples, and $\mu = 1.3282$ and $\sigma = 0.5002$ using 10^7 samples.

A convergence of this sampling is shown in Figure 20, from which we can see how poorly the Monte Carlo method converges: one needs a very large number of the sampling points in order to gain an accurate approximation.

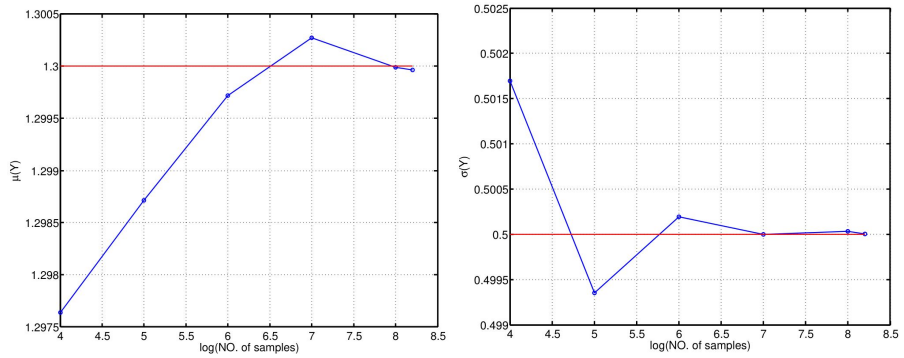


Fig. 20: Convergence of the mean (top) and standard deviation (bottom) as a function of the number of sampling points, using the Monte Carlo method.

6.3.2 An example of non-linear function:

we generate data using the following non-linear function:

$$Y = sin(X_1)/cos(X_2) + X_2X_2, \tag{14}$$

with $X_1 \sim \mathcal{N}(\mu_1, \sigma_1^2)$ and $X_2 \sim \mathcal{N}(\mu_2, \sigma_2^2)$. For $\mu_1 = 0.3$, $\mu_2 = 0.5$, $\sigma_1 = 0.3$ and $\sigma_2 = 0.2$, the convergence of the PCE method together with the convergence of Monte Carlo method are shown in Figure 21, from which it can be seen that the PCE is much cheaper.

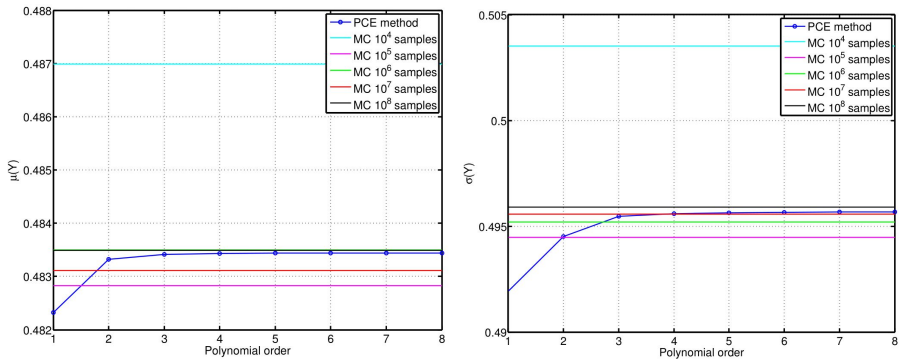


Fig. 21: Convergence of the mean (top) and standard deviation (bottom) as a function of the polynomial order of the PCE method. MC denotes Monte Carlo mehtod.

One should notice that assuming a Gaussian input (or other distribution) is the prerequisite for using the PCE method, while Monte Carlo method needs no assumption of the input variables. This is one essential reason why the PCE method is more efficient than the Monte Carlo method.

6.4 Python code for prediction on uniform grids using PCE method

```
import numpy as np
import chaospy as cp
import pandas as pd
import GPy
####################
dim_in=2
dim_out=2
dim=dim_in+dim_out
####################
pd.set_option('precision',16)
data = pd.read_csv('train.txt',header=None)

x = data.iloc[:, 0:dim_in].values
y = data.iloc[:, dim_in:dim].values

rbf = GPy.kern.RBF(input_dim=dim_in, variance=1, lengthscale=1)
gp = GPy.models.GPRegression(x, y, rbf)

gp.optimize()
gp = GPy.models.GPRegression(x, y, rbf,noise_var=1.e-10)

poly_order=4
def pce(distribution):
    nodes,weights = cp.generate_quadrature(poly_order,
    distribution,rule="Gaussian")

    y_pred, var = gp.predict(np.transpose(nodes))

    f1=y_pred[:,0]
    f2=y_pred[:,1]

    polynomials = cp.orth_ttr(order=poly_order,dist=distribution)

    model_approx = cp.fit_quadrature(polynomials,nodes,weights,f1)
   #model_approx = cp.fit_quadrature(polynomials,nodes,weights,f2)

    mean = cp.E(model_approx, distribution)
    deviation = cp.Std(model_approx, distribution)

    print("mean, deviation=",mean,deviation)

    file=open("pce_predict.txt","a")
    file.write("%.16f  %.16f\n" % (mean,deviation))
    file.close()

xset = np.linspace(0, 1, 100)
yset = np.linspace(0, 1, 100)
for xm in xset:
    for ym in yset:
        c0 = cp.Normal(xm, 0.025)
```

```
        c1 = cp.Normal(ym, 0.025)
        distribution = cp.J(c0,c1)
        pce(distribution)
```

6.5 Python code for optimisation based on the PCE model

In this section, we only provide the Python code for solving Problem 1, which is also a template for Problems 2 and 3.

```
import numpy as np
import pandas as pd
import GPy
from scipy.optimize import minimize
####################
dim_in=2
dim_out=4
dim=dim_in+dim_out
####################
def gpr(x):
    x=x.reshape(1,len(x))
    yp = gp.predict(x)[0]
    return yp[0,0]*omega+yp[0,1]*(1.-omega)
def gpr_mean(x):
    x=x.reshape(1,len(x))
    yp = gp.predict(x)[0]
    return [yp[0,0],yp[0,1]]
def gpr_sigma(x):
    x=x.reshape(1,len(x))
    yp = gp.predict(x)[0]
    return [yp[0,2],yp[0,3]]

pd.set_option('precision',16)
data = pd.read_csv('pce_predict.txt',header=None)

x_train = data.iloc[:, 0:dim_in].values
y_train = data.iloc[:, dim_in:dim].values

rbf = GPy.kern.RBF(input_dim=dim_in, variance=1, lengthscale=1)
gp = GPy.models.GPRegression(x_train, y_train, rbf)

'''Run optimization'''
gp.optimize()

'''Obtain optimized kernel parameters'''
noise=gp.Gaussian_noise.variance
print("optimized noise: \n",noise[0])
l = gp.rbf.lengthscale.values
sigma_f = np.sqrt(gp.rbf.variance.values)
print("optimized kernel parameters: \n",l,sigma_f)

rbf = GPy.kern.RBF(input_dim=dim_in, variance=sigma_f**2,
 lengthscale=l)
'''Fix the noise variance to known value'''
gp.Gaussian_noise.variance = noise
gp.Gaussian_noise.variance.fix()
gp = GPy.models.GPRegression(x_train, y_train, rbf)

x0=[0.5,0.5]
bb=((0,1),(0,1))
sigma1=0.08
```

```
sigma2=0.02

from scipy.optimize import NonlinearConstraint
nonlinear_constraint = NonlinearConstraint(gpr_sigma,
[-np.inf,-np.inf], [sigma1,sigma2])

OmegaSet = np.linspace(0, 1, 100)
f=open("pareto.txt","w+")
for omega in OmegaSet:
    #res=minimize(gpr, x0, bounds=bb})
    res = minimize(gpr, x0, method='trust-constr',
    constraints=[nonlinear_constraint],
    options={'xtol': 1e-12,'verbose': 1}, bounds=bb)
    print('minimum:', omega, res.x, res.fun,'\n')
    mean=gpr_mean(res.x)
    f.write("%.10f %.10f %.10f %.10f\n" %
    (res.x[0],res.x[1],mean[0],mean[1]))

f.close()
```

Declarations

Conflicts of interests On behalf of all authors, the corresponding author states that there is no conflict of interest.

Replication of results All the Python code of implementing the numerical tests in this paper are attached to the appendices.

Data availability The datasets generated during and/or analysed during the current study are available in the public GitHub repository: https://github.com/yongxingwang.

References

1. Askey, R., Wilson, J.A.: Some basic hypergeometric orthogonal polynomials that generalize Jacobi polynomials, vol. 319. American Mathematical Soc. (1985)
2. Babaei, M., Alkhatib, A., Pan, I.: Robust optimization of subsurface flow using polynomial chaos and response surface surrogates. Computational Geosciences **19**(5), 979–998 (2015)
3. Ben-Tal, A., El Ghaoui, L., Nemirovski, A.: Robust optimization, vol. 28. Princeton University Press (2009)
4. Ben-Tal, A., Nemirovski, A.: Robust optimization–methodology and applications. Mathematical programming **92**(3), 453–480 (2002)
5. Bertsimas, D., Brown, D.B., Caramanis, C.: Theory and applications of robust optimization. SIAM review **53**(3), 464–501 (2011)
6. Bertsimas, D., Gupta, V., Kallus, N.: Data-driven robust optimization. Mathematical Programming **167**(2), 235–292 (2018)
7. Beyer, H.G., Sendhoff, B.: Robust optimization–a comprehensive survey. Computer Methods in Applied Mechanics and Engineering **196**(33-34), 3190–3218 (2007)
8. Beyreuther, M., Barsch, R., Krischer, L., Megies, T., Behr, Y., Wassermann, J.: Obspy: A python toolbox for seismology. Seismological Research Letters **81**(3), 530–533 (2010)
9. Bodla, K.K., Murthy, J.Y., Garimella, S.V.: Optimization under uncertainty applied to heat sink design. Journal of Heat Transfer **135**(1) (2013)
10. Chen, P.C., Nikitopoulos, D.E., Soper, S.A., Murphy, M.C.: Temperature distribution effects on micro-cfpcr performance. Biomedical Microdevices **10**(2), 141–152 (2008)
11. Chiou, J., Matsudaira, P., Sonin, A., Ehrlich, D.: A closed-cycle capillary polymerase chain reaction machine. Analytical Chemistry **73**(9), 2018–2021 (2001)

12. Dodson, M., Parks, G.T.: Robust aerodynamic design optimization using polynomial chaos. Journal of Aircraft **46**(2), 635–646 (2009)
13. Duryodhan, V., Singh, A., Singh, S.G., Agrawal, A.: A simple and novel way of maintaining constant wall temperature in microdevices. Scientific Reports **6**, 18230 (2016)
14. Ernst, O.G., Mugler, A., Starkloff, H.J., Ullmann, E.: On the convergence of generalized polynomial chaos expansions. ESAIM: Mathematical Modelling and Numerical Analysis **46**(2), 317–339 (2012)
15. Feinberg, J., Langtangen, H.P.: Chaospy: An open source tool for designing methods of uncertainty quantification. Journal of Computational Science **11**, 46–57 (2015)
16. Field Jr, R.V., Grigoriu, M.: Convergence properties of polynomial chaos approximations for L_2 random variables. Tech. rep., Sandia National Laboratories (2007)
17. Frey, O., Bonneick, S., Hierlemann, A., Lichtenberg, J.: Autonomous microfluidic multichannel chip for real-time PCR with integrated liquid handling. Biomedical Microdevices **9**(5), 711–718 (2007)
18. Gao, H., Jézéquel, L., Cabrol, E., Vitry, B.: Multi-objective robust optimization of chassis system with polynomial chaos expansion method. Engineering Optimization **53**(9), 1483–1503 (2021)
19. Ghanem, R.G., Spanos, P.D.: Stochastic finite element method: Response statistics. In: Stochastic finite elements: a spectral approach, pp. 101–119. Springer (1991)
20. Ghisu, T., Lopez, D.I., Seshadri, P., Shahpar, S.: Gradient-enhanced least-square polynomial chaos expansions for uncertainty quantification and robust optimization. In: AIAA AVIATION 2021 FORUM, p. 3073 (2021)
21. Gui, L., Ren, C.L.: Numeric simulation of heat transfer and electrokinetic flow in an electroosmosis-based continuous flow PCR chip. Analytical Chemistry **78**(17), 6215–6222 (2006)
22. Hamad, H.S., Kapur, N., Khatir, Z., Querin, O., Thompson, H.M., Wang, Y., Wilson, M.: Computational fluid dynamics analysis and optimisation of polymerase chain reaction thermal flow systems. Applied Thermal Engineering **183**, 116122 (2021)
23. Hashimoto, M., Chen, P.C., Mitchell, M.W., Nikitopoulos, D.E., Soper, S.A., Murphy, M.C.: Rapid PCR in a continuous flow device. Lab on a Chip **4**(6), 638–645 (2004)
24. Havinga, J., van den Boogaard, A.H., Klaseboer, G.: Sequential improvement for robust optimization using an uncertainty measure for radial basis functions. Structural and Multidisciplinary Optimization **55**(4), 1345–1363 (2017)
25. Hensman, J., Fusi, N., Andrade, R., Durrande, N., Saul, A., Zwiessele, M., Lawrence, N.: Gpy: A Gaussian process framework in python (2012)
26. Ho, S.L., Yang, S.: A fast robust optimization methodology based on polynomial chaos and evolutionary algorithm for inverse problems. IEEE Transactions on Magnetics **48**(2), 259–262 (2012)
27. Hopfe, C.J., Emmerich, M.T., Marijt, R., Hensen, J.: Robust multi-criteria design optimisation in building design. Proceedings of Building Simulation and Optimization, Loughborough, UK pp. 118–125 (2012)
28. Ishigami, T., Homma, T.: An importance quantification technique in uncertainty analysis for computer models. In: [1990] Proceedings. First International Symposium on Uncertainty Modeling and Analysis, pp. 398–403. IEEE (1990)
29. Kaintura, A., Dhaene, T., Spina, D.: Review of polynomial chaos-based methods for uncertainty quantification in modern integrated circuits. Electronics **7**(3), 30 (2018)
30. Kala, Z.: Benchmark of goal oriented sensitivity analysis methods using ishigami function. International Journal of Mathematical and Computational Methods **3** (2018)
31. Kersaudy, P., Sudret, B., Varsier, N., Picon, O., Wiart, J.: A new surrogate modeling technique combining kriging and polynomial chaos expansions–application to uncertainty analysis in computational dosimetry. Journal of Computational Physics **286**, 103–117 (2015)
32. Korondi, P.Z., Marchi, M., Parussini, L., Poloni, C.: Multi-fidelity design optimisation strategy under uncertainty with limited computational budget. Optimization and Engineering **22**(2), 1039–1064 (2021)
33. Kuss, M.: Gaussian process models for robust regression, classification, and reinforcement learning. Ph.D. thesis, Technische Universität Darmstadt Darmstadt, Germany (2006)
34. Li, W., Gao, L., Garg, A., Xiao, M.: Multidisciplinary robust design optimization considering parameter and metamodeling uncertainties. Engineering with Computers pp. 1–18 (2020)

35. McGibbon, R.T., Hernández, C.X., Harrigan, M.P., Kearnes, S., Sultan, M.M., Jastrzebski, S., Husic, B.E., Pande, V.S.: Osprey: Hyperparameter optimization for machine learning. Journal of Open Source Software **1**, 34 (2016)
36. Miralles, V., Huerre, A., Malloggi, F., Jullien, M.C.: A review of heating and temperature control in microfluidic systems: techniques and applications. Diagnostics **3**(1), 33–67 (2013)
37. Mitchell, M., Liu, X., Bejat, Y., Nikitopoulos, D., Soper, S., Murphy, M.: Microfluidics, biomems, and medical microsystems. Proc. SPIE–Int. Soc. Opt. Eng., San Jose, CA, USA (2003)
38. Murphy, K.P.: Machine learning: a probabilistic perspective. MIT press (2012)
39. Onorato, G., Loeven, G., Ghorbaniasl, G., Bijl, H., Lacor, C.: Comparison of intrusive and non-intrusive polynomial chaos methods for cfd applications in aeronautics. In: V European Conference on Computational Fluid Dynamics ECCOMAS, Lisbon, Portugal, pp. 14–17 (2010)
40. Park, G.J., Lee, T.H., Lee, K.H., Hwang, K.H.: Robust design: an overview. AIAA journal **44**(1), 181–191 (2006)
41. Park, J., Park, H.: Thermal cycling characteristics of a 3D-printed serpentine microchannel for DNA amplification by polymerase chain reaction. Sensors and Actuators A: Physical **268**, 183–187 (2017)
42. Phadke, M.S.: Quality engineering using robust design. Prentice Hall PTR (1995)
43. Ren, C., Xiong, F., Mo, B., Chawdhury, A., Wang, F.: Design sensitivity analysis with polynomial chaos for robust optimization. Structural and Multidisciplinary Optimization **63**(1), 357–373 (2021)
44. Rosenbrock, H.: An automatic method for finding the greatest or least value of a function. The Computer Journal **3**(3), 175–184 (1960)
45. Sacks, J., Welch, W.J., Mitchell, T.J., Wynn, H.P.: Design and analysis of computer experiments. Statistical Science pp. 409–423 (1989)
46. Schaerli, Y., Wootton, R.C., Robinson, T., Stein, V., Dunsby, C., Neil, M.A., French, P.M., DeMello, A.J., Abell, C., Hollfelder, F.: Continuous-flow polymerase chain reaction of single-copy DNA in microfluidic microdroplets. Analytical Chemistry **81**(1), 302–306 (2009)
47. Schevenels, M., Lazarov, B.S., Sigmund, O.: Robust topology optimization accounting for spatially varying manufacturing errors. Computer Methods in Applied Mechanics and Engineering **200**(49-52), 3613–3627 (2011)
48. Schobi, R., Sudret, B., Wiart, J.: Polynomial-chaos-based kriging. International Journal for Uncertainty Quantification **5**(2) (2015)
49. Shahbaz, M., Han, Z.H., Song, W., Aizud, M.N.: Surrogate-based robust design optimization of airfoil using inexpensive monte carlo method. In: 2016 13th International Bhurban Conference on Applied Sciences and Technology (IBCAST), pp. 497–504. IEEE (2016)
50. Shang, C., Huang, X., You, F.: Data-driven robust optimization based on kernel learning. Computers & Chemical Engineering **106**, 464–479 (2017)
51. Simpson, T., Toropov, V., Balabanov, V., Viana, F.: Design and analysis of computer experiments in multidisciplinary design optimization: a review of how far we have come-or not. In: 12th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, p. 5802 (2008)
52. Sudret, B.: Polynomial chaos expansions and stochastic finite element methods. Risk and Reliability in Geotechnical Engineering pp. 265–300 (2015)
53. Taguchi, G.: Introduction to quality engineering: designing quality into products and processes (1986)
54. Tootkaboni, M., Asadpoure, A., Guest, J.K.: Topology optimization of continuum structures under uncertainty–a polynomial chaos approach. Computer Methods in Applied Mechanics and Engineering **201**, 263–275 (2012)
55. Tsui, K.L.: An overview of Taguchi method and newly developed statistical methods for robust design. Iie Transactions **24**(5), 44–57 (1992)
56. Wang, F., Xiong, F., Chen, S., Song, J.: Multi-fidelity uncertainty propagation using polynomial chaos and gaussian process modeling. Structural and Multidisciplinary Optimization **60**(4), 1583–1604 (2019)
57. Wang, R., Work, D.: Application of robust optimization in matrix-based lci for decision making under uncertainty. The International Journal of Life Cycle Assessment **19**(5), 1110–1118 (2014)

58. Wiener, N.: The homogeneous chaos. American Journal of Mathematics **60**(4), 897–936 (1938)
59. Williams, C.K., Rasmussen, C.E.: Gaussian processes for machine learning, vol. 2. MIT press Cambridge, MA (2006)
60. Wu, X., Zhang, W., Song, S.: Robust aerodynamic shape design based on an adaptive stochastic optimization framework. Structural and Multidisciplinary Optimization **57**(2), 639–651 (2018)
61. Xiu, D., Karniadakis, G.E.: The wiener–askey polynomial chaos for stochastic differential equations. SIAM journal on scientific computing **24**(2), 619–644 (2002)
62. Yang, S., Xiong, F., Wang, F.: Polynomial chaos expansion for probabilistic uncertainty propagation. Uncertainty Quantification and Model Calibration (2017)
63. Zhou, Q., Wang, Y., Choi, S.K., Jiang, P., Shao, X., Hu, J., Shu, L.: A robust optimization approach based on multi-fidelity metamodel. Structural and Multidisciplinary Optimization **57**(2), 775–797 (2018)