CHALMERS
UNIVERSITY OF TECHNOLOGY

Machine learning surrogates for the optimization of curing ovens

Citation for the original published paper (version of record):
Parsons, Q., Nowak, D., Bortz, M. et al (2024). Machine learning surrogates for the optimization of curing ovens. Engineering Applications of Artificial Intelligence, 133(Part C).
http://dx.doi.org/10.1016/j.engappai.2024.108086

Engineering Applications of Artificial Intelligence 133 (2024) 108086

Contents lists available at ScienceDirect

Engineering Applications of Artificial Intelligence

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

Research paper

Machine learning surrogates for the optimization of curing ovens

Quentin Parsons [a], Dimitri Nowak [a,*], Michael Bortz [a], Tomas Johnson [b], Andreas Mark [b], Fredrik Edelvik [b]

[a] Optimization Department, Fraunhofer Institute for Industrial Mathematics ITWM, Fraunhofer-Platz 1, Kaiserslautern, 67663, Germany
[b] Computational Engineering and Design, Fraunhofer-Chalmers Centre for Industrial Mathematics, Chalmers Science Park, Gothenburg, SE-412 88, Sweden

ARTICLE INFO

Keywords:
Surrogate modelling
Response surface modelling
Metamodelling
Design optimization
Non-convex optimization
Multi-criteria optimization
Machine learning
Supervised learning
Neural networks
Deep learning
Non-linear regression
Linear regression
Gaussian processes
CFD simulations

ABSTRACT

We investigate how to set the inlet temperature, and arrange a set of vehicle parts inside a paint curing oven, so as to maximize a non-convex, non-linear objective function. Standard methods for solving this kind of problem require a large number of objective function evaluations, each of which depends on a computationally expensive (minutes/hours) CFD simulation.

We replace the CFD solver with machine learning surrogates that can approximate the data required for an objective function evaluation extremely quickly (sub-second). We develop i) simulation surrogates that produce simulations that are structurally identical to their CFD-generated counterparts, and ii) objective function surrogates that learn an objective function directly.

We consider elementary learners (simple neural networks, non-linear regressions, Gaussian processes) and develop various techniques to use and combine them to solve single- and multi-criteria optimization problems.

We combine our surrogates in a configuration resembling a stack ensemble, and use it to solve the optimization problem at greatly reduced computational cost. We are thus able to explore multiple local maxima, and obtain solutions with higher objective function values than with traditional methods. Finally, we propose an approach that allows practitioners to throttle the computational effort until a satisfactory solution quality is achieved.

1. Introduction

We study the problem of how best to set the inlet temperature and arrange a set of vehicle parts inside a paint curing oven. Problems of this sort are typically solved as follows (see Nowak et al., 2021): (i) a suitable objective function is defined that encodes something to be optimized (minimum energy, maximum throughput, etc.), and (ii) an appropriate optimization algorithm is selected and applied to find configurations that optimize the objective function. Such algorithms (Johnson, 2007) evaluate the objective function repeatedly as they iterate towards a solution.

The objective functions for this particular optimization problem have two remarkable properties: (i) they lead to continuous, non-convex optimization problems, and (ii) they depend on the conditions inside the oven throughout the curing process. As such, each objective function evaluation performed during the optimization requires a complete and computationally expensive **computational fluid dynamics** (CFD) simulation (and it is easy for a single optimization to require tens to hundreds of such evaluations). Solving such problems can therefore be prohibitively time consuming. Moreover, it is generally impossible to determine whether the resulting solutions are global optima, and it may be impractical to find and compare multiple (local) solutions.

Our plan is to replace expensive, but accurate, CFD simulations in the optimization process, with cheaper, but approximate, **machine learning** (ML) surrogate approximations. We have two main goals: (i) to solve these problems at lower computational cost, and (ii) to use these capabilities to find multiple local (i.e. better) solutions.

1.1. Literature

Surrogate modelling, also known as response surface modelling or metamodelling, is a technique used to approximate the behaviour of a complex system or process. It is used when direct evaluation of the system is computationally intensive or infeasible, or when the system is stochastic or otherwise difficult to model accurately. The technique is used in a wide range of applications in various fields, including engineering, computer science and data science. In engineering (Forrester et al., 2008), surrogate models can be used to optimize the design of products or processes by predicting the performance of different design

* Corresponding author.
E-mail address: dimitri.nowak@itwm.fraunhofer.de (D. Nowak).

https://doi.org/10.1016/j.engappai.2024.108086
Received 16 January 2023; Received in revised form 18 January 2024; Accepted 12 February 2024
Available online 9 April 2024

configurations. In computer science, surrogate models can be used to speed up the optimization of computer programs using evolutionary algorithms. Tong et al. (2021) presents a taxonomy of surrogate models, and evaluates their performance on various optimization tasks. In data science (Dawson-Elli et al., 2018), surrogate models can be used to model the relationship between input and output variables in a data set. Surrogate modelling techniques include Kriging, artificial **neural networks** (NNs), and polynomial regression (Koziel and Leifsson, 2013). Of particular relevance to this work are efforts to use surrogates in physical simulations (Heese et al., 2019; Ludl et al., 2022; Forte et al., 2019). Similar attempts to solve **multi-criteria optimization** (MCO) problems encountered in engineering can be found in Bortz et al. (2014), Asprion and Bortz (2018). Alternative heuristic approaches are proposed in Huang et al. (2023), Sharma et al. (2022), Mir et al. (2020).

1.2. Overview and approach

This work is an extension of Nowak et al. (2021). The goal of that paper was to find accurate, physically realistic solutions to an MCO problem, using fine-grid simulations generated by the IPS IBOFlow (Immersed Boundary Octree Flow Solver) CFD solver (Fraunhofer Chalmers Research Centre for Industrial Mathematics, 0000; Mark et al., 2013; Andersson et al., 2018). We replace IBOFlow in the optimization process with various ML surrogates. We focus on problem solving techniques, and are less interested in specific solutions. As such, we work with coarse-grid simulations that are significantly computationally cheaper,[1] but less physically accurate than those in Nowak et al. (2021). We can thus create much larger training sets, and conduct much larger optimization experiments involving many more simulations than would otherwise be practical.

We develop two types of surrogate: (i) Simulation surrogates are trained on exact/IBOFlow simulations, and produce 'synthetic' simulations that are structurally indistinguishable from IBOFlow simulations. Objective functions can be evaluated on these synthetic simulations in the same way as on their IBOFlow counterparts. We consider fully connected feed-forward NNs and nonlinear regression simulation surrogates. (ii) Objective function surrogates learn an objective function directly. We use Gaussian processes.

We use bespoke, but *simple* ML surrogates (Bishop, 2006) trained on small to modest training sets to reduce the computational cost of solving optimization problems. Rather than learning the entire CFD solution (fluid velocity, temperature, pressure, etc.) as in Thuerey et al. (2020), Beck et al. (2018), Farimani et al. (2017), Wiewel et al. (2019), our simulation surrogates learn only those aspects of the simulation that are inputs to the objective functions of interest, namely the minimum, average, and maximum temperatures of each part in the oven during the curing process.

1.3. Summary and novelty of our contribution

Our contribution comprises three main parts: (i) we develop a comprehensive framework for evaluating and comparing our surrogates; (ii) we propose methods for using and combining them to solve challenging optimization problems; (iii) we conduct numerical experiments and propose a practical approach to solving MCO problems that gives the practitioner control over the computational effort.

The novelty of this case study is twofold. First, we believe that our approach to solving this particular class of oven curing problem is new. Second, we propose an approach in which all of our learners are combined to solve an MCO problem in a configuration resembling a stack ensemble. We are not aware of this approach ever having been applied to an MCO problem before.

[1] At least 10× cheaper than those in Nowak et al. (2021).

The remainder of this work is organized as follows: in Section 2, we introduce the optimization problems and our notation. In Section 3, we discuss objective functions and our approach to solving the corresponding optimization problems. We describe our ML surrogates in Section 4. We review our data sets (train/validate/test) in Section 4.1. We discuss NN surrogates in Section 4.2, and a nonlinear regression surrogate in Section 4.3. In Section 4.4 we evaluate the accuracy of the simulations produced by our simulation surrogates. In Section 4.5, we develop an objective function surrogate. In Section 4.6 we evaluate how well our surrogates capture the objective functions of interest. In Section 5, we use our surrogates to solve a series of optimization problems of increasing complexity. In Section 5.1, we review our approach to selecting **initial guesses** (IGs), and propose several solution approaches in Section 5.2. We solve a single objective problem in Section 5.3, and an MCO problem in Section 5.4. We conclude in Section 6. *This paper has an extensive appendix; the main text includes summaries and highlights of our experimental results.*

2. Context, notation

An **oven configuration** (OC) is a set of part centre coordinates, $\boldsymbol{X} = (x, y, z)$, and the oven inlet temperature, Θ^{inlet}.[2] We constrain each part to the mid-plane of the oven, i.e. we fix (and suppress) x in $\boldsymbol{X}$. We identify each OC of n parts with a vector $\boldsymbol{x} := \left[(y_1, z_1), \ldots, (y_n, z_n), \Theta^{\text{inlet}}\right] \in \mathbb{R}^{2n+1}$, comprising n coordinate pairs (y, z) for the positions of the part centres, and an oven inlet temperature. In a *valid* OC the parts are inside the oven, and well separated (they do not overlap or touch), and $\Theta^{\text{inlet}} \in \left[\Theta^{\text{inlet}}_{\text{min}}, \Theta^{\text{inlet}}_{\text{max}}\right]$. $\boldsymbol{V} \subset \mathbb{R}^{2n+1}$ is the set of valid OCs. The simulation time horizon T is the amount of time that the parts are inside the oven.

IBOFlow admits a pair of non-negative integer refinement levels, $\boldsymbol{r} = (r_g, r_o)$: r_g is the grid refinement level, for discretizing the background fluid, and r_o is the object refinement level, for the vehicle parts inside the fluid. Higher refinement levels correspond to finer spatial meshes, and smaller time steps.

A **simulation**, $S(\boldsymbol{x}, \boldsymbol{r}, T)$, is a function that maps a valid OC, $\boldsymbol{x}$, a refinement level specification, $\boldsymbol{r}$, and a time horizon, T, to a set of temperature functions, Θ_i, that describe the minimum, mean and maximum temperatures of each part in the oven. Each temperature function is represented as a set of temperatures over a set of time steps[3]

$$t_1 < \cdots < T \le t_m =: \left\{t_j\right\}_{j=1}^{m}.$$

We write

$$S(\boldsymbol{x}, \boldsymbol{r}, T) = \left\{\boldsymbol{x}, \left(\Theta_i^{\min}\left(\boldsymbol{x}, t_j\right)_{j=1}^{m}, \Theta_i^{\text{mean}}\left(\boldsymbol{x}, t_j\right)_{j=1}^{m}, \Theta_i^{\max}\left(\boldsymbol{x}, t_j\right)_{j=1}^{m}\right)_{i=1}^{n}\right\}. \tag{1}$$

Vehicle parts do not move during curing ($\boldsymbol{x}$ is fixed). Likewise, Θ^{inlet} is set and fixed for each simulation.

We refer to simulations generated by IBOFlow as *IBOFlow simulations*, and denote them by $S_{\text{IBF}}(\boldsymbol{x}, \boldsymbol{r}, T)$. Simulations generated by ML surrogates are called *synthetic simulations*, and denoted by $S_{\text{surr.}}(\boldsymbol{x}, \boldsymbol{r}, T)$. We study two types of surrogate:

Simulation surrogates : emulate IBOFlow. For a given $\boldsymbol{x}$ and a set of time steps $\left\{t_i\right\}_{i=1}^{m}$, these objects output a complete (synthetic) simulation.

Simulation surrogates are trained on pairs of the form $\left\{\left(\boldsymbol{x}_i, S_{\text{IBF}}\left(\boldsymbol{x}_i\right)\right)\right\}$ (each target is an IBOFlow simulation), for a varying set of OCs, and a common $\boldsymbol{r}$ and T. Even though $\boldsymbol{r}$ and T are not explicit features, each surrogate encapsulates an $(\boldsymbol{r}, T)$

[2] We fix and ignore the rotation vector, $\boldsymbol{R}$.

[3] IBOFlow excludes t_0 from its output, i.e. the simulations start at $t_1 > 0$. The temporal discretization $t_{j+1} - t_j$ (and consequently $t_1 > 0$) is calculated by IBOFlow, and varies within and across simulations.

pair. A synthetic simulation is an approximation of an IBOFlow simulation.

We cannot *a priori* fix the time steps $\{t_i\}_{i=1}^{m}$ for which IBOFlow calculates temperatures; they are usually *not equally spaced*. However, we can specify the time steps in a synthetic simulation.[4]

Objective function surrogates : learn an objective function. For a given OC $\boldsymbol{x}$, these objects estimate the value of an objective function, *without* generating a simulation. Objective function surrogates are trained on pairs $\{(\boldsymbol{x}_i, f(\boldsymbol{x}_i, S_{\text{IBF}}(\boldsymbol{x}_i)))\}$, for an objective function f that takes a simulation as input (which in turn depends on an OC, $\boldsymbol{x}$). As with simulation surrogates, $\boldsymbol{r}$ and T are implicit.

We set $\boldsymbol{r} = (0, 0)$ (no grid or object refinements). As in Nowak et al. (2021), we set $T := 2000$ (s) so that the temperatures of the parts are stationary at the end of the simulation. We can thus suppress $\boldsymbol{r}$ and T in our notation, regard each simulation as a function of $\boldsymbol{x}$ only, and maximize scalar objective functions of the form $f(\boldsymbol{x}) = f(\boldsymbol{x}, S(\boldsymbol{x}))$. Clearly, the value of $f(\boldsymbol{x})$ depends on whether $S(\boldsymbol{x})$ is an IBOFlow simulation or a synthetic, and whether f is an exact objective function, or a surrogate. Consequently, we define

$$f_{\text{exact}}(\boldsymbol{x}) := f\left(\boldsymbol{x}, S_{\text{IBF}}(\boldsymbol{x})\right) \quad \text{and} \quad f_{\text{synth.}}(\boldsymbol{x}) := f\left(\boldsymbol{x}, S_{\text{surr.}}(\boldsymbol{x})\right) \tag{2}$$

for an objective function, f. We use $f_{\text{surr.}}(\boldsymbol{x})$ for the value of the surrogate objective function $f_{\text{surr.}}$ (of f) at $\boldsymbol{x}$.[5] We consider problems of the form

$$(\mathbf{P}_{\text{exact}}) \quad \text{Find} \quad \boldsymbol{x} \in \boldsymbol{V} \quad \text{s.t.} \quad f_{\text{exact}}(\boldsymbol{x}) \to \max, \tag{3a}$$

$$(\mathbf{P}_{\text{synth.}}) \quad \text{Find} \quad \boldsymbol{x} \in \boldsymbol{V} \quad \text{s.t.} \quad f_{\text{synth.}}(\boldsymbol{x}) \to \max, \tag{3b}$$

$$(\mathbf{P}_{\text{surr.}}) \quad \text{Find} \quad \boldsymbol{x} \in \boldsymbol{V} \quad \text{s.t.} \quad f_{\text{surr.}}(\boldsymbol{x}) \to \max, \tag{3c}$$

subject to additional constraints. $\boldsymbol{V}$ is the decision space of the problem. A major difficulty in solving such problems is that function evaluations $f(\boldsymbol{x})$ may require a full simulation $S(\boldsymbol{x})$, and high-precision IBOFlow simulations are extraordinarily computationally expensive (wall time minutes to hours). Conversely, we expect trained ML surrogates to be able to generate simulations or estimate objective function values essentially for free (wall time $\ll 1$ second). Consequently, we have good reason to replace IBOFlow with a surrogate that can solve (3b) or (3c) inexpensively, and thus find an OC $\boldsymbol{x}$ that is a (potentially approximate) solution of (3a).

2.1. Oven properties

Fig. 1 depicts three vehicle parts inside the oven. They are the **cab corner** (CC), the **large pc pet** (LPP) and the **small pc pet** (SPP). The parts have a small extent in the x-direction (out of the page), and a much larger y- and z-extent. The oven has dimensions $\Delta x \times \Delta y \times \Delta z = 1 \times 2 \times 1.6$. The centres of the parts are in the x mid-plane.[6]

The oven has operational parameters

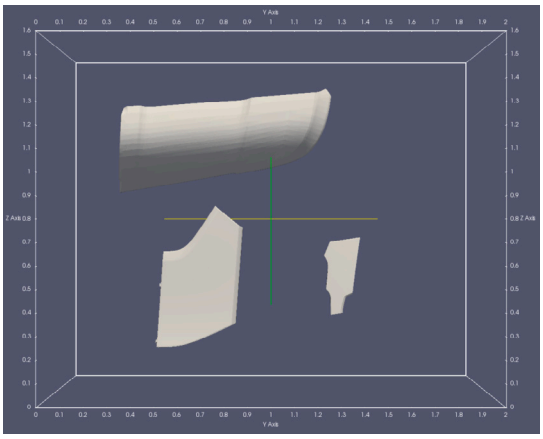
$$\Theta^{\text{inlet}}_{\text{min}} := 355\,\text{K} \le \Theta^{\text{inlet}} \le 380\,\text{K} =: \Theta^{\text{inlet}}_{\text{max}}. \tag{4}$$

The initial temperature (of all parts) is 293.15 K (room temperature of 20 °C).

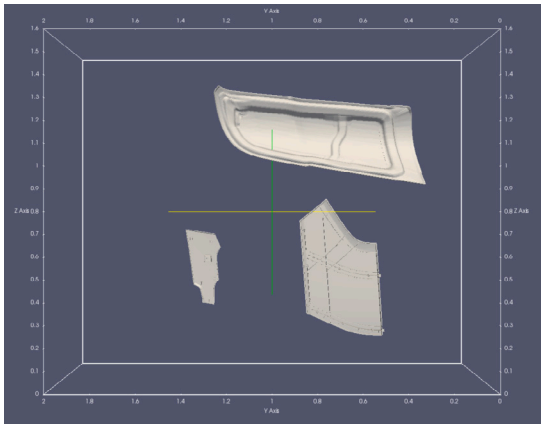
[4] Surrogates are trained on (discrete) sets of t_i values (from IBOFlow), but can predict temperatures for any $t \in (0, T]$.

[5] We never consider evaluations of the form $f_{\text{surr.}}(S(\boldsymbol{x}))$, i.e. surrogate objective functions of simulations. Surrogate objective functions are defined exclusively on OCs.

[6] Fig. 1 assumes an origin at the bottom left, but IBOFlow places the origin in the middle of the oven. Depending on this choice, the parts are restricted to $x = 0.5$ or $x = 0$.



(a) View from above



(b) View from below

Fig. 1. The three parts inside the oven: CC (largest part), LPP (medium-sized part) and SPP (smallest part).

2.2. Scaling

It is clear that the inputs of the problem span several orders of magnitude. We therefore linearly re-scale the part coordinates (y, z), Θ^{inlet}, and T, to the unit hypercube. As a result, all of the inputs to our objective functions are in $[0, 1]$. The outputs (the temperatures of the parts) are not scaled. *We use the same notation for scaled inputs (e.g. Θ^{inlet}) as their un-scaled counterparts.*

3. Optimization problems

Constraints

Given the complexity of the shapes of the parts of interest (cf. Fig. 1), it is hopeless to define $\boldsymbol{V}$ explicitly as a set of functions of $\boldsymbol{x}$. We proceed implicitly by penalizing invalid OCs: in our maximization problems, we set $f(\boldsymbol{x}) := f_{\text{min}}$ if $\boldsymbol{x} \notin \boldsymbol{V}$. We will also encounter valid OCs ($\boldsymbol{x} \in \boldsymbol{V}$) with invalid objective function values, e.g. a zero curing time (see below). For these, we set $f(\boldsymbol{x}) := f_{\text{min}}$, and regard these OCs as *invalid for the objective*.

3.1. Objective functions

Inlet temperature

We use the inlet temperature as a proxy for the energy needed for curing, which we seek to minimize. We use the symbol Θ^{inlet} for both

the inlet temperature, and the associated objective function, and define

$$\Theta^{\text{inlet}}(\boldsymbol{x}) := \begin{cases} \Theta^{\text{inlet}}(\boldsymbol{x}), & \boldsymbol{x} \in \boldsymbol{V} \\ 1, & \text{otherwise.} \end{cases} \tag{5}$$

The optimization problem can be written as

$$(\mathbf{P}_{\Theta^{\text{inlet}}}) \quad \text{Find} \quad \boldsymbol{x} \in \boldsymbol{V} \quad \text{s.t.} \quad \Theta^{\text{inlet}}(\boldsymbol{x}) \to \min. \tag{6}$$

(Equivalently, we could seek $\boldsymbol{x}$ such that $-\Theta^{\text{inlet}}(\boldsymbol{x}) \to \max$.) (5) is trivially a function of $\boldsymbol{x}$, but does not depend on a simulation. Consequently, any $\boldsymbol{x} \in \boldsymbol{V}$ with $\Theta^{\text{inlet}} = \Theta^{\text{inlet}}_{\min}$ (which scales to zero) is a non-unique solution of (6). $\Theta^{\text{inlet}}(\boldsymbol{x}) \in [0,1]$ for any $\boldsymbol{x}$, and $\Theta^{\text{inlet}}(\boldsymbol{x}) = 1$ if $\boldsymbol{x} \notin \boldsymbol{V}$, by virtue of Section 2.2.

Curing time

The **curing time** (CT) of a part is the amount of time its temperature is between the minimum and maximum curing temperatures, $\Theta^{\text{curing}}_{\min} := 350\,\text{K}$ and $\Theta^{\text{curing}}_{\max} := 375\,\text{K}$ (part of the paint specification). We define the highest minimum temperature of the ith part as

$$\Theta^{\max,\min}_i(\boldsymbol{x}) := \max_{t_j} \left\{ \Theta^{\min}_i(\boldsymbol{x}, t_j) \right\}, \tag{7a}$$

and its highest maximum temperature as

$$\Theta^{\max,\max}_i(\boldsymbol{x}) := \max_{t_j} \left\{ \Theta^{\max}_i(\boldsymbol{x}, t_j) \right\}. \tag{7b}$$

The raw CT of part i (in OC $\boldsymbol{x}$) is

$$C^{\text{raw}}_i(\boldsymbol{x}) := \begin{cases} \int_0^T \mathbb{1}\left\{ c_i(\boldsymbol{x}, t) \right\} \mathrm{d}t & \boldsymbol{x} \in \boldsymbol{V}, \text{ and } \Theta^{\max,\max}_i(\boldsymbol{x}) \le \Theta^{\text{curing}}_{\max} \\ 0 & \text{otherwise,} \end{cases} \tag{8a}$$

where $\mathbb{1}\{\cdot\}$ is the indicator function, and

$$c_i(\boldsymbol{x}, t) := \left\{ \Theta^{\min}_i(\boldsymbol{x}, t) \ge \Theta^{\text{curing}}_{\min} \right\}. \tag{8b}$$

A consequence of (8) is that part i has zero CT if

$$\Theta^{\max,\min}_i(\boldsymbol{x}) < \Theta^{\text{curing}}_{\min},$$

that is, $\Theta^{\min}_i$ must reach $\Theta^{\text{curing}}_{\min}$. Part i is *burned* and $C^{\text{raw}}_i(\boldsymbol{x}) := 0$ if

$$\Theta^{\max,\max}_i(\boldsymbol{x}) > \Theta^{\text{curing}}_{\max},$$

that is, $\Theta^{\max}_i$ must not exceed $\Theta^{\text{curing}}_{\max}$. A part is *cured* if its (raw) CT exceeds $C_{\min} := 1000$ (s) (also part of the paint specification, cf. Nowak et al., 2021), which scales to $C_{\min} = \frac{1}{2}$ (cf. Section 2.2). We insist that

$$C^{\text{raw}}_i(\boldsymbol{x}) \ge C_{\min} = \frac{1}{2}. \tag{9}$$

We encode (9) by defining the (net) CT of part i (in OC $\boldsymbol{x}$) as

$$C_i(\boldsymbol{x}) := \begin{cases} C^{\text{raw}}_i(\boldsymbol{x}), & C^{\text{raw}}_i(\boldsymbol{x}) \ge C_{\min} = \frac{1}{2} \\ 0, & \text{otherwise.} \end{cases} \tag{10}$$

We define the *simulation CT* as

$$C(\boldsymbol{x}) := \min_{1 \le i \le n} C_i(\boldsymbol{x}). \tag{11}$$

For a given T, this is the amount of time spent doing useful work, namely curing parts. We use $C(\boldsymbol{x})$ as a proxy for oven throughput, and seek to maximize it.

We set $C(\boldsymbol{x}) = 0$ if $\boldsymbol{x} \notin \boldsymbol{V}$, or (9) fails i.e. $\boldsymbol{x}$ is invalid for the objective, and one or more parts is insufficiently cured/burnt. Hence, $C(\boldsymbol{x}) \in [0,1]$, by virtue of Section 2.2.

For a given $\boldsymbol{x}$, the value of $C(\boldsymbol{x})$ will depend on whether it is evaluated on an IBOFlow or synthetic simulation. We therefore pose the maximization problems (cf. (3)),

$$(\mathbf{P}_{C_{\text{exact}}}) \quad \text{Find} \quad \boldsymbol{x} \in \boldsymbol{V} \quad \text{s.t.} \quad C_{\text{exact}}(\boldsymbol{x}) \to \max, \tag{12a}$$

$$(\mathbf{P}_{C_{\text{synth.}}}) \quad \text{Find} \quad \boldsymbol{x} \in \boldsymbol{V} \quad \text{s.t.} \quad C_{\text{synth.}}(\boldsymbol{x}) \to \max. \tag{12b}$$

We also define a surrogate CT objective function, $C_{\text{surr.}}$ (see Section 4.5), and solve problems of the form

$$(\mathbf{P}_{C_{\text{surr.}}}) \quad \text{Find} \quad \boldsymbol{x} \in \boldsymbol{V} \quad \text{s.t.} \quad C_{\text{surr.}}(\boldsymbol{x}) \to \max. \tag{12c}$$

We set $f_{\min} = 0$ for these objective functions.

We can think of (11) as defining a function

$$C(\boldsymbol{x}) := C\left(\left\{ \Theta^{\max,\max}_i(\boldsymbol{x}), \Theta^{\max,\min}_i(\boldsymbol{x}), C^{\text{raw}}_i(\boldsymbol{x}) \right\}_{i=1}^{n} \middle| \, \Theta^{\text{curing}}_{\min}, \Theta^{\text{curing}}_{\max} \right), \tag{13}$$

for given $\Theta^{\text{curing}}_{\min}$ and $\Theta^{\text{curing}}_{\max}$, with each C^{raw}_i as in (8a). The CT thus has two parts: a regression part (a value) and a classification part (valid/invalid $\boldsymbol{x}$). We measure the inputs on the right side of (13) for a simulation, and inject them into a deterministic function that evaluates the two parts. This perspective is useful for evaluating our surrogates.

MCO objectives

We expect solutions of (6) to have small Θ^{inlet}, and solutions of (12) to have large Θ^{inlet}. We combine these objectives in an MCO problem to maximize the number of cured parts per unit of input energy. To this end, we consider the composite weighted sum objective function

$$M(\boldsymbol{x}; \theta_{\text{IN}}, \theta_{\text{CT}}) := \begin{cases} \theta_{\text{CT}} C(\boldsymbol{x}) - \theta_{\text{IN}} \Theta^{\text{inlet}}(\boldsymbol{x}), & \boldsymbol{x} \in \boldsymbol{V}, \text{ (9) holds for each } i, \\ -\theta_{\text{IN}}, & \text{otherwise,} \end{cases} \tag{14}$$

for weights $\theta_{\text{IN}}, \theta_{\text{CT}} \ge 0$. For $\theta_{\text{CT}} > 0$, $M(\boldsymbol{x}; \theta_{\text{IN}}, \theta_{\text{CT}})$ depends on the simulation type (IBOFlow, synthetic), because the CT does. We therefore consider 2-parameter families of constrained maximization problems of the form (cf. (12), (6))

$$(\mathbf{P}_M)_{(\theta_{\text{IN}}, \theta_{\text{CT}})} \quad \text{Find} \quad \boldsymbol{x} \in \boldsymbol{V} \quad \text{s.t.} \quad M(\boldsymbol{x}; \theta_{\text{IN}}, \theta_{\text{CT}}) \to \max. \tag{15}$$

For MCO objectives, we set $f_{\min} = -\theta_{\text{IN}}$, i.e. $M(\boldsymbol{x}; \theta_{\text{IN}}, \theta_{\text{CT}}) = -\theta_{\text{IN}}$ if $\boldsymbol{x} \notin \boldsymbol{V}$, or $\boldsymbol{x}$ is invalid for the CT objective.

4. Surrogates

4.1. Data (train/validate/test)

We generate nine quasi-random Sobol sequences (Burkardt, 2020) of vectors (three sequences in each of $\mathbb{R}^3$ (single part coordinates plus Θ^{inlet}), $\mathbb{R}^5$ (two part coordinates plus Θ^{inlet}), and $\mathbb{R}^7$ (three part coordinates plus Θ^{inlet})), each of which is identified with a set of OCs. These are validated by IBOFlow, and simulations executed for each $\boldsymbol{x} \in \boldsymbol{V}$. Results are organized into our training set $\boldsymbol{V}_{\text{train}}$ (2 186 1-part [CC, LPP, SPP], 2 199 2-part [CC/LPP, CC/SPP, LPP/SPP], and 3 928 3-part simulations), validation set $\boldsymbol{V}_{\text{valid.}}$ (206 3-part simulations) and test set $\boldsymbol{V}_{\text{test}}$ (398 3-part simulations).[7] Only ~2% of the 3-part OCs are valid.

4.2. Simulation surrogate: NNs

For simplicity, we consider only fully connected feed-forward NNs (Géron, 2019, Abadi et al., 2015) that learn the functions $\Theta^{\min}_i(\boldsymbol{x}, t_j), \Theta^{\text{mean}}_i(\boldsymbol{x}, t_j), \Theta^{\max}_i(\boldsymbol{x}, t_j)$. We consider two models of the oven with different input features.

[7] In particular, the validation and test sets contain only 3-part simulations.

Variable-(1,2,3)-part features

In a **variable-part** NN, we model an oven that contains *up to* three parts (CC ($i = 1$), LPP ($i = 2$) and SPP ($i = 3$)). We train such networks on all of $\boldsymbol{V}_{\text{train}}$. For our optimization application, we are interested in 3-part simulations and hope that the auxiliary learning task (i.e. for less than three parts) will have a regularizing effect and improve generalization. Each input has 11 features: simulation time t, Θ^{inlet}, and 3 components for each of the 3 parts: an indicator $\mathbb{1}_i$ indicating whether part i is present, and its coordinates (y, z). If part i is not present, $\mathbb{1}_i = 0$ and $(y_i, z_i) = \mathbf{0}$.

Fixed-(3)-part features

In a **fixed-part** NN, we model an oven that contains three parts. Each input has 8 features: t, Θ^{inlet} and the (y, z) coordinates of each part. We train these networks on the 3-part simulations in $\boldsymbol{V}_{\text{train}}$.

Outputs, loss function

In both cases, we learn three output temperatures $(\Theta_i^{\min}, \Theta_i^{\text{mean}}, \Theta_i^{\max})$ for each of three parts $i = 1, 2, 3$, i.e. nine outputs. For parts that are not present (i.e. $\mathbb{1}_i = 0$), we set $\Theta_i^{\min} = \Theta_i^{\text{mean}} = \Theta_i^{\max} = 0$. Our variable-part NNs learn a function

$$\mathcal{N}_V : \mathbb{R}^{3n+2} \ni (\mathbb{1}, \boldsymbol{x}, t) \to (\boldsymbol{\Theta}_1, \boldsymbol{\Theta}_2, \boldsymbol{\Theta}_3) \in \mathbb{R}^{3n}, \tag{16a}$$

and our fixed-part networks learn a function

$$\mathcal{N}_3 : \mathbb{R}^{2n+2} \ni (\boldsymbol{x}, t) \to (\boldsymbol{\Theta}_1, \boldsymbol{\Theta}_2, \boldsymbol{\Theta}_3) \in \mathbb{R}^{3n}. \tag{16b}$$

Our (training) loss function is

$$\text{MSE}_{\text{train}} := \frac{1}{M} \sum_{\boldsymbol{x}_i} \sum_{t_{ij}} \left\| \mathcal{N}(\boldsymbol{x}_i, t_{ij}) - S_{\text{IBF}}(t_{ij}, \boldsymbol{x}) \right\|_2^2, \tag{17}$$

where $\boldsymbol{x}_i$ is the ith OC in $\boldsymbol{V}_{\text{train}}$, t_{ij} is the jth time step in the ith simulation, and M is the total number of temperature measurements (all simulations, time steps, parts, temperature types [min/mean/max]) in the training set. $S_{\text{IBF}}(t_{ij}, \boldsymbol{x}) \in \mathbb{R}^{3n}$ is a vector of (exact) temperatures in a simulation, and $\mathcal{N}(\boldsymbol{x}_i, t_{ij})$ is the corresponding NN prediction (cf. (16)).

Hyper-parameters, training

We experiment with different network architectures (width: number of neurons per input feature, and depth: number of layers), activation functions (ReLU, ELU) and corresponding weight initializations, learning rate scheduling approaches (piece-wise constant, exponential, performance, 1-cycle scheduling, etc.). We train hundreds of NNs to select good values for the hyper-parameters. We use stochastic gradient descent with Nesterov Momentum, and a batch size of 32 during training.

We want to understand how the size of the training set affects accuracy. To this end, we train our variable-part NNs on all 1,2-part simulations in $\boldsymbol{V}_{\text{train}}$, plus a variable number of 3-part simulations,

$$N_3 \in \{500, 1000, 1\,500, 2000, 2\,500, 3000, 3\,500, 3\,928\}$$

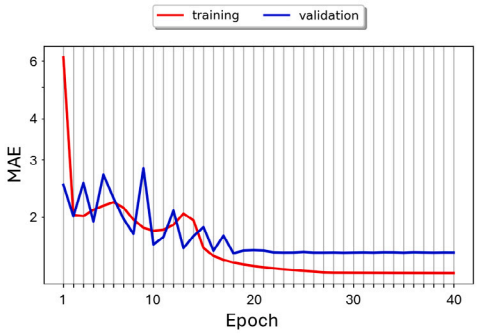
in a *training programme*. For each N_3, we start training with the best NN (i.e. with the lowest training error) from the previous, smaller training set. Validation is always done with the (fixed) $\boldsymbol{V}_{\text{valid.}}$.

The training of fixed-part NNs is the same, except that they are trained only on 3-part simulations.

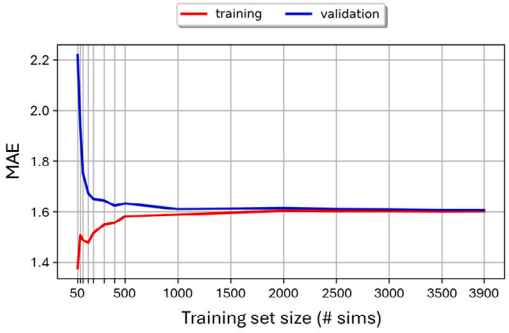
Results

Training is surprisingly difficult and time-consuming (days per training programme). We are forced to use *extremely* small learning rates (10^{-7} to 10^{-5}, as larger rates consistently lead to divergence) over large numbers of epochs (hundreds per programme). We present sample learning curves from this process in Fig. 2(a).

We observe the smallest validation error for two NNs, which we designate NN-3 (3 parts (fixed)) and NN-V (variable parts):



(a) NN learning curves (MAE vs epoch) of NN-3 with $N_3 = 500$ for the first part of an NN 'training programme'.



(b) RR learning curves (CV average MAE vs training set size) for $(p_\Theta, p_\Delta, p_\tau) = (0, 3, 3)$.

Fig. 2. Illustrative/sample NN and RR learning curves. In both cases, we measure learning performance with the training/validation MAE (all parts/all temperature components). For NNs, we use the number of training epochs as a proxy for the learning experience. For RRs, we use N_3: the training set size.

NN-3 ELU activation function, eight (hidden) layers, with three neurons per layer per input feature (fixed width of 24; 4 641 trainable parameters). Trained on the 3-part simulations in $\boldsymbol{V}_{\text{train}}$, its validation MAE is 1.190 K.

NN-V ReLU activation function, eight (hidden) layers, with three neurons per layer per input feature (fixed width of 33; 8 556 trainable parameters). Trained on all of $\boldsymbol{V}_{\text{train}}$, its validation MAE is 1.196 K.

4.3. Simulation surrogate: non-linear regressions

Based on our observations of part temperature profiles, we propose an *ansatz* for the time evolution of the min/mean/max temperatures of part i,

$$\Theta_i(t) = \Theta_{i,0} + \Delta_i \left(1 - \exp\left(-\frac{t}{\tau_i}\right)\right), \quad 0 \leq t \leq T. \tag{18}$$

Each of these functions has three parameters: $\Theta_{i,0}$, Δ_i and τ_i. $\Theta_{i,0} = 293.15$ is the initial temperature (at $t = 0$) for each part/temperature (cf. Section 2.1). Δ_i is the temperature increment due to oven heating, i.e. the steady state temperature is $\Theta_{i,0} + \Delta_i$. We need ($3 \times$ min/mean/max =) nine such functions for each 3-part simulation.

Two-stage training

Stage 1. We use non-linear regression (Virtanen et al., 2020[8]) to find the best-fit parameters of (18) for each simulation in a training set.

The result is nine sets of training data $\{(y_i, z_i, \Theta^{\text{inlet}}); \Theta_{i,0}, \Delta_i, \tau_i\}$ (three temperatures for each part), in which the best-fit parameters are associated with the position (y_i, z_i) of the part, and Θ^{inlet}.

Stage 2. We use linear regression, as in Pedregosa et al. (2011), to fit each set of parameters from Stage 1 (for each part), to a polynomial in y_i, z_i, and Θ^{inlet},

$$\Theta_{i,0} := 293.15, \tag{19a}$$

$$\Delta_i = \Delta_i\left(y_i, z_i, \Theta^{\text{inlet}}\right) = \sum_{m,n,o} \widetilde{\Delta}_{m,n,o} y_i^m z_i^n \left(\Theta^{\text{inlet}}\right)^o, \tag{19b}$$

$$\tau_i = \tau_i\left(y_i, z_i, \Theta^{\text{inlet}}\right) = \sum_{m,n,o} \widetilde{\tau}_{m,n,o} y_i^m z_i^n \left(\Theta^{\text{inlet}}\right)^o. \tag{19c}$$

$\Theta_{i,0}$ is a constant initial temperature function. The temperature evolution of part i in (19) depends on its position (y_i, z_i), and Θ^{inlet}, but not on the positions of the other parts in the oven. We characterize our fitted nonlinear regressions by a triple $(p_\Theta \equiv 0, p_\Delta, p_\tau)$ of the fitted polynomial orders. We use second and third order polynomials for Δ_i and τ_i based on informal numerical experiments and visualizations.

We fit functions Δ_i, τ_i for each part/temperature combination.[9] All of the fitted Δ polynomials (all parts/temperatures) have the same order, as do the τ polynomials.

Whereas we train our NNs on thousands of time-stamped temperature observations per simulation, the regression training process extracts one set of labelled training data $\{(y_i, z_i); \Theta_{i,0}, \Delta_i, \tau_i\}$ from each 3-part simulation.

Training regressions is *significantly* less computationally demanding than training NNs,[10] and so we combine the 3-part simulations from $\boldsymbol{V}_{\text{train}}$ and $\boldsymbol{V}_{\text{valid.}}$, and use k-fold **cross-validation** (CV) for each training set size, $N_3 \in \{50, 75, 100, 150, 200, 300, 400, 500, 1000, 1500, 2000, 2500, 3000, 3500, 3900\}$.

We vary $k \geq 6$ to accommodate the training set size and our available data.

Results

Training and validation errors appear to plateau above $\sim$ 1000 simulations (cf. Fig. 2(b)). Adding more training data has no significant effect on accuracy because of the bias of the *ansatz* (18). The $(p_\Theta, p_\Delta, p_\tau) = (0, 3, 2)$ and $(p_\Theta, p_\Delta, p_\tau) = (0, 3, 3)$ results are virtually indistinguishable. Going forward, we work with a $(0, 3, 3)$ regression trained on 3 900 simulations (cf. Fig. 2(b)), with a validation MAE of 1.58 K and a MAPE of 0.45%. We refer to this surrogate as 'RR'.

4.4. Simulation surrogates: accuracy over the test set

We now compare the synthetic simulations to their IBOFlow counterparts in (the unseen) $\boldsymbol{V}_{\text{test}}$. Summary results appear in Table 1.

The two NNs are more accurate than RR. The errors for the (smaller) LPP, SPP parts are much higher than for the CC. All of the surrogates produce large errors for the LPP $\Theta^{\min}$ function; RR also performs poorly with the SPP.

The MAPEs of $\sim$ 0.3% for NN-3/V are impressive, but histograms of the errors (see σ_{MAE} in Table 1) suggest a significant spread, which is likely to be a problem for the CT objective. The RR errors have a particularly large spread, suggesting a high probability of large errors.

Table 1
Summary test-set performance.

Surrogate	RMSE	MAE	σ_{MAE}	MAPE
NN-3	1.86	1.16 K	1.45 K	0.33%
NN-V	1.92	1.23 K	1.48 K	0.35%
RR	2.36	1.64 K	1.71 K	0.46%

σ_{MAE} is the standard deviation of the MAE. NN-3 is the most accurate surrogate (by a small margin).

Error heat maps of the oven suggest that errors are generally larger for small y, suggesting that temperature profiles are harder to learn when parts are centred in that part of the oven. The effect is most severe for the LPP and SPP parts under the RR surrogate. Errors in predictions of the temperatures of the CC part are generally unaffected by its location.

Plots over the simulation horizon show that errors increase for early times, peak near $t = 0.2$, and then decrease. The problem is that this peak is near the onset of curing (i.e. when temperatures enter the $\left[\Theta_{\min}^{\text{curing}}, \Theta_{\max}^{\text{curing}}\right]$ interval), which could severely affect the accuracy of CT estimates. NN-3/V errors tend to decay from the peak, but RR errors increase again as $t \to 1$.

All of the surrogates tend to become less accurate as Θ^{inlet} increases. We expect maximizers of the CT objective function to have high Θ^{inlet} values, which means that our surrogates might perform poorly as we approach these maximizers.

4.5. A Gaussian process CT objective function surrogate

Instead of learning the CT directly, we train a Gaussian process (Rasmussen, 2003) to produce a vector of its inputs, as a function of the OC (cf. (13)). We learn

$$\mathbb{R}^{2n+1} \ni \boldsymbol{x} \mapsto \left\{\Theta_i^{\max,\max}(\boldsymbol{x}), \Theta_i^{\max,\min}(\boldsymbol{x}), C_i^{\text{raw}}(\boldsymbol{x})\right\}_{i=1}^{n} \in \mathbb{R}^{3n}, \tag{20}$$

($\mathbb{R}^7 \to \mathbb{R}^9$ for us). To estimate the CT, we put the output of (20) into (13).

Remark. We have chosen not to learn the scalar CT for several reasons. First, this function is extraordinarily complicated, and we estimate that we could achieve higher accuracy by learning its (simpler) inputs. As we note in Section 3.1, the CT includes a classification part and a regression part. It would be difficult to validate an OC for the CT objective using a single (learned) scalar output. But we can unambiguously validate OCs for the CT based on its (learned/surrogate) inputs. We can thus compare this surrogate with our simulation surrogates.

Training

We train a set of Gaussian process regressions on training sets of size $N_3 \in \{100, 200, 500, 1000, 1500, 2000, 2500, 3000, 3500\}$, using 8-fold cross-validation for $N_3 = 3500$, and 6-fold cross-validation for the others. We experiment with various covariance kernel functions, and opt for an isotropic Matérn kernel with $\nu = 0.5$.

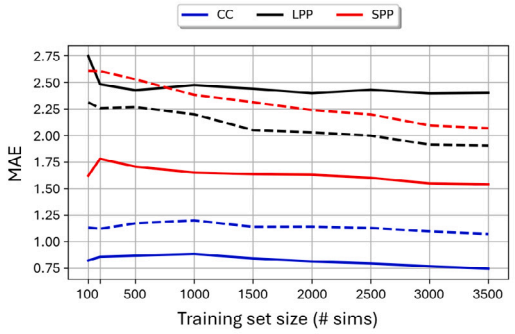
Results

Fitting a Gaussian process involves maximizing a log-likelihood function, which is quite different from the MSE loss function in (17). Nevertheless, we are able to construct learning curves similar to those for our simulation surrogates by evaluating a validation version of (17). Training errors for Gaussian processes are essentially zero as a result of the fitting process, and are therefore omitted. We show validation errors (averaged across the CV folds) in Fig. 3. Consistent with our work

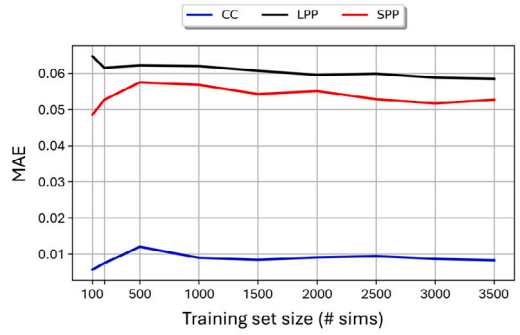
[8] https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.curve_fit.html

[9] We fit 2 functions (Δ_i, τ_i) × 3 parts × 3 temperatures = 18 functions.

[10] Seconds to train a regression; hours for an NN.



(a) MAE validation errors (in K) vs training set size. Solid lines represent $\Theta^{\max,\max}$ errors; dashed lines represent $\Theta^{\max,\min}$ errors.



(b) MAE validation errors (in scaled/unit seconds) vs training set size, for C^{raw}.

Fig. 3. GP learning curves. The GP surrogate does not learn complete time-dependent part temperature profiles, and so the MAEs relate to inputs to the CT objective function. Moreover, the GP fitting process yields extremely small (essentially zero) training errors. Direct comparisons with NNs/RRs (cf. Fig. 2) are therefore not appropriate. For the GP surrogate, we use the training set size as a proxy for learning experience, and depict validation errors (MAE) from k-fold CV, for per-part CT inputs (cf. (20)). Errors for the CC part are significantly smaller than the others. Errors appear to decline very slowly as the learning experience is increased.

on simulation surrogates, we present MAE results. We split the errors for $\Theta^{\max,\max}$, $\Theta^{\max,\min}$ and raw CT as they are of different magnitudes.

The errors associated with the smaller parts (LPP, SPP) are much larger than the errors associated with the large part (CC). The decay in the errors as a function of training set size is small, i.e. we would need *much* more data to improve these results. For further evaluation and our optimization experiments, we train a new Gaussian process on $\boldsymbol{V}_{\text{train}} \cup \boldsymbol{V}_{\text{valid.}}$, which we refer to as 'GP'.

4.6. Optimization surrogate performance

We now evaluate how well our surrogates capture the CT objective function. For our simulation surrogates, we generate synthetic simulations corresponding to each member of $\boldsymbol{V}_{\text{test}}$. We calculate and compare the CT of each synthetic simulation and the corresponding member of $\boldsymbol{V}_{\text{test}}$. For our CT objective function surrogate, we calculate and compare the surrogate and exact CT function values for each IBOFlow simulation in $\boldsymbol{V}_{\text{test}}$.

Objective function value comparison

We compare the value of the CT objective function of each member of $\boldsymbol{V}_{\text{test}}$ with that of the corresponding synthetic. Whereas an IBOFlow simulation may be valid for the CT objective function ($C(\boldsymbol{x}) \neq 0$), its synthetic counterpart may not be ($C(\boldsymbol{x}) = 0$). Similarly, the GP CT surrogate may classify a valid (for CT) member of $\boldsymbol{V}_{\text{test}}$ as invalid (for the surrogate CT). We present confusion matrices for our surrogates for the classification part of CT in Table 2. A summary of the comparative results for the regression part can be found in Table 3. MAPE figures (arguably the most informative) are defined only for the true-positive set of Table 2 (IBOFlow and synthetic simulation valid for CT), as the CT may be zero elsewhere. Results are in the right half of Table 3.

Our investigation has revealed one of the main difficulties in working with the CT objective function. One could imagine a synthetic simulation in which the coldest part has $\Theta^{\max} = \Theta^{\text{curing}}_{\min} - \frac{\varepsilon}{2}$, with $C = 0$, but the corresponding temperature in the IBOFlow simulation is $\Theta^{\text{curing}}_{\min} + \frac{\varepsilon}{2}$, and $C > \frac{1}{2}$. The corresponding surrogate might be extremely accurate, and produce synthetics that are very similar to their IBOFlow counterparts, but they could have *very* different CT values. This kind of situation would seriously affect the usefulness of surrogates in an optimization context, especially for maximizers at/near the boundary of validity.

Sign-test comparison

We do not expect $f_{\text{exact}}(\boldsymbol{x}) = f_{\text{synth.}}(\boldsymbol{x})$ or $f_{\text{exact}}(\boldsymbol{x}) = f_{\text{surr.}}(\boldsymbol{x})$ (cf. (3)), but this is not a problem *per se*. To be of use in an optimization problem, we ideally require

$$f_{\text{exact}}(\boldsymbol{x}_1) < f_{\text{exact}}(\boldsymbol{x}_2) \iff \begin{cases} f_{\text{synth.}}(\boldsymbol{x}_1) < f_{\text{synth.}}(\boldsymbol{x}_2), \\ f_{\text{surr.}}(\boldsymbol{x}_1) < f_{\text{surr.}}(\boldsymbol{x}_2), \end{cases} \tag{21}$$

for OCs $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$. We are not aware of analytical techniques to investigate (21), and so we proceed numerically. We compare

$$\text{sign}\left(f_{\text{synth.}}(\boldsymbol{x}_i) - f_{\text{synth.}}(\boldsymbol{x}_j)\right) = \text{sign}\left(f_{\text{exact}}(\boldsymbol{x}_i) - f_{\text{exact}}(\boldsymbol{x}_j)\right), \tag{22}$$

and similarly for $f_{\text{surr.}}$, for each pair of OCs $\boldsymbol{x}_i \neq \boldsymbol{x}_j$ in $\boldsymbol{V}_{\text{test}}$, for the CT and an MCO objective function (cf. (14)) with $\theta_{\text{IN}} = \theta_{\text{CT}} = \frac{1}{2}$. We plot the proportion of points for which (22) is true as a function of the Euclidean distance, d_{ij}, between pairs of OCs, $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$, in Fig. 4.

For large d_{ij}, the OCs $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ are far apart, and we investigate how well the surrogate captures the large-scale structure of the objective function. On the other hand, if d_{ij} is small, we measure how well our surrogates capture the fine structure of the objective function. This is important because we expect successive OCs generated within an optimization sequence to get closer together as they approach a

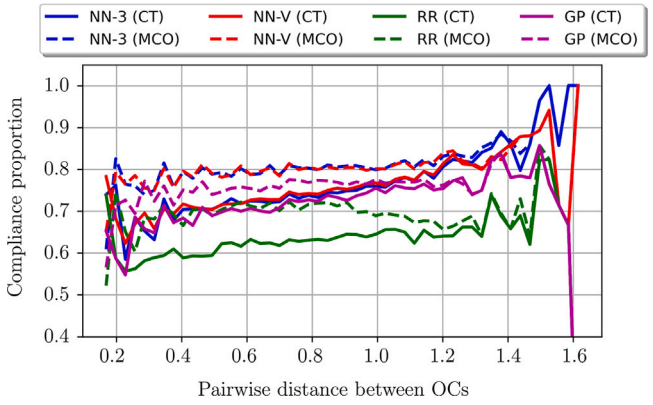


Fig. 4. Sign-test compliance results. As discussed in the main text, we do not expect or require $f_{\text{exact}}(\boldsymbol{x}) = f_{\text{synth.}}(\boldsymbol{x})$ or $f_{\text{exact}}(\boldsymbol{x}) = f_{\text{surr.}}(\boldsymbol{x})$. We are prepared to accept the condition in (22) i.e. that the *signs* of differences between pairs of exact/surrogate objective function values match. Here we depict the proportion of times (22) holds for each surrogate as a function of the Euclidean distance between pairs of sampled OCs. Solid lines are for CT, and dashed lines for the MCO objective (with $\theta_{\text{IN}} = \theta_{\text{CT}} = \frac{1}{2}$). For NN-3/V, (22) holds in about 70%–80% of sampled cases for CT, and >80% for MCO. For GP, compliance is 60%–80% for CT, and 70%–80% for MCO. For RR (worst performer) compliance is 60%–65% for CT, and ~70% for MCO. According to these graphs, all of the surrogates are less accurate at close range, by this measure. Moreover, compliance is generally better for the MCO objective than for CT.

Table 2
CT classification part confusion matrices.

		Predicted (synthetics)							
		Valid				Invalid			
		NN-3	NN-V	RR	GP	NN-3	NN-V	RR	GP
True	**Val.** (0.54)	0.49	0.5	0.5	0.46	0.043	0.035	0.04	0.078
	Inv. (0.46)	0.088	0.093	0.18	0.073	0.37	0.37	0.28	0.39

54% of the simulations in $\boldsymbol{V}_{\text{test}}$ ($|\boldsymbol{V}_{\text{test}}| = 398$) are valid; 46% are invalid. True-positive proportions (top left) and true-negative proportions (bottom right) are shaded green; false-negatives (top right) and false-positives (bottom left) are not shaded. NN-3/NN-V/RR synthetics yield the correct classification in 86%/87%/78% of cases. For GP (on $\boldsymbol{V}_{\text{test}}$) the figure is 85%.

Table 3
CT test performance.

	All		True positives		
Surrogate	RMSE	MAE	RMSE	MAE	MAPE
NN-3	0.227	0.100	0.048	0.040	6.498%
NN-V	0.223	0.098	0.048	0.040	6.449%
RR	0.293	0.157	0.053	0.044	7.018%
GP	0.241	0.111	0.052	0.042	6.858%

Four surrogates on $\boldsymbol{V}_{\text{test}}$ ($|\boldsymbol{V}_{\text{test}}| = 398$), and on the true-positive sets of Table 2 (top left). On the true-negative sets (bottom right in Table 2), the simulation surrogates produce synthetics with objective function values equal to their IBOFlow counterparts. The GP surrogate function value is also exact (= $f_{\min}$ in Section 3). The NN-V surrogate seems to be the most accurate. The MAPEs look quite high.

maximizer. According to Fig. 4, all of our surrogates are *less* accurate at close range. Compliance for the MCO objective is generally better than for CT. NN-3/V outperform the RR and GP surrogates (RR is worst). These results look quite poor, but we will see that we can still use our surrogates to solve optimization problems.

5. Optimization experiments

We consider two approaches: (i) use a sequence of surrogate simulations/evaluations to get close(r) to a maximizer, followed by a sequence of IBOFlow simulations to finish the process; (ii) use only surrogates. The second approach significantly reduces the computational effort, at the risk of degraded solution quality.

5.1. Initial guesses

We start each optimization from a *set* of k IGs (i.e. OCs) to reduce the risk of getting stuck in a local maximum. The IG selection process proceeds as follows: we maximize the objective function over $\boldsymbol{V}_{\text{data}}$ ($\equiv \boldsymbol{V}_{\text{train}} \cup \boldsymbol{V}_{\text{valid.}} \cup \boldsymbol{V}_{\text{test}}$) and call the corresponding OC $\boldsymbol{x}_1 \in \mathbb{R}^{2n+1}$. We use $\boldsymbol{V}_{\text{data}}^{\text{valid}}$ to denote the subset of $\boldsymbol{V}_{\text{data}}$ that is valid for the objective function. We extract the remaining $k-1$ IGs from $\boldsymbol{V}_{\text{data}}^{\text{valid}} \setminus \{\boldsymbol{x}_1\}$, so that the k IGs are as well separated as possible. To do this, we maximize the minimum Euclidean distance (treating OCs as vectors in $\mathbb{R}^{2n+1}$) between any pair of OCs in a set of k OCs from $\boldsymbol{V}_{\text{data}}^{\text{valid}}$ (containing $\boldsymbol{x}_1$). We seek C_k, a set of k OCs that includes $\boldsymbol{x}_1$, and solves

$$(\mathbf{P}_{\text{IC}})_k \quad \text{Find} \quad C_k \subset \boldsymbol{V}_{\text{data}}^{\text{valid}} \quad \text{s.t.} \quad \min_{\boldsymbol{x}_i, \boldsymbol{x}_j \in C_k} \left\| \boldsymbol{x}_i - \boldsymbol{x}_j \right\|_2 \to \max . \tag{23}$$

Our numerical approach is simple: we draw a large number of sets of $k-1$ OCs from $\boldsymbol{V}_{\text{data}}^{\text{valid}} \setminus \{\boldsymbol{x}_1\}$, and choose the set for which $\min_{\boldsymbol{x}_i, \boldsymbol{x}_j \in C_k} \left\| \boldsymbol{x}_i - \boldsymbol{x}_j \right\|_2$ is the highest.

5.2. Optimization approaches

We consider nine approaches. For each, we perform one or more NLopt COBYLA (Constrained Optimization BY Linear Approximations; see Johnson, 2007, Powell, 1994a, Powell, 1994a, 1998)[11] optimization sequences, with relative and absolute x tolerances of 10^{-3}. We use IBOFlow or a surrogate to generate simulations or estimate objective function values. The computational cost of an IBOFlow simulation *significantly* exceeds that of using a surrogate.[12] We therefore use the number of IBOFlow simulations as a proxy for the computational cost of each method. The first two methods are IBOFlow baselines:

IBF(1) (results in **black**). We use IBOFlow with the exact objective function. We start a COBYLA sequence from each IG.

IBF(2) (results in grey). Two consecutive COBYLA sequences (Stage-1 and Stage-2) are run with IBOFlow and the exact objective function. The result of Stage-1 is the IG for Stage-2. We consider IBF(2) as the worst-case cost for the two-stage surrogate approaches (see below). We expect that it might also represent a bound on the expected solution quality.

The next four approaches are similar to IBF(2), except that we use a surrogate in Stage-1. The surrogate is termed *successful* if Stage-1 yields an OC with a higher exact objective function value than the IG. For a given surrogate (simulation/objective function), any COBYLA sequence will increase the (surrogate/synthetic) objective function value. But the resulting OC may correspond to an (exact) IBOFlow OC/simulation that is either invalid for the (exact) objective function, or is valid, but has a lower (exact) objective function value than the IG. We say that the surrogate *fails* in such cases. We make two adjustments to identify and deal with these failures. (i) Each surrogate COBYLA sequence is preceded/succeeded by an IBOFlow simulation/function evaluation of the IG/result. Each such sequences therefore incurs a computational cost of +2 IBOFlow simulations. (ii) If the post-sequence evaluation shows that the surrogate has failed, we run an IBF(1) sequence (which is guaranteed to succeed) from the Stage-1 IG.

NN-3 (results in blue). An NN-3 simulation surrogate generates synthetic simulations for each step of Stage-1.

NN-V (red). An NN-V surrogate generates the Stage-1 simulations.

RR (dark green). An RR surrogate generates the Stage-1 simulations.

GP (magenta). A GP surrogate estimates the CT for each OC in Stage-1. No simulations are generated.

Finally, we examine three ensemble-type **best-surrogate** BS(·) approaches. We use all of the surrogates in a series of Stage-1 maximizations, and then (optionally) perform a second maximization with IBOFlow.

BS(1) (pink). We use each surrogate to run Stage-1 from each IG, and select the result of the successful surrogate with the highest exact function value. If all of the surrogates fail for an IG, we discard the IG. If no surrogate succeeds for any IG (all surrogates fail from every IG[13]), we run IBF(1) from each IG.

[11] We use COBYLA for consistency with Nowak et al. (2021).

[12] Our surrogates are unable to validate that an OC is valid viz. that $\boldsymbol{x} \in \boldsymbol{V}$. IBOFlow performs all of the OC validations.

[13] We never encountered this condition in our experiments

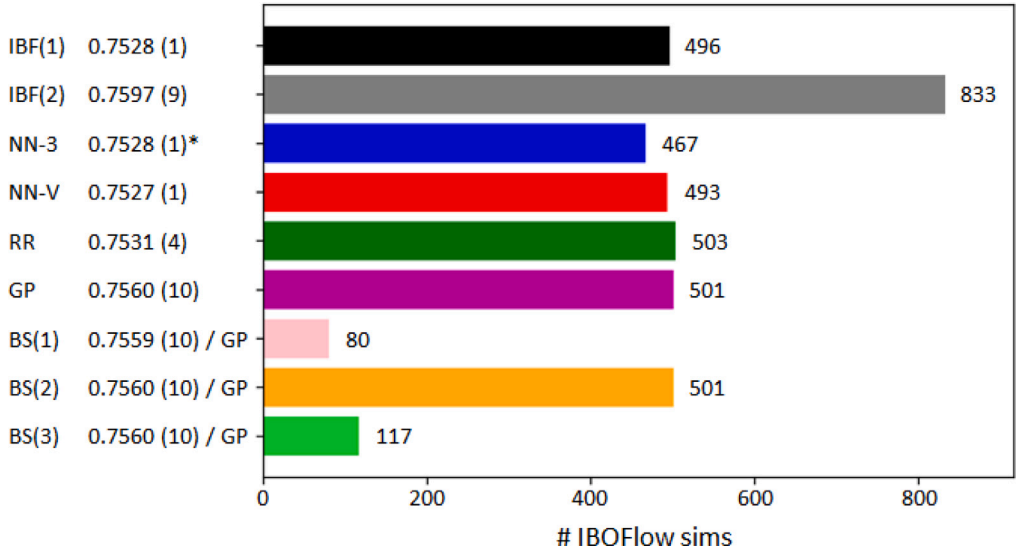


Fig. 5. Summary of results. The second column contains the highest CT value, and its IG. The horizontal bars show the total number of IBOFlow simulations for each approach, from ten IGs. IBF(2) yields the highest CT, from IG-9 (second is GP, from IG-10). All of the highest CT values (per approach) are within 1% of the maximum. BS(3) yields the same maximum as the more expensive BS(2). BS(1) is by far the cheapest approach. *NN-3 is the cheapest standalone surrogate. Its best result comes from IG-1, but NN-3 fails from this IG, which means that this result comes from an IBF(1) sequence. Its best success result (i.e. not requiring an IBF(1) sequence) is 0.7359, from IG-4.

BS(2) (orange). Using the BS(1) results as IGs, we execute an IBF(1) sequence from each sequence for which one or more surrogates was successful.

BS(3) (lime green). A variant of BS(2), in which the Stage-2 IBF(1) sequence is executed only from the result of the best-performing sequence/IG (with the highest objective function value) in Stage-1.

5.3. Single objective optimization: CT

We experiment with the CT objective function (12), with $k = 10$ IGs. Each synthetic simulation (NN-3, NN-V, RR) has 10^4 time-steps.

Results

The best results for each approach, and the total number of simulations (all IGs) appear in Fig. 5. Surrogate failures (the number of IGs where the surrogate failed) are: NN-3: 4, NN-V: 3, RR: 7 and GP: 3, for an overall failure rate of 43%.

Discussion

Our first observation is that while IG-1 is a good guess, it does not always lead to the highest CT. Indeed, our best result comes from IG-9, which provides at least some justification for our (expensive!) multiple IG approach. IBF(2) is the most expensive method, but does not yield the highest CT from every IG.

The NN-3/V surrogates yield a small reduction in the total number of IBOFlow simulations compared to IBF(1) (NN-3 by ∼6%, and NN-V by <1%). RR and GP are slightly more expensive than IBF(1). BS(1) is the cheapest method, and is able to produce objective function values within 1% of the best IBF(2) solution. BS(2) is more expensive than BS(1), but does not perform better than the less expensive BS(3) (which finds the same maximum). Interestingly, the best surrogate (GP) was not the most accurate in Section 4.6 (cf. Fig. 4).

It would be strange to use IBF(2) in practice, and we are pleased that our surrogates achieve higher objective function values from several IGs, at *substantially* lower computational cost. Without IBF(2), IBF(1) would give the best results from IG-1, IG-2 and IG-3, and NN-V would be the best approach from IG-9. The best overall method would be GP, from IG-10.

5.4. MCO

We use the Sandwiching algorithm (see Section 5 of Andersson et al., 2018) to solve MCO problems. This algorithm chooses a set of weights $\{\theta_{\text{IN}}, \theta_{\text{CT}}\}$ in (15) and solves a set of weighted sum problems to estimate the Pareto front. We assume a concave Pareto front (cf. Nowak et al., 2021). Solving an MCO problem therefore involves solving a maximization problem from multiple IGs (as in the previous section), for each Pareto point. We choose $p = 7$ Pareto points, and solve each component problem from $k = 6$ IGs.

We use the approaches described in Section 5.2 for the component problems, but instead of IBF(2), we maximize $(\mathbf{P}_M)$ over $\boldsymbol{V}_{\text{data}}$ for each Pareto point. We call this method 'Set Max.', and assign it the colour cyan.

Set Max. For each $\{\theta_{\text{IN}}, \theta_{\text{CT}}\}$ from the sandwiching algorithm, solve (cf. (15))

$$(\mathbf{P}_M)_{(\theta_{\text{IN}},\theta_{\text{CT}})} \quad \text{Find} \quad \boldsymbol{x} \in \boldsymbol{V}_{\text{data}} \quad \text{s.t.} \quad M\left(\boldsymbol{x}; \theta_{\text{IN}}, \theta_{\text{CT}}\right) \to \max. \tag{24}$$

To do this, we evaluate each (exact) $M\left(\cdot; \theta_{\text{IN}}, \theta_{\text{CT}}\right)$ on each $\boldsymbol{x} \in \boldsymbol{V}_{\text{data}}$, and choose the $\boldsymbol{x}$ with the highest MCO objective function value.

We do *not* use the sandwiching algorithm to solve nine MCO problems, with different sets of weights. To isolate the effect(s) of our methods, we:

(i) use Set Max., with sandwiching.
(ii) use IBF(1), with sandwiching. We find a different set of IGs for each Pareto point (cf. Section 5.1), based on the MCO objectives $M\left(\cdot; \theta_{\text{IN}}, \theta_{\text{CT}}\right)$, with weights from the sandwiching algorithm.
(iii) use the remaining methods to solve the same problems (the same IGs and weights) as were solved by IBF(1).

We can compare the results from IBF(1) with those from NN-3/V, RR, GP, and BS(·), as they solve the same problems.

Results

The MCO weights selected by the sandwiching algorithm are in Table 4. The MCO objective function values calculated by the various methods are in Table 5 and Fig. 6. The computational cost (# of IBOFlow simulations) of the individual methods is summarized in Fig. 7.

Table 4
MCO sandwiching weights.

Point	Weights: $(\theta_{IN}, \theta_{CT})$ Set Max.	IBF(1)	Comment
(1)	$(1-10^{-6}, 10^{-6})$		Very small CT weighting
(2)	$(10^{-6}, 1-10^{-6})$		See the problem in Section 5.3
(3)	$(1, 1)$		Equal weightings
(4)	$(0.201, 0.980)$	$(0.188, 0.982)$	Large CT weighting
(5)	$(0.294, 0.956)$	$(0.273, 0.962)$	Large CT weighting
(6)	$(0.0981, 0.995)$	$(0.0940, 0.996)$	Large CT weighting
(7)	$(0.334, 0.943)$	$(0.0757, 0.997)$	Large CT weighting

Set Max. and IBF(1). The Set Max. and IBF(1) weights for points (1)-(3) are identical. IBF(1) points (6) and (7) have similar weights. The Set Max. and IBF(1) weights for point (7) are very different.

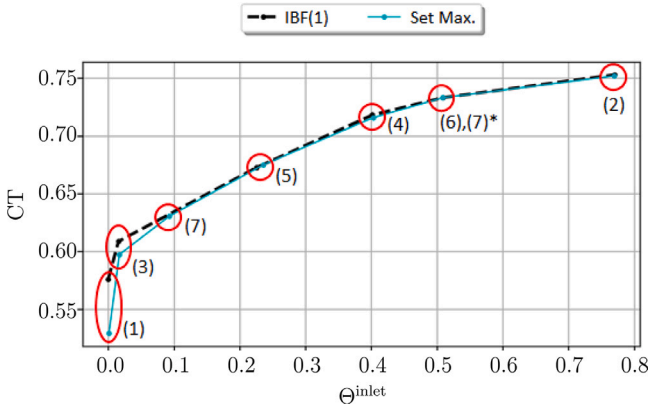


Fig. 6. (Θ^{inlet}, CT) Pareto fronts for IBF(1) and Set Max. with sandwiching. We regard these two solutions as baselines: Set Max. with sandwiching is the cheapest, simplest technique (by far), and does not use a formal optimization algorithm. IBF(1) is almost at the other complexity extreme, and depends on a large number of exact simulations from IBOFlow. In this graph up/left is 'better' (higher CT for given Θ^{inlet}). This graph says that all of the engineering and computational expense of IBF(1) yields the small difference (additional height) of the dashed black line compared to the blue one. The objective of this study is to find Pareto fronts that cost less than IBF(1), and are (at least) above/left of the blue front (if not the black one too). The solution approximation qualities of Set Max. and IBF(1) are 0.0224 and 0.0162. The Pareto points are labelled as in Table 4; *IBF(1) point (7) is close to point (6) and far from Set Max. point (7).

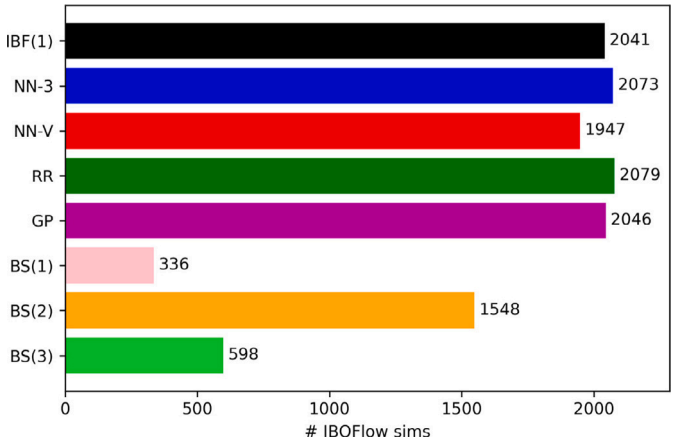


Fig. 7. Total number of IBOFlow simulations. NN-V is slightly cheaper than IBF(1). BS(1) is the cheapest.

Discussion

In Fig. 6, the IBF(1) front is higher and to the left of the Set Max. front, and has a higher solution approximation quality. Pareto point comparisons between Set Max. and IBF(1) are generally not possible (different weights for points (4)–(7), cf. the cyan cells in Tables 4 and 5), but it seems that most of the improvement is due to points (1) and (3). The improvement in solution quality comes at the incremental computational cost of using IBF(1), which is one of our most expensive methods (cf. Fig. 7; recall that we use $\boldsymbol{V}_{data}$ to find IGs for IBF(1)). The IBF(1) results are only slightly better than those from Set Max.

The standalone surrogates rarely beat IBF(1), and some of the improvements are small (at the sixth or seventh significant digit). The more significant improvements are: NN-V and RR perform better for point (1), and GP performs better for point (3). The reason for this seems to be that our surrogates often fail from IG-1 (cf. the un-shaded cells in Table 5), yet this IG led to the best results for the majority of IBF(1) Pareto points. BS(1) is cheap (cf. Fig. 7) and performs quite well on points (2) and (6). BS(3) performs quite well on points (1), (3) and (5). BS(2), the most expensive BS(·) method, performs quite well for most points; for points (4) and (7), it beats BS(3), but not IBF(1).

We are surprised that the results in Fig. 4 have no effect here. In that graph, we saw improved sign-test compliance for an MCO objective combining CT and Θ^{inlet}. We had hoped that this would lead to greater computational savings if we increased the Θ^{inlet} proportion in MCO problems, but this is evidently not the case.

Comparing Table 5 and Fig. 7, we see that higher IBOFlow simulation counts generally lead to better/higher (exact) MCO objective function values. Based on this observation, and our other results, we propose an approach that allows the practitioner to match computational effort with the desired solution quality:

(1) Solve the MCO problem with Set Max. with sandwiching. For sufficiently large $\boldsymbol{V}_{data}$, these solutions might be satisfactory, and no further effort would be required.
(2) Find a set of IGs for each Pareto point returned by the Set Max. method, i.e. treat each Set Max. solution as $\boldsymbol{x}_1$ in a set of IGs (cf. Section 5.1). Run BS(1) for each point. Keep solutions with a higher objective function value (compared to the Set Max. solution).
(3) Run BS(3) for the Pareto points for which step (2) did not result in an increase. Keep those solutions with higher objective function values.
(4) Run BS(2) for the Pareto points for which step (3) did not result in an increase. Keep those solutions with higher objective function values.
(5) If additional solution quality is required, or there are points for which every surrogate has failed, run IBF(1) from the Set Max. Pareto point.
(6) If necessary, run further IBF(1) sequences from all remaining Pareto point/IG combinations for which one or more surrogates failed.

6. Conclusion

6.1. Summary of our contribution

We studied single objective and MCO problems and proposed various solution techniques using ML surrogates. We developed two types of surrogate: (i) simulation surrogates (two NNs and an RR), and (ii) objective function surrogates (a GP). While our surrogates were bespoke, they were not particularly novel, and our contribution lies elsewhere. First, we developed a framework for assessing, evaluating and comparing surrogates as standalone objects, and in terms of the accuracy of their approximations of a challenging objective function. Second, we proposed methods for using surrogates to solve optimization problems, as standalone objects, and in a stack-ensemble-like combination. A typical ensemble approach would entail combining multiple learners to

Table 5
MCO objective function values.

	Solution Point						
Approach	**(1)**	**(2)**	**(3)**	**(4)**	**(5)**	**(6)**	**(7)**
Set Max.	−9.91e−4	0.752	0.581	0.620	0.576	0.680	0.563*
IBF(1)	5.76e−7 (6)	0.753	0.593	0.630	0.586	0.682	0.693
NN-3	–	–	0.593	–	–	0.682 (5)	–
NN-V	5.93e−7 (3)	0.753	–	–	–	–	–
RR	5.96e−7 (2)	–	0.593	–	–	–	–
GP	–	0.753	0.595	–	–	0.682	–
BS(1)	5.02e−7 (3) NN-V	0.752 GP	0.581 NN-3	0.597 (5) NN-3	0.572 (4) NN-V	0.682 GP	0.684 (2) NN-V
BS(2)	5.93e−7 (3) NN-V	0.753 GP	0.593 NN-3	0.623 (3) GP	0.584 (4) NN-V	0.682 (5) NN-3	0.688 (5) NN-V
BS(3)	5.93e−7 (3) NN-V	0.753 GP	0.593 NN-3	0.597 (5) NN-3	0.584 (4) NN-V	0.682 GP	0.687 (2) NN-V

Solutions from an IG *other* than IG-1 are shaded yellow (with the IG# in brackets). For the BS(·) methods, we name the best surrogate. Dashes indicate surrogate failures (and a reversion to IBF(1)). *Comparing the point (7) Set Max. results with the others is inappropriate (cf. point (7) weights in Table 4).

produce more accurate simulations/objective function approximations: we applied the idea to the MCO problem itself. We believe that this technique is novel, and are not aware of it having been applied to an MCO problem before. It has certainly never been applied to this specific paint curing problem. Finally, we conducted numerical experiments and presented evidence that these techniques can be used to reduce computational cost *and* improve solution quality. In particular, we were able to examine and compare more candidate solutions of the optimizations problems than would otherwise have been practical. We were unable to pick a clear winner from our set of surrogates. We used our experimental results to propose a practical, incremental approach to solving MCO problems using all available surrogates, with some control over the computational cost.

6.2. Observations and conclusions

There is a sense in which this entire manuscript is about the gap between the cyan and black lines in Fig. 6. This gap may have commercial value, but in practice, it might be difficult to justify the effort and cost of such a project for such small objective function value improvements. In any case, care should be taken to ensure that ML projects have a viable business case.

We should be careful celebrating our IBOFlow simulation count results. Figs. 5 and 7 suggest that BS(1) is *much* cheaper than IBF(1). However, we need to take into account the computational (and other) costs of creating $\boldsymbol{V}_{\text{data}}$ (we had $|\boldsymbol{V}_{\text{data}}| \sim 9000$ simulations), and of developing the surrogates used in BS(1). To achieve a net benefit, the savings from using the surrogates should exceed their development and training cost, including training data creation.[14] We could also use our surrogates in other ways to extract more value from them. For example, objective functions are usually defined and derived in an iterative process, whereby business priorities are translated into abstract mathematical objective functions. Typically, candidate objective functions are evaluated based on the solutions to their optimization problems, and the best/most appropriate objective function is then used to 'solve the problem'. We inherited the CT objective function from Nowak et al. (2021), but in a new project, we would create a set like $\boldsymbol{V}_{\text{data}}$ as early as possible, and use it, together with the Set Max. method, and one or more easily/cheaply trained simulation surrogates (like our GP surrogate) to quickly, but comprehensively evaluate different objective functions.

6.3. Limitations of this case study and future opportunities

There are several opportunities for extending this case study. These include (i) the application of other ML learners in more sophisticated ensemble configurations, (ii) different approaches to training the learners we used, (iii) using fine-grid training data (from IBOFlow), (iv) the application of alternate optimization (potentially gradient-based) solvers, (v) a more sophisticated approach to selecting IGs, and (vi) design, training and application of OC classification surrogates. We discuss these in the rest of this section.

It is easy to conceive of more sophisticated ML learners, and different (potentially better) approaches for designing and training the ones we considered. Our BS(·) methods resemble a simple stack ensemble (applied to the optimization problem), but more sophisticated ensemble approaches (e.g. XGBoost, Chen and Guestrin, 2016) could be applied to the ML learning tasks.

We worked exclusively with coarse, computationally cheap IBOFlow simulations. This allowed us to investigate the effects of training set size on surrogate accuracy for moderately large training sets. We were also able to run optimization experiments involving long optimization sequences, and a large number of IBOFlow simulations. Physical experiments (cf. Nowak et al., 2021) confirm that these coarse simulations are not very accurate. We suspect that our training temperature profiles suffered from various non-physical artefacts (e.g. we observed 'kinks' in many LPP and SPP temperature profiles), which were likely due to the small number of grid points used to represent these objects in the coarse discretization. These effects likely made the learning task more difficult than it otherwise would have been, especially for the RR surrogate. This raises the question (and risk) as to how much we can extrapolate these results to problems involving fine-grid IBOFlow simulations. We are confident that our results are meaningfully representative of what we would observe if we were to tackle optimization problems involving

[14] We used $\boldsymbol{V}_{\text{data}}$ to find IGs for the IBF(·) methods. We could have used a different technique, but that would also have incurred a cost.

fine-grid (higher r_o and r_g) simulations. In any case, working with fine-grid simulations to produce physically realistic results is an important next step.

We developed a deep, flexible software stack spanning the generation and management of CFD training data, feature extraction, training and evaluation of surrogates, and the use of surrogates to solve single- and multi-objective optimization problems. We used a Microsoft Windows IBOFlow executable to generate train/test simulations, IBOFlow is based on the method introduced in Mark and van Wachem (2008). All of the code for this paper was written in Python on a Microsoft Windows machine. Our NNs were built using TensorFlow (Abadi et al., 2015), we used scipy (https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.curve_fit.html) for the non-linear regression and scikit-learn (Pedregosa et al., 2011) for the Gaussian processes. Finally, we used the NLopt library for the COBYLA optimization (Johnson, 2007; Powell, 1994a). It should be straightforward to add new surrogates and solvers (like derivative-based optimization methods), and to work with different objective functions.

The CT objective function was particularly difficult to work with. Our surrogate learning curves suggest that we might see improvements if we had (significantly) more training data.[15] However, this approach would be unlikely to scale for situations involving fine-grid simulations. We should try to work with the smallest viable training sets, because one of our goals is to solve these problems as cheaply as possible. In this respect, it might be possible to pre-train NNs on a large set of (cheap) coarse-grid simulations and then refine the training on a smaller set of (expensive) fine-grid simulations. We do not recommend generating large, expensive data sets specifically so as to use ML methods, if cheaper (and more precise) methods (cf. Nowak et al., 2021) are available.

Our use of multiple IGs seemed only marginally beneficial, as we found maximizers from IGs other than $\boldsymbol{x}_1$ in Section 5.1. Our approach to finding IGs was simple, but quite crude. A more sophisticated approach could lead to IGs that are better separated, which could lead to higher quality solutions.

Our numerical results are suggestive, but it is difficult to draw general conclusions from them. It might be beneficial to perform an exercise using each element of $\boldsymbol{V}_{\text{data}}$ (or even $\boldsymbol{V}_{\text{test}}$) as an IG for optimization. We could then develop metrics to understand how surrogate failure rates, solution quality and computational effort vary with IG.

We used the number of IBOFlow simulations as a proxy for the computational cost of our techniques. Once we started using our surrogates, total computational cost was dominated by the cost of the IBOFlow candidate OC validations. Using classification surrogates to perform these validations would reduce this runtime computational cost (but incur an additional training/development cost, and introduce another approximation error).

Abbreviations and symbols

We list important abbreviations and symbols in Table 6.

Table 6
Abbreviations and symbols.

Symbol	Description
CFD	Computational fluid dynamics.
ML	Machine Learning.
Ω	Volume occupied by the oven interior in $\mathbb{R}^3$.
n	The number of vehicle parts inside the oven. All our experiments involve $n = 3$ — see Fig. 1.
$\boldsymbol{X}$	Vector (in $\mathbb{R}^{2n}$) representing the position of the vehicle parts in the oven.
Θ^{inlet}	Oven inlet temperature: the setting on a temperature control dial for the oven. We use the same symbol for the objective function with the same name.
OC	An oven configuration. A complete OC encapsulates the position and rotation of each part in the oven, and Θ^{inlet}. We fix and suppress the rotation vectors and constrain each part to be in the mid-plane ($x = 0$ in each position vector).
$\boldsymbol{x}$	Vector (in $\mathbb{R}^{2n+1}$) representing an OC. Contains the positions of the parts (two components per part in the mid-plane) and Θ^{inlet}.
$\boldsymbol{V} \subset \mathbb{R}^{2n+1}$	The set of valid OCs (parts well separated and inside the oven).
T	Total simulation time horizon.
$\boldsymbol{r}$	IBOFlow simulation refinement levels. A pair of the form $\boldsymbol{r} = (r_g, r_o)$: r_g is the grid refinement, r_o the object refinement.
CC	Cab corner (the large vehicle part in Fig. 1).
LPP	Large pc pet (the medium-sized vehicle part in Fig. 1).
SPP	Small pc pet (the small vehicle part in Fig. 1).
NN	Neural network.
NN-3	Fixed-part NN simulation surrogate. Trained on 3-part simulations, only.
NN-V	Variable-part NN simulation surrogate. Trained on 1,2,3-part simulations.
RR	Non-linear regression simulation surrogate. Generates temperature evolution profiles according to the *ansatz* in (18).
GP	Gaussian process objective function surrogate (learns the inputs of the CT objective function, cf. (20)).
IG	Initial-Guess (for the solution of an optimization problem).
CT	Curing-Time (objective function).
MCO	Multi-criteria optimization.
$\boldsymbol{V}_{\text{train}}$, $\boldsymbol{V}_{\text{valid.}}$, $\boldsymbol{V}_{\text{test}}$	ML training, validation and test simulation sets. We define $\boldsymbol{V}_{\text{data}} := \boldsymbol{V}_{\text{train}} \cup \boldsymbol{V}_{\text{valid.}} \cup \boldsymbol{V}_{\text{test}}$.
N_3	The number of three-part (i.e. having $n = 3$) simulations in a training set.
MSE	Mean squared error.
MAE	Mean absolute error.
MAPE	Mean absolute percentage error.
k	Number of IGs used to start an optimization problem.
p	Number of Pareto points required to solve each MCO problem.

CRediT authorship contribution statement

Quentin Parsons: Data curation and numerical experiments, Formal analysis, Methodology: esp. the optimization stack ensemble, Software: C++/Python data pipelines, all ML design and training, optimization, Visualization, Writing – original draft, Writing – review & editing. **Dimitri Nowak:** Conceptualization, Formal analysis, Methodology, Project administration, Writing – original draft, Writing – review & editing. **Michael Bortz:** Conceptualization, Formal analysis, Methodology, Supervision, Writing – review & editing. **Tomas Johnson:** Software, Writing – review & editing. **Andreas Mark:** Funding acquisition, Software: IBOFlow, Writing – review & editing. **Fredrik Edelvik:** Funding acquisition, Supervision, Writing – review & editing.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Andreas Mark reports financial support was provided by Sweden's Innovation Agency. Andreas Mark reports financial support was provided by FFI Sustainable Production Technology program. Andreas Mark reports financial support was provided by The project Virtual PaintShop - Simulation of Oven Curing.

Data availability

The data that has been used is confidential.

[15] This would also make $\boldsymbol{V}_{\text{data}}$ larger, and the Set Max. results even better.

Acknowledgement

This work was supported in part by the Swedish Governmental Agency for Innovation Systems, VINNOVA, through the FFI Sustainable Production Technology program and the project Virtual PaintShop - Simulation of Oven Curing.

Appendix A. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.engappai.2024.108086.

References

Abadi, Martín, Agarwal, Ashish, Barham, Paul, Brevdo, Eugene, Chen, Zhifeng, Citro, Craig, Corrado, Greg S., Davis, Andy, Dean, Jeffrey, Devin, Matthieu, Ghemawat, Sanjay, Goodfellow, Ian, Harp, Andrew, Irving, Geoffrey, Isard, Michael, Jia, Yangqing, Jozefowicz, Rafal, Kaiser, Lukasz, Kudlur, Manjunath, Levenberg, Josh, Mané, Dandelion, Monga, Rajat, Moore, Sherry, Murray, Derek, Olah, Chris, Schuster, Mike, Shlens, Jonathon, Steiner, Benoit, Sutskever, Ilya, Talwar, Kunal, Tucker, Paul, Vanhoucke, Vincent, Vasudevan, Vijay, Viégas, Fernanda, Vinyals, Oriol, Warden, Pete, Wattenberg, Martin, Wicke, Martin, Yu, Yuan, Zheng, Xiaoqiang, 2015. TensorFlow: Large-scale machine learning on heterogeneous systems.

Andersson, Tommy, Nowak, Dimitri, Johnson, Tomas, Mark, Andreas, Edelvik, Fredrik, Küfer, Karl-Heinz, 2018. Multiobjective optimization of a heat-sink design using the sandwiching algorithm and an immersed boundary conjugate heat transfer solver. J. Heat Transfer 140 (10).

Asprion, Norbert, Bortz, Michael, 2018. Process modeling, simulation and optimization: From single solutions to a multitude of solutions to support decision making. Chem. Ing. Tech. 90 (11), 1727–1738.

Beck, Andrea D., Flad, David G., Munz, Claus-Dieter, 2018. Deep neural networks for data-driven turbulence models. arXiv preprint arXiv:1806.04482.

Bishop, Christopher M., 2006. Pattern Recognition and Machine Learning. Springer.

Bortz, Michael, Burger, Jakob, Asprion, Norbert, Blagov, Sergej, Böttcher, Roger, Nowak, U., Scheithauer, A., Welke, Richard, Küfer, K.-H., Hasse, Hans, 2014. Multi-criteria optimization in chemical process design and decision support by navigation on Pareto sets. Comput. Chem. Eng. 60, 354–363.

Burkardt, John, sobol C++, Original FORTRAN77 version by Bennett Fox; C++ version by John Burkardt, https://people.sc.fsu.edu/~jburkardt/cpp_src/sobol/sobol.html.

Chen, Tianqi, Guestrin, Carlos, 2016. XGBoost: A scalable tree boosting system. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. KDD '16, ACM, New York, NY, USA, pp. 785–794. http://dx.doi.org/10.1145/2939672.2939785.

Dawson-Elli, Neal, Lee, Seong Beom, Pathak, Manan, Mitra, Kishalay, Subramanian, Venkat R., 2018. Data science approaches for electrochemical engineers: An introduction through surrogate model development for lithium-ion batteries. J. Electrochem. Soc. 165 (2), A1–A15. http://dx.doi.org/10.1149/2.1391714jes.

Farimani, Amir Barati, Gomes, Joseph, Pande, Vijay S., 2017. Deep learning the physics of transport phenomena. arXiv preprint arXiv:1709.02432.

Forrester, A.I.J., Sobester, A., Keane, A.J., 2008. Engineering Design Via Surrogate Modelling: A Practical Guide. John Wiley & Sons.

Forte, Esther, Jirasek, Fabian, Bortz, Michael, Burger, Jakob, Vrabec, Jadran, Hasse, Hans, 2019. Digitalization in thermodynamics. Chem. Ing. Tech. 91 (3), 201–214.

Fraunhofer Chalmers Research Centre for Industrial Mathematics, 0000. IPS IBOFlow, http://www.fcc.chalmers.se/software/ips/iboflow/.

Géron, Aurélien, 2019. Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFLow: Concepts, Tools, and Techniques To Build Intelligent Systems, second ed. O'Reilly Media.

Heese, Raoul, Walczak, Michał, Seidel, Tobias, Asprion, Norbert, Bortz, Michael, 2019. Optimized data exploration applied to the simulation of a chemical process. Comput. Chem. Eng. 124, 326–342.

Huang, Qirui, Ding, Huan, Razmjooy, Navid, 2023. Optimal deep learning neural network using ISSA for diagnosing the oral cancer. Biomed. Signal Process. Control 84, 104749.

Johnson, Steven G., The NLopt nonlinear-optimization package, http://github.com/stevengj/nlopt.

Koziel, Slawomir, Leifsson, Leifur, 2013. Surrogate-Based Modeling and Optimization. Springer New York, New York, NY, http://dx.doi.org/10.1007/978-1-4614-7551-4.

Ludl, Patrick Otto, Heese, Raoul, Höller, Johannes, Asprion, Norbert, Bortz, Michael, 2022. Using machine learning models to explore the solution space of large nonlinear systems underlying flowsheet simulations with constraints. Front. Chem. Sci. Eng. 16 (2), 183–197.

Mark, Andreas, Svenning, Erik, Edelvik, Fredrik, 2013. An immersed boundary method for simulation of flow with heat transfer. Int. J. Heat Mass Transfer 56 (1–2), 424–435.

Mark, Andreas, van Wachem, Berend G.M., 2008. Derivation and validation of a novel implicit second-order accurate immersed boundary method. J. Comput. Phys. 227 (13), 6660–6680. http://dx.doi.org/10.1016/j.jcp.2008.03.031, https://www.sciencedirect.com/science/article/pii/S0021999108001770.

Mir, Mahdi, Dayyani, Mohammad, Sutikno, Tole, Mohammadi Zanjireh, Morteza, Razmjooy, Navid, 2020. Employing a Gaussian particle swarm optimization method for tuning multi input multi output-fuzzy system as an integrated controller of a micro-grid with stability analysis. Comput. Intell. 36 (1), 225–258.

Nowak, Dimitri, Johnson, Tomas, Mark, Andreas, Ireholm, Charlotte, Pezzotti, Fabio, Erhardsson, Lars, Ståhlberg, Daniel, Edelvik, Fredrik, Küfer, Karl-Heinz, 2021. Multicriteria optimization of an oven with a novel ϵ-constraint-based sandwiching method. J. Heat Transfer 143 (1), 012101.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E., 2011. Scikit-learn: Machine learning in python. J. Mach. Learn. Res. 12, 2825–2830.

Powell, M.J.D., 1994a. A direct search optimization method that models the objective and constraint functions by linear interpolation. In: Gomez, S., Hennart, J.-P. (Eds.), Advances in Optimization and Numerical Analysis. In: Mathematics and Its Applications, 275, Springer, pp. 51–67. http://dx.doi.org/10.1007/978-94-015-8330-5_4.

Powell, Michael J.D., 1998. Direct search algorithms for optimization calculations. Acta Numer. 7, 287–336.

Rasmussen, Carl Edward, 2003. Gaussian processes in machine learning. In: Summer School on Machine Learning. Springer, pp. 63–71.

Sharma, Haresh Kumar, Majumder, Saibal, Biswas, Arindam, Prentkovskis, Olegas, Kar, Samarjit, Skačkauskas, Paulius, 2022. A study on decision-making of the Indian railways reservation system during COVID-19. J. Adv. Transp. 2022, 1–10.

Thuerey, Nils, Weißenow, Konstantin, Prantl, Lukas, Hu, Xiangyu, 2020. Deep learning methods for Reynolds-averaged Navier–Stokes simulations of airfoil flows. AIAA J. 58 (1), 25–36.

Tong, Hao, Huang, Changwu, Minku, Leandro L., Yao, Xin, 2021. Surrogate models in evolutionary single-objective optimization: A new taxonomy and experimental study. Inform. Sci. 562, 414–437. http://dx.doi.org/10.1016/j.ins.2021.03.002.

Virtanen, Pauli, Gommers, Ralf, Oliphant, Travis E., Haberland, Matt, Reddy, Tyler, Cournapeau, David, Burovski, Evgeni, Peterson, Pearu, Weckesser, Warren, Bright, Jonathan, van der Walt, Stéfan J., Brett, Matthew, Wilson, Joshua, Millman, K. Jarrod, Mayorov, Nikolay, Nelson, Andrew R.J., Jones, Eric, Kern, Robert, Larson, Eric, Carey, C.J., Polat, İlhan, Feng, Yu, Moore, Eric W., VanderPlas, Jake, Laxalde, Denis, Perktold, Josef, Cimrman, Robert, Henriksen, Ian, Quintero, E.A., Harris, Charles R., Archibald, Anne M., Ribeiro, Antônio H., Pedregosa, Fabian, van Mulbregt, Paul, SciPy 1.0 Contributors, 2020. SciPy 1.0: Fundamental algorithms for scientific computing in Python. Nature Methods 17, 261–272. http://dx.doi.org/10.1038/s41592-019-0686-2.

Wiewel, Steffen, Becher, Moritz, Thuerey, Nils, 2019. Latent space physics: Towards learning the temporal evolution of fluid flow. Comput. Graph. Forum 38 (2), 71–82.