Modular surrogate-based optimization framework for expensive computational simulations

Panzo, Antonín; Chen, Boyang

Publication date
2020

Document Version
Final published version

Published in
CEUR Workshop Proceedings

Modular Surrogate-based Optimization Framework for Expensive Computational Simulations

Antonín Panzo[1,2], Boyang Chen[1]

[1] Delft University of Technology, Department of Aerospace Structures and Materials
Kluyverweg 1, 2629 HS Delft, The Netherlands
B.Chen-2@tudelft.nl
[2] Škoda Transportation, Research & Development Department
Bucharova 1314/8, 158 00 Praha 5, Czech Republic

Abstract: In practical applications, the use of computational modeling has been industry-wide adopted to speed up product development as well as reduce physical testing costs. Such models of complex or large systems are, however, often computationally expensive, hence solution times of hours or more are not uncommon. Additionally, as these models are typically evaluated using black-box solvers, the direct study of relations between design parameters renders demanding in terms of computational time and provides poor engineering insight and understanding.

To address this, a modular framework integrating computation automation with the use of surrogate-based modeling, optimization and visualization techniques is presented. The framework is built in the Python programming language. Its use is illustrated on a study of the side impact response of a car body using an artificial neural network as a surrogate together with the NSGA-III genetic algorithm for optimization.

1 Introduction

Nowadays, solving of many engineering problems, such as the development of transport vehicles, involves the use of computationally expensive simulations, such as computational fluid dynamics (CFD), the finite element method (FEM), or others, as well as their combinations, e.g. the fluid-structure interaction (FSI). These will be further addressed as *simulations*. In practice, typically, such simulations suffer from two major drawbacks. Firstly, even thought the computational power available to an engineer for running such simulations has been gradually increasing over years, in practice the requirements on the model accuracy had increased as well [48], and as such, there is still a large quantity of simulations that take hours, days and occasionally even more, just to evaluate a single design [43]. Secondly, they are often of a black-box nature, meaning that given an input $\vec{X}$, an output $\vec{Y}$ is returned to the user without an accompanying explicit relation between the two [41].

Within a practically infinite space of design choices for the system under study, the task of an engineer is to identify the most influential design parameters and understand their degree of influence. Through the use of simulations, often the next goal is to meet a set of threshold requirements on selected targets, while minimizing or maximizing a certain property of the overall system, such as cost, weight, etc. Mathematically, this can be formulated as a multi-objective optimization problem with the goal to:

$$\text{minimize } f_m(\vec{X}) \quad m = 1,...,P \tag{1}$$

subject to

$$g_j(\vec{X}) \leq 0 \quad j = 1,...,Q \tag{2}$$

$$h_k(\vec{X}) = 0 \quad k = 1,...,R \tag{3}$$

$$X_i^L \leq X_i \leq X_i^U \quad i = 1,...,n_{in} \tag{4}$$

where f are the objective functions, g and h are the inequality and equality constraints, respectively, and $\vec{X}$ is the vector of design variables in the design search space bounded by X_i^L and X_i^U from the lower and upper side, respectively [47]. In case of multi-objective optimization when $m \geq 2$, the notion of singular optimality needs to be augmented as there is typically not a single solution $\vec{X}^*$ minimizing all of the objectives of Equation 1 at the same time. In such cases, the concept of so-called Pareto optimality is considered [36], which can be loosely described as: "for each solution that is contained in the Pareto set, one can only improve one objective by accepting a trade-off in at least one other objective" [39].

Due to said high computational costs and the black-box nature of simulations, multi-objective parametric studies and optimizations often become too lengthy and impractical for actual application [42, 41]. The lack of derivatives required for derivative-based optimization methods, such as in Liu and Reynolds [35] or Peitz and Dellnitz [39], can be addressed by the use of derivative-free optimization (DFO) methods, such as genetic, evolutionary or swarm algorithms [27]. However, these methods compensate for the lack of gradients by evaluating on larger sets of candidate solutions, which leaves the issue of high computational cost unresolved. One way to resolve that

is by using a model of lower fidelity requiring less computational resources [33]. However, the accuracy of such lower fidelity models can become unacceptably low. As an alternative, a surrogate-based approach can be adopted [20, 5, 12].

A surrogate model, also addressed as a metamodel, is a mathematical model that upon training approximates the response of the original model based on a sample set from the design space [31]. Such computationally cheap surrogate can then be called during the optimization defined by Equation 1-4 instead of the original, expensive simulation. In addition, the use of a surrogate model opens up several new opportunities, which include:

1. using the surrogate as a substructure of a larger model [34]
2. cheap numerical evaluation of gradients using the surrogate
3. repeating the optimization for various formulations of the problem (within the bounds of validity of the surrogate) - compare various formulations of the same car side impact problem: Youn et al. [54], Guo, Wang, and Wang [26], and Tanabe and Ishibuchi [46]
4. design exploration through sensitivity studies and visualization [49, 38]

On the other hand, a disadvantage is the added complexity of the overall task due to introduction of extra tuning parameters related to the surrogate model and its training.

Due to the often black-box nature of simulations, it is not possible to a-priori quantify the sample set size required for an accurate training of the surrogate [4]. Therefore, for a purely optimization-focused approach, in general it is not guaranteed that such surrogate-based approach is indeed computationally cheaper than optimizing directly using the original model. Nevertheless, it has been established that for budget-limited tasks with expensive simulations, it is indeed the case [44, 24, 38]. This is the case especially for multi-objective optimization, when the exploration of the full Pareto front is required. Confirming results of an empirical comparison can be found e.g. in Voutchkov and Keane [50] on benchmark problems and in Bessa and Pellegrino [3] on a practical problem of the design of ultra-thin carbon fiber deployable shells.

Example uses of surrogate-based optimization (SBO) from the area of structural optimization include the design of 3D weaving composite stiffened panels by Fu, Ricci, and Bisagni [21], hypersonic vehicle's metallic thermal protection by Guo et al. [25] or a wellhead connector for a subsea Christmas tree by Zeng et al. [55]. The SBO methodology has been already integrated into various commercial software packages, such as OptiSLang[1] or HEEDS[2]. This can be often practical, but not always. Firstly, the user must rely only on the available capabilities of such software packages, which can sometimes be insufficient for the application at hand. Secondly, especially for individuals or smaller businesses, the license price for such packages can be too expensive, thus inaccessible. Last but not least, for research purposes, commercial solutions often offer only a limited insight into the used methods and source code, leading to limited customabilizity, a feature often required for research.

In this paper, the core of a developed Python open-source SBO framework is presented. In Section 2, the different elements of the proposed framework are presented. Next, in Section 3, the use of the framework is demonstrated on a benchmark problem. Finally, the capabilities of the framework are summarized in Section 4 together with an outline of suggested further developments.

2 Proposed method

This framework integrates elements of surrogate-based modeling and automated model selection together with derivative-free optimization. One of the driving ideas behind it is modularity, such that it could be readily customized for the user's particular needs. Upon completion of the master thesis project, the framework will be made publicly available at `github.com/apanzo/optimization`.

The top-level illustration of the framework is shown in Figure 1. As a starting point, the simulation is prepared, and the framework settings, such as the selection of a surrogate model and optimization algorithm, are set. Both the use of a surrogate and performing optimization are optional. The surrogate loop contains sample selection, model evaluation, results storage and retrieval, optimization and training of the surrogate, and finally accessing its convergence. The optimization loop consists of the actual optimization, and, in case that a surrogate model is used, a verification of the obtained results against the original model. In the following subsections, key parts of the framework are discussed in the order from optimization start to end as indicated in Figure 1.

2.1 Sampling strategy

Considering the expensiveness of the simulations addressed by this framework, one of the keys of success is the smart selection of the sample points within the explored design space, such that the surrogate model obtains a sufficiently representative sample to be trained on. For this purpose, simpler sampling strategies such as the grid search or random search are not suitable. The former is expensive as it does not benefit from sampling in multiple dimensions, as the sample projection on each of the input's

[1] ANSYS OptiSLang, `https://www.dynardo.de/en/software/optislang/ansys-optislang.html` [Accessed on 03/08/2020]

[2] HEEDS MDO, `https://www.redcedartech.com/index.php/solutions/heeds-software` [Accessed on 03/08/2020]

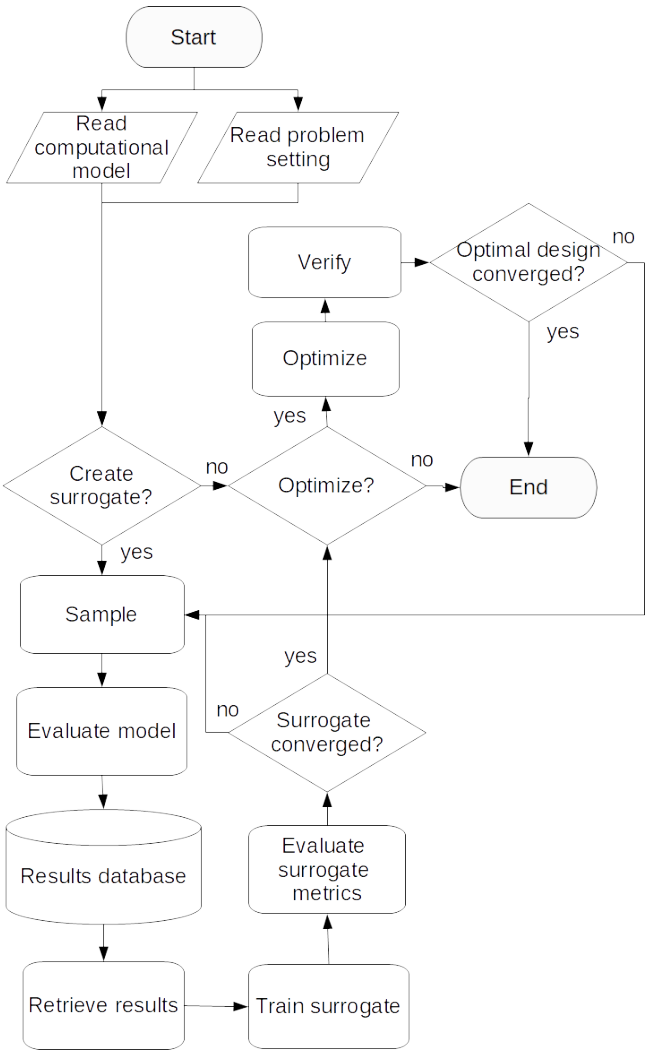

Figure 1: Diagram of the proposed framework

axes is independent of the number of inputs dimensions, and in the case of re-sampling with a finer sample, either the new sample discards all of the previous samples, or the re-sampling is to be done on an integer multiple of samples per input dimension of the current sample. The random search does not suffer from those, but its weakness lies in its poor space-filling property, thus a large amount of samples is required to generate a sufficiently representative sample.

As a baseline within this framework, quasi-random sampling strategies are implemented. In particular, the two available sampling methods are the cell centered Latin hypercube sampling (LHS) [14, 18] and the original Halton sequence sampling [13]. The principal logic behind the LHS method is that it is a sparse grid method where the projection of the sample on each of the input's axes contains only one sample per uniform interval. As such, the sample size is significantly reduced compared to the full grid. On the contrary, the disadvantage of the poor resampling pertains. This is not the case for the Halton sequence, which is a multi-dimensional version of the van Der Corput sequence. It is defined as

$$P = \{\phi_{q_1}(Z), ..., \phi_{q_n}(Z)\}$$

with bases $q_1, ..., q_n$ that are mutually coprime and in practice taken as the first s primes [13, 22]. $\phi_q(Z)$, the inverse radix number, is obtained from an integer

$$Z = \sum_{i=0}^{r} a_i q^i$$

where q is the basis and the largest exponent is $r = \text{int}\left(\frac{\ln Z}{\ln q}\right)$, and $0 \leq a_i < q$ are the coefficients [13, 11], as

$$\phi_q(Z) = \sum_{i=0}^{r} a_i q^{-(i-1)}$$

The advantage is that each point of the sequence is generated independently of the previous point. An example is shown in Figure 2. It is noted however that the original method is suitable only up to 8 input dimensions, as in higher dimensions, spurious correlations between the inputs occur, considerably deteriorating the sample's quality [13]. This can be leveraged by using its modified versions that address this issue [19, 37].

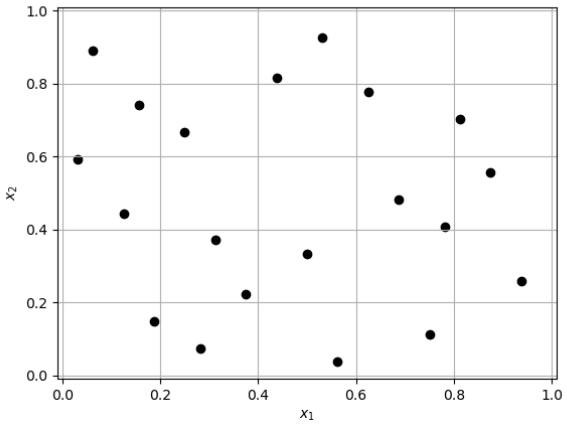

Figure 2: Example of a Halton sample on the unit range

To take it one step further, considering that these, so called static sampling methods, only take into account the space-filling criteria, that is the input's quality, and not the nature of the response, that is the output's quality, an adaptive sampling method that does so is implemented as well. As an example, if S is the full design space and the response of a model is flat everywhere apart from the region $D \in S$, where there is a valley in the response, only a few samples are required outside D, while in D more samples are required to train the surrogate. The selected method is from Eason and Cremaschi [17], which determines the new samples based on a balance of the variance in predictions of the K models from cross-validation, explained

further in subsection 2.4, for exploitation and the distance to the nearest sample for exploration, each normalized by its maximum. The new sample is chosen where the sum of the two criteria is the largest.

2.2 External evaluator

For the practical use of the framework, it has to be able to interact with external solvers that evaluate the simulations. This independence is one of the main strong points of the framework, as it allows to integrate results from different solvers. As an example, for a FSI study, different FEM and CFD solvers can be used and their results integrated within this framework.

The implementation of different evaluators is application dependent, therefore the interaction with each new solver has to be customized according to its specific interface. In the course of development of this framework, a custom ANSYS evaluator has been integrated. For its use, the procedure is to firstly build a parametric model inside the ANSYS Workbench environment[3], with the input and output parameters defined. These parameters are passed over by name to the framework together with the path to the prepared model. The rest of the interaction is automated. After determining the sample points, firstly the framework checks that there are available free licenses for carrying out the computations. If there are available licenses, a request is sent to ANSYS Workbench to evaluate the selected sample points. Once all the samples have been evaluated, the framework retrieves the defined output parameters and integrates them within its internal database.

2.3 Surrogate model

Among others, the most commonly used surrogate-modeling techniques are kriging, radial basis functions (RBF), artificial neural networks (ANN) or support vector machines (SVM) [17, 12]. The former two are integrated within the framework using the Surrogate Modeling Toolbox (SMT) library [10] natively, while the ANN is integrated as a custom class into SMT using the TensorFlow library [1]. In the course of the reported research, the ANN has been used due to previous experience in the research group, therefore it is elaborated upon closer in this section.

In the general sense, ANNs are one of the subclasses of the broader field of machine learning (ML), which in turn is a subclass of artificial intelligence (AI). Most commonly, the ANNs are trained using a form of supervised learning, that is by providing both the input and output data during the training process. They can be used both for classification, that is categorization of the input content into different sub-classes with common features, as well as regression. For the purpose of optimization, regression is typically used and the task of learning is to obtain the unknown relation between input and output data.

The core concept of an ANN is a single neuron. Inspired by the function of a real neuron inside the living brain, the mathematical model from input x to output y consists of performing a weighted summation of all the inputs

$$z = \sum_{i=1}^{n_{in}} w_i x_i \tag{5}$$

and applying a non-linear transformation

$$y = g(z)$$

with g the so-called activation function. Common activation functions are the logistic sigmoid

$$g(z) = \frac{1}{1+e^{-(z)}},$$

rectified linear unit (ReLU)

$$g(z) = \max(0, z)$$

[23], or Swish

$$g(z) = z\frac{1}{1+e^{-(z)}}$$

[40] functions.

To provide a flexible range of outputs based on the neuron's activations, the neurons are organized into layers. The typical neural network architecture for regression is the multi-layer perceptron (MLP), consisting of the first and last layers containing the amount of neurons equal to the number of input and output dimensions, respectively, and a selected number of intermediate, hidden layers, which can each contain an arbitrary amount of neurons. Building upon the Kolmogorov's theorem [32], Hecht-Nielsen, Ne, and Corporation [28] proved that using specific activation functions, even an ANN with a single hidden layer of neurons can approximate any continuous function. However, it is not guaranteed that such ANN can actually *learn* such representation [6], and in practice, using multiple hidden layers of neurons can simplify the learning process [29].

The training of the network is equivalent to the determination of appropriate weights w from Equation 5 for each neuron, such that the desired mapping $x \rightarrow y$ from the input data to the output data is obtained. The common training method is the backpropagation method, where the prediction error E after the forward pass is propagated back to the contribution to that error from each neuron, and their weights are updated in each training iteration. This is schematically illustrated in Figure 3.

Independent of the selected surrogate model, to obtain an accurate model, a step of crucial importance is the selection of the model's hyperparameters, that is the parameters that are not directly learned through the training process. This is specific for each of the surrogate models. As an example, in the case of the ANN, these are

[3] ANSYS Workbench Platform, `https://www.ansys.com/-/media/Ansys/corporate/resourcelibrary/brochure/workbench-platform-121.pdf` [Accessed on 05/08/2020]

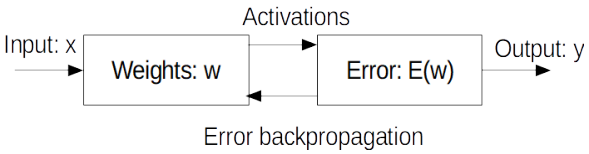

Figure 3: Illustration of backpropragation

the amount of hidden layers and neurons in each of them, the learning rate parameter, regularization parameters or the activation function choice. For this purpose, model-specific methods, such as the neuron pruning [56], where the neurons of low significance are removed, as well as model-independent, such as a random search, Bayesian optimization or meta-heuristic optimization approaches, can be used.

At this stage, a random search and Bayesian optimization (BO) [45] are included for the ANN using Keras Tuner[4]. A hypermodel is defined with default hyperparameters stored in a configuration file. Within, these hyperparameters can be either changed manually, or specified to be subject of automatic tuning. For ANNs, the tunable hyperparameters are: number of hidden layers, number of neurons in each layer, initial learning rate, activation function and the regularization parameter. The current implementation of the BO allows only setting a fixed optimization budget, therefore the selected option is to start with an initial random sample of 3x times the number of optimized hyperparameters, and stop at 10x times the number. The acquisition function is the upper-confidence bound [45]. The frequency of hyperparameter optimization is set as at the first iteration, and then every time the sample set size increases by 50%.

2.4 Surrogate's convergence

A metric of the surrogate's accuracy of choice is to be tracked to determine the convergence of the sample set size. In our approach, the tracked metric is the mean absolute error (MAE)

$$E_{MAE} = \frac{\sum_{i=1}^{N} |y_{pred} - y_{train}|}{N}$$

where y_{pred} is the prediction of the surrogate, y_{train} is the training value and N is the total number of samples. Since the amount of data is relatively low, to perform validation of the trained surrogate model, the K-fold cross-validation approach is adopted [9]. In each iteration, the data is split into a set of K-folds, and K surrogates are trained each time leaving out a different fold of the data. This holdout set is used to calculate the generalization error of the model on previously unseen data. With increasing size of the data set, leaving out the holdout set will affect the surrogate's prediction less and less, so the average E_{MAE} over the K surrogates will decrease. The resampling loop is terminated when a satisfactory level of E_{MAE} is attained.

[4]keras-tuner, `https://github.com/keras-team/keras-tuner` [Accessed on 06/08/2020]

2.5 Optimization algorithm

Once the surrogate model has been trained to a desired level of accuracy, the optimization run can be started. Within this framework, various population-based optimization algorithms are provided by the Pymoo library [7]. Pymoo provides implementations of a series of population-based algorithms, including:

- Differential Evolution
- Genetic Algorithm
- BRKGA
- Nelder Mead
- Pattern Search
- CMAES
- NSGA-II and NSGA-III
- RNSGA-II and RNSGA-III
- UNSGA-III
- MOEA/D.

Among those, the most commonly used in the industry is the NSGA-II method [15] and its updated NSGA-III version [16] adapted for the many-objective cases of $m > 2$, where an even exploration of the Pareto-front is accomplished by the use of reference directions. A generic scheme of a genetic algorithm is shown in Figure 4. The idea behind is similar to Darwinian evolution, that is that once the initial population of solutions is generated (Step 1), the fitness of the individuals is evaluated (Step 2) and only the fittest survive in each generation (Step 3). From those, a selection parent selection takes places (Step 4), from which new offsprings are generated through crossover (Step 5). Additionally, a random mutation is applied to their genome (Step 6) to maintain diversity within the population and explore the design space better. At the end of each iteration, the convergence criterion is evaluated, and either the optimization stops, or continues with a new generation [52].

The available termination criteria are the maximum:

- number of evaluations
- number of generations
- time
- design space tolerance
- objective space tolerance.

Due to the computational cheapness of the surrogate, the optimization process is not limited by the computational cost of evaluating the simulation, such as the maximal number of its evaluations. Therefore, the selection of the

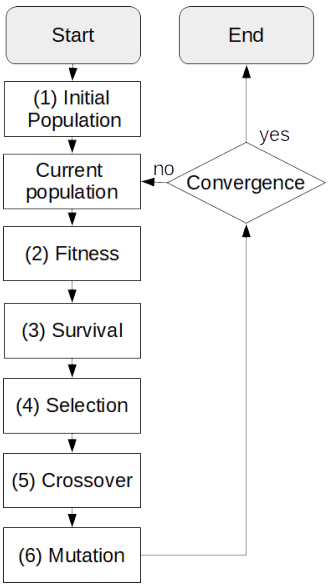

Figure 4: A generic flowchart of a genetic algorithm

termination criterion can focus purely on the convergence of the optimization. In this case the last two criteria are of main importance. A threshold tolerance on the metrics tracking the evolution of solutions in the design or objective space is defined, and if the solutions in the specified amount of last iterations do not improve over this threshold, the optimization is terminated. For robustness purposes, also the maximal amount of evaluations or generations can be specified, e.g. for cases when the optimal solutions oscillate around some value, and thus in the defined window would otherwise never have converged.

It is also possible to enter the optimization directly using the original model without constructing the surrogate. In this case however, for an expensive simulation, the choices on the optimization algorithm and its components, especially the population size and number of generated offsprings in each generation, have to be more carefully selected with respect to the number of required simulations.

2.6 Result verification

Once an optimum (set) has been found, given that this optimum has been found based on the surrogate, it is important to verify it's accuracy. In particular, it could be that the algorithm has found an optimum that lies in a spurious extremum caused by the surrogate's inaccuracy which does not pertain to the original model. Therefore, by submitting a sample from the Pareto optimal set back to an evaluation using the original model, it can be verified whether the obtained solutions indeed pertain to the original model. If the mean or maximal error within this verification exceeds a pre-defined tolerance, the surrogate is retrained on an enlarged sample, and the optimization occurs again, as seen in Figure 1.

2.7 Implementation notes

As an important implementation note, the data is normalized within the internal data flow. This is done such that common settings can be used across different problems, as well as for a single problem to treat inputs and outputs of different magnitudes comparably. Not doing so would mean that for example when the prediction error is summed, the weight in kilograms would completely dominate deformation in millimeters, which is undesirable. Therefore, all the inputs and outputs are normalized by dividing each quantity with its absolute maximal value. In such a way, all the data is guaranteed to lie within the [-1,1] range, while maintaining its sign, conforming to the constraint violation formulation in Equation 2.

3 Illustrative example

As an illustrative example of a practical problem, the results on a parametrized nonlinear FEM simulation of a car side impact problem [30] ran using the proposed framework are presented in this section.

3.1 Model description

The design space is defined by 7 input parameters

$$\begin{aligned} 0.5 \leq x_1 &\leq 1.5 \\ 0.45 \leq x_2 &\leq 1.35 \\ 0.5 \leq x_3 &\leq 1.5 \\ 0.5 \leq x_4 &\leq 1.5 \\ 0.875 \leq x_5 &\leq 2.625 \\ 0.4 \leq x_6 &\leq 1.2 \\ 0.4 \leq x_7 &\leq 1.2 \end{aligned}$$

that represent the thicknesses of structural members such as B-pillars, beams or the floor. A single solution was, at the time of the problem's publication, reported to take about 20h [54]. Even thought this time has likely reduced with today's hardware, it is still a good illustrative example of surrogate-based modeling.

For the purpose of benchmarking, the response has been parametrized by analytical expressions, which are presented hereafter [30]. The 3 objectives of the design study are to minimize the structural weight, impact force on the passenger and average velocity of the side members that absorbs the impact load, represented as

$$\begin{aligned} f_1(\vec{x}) \equiv W = 1.98 + 4.9x_1 + 6.67x_2 + 6.98x_3 + 4.01x_4 \\ +1.78x_5 + 0.00001x_6 + 2.73x_7 \end{aligned}$$

$$f_2(\vec{x}) = F$$

$$f_3(\vec{x}) = 0.5(V_{MBP} + V_{FD})$$

respectively. In addition, there are 10 constraints that take into account criteria such as the abdomen load

$$g_1(\vec{x}) = 1.16 - 0.3717x_2x_4 - 0.0092928x_3 \leq 1$$

viscous criteria

$$g_2(\vec{x}) = 0.261 - 0.0159x_1x_2 - 0.06486x_1 - 0.019x_2x_7 + 0.0144x_3x_5 + 0.0154464x_6 \leq 0.32$$

$$g_3(\vec{x}) = 0.214 + 0.00817x_5 - 0.045195x_1 - 0.0135168x_1 + 0.03099x_2x_6 - 0.018x_2x_7 + 0.007176x_3 + 0.023232x_3 - 0.00364x_5x_6 - 0.018x_22 \leq 0.32$$

$$g_4(\vec{x}) = 0.74 - 0.61x_2 - 0.031296x_3 - 0.031872x_7 + 0.227x_2^2 \leq 0.32$$

rib deflections

$$g_5(\vec{x}) = 28.98 + 3.818x_3 - 4.2x1x_2 + 1.27296x_6 - 2.68065x_7 \leq 32$$

$$g_6(\vec{x}) = 33.86 + 2.95x_3 - 5.057x_1x_2 - 3.795x_2 - 3.4431x_7 + 1.45728 \leq 32$$

$$g_7(\vec{x}) = 46.36 - 9.9x_2 - 4.4505x_1 \leq 32$$

pubic symphysis force

$$g_8(\vec{x}) \equiv F = 4.72 - 0.5x_4 - 0.19x_2x_3 \leq 4$$

and velocities of structural members at impact

$$g_9(\vec{x}) \equiv V_{MBP} = 10.58 - 0.674x_1x_2 - 0.67275x_2 \leq 9.9$$

$$g_{10}(\vec{x}) \equiv V_{FD} = 16.45 - 0.489x_3x_7 - 0.843x_5x_6 \leq 15.7$$

[54].

3.2 Setup and user interaction

The format of the user input using the JSON formatting[5] is shown below:

```
{
 "data": {
 "problem": "carside",
 "evaluator": "benchmark",
 "sampling": "halton",
 "default_sample_coef": 10,
 "resampling": "geometric",
 "resampling_param": 1.2,
 "adaptive": null,
 "convergence": "mae",
 "convergence_relative": false,
 "convergence_limit": 0.03
 },
 "surrogate": {
 "surrogate": "ann",
 "reoptimization_ratio": 1.5,
 "validation": "kfold",
 "validation_param": 5,
 "append_verification": false
 },
 "optimization": {
 "algorithm": "nsga3",
 "termination": "f_tol",
 "termination_val": [0.001,5,5,150,
    null],
 "constrained": true,
 "error_limit": 5,
 "error": "mean"
 }
}
```

To translate it, the carside problem is evaluated from the benchmark problems set. The initial sample is 10x the number of input dimensions, thus 70, and in every additional iteration, the sample set is increased by 20%. The convergence metric for the model is the MAE metric, with a maximal value of 0.03. The ANN is the used surrogate model and its performance metric is calculated using 5-fold cross validation. The hyperparameters are re-optimized when the amount of samples increases by 50%. The optimization uses the NSGA-III algorithm and its termination is based upon the objective space tolerance, which terminates, if there is no change by more than 0.001 in the tracked metrics: changes of the objective functions, inverted generational distance (IGD) [2] and the nadir point, over the last 5 generations. This convergence check is performed every 5 generations. Additionally, a maximum of 150 generations is allowed during a single optimization run. The verification mean error limit is set as 5%.

Table 1: ANN hyperparameters specification[6]

Optimizer	adam
Activation function	relu
Learning rate	0.01
Regularization	None
Weight initialization	he_normal
Bias initialization	zeros
Max epochs	1000
Loss	mse
Early stopping delta	0.0001
Early stopping patience	50

Additionally, not present in the input file, the ANN's hyperparameters are defined as in Table 1. The output layer function is linear. The number of hidden layers and neurons per hidden layer are optimized using BO between [2,10] and [6,200], respectively. For optimization,

[5] JSON, https://www.json.org/ [Accessed on 07/08/2020]

[6] Module: tf.keras | TensorFlow Core v2.2.0-rc1, https://www.tensorflow.org/versions/r2.2/api_docs/python/tf/keras [Accessed on 07/08/2020]

the algorithm is initiated with default settings[7]. The reference directions are obtained using an energy method from Blank et al. [8] and their number, which also defines the population size, is set at 30x the number of objectives, that is 90 in this case. The amount of generated offsprings is equal to the population size.

3.3 Results

In Figure 5, the comparison of training data with the prediction of the final trained ANN is shown.

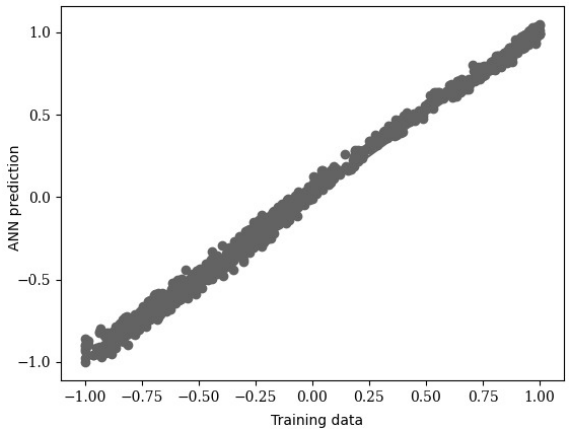

Figure 5: Comparison of the ANN's trained prediction against the training output data

The sample set converged after 3 surrogate training iterations, that is with 101 samples evaluated. This is just over the defined population size in one generation of the genetic algorithm. The optimization took 70 generations to converge, which amount to 9000 evaluations of the model. That as a factor of 90x more than the required surrogate training sample size, confirming the advantage of using the surrogate. It is noted however, that thus far the selected optimization parameters are determined empirically on a trial-and-error basis.

As a validation of the result, the numerical values of the Pareto optimal solutions are unfortunately not presented in Jain and Deb [30] nor other paper, however, the qualitative comparison with their solutions in the objective space as shown in Figure 6 indicates that the ranges of all 3 objectives fall within nearly identical ranges. The final mean verification error between the Pareto optimal set and the original model, as discussed in subsection 2.6, was 2.10%.

4 Conclusion

In this paper, a working core example of a modular surrogate-based optimization framework was presented. A more detailed elaboration on the ANN surrogate was

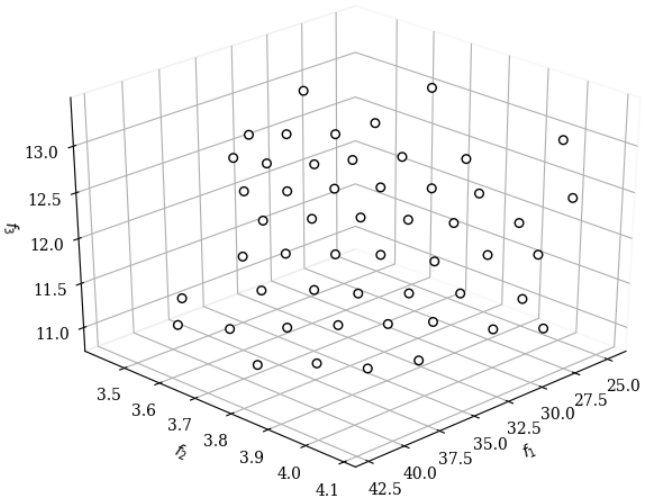

Figure 6: Scatter plot of the Pareto optimal set

drawn, together with a top-level description of each of the key submodules. An illustrative working example was shown, that demonstrated the interaction and use of the framework together with the efficiency of the surrogate based approach. Compared to commercial solutions, the modularity makes it suitable for research purposes, while the open-source nature is beneficiary for small companies or users with advanced customization needs.

On top of the correct implementation of the methods, the main challenges related to its use include the selection of a particular surrogate and optimization algorithm or the termination, model selection and convergence criteria. On a fully general scale, the no free lunch (NFL) theorems [51] prove that an universal optimal choice is impossible. However, if the set of problems is limited to an expected class of practical engineering problems, at least a good baseline starting point is possible [53]. Thus, the main aim of the follow-up work is to perform testing on a wide range of problems with a range of framework setups to thoroughly establish the influence of the tunable parameters of the surrogate model as well as of the optimization algorithm.

Further suggestions include a quantitative comparative study of the performance of different proposed adaptive sampling methods, which show the largest potential on the overall SBO performance, as the expensive simulations remains main bottleneck overall. Beyond that, software-wise, additional suggested features include a graphical user interface (GUI), inclusion of more surrogate models from SMT, such as the RBF, extending the amount of supported evaluators e.g. for Abaqus, and others. Finally, for practice oriented research, a composite layup optimization study in terms of the number of plies and their ply angles is an interesting area of possible application of this framework within the field of advanced structures.

[7] pymoo - NSGA-III, `http://pymoo.org/algorithms/nsga3.html` [Accessed on 07/08/2020]

References

[1] Martin Abadi et al. "TensorFlow: A system for large-scale machine learning". In: *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*. 2016, pp. 265–283. URL: https://www.usenix.org/system/files/conference/osdi16/osdi16-abadi.pdf.

[2] Charles Audet et al. "Performance indicators in multiobjective optimization". working paper or preprint. Feb. 2020. URL: https://hal.archives-ouvertes.fr/hal-02464750.

[3] M. A. Bessa and S. Pellegrino. "Design of ultra-thin shell structures in the stochastic post-buckling range using Bayesian machine learning and optimization". In: *International Journal of Solids and Structures* 139-140 (May 2018), pp. 174–188. DOI: 10.1016/j.ijsolstr.2018.01.035.

[4] M. A. Bessa et al. "A framework for data-driven analysis of materials under uncertainty: Countering the curse of dimensionality". In: *Computer Methods in Applied Mechanics and Engineering* 320 (June 2017), pp. 633–667. DOI: 10.1016/j.cma.2017.03.037.

[5] Atharv Bhosekar and Marianthi Ierapetritou. "Advances in surrogate based modeling, feasibility analysis, and optimization: A review". In: *Computers & Chemical Engineering* 108 (Jan. 2018), pp. 250–267. DOI: 10.1016/j.compchemeng.2017.09.017.

[6] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer-Verlag New York Inc., Aug. 17, 2006. 738 pp. ISBN: 0387310738.

[7] Julian Blank and Kalyanmoy Deb. "pymoo: Multi-objective Optimization in Python". In: *IEEE Access* 8 (2020), pp. 89497–89509. DOI: 10.1109/access.2020.2990567.

[8] Julian Blank et al. "Generating Well-Spaced Points on a Unit Simplex for Evolutionary Many-Objective Optimization". In: *IEEE Transactions on Evolutionary Computation* (2020), pp. 1–1. DOI: 10.1109/tevc.2020.2992387.

[9] Avrim Blum, Adam Kalai, and John Langford. "Beating the Hold-Out: Bounds for K-fold and Progressive Cross-Validation". In: *Proceedings of the twelfth annual conference on Computational learning theory - COLT '99*. ACM Press, 1999. DOI: 10.1145/307400.307439.

[10] Mohamed Amine Bouhlel et al. "A Python surrogate modeling framework with derivatives". In: *Advances in Engineering Software* 135 (Sept. 2019), p. 102662. DOI: 10.1016/j.advengsoft.2019.03.005.

[11] Luis E. Garza Castañón, Saúl Montes de Oca, and Rubén Morales-Menéndez. "An Application of Random and Hammersley Sampling Methods to Iris Recognition". In: *Advances in Applied Artificial Intelligence*. Ed. by M. Ali and R. Dapoigny. Vol. 4031. LNAI. Springer Berlin Heidelberg, 2006, pp. 520–529. DOI: 10.1007/11779568_56.

[12] Kai Cheng et al. "Surrogate-assisted global sensitivity analysis: an overview". In: *Structural and Multidisciplinary Optimization* 61.3 (Jan. 2020), pp. 1187–1213. DOI: 10.1007/s00158-019-02413-5.

[13] H. Chi, M. Mascagni, and T. Warnock. "On the optimal Halton sequence". In: *Mathematics and Computers in Simulation* 70.1 (Sept. 2005), pp. 9–21. DOI: 10.1016/j.matcom.2005.03.004.

[14] Keith Dalbey and George Karystinos. "Fast Generation of Space-filling Latin Hypercube Sample Designs". In: *13th AIAA/ISSMO Multidisciplinary Analysis Optimization Conference*. American Institute of Aeronautics and Astronautics, Sept. 2010. DOI: 10.2514/6.2010-9085.

[15] K. Deb et al. "A fast and elitist multiobjective genetic algorithm: NSGA-II". In: *IEEE Transactions on Evolutionary Computation* 6.2 (Apr. 2002), pp. 182–197. DOI: 10.1109/4235.996017.

[16] Kalyanmoy Deb and Himanshu Jain. "An Evolutionary Many-Objective Optimization Algorithm Using Reference-Point-Based Nondominated Sorting Approach, Part I: Solving Problems With Box Constraints". In: *IEEE Transactions on Evolutionary Computation* 18.4 (Aug. 2014), pp. 577–601. DOI: 10.1109/tevc.2013.2281535.

[17] John Eason and Selen Cremaschi. "Adaptive sequential sampling for surrogate model generation with artificial neural networks". In: *Computers & Chemical Engineering* 68 (Sept. 2014), pp. 220–232. DOI: 10.1016/j.compchemeng.2014.05.021.

[18] Ke Fang, Yuchen Zhou, and Ping Ma. "An adaptive sequential experiment design method for model validation". In: *Chinese Journal of Aeronautics* 33.6 (June 2020), pp. 1661–1672. DOI: 10.1016/j.cja.2019.12.026.

[19] Henri Faure and Christiane Lemieux. "Generalized Halton sequences in 2008". In: vol. 19. 4. Association for Computing Machinery (ACM), Oct. 2009, pp. 1–31. DOI: 10.1145/1596519.1596520.

[20] Alexander I. J. Forrester and Andy J. Keane. "Recent advances in surrogate-based optimization". In: *Progress in Aerospace Sciences* 45.1-3 (Jan. 2009), pp. 50–79. DOI: 10.1016/j.paerosci.2008.11.001.

[21] Xinwei Fu, Sergio Ricci, and Chiara Bisagni. "Minimum-weight design for three dimensional woven composite stiffened panels using neural networks and genetic algorithms". In: *Composite Structures* 134 (Dec. 2015), pp. 708–715. DOI: 10.1016/j.compstruct.2015.08.077.

[22] Sushant S. Garud, Iftekhar A. Karimi, and Markus Kraft. "Design of computer experiments: A review". In: *Computers & Chemical Engineering* 106 (Nov. 2017), pp. 71–95. DOI: 10.1016/j.compchemeng.2017.05.010.

[23] Xavier Glorot and Yoshua Bengio. "Understanding the difficulty of training deep feedforward neural networks". In: *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*. Ed. by Yee Whye Teh and Mike Titterington. Vol. 9. Proceedings of Machine Learning Research. Chia Laguna Resort, Sardinia, Italy: PMLR, May 2010, pp. 249–256. URL: http://proceedings.mlr.press/v9/glorot10a.html.

[24] Dirk Gorissen et al. "Adaptive Distributed Metamodeling". In: *Lecture Notes in Computer Science*. Ed. by M. Daydé et al. Vol. 4395. LNCS. Springer Berlin Heidelberg, 2007, pp. 579–588. DOI: 10.1007/978-3-540-71351-7_45.

[25] Qi Guo et al. "Thermo-mechanical optimization of metallic thermal protection system under aerodynamic heating". In: *Structural and Multidisciplinary Optimization* (Aug. 2019). DOI: 10.1007/s00158-019-02379-4.

[26] Xiaofang Guo, Yuping Wang, and Xiaoli Wang. "An objective reduction algorithm using representative Pareto solution search for many-objective optimization problems". In: *Soft Computing* 20.12 (July 2015), pp. 4881–4895. DOI: 10.1007/s00500-015-1776-4.

[27] Warren Hare, Julie Nutini, and Solomon Tesfamariam. "A survey of non-gradient optimization methods in structural engineering". In: *Advances in Engineering Software* 59 (May 2013), pp. 19–28. DOI: 10.1016/j.advengsoft.2013.03.001.

[28] Robert Hecht-Nielsen, Hecht-Nielsen Ne, and urocomputer Corporation. "Kolmogorov's Mapping Neural Network Existence Theorem". In: *Proceedings of the international conference on Neural Networks*. Vol. 3. IEEE Press New York. 1987, pp. 11–14.

[29] Guang-Bin Huang. "Learning capability and storage capacity of two-hidden-layer feedforward networks". In: *IEEE Transactions on Neural Networks* 14.2 (Mar. 2003), pp. 274–281. DOI: 10.1109/tnn.2003.809401.

[30] Himanshu Jain and Kalyanmoy Deb. "An Evolutionary Many-Objective Optimization Algorithm Using Reference-Point Based Nondominated Sorting Approach, Part II: Handling Constraints and Extending to an Adaptive Approach". In: *IEEE Transactions on Evolutionary Computation* 18.4 (Aug. 2014), pp. 602–622. DOI: 10.1109/tevc.2013.2281534.

[31] Ruichen Jin, Wei Chen, and Agus Sudjianto. "On Sequential Sampling for Global Metamodeling in Engineering Design". In: *Volume 2: 28th Design Automation Conference*. ASMEDC, Jan. 2002. DOI: 10.1115/detc2002/dac-34092.

[32] A. N. Kolmogorov. "On the representation of continuous functions of many variables by superposition of continuous functions of one variable and addition". In: *Doklady Akademii Nauk SSSR* 114.5 (1957), pp. 953–956. URL: http://mi.mathnet.ru/eng/dan22050.

[33] Slawomir Koziel and Stanislav Ogurtsov. "Multi-Objective Design of Antennas Using Variable-Fidelity Simulations and Surrogate Models". In: *IEEE Transactions on Antennas and Propagation* 61.12 (Dec. 2013), pp. 5931–5939. DOI: 10.1109/tap.2013.2283599.

[34] L. Lanzi, C. Bisagni, and S. Ricci. "Crashworthiness optimization of helicopter subfloor based on decomposition and global approximation". In: *Structural and Multidisciplinary Optimization* 27.5 (June 2004). DOI: 10.1007/s00158-004-0394-z.

[35] Xin Liu and Albert C. Reynolds. "Gradient-based multi-objective optimization with applications to waterflooding optimization". In: *Computational Geosciences* 20.3 (Sept. 2015), pp. 677–693. DOI: 10.1007/s10596-015-9523-6.

[36] R. T. Marler and J. S. Arora. "Survey of multi-objective optimization methods for engineering". In: *Structural and Multidisciplinary Optimization* 26.6 (Apr. 2004), pp. 369–395. DOI: 10.1007/s00158-003-0368-6.

[37] Art B. Owen. "A randomized Halton algorithm in R". In: June 9, 2017. arXiv: 1706.02808v2 [stat.CO].

[38] Pramudita Satria Palar et al. "On the use of surrogate models in engineering design optimization and exploration". In: *Proceedings of the Genetic and Evolutionary Computation Conference Companion*. ACM, July 2019. DOI: 10.1145/3319619.3326813.

[39] Sebastian Peitz and Michael Dellnitz. “Gradient-Based Multiobjective Optimization with Uncertainties”. In: *NEO 2016*. Springer International Publishing, Sept. 2017, pp. 159–182. DOI: 10.1007/978-3-319-64063-1_7.

[40] Prajit Ramachandran, Barret Zoph, and Quoc V. Le. “Searching for Activation Functions”. In: *CoRR* (Oct. 16, 2017). arXiv: http://arxiv.org/abs/1710.05941v2 [cs.NE].

[41] Songqing Shan and G. Gary Wang. “Survey of modeling and optimization strategies to solve high-dimensional design problems with computationally-expensive black-box functions”. In: *Structural and Multidisciplinary Optimization* 41.2 (Aug. 2009), pp. 219–241. DOI: 10.1007/s00158-009-0420-2.

[42] T. W. Simpson et al. “Metamodels for Computer-based Engineering Design: Survey and recommendations”. In: *Engineering with Computers* 17.2 (July 2001), pp. 129–150. DOI: 10.1007/pl00007198.

[43] Prashant Singh, Dirk Deschrijver, and Tom Dhaene. “A balanced sequential design strategy for global surrogate modeling”. In: *2013 Winter Simulations Conference (WSC)*. IEEE, Dec. 2013. DOI: 10.1109/wsc.2013.6721594.

[44] J. Sobieszczanski-Sobieski. *Structural Optimization: Challenges and Opportunities*. NASA (National Aeronautics and Space Administration), 1984.

[45] Niranjan Srinivas et al. “Gaussian Process Optimization in the Bandit Setting: No Regret and Experimental Design”. In: *Proceedings of the 27th International Conference on International Conference on Machine Learning*. ICML’10. Haifa, Israel: Omnipress, 2010, pp. 1015–1022. ISBN: 9781605589077.

[46] Ryoji Tanabe and Hisao Ishibuchi. “An easy-to-use real-world multi-objective optimization problem suite”. In: vol. 89. Elsevier BV, Apr. 2020, p. 106078. DOI: 10.1016/j.asoc.2020.106078.

[47] G. N. Vanderplaats. “Thirty years of modern structural optimization”. In: *Advances in Engineering Software* 16.2 (Jan. 1993), pp. 81–88. DOI: 10.1016/0965-9978(93)90052-u.

[48] S. Venkataraman and R. T. Haftka. “Structural optimization complexity: what has Moore?s law done for us?” In: *Structural and Multidisciplinary Optimization* 28.6 (Sept. 2004), pp. 375–387. DOI: 10.1007/s00158-004-0415-y.

[49] Felipe A. C. Viana et al. “Special Section on Multidisciplinary Design Optimization: Metamodeling in Multidisciplinary Design Optimization: How Far Have We Really Come?” In: *AIAA Journal* 52.4 (Apr. 2014), pp. 670–690. DOI: 10.2514/1.j052375.

[50] Ivan Voutchkov and Andy Keane. “Multi-Objective Optimization Using Surrogates”. In: *Computational Intelligence in Optimization*. Ed. by Y. Tenne and C.-K. Goh. Springer Berlin Heidelberg, 2010, pp. 155–175. DOI: 10.1007/978-3-642-12775-5_7.

[51] D. H. Wolpert and W. G. Macready. “No free lunch theorems for optimization”. In: *IEEE Transactions on Evolutionary Computation* 1.1 (Apr. 1997), pp. 67–82. DOI: 10.1109/4235.585893.

[52] Xin-She Yang. *Nature-Inspired Optimization Algorithms*. Elsevier, 2014. ISBN: 978-0-12-416743-8.

[53] Xin-She Yang. “Nature-inspired optimization algorithms: Challenges and open problems”. In: *Journal of Computational Science* (Mar. 2020), p. 101104. DOI: 10.1016/j.jocs.2020.101104. eprint: 2003.03776v1.

[54] B. D. Youn et al. “Reliability-based design optimization for crashworthiness of vehicle side impact”. In: *Structural and Multidisciplinary Optimization* 26.3-4 (Feb. 2004), pp. 272–283. DOI: 10.1007/s00158-003-0345-0.

[55] Wei Zeng et al. “Design Optimization of a VX Gasket Structure for a Subsea Connector Based on the Kriging Surrogate Model-NSGA-II Algorithm Considering the Load Randomness”. In: *Algorithms* 12.2 (Feb. 2019), p. 42. DOI: 10.3390/a12020042.

[56] Michael Zhu and Suyog Gupta. “To Prune, or Not to Prune: Exploring the Efficacy of Pruning for Model Compression”. In: *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Workshop Track Proceedings*. 2018. arXiv: https://arxiv.org/abs/1710.01878v2 [cs.ML].