



HAL
open science

SMT 2.0: A Surrogate Modeling Toolbox with a focus on hierarchical and mixed variables Gaussian processes

Paul Saves, Rémi Lafage, Nathalie Bartoli, Youssef Diouane, Jasper Bussemaker, Thierry Lefebvre, John T Hwang, Joseph Morlier, Joaquim R.R.A. Martins

► To cite this version:

Paul Saves, Rémi Lafage, Nathalie Bartoli, Youssef Diouane, Jasper Bussemaker, et al.. SMT 2.0: A Surrogate Modeling Toolbox with a focus on hierarchical and mixed variables Gaussian processes. *Advances in Engineering Software*, 2024, 188 (103571), pp.103571. 10.1016/j.advengsoft.2023.103571 . hal-04331916

HAL Id: hal-04331916

<https://hal.science/hal-04331916>

Submitted on 11 Dec 2023

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.



Distributed under a Creative Commons Attribution - NoDerivatives 4.0 International License

SMT 2.0: A Surrogate Modeling Toolbox with a focus on Hierarchical and Mixed Variables Gaussian Processes

Paul Saves^{a,b}, Rémi Lafage^a, Nathalie Bartoli^a, Youssef Diouane^c, Jasper Bussemaker^d, Thierry Lefebvre^a, John T. Hwang^e, Joseph Morlier^{f,b}, Joaquim R. R. A. Martins^g

^aONERA/DTIS, Université de Toulouse, Toulouse, France

^bISAE-SUPAERO, Université de Toulouse, Toulouse, France

^cPolytechnique Montréal, Montreal, QC, Canada

^dGerman Aerospace Center (DLR), Institute of System Architectures in Aeronautics, Hamburg, Germany

^eUniversity of California San Diego, Department of Mechanical and Aerospace Engineering, La Jolla, CA, USA

^fICA, Université de Toulouse, ISAE-SUPAERO, INSA, CNRS, MINES ALBI, UPS, Toulouse, France

^gUniversity of Michigan, Department of Aerospace Engineering, Ann Arbor, MI, USA

Abstract

The Surrogate Modeling Toolbox (SMT) is an open-source Python package that offers a collection of surrogate modeling methods, sampling techniques, and a set of sample problems. This paper presents SMT 2.0, a major new release of SMT that introduces significant upgrades and new features to the toolbox. This release adds the capability to handle mixed-variable surrogate models and hierarchical variables. These types of variables are becoming increasingly important in several surrogate modeling applications. SMT 2.0 also improves SMT by extending sampling methods, adding new surrogate models, and computing variance and kernel derivatives for Kriging. This release also includes new functions to handle noisy and use multi-fidelity data. To the best of our knowledge, SMT 2.0 is the first open-source surrogate library to propose surrogate models for hierarchical and mixed inputs. This open-source software is distributed under the New BSD license ¹.

Email addresses: paul.saves@onera.fr (Paul Saves), remi.lafage@onera.fr (Rémi Lafage), nathalie.bartoli@onera.fr (Nathalie Bartoli), youssef.diouane@polymtl.ca (Youssef Diouane), jasper.bussemaker@dlr.de (Jasper Bussemaker), thierry.lefebvre@onera.fr (Thierry Lefebvre), jhwang@eng.ucsd.edu (John T. Hwang), joseph.morlier@isae-supaero.fr (Joseph Morlier), jraram@umich.edu (Joaquim R. R. A. Martins)

¹<https://github.com/SMTorg/SMT>

Keywords: surrogate modeling, Gaussian process, Kriging, hierarchical problems, hierarchical and mixed-categorical inputs, meta variables

1. Motivation and significance

With the increasing complexity and accuracy of numerical models, it has become more challenging to run complex simulations and computer codes [56, 47]. As a consequence, surrogate models have been recognized as a key tool for engineering tasks such as design space exploration, uncertainty quantification, and optimization [38]. In practice, surrogate models are used to reduce the computational effort of these tasks by replacing expensive numerical simulations with closed-form approximations [57, Ch. 10]. To build such a model, we start by evaluating the original expensive simulation at a set of points through a Design of Experiments (DoE). Then, the corresponding evaluations are used to build the surrogate model according to the chosen approximation, such as Kriging, quadratic interpolation, or least squares regression.

The Surrogate Modeling Toolbox (SMT) is an open-source framework that provides functions to efficiently build surrogate models [9]. Kriging models (also known as Gaussian processes) that take advantage of derivative information are one of SMT's key features [6]. Numerical experiments have shown that SMT achieved lower prediction error and computational cost than Scikit-learn [62] and UQLab [52] for a fixed number of points [27]. SMT has been applied to rocket engine coaxial-injector optimization [48], aircraft engine consumption modeling [60], numerical integration [26], multi-fidelity sensitivity analysis [25], high-order robust finite elements methods [45, 49], planning for photovoltaic solar energy [18], wind turbines design optimization [39], porous material optimization for a high pressure turbine vane [79], chemical process design [75] and many other applications.

In systems engineering, architecture-level choices significantly influence the final system performance, and therefore, it is desirable to consider such choices in the early design phases [14]. Architectural choices are parameterized with discrete design variables; examples include the selection of technologies, materials, component connections, and number of instantiated elements. When design problems include both discrete variables and continuous variables, they are said to have *mixed variables*.

When architectural choices lead to different sets of design variables, we have *hierarchical* variables [37, 11]. For example, consider different aircraft propulsion architectures [28]. A conventional gas turbine would not require a variable to represent a choice in the electrical power source, while hybrid

or pure electric propulsion would require such a variable. The relationship between the choices and the sets of variables can be represented by a hierarchy.

Handling hierarchical and mixed variables requires specialized surrogate modeling techniques [12]. To address these needs, **SMT 2.0** is offering researchers and practitioners a collection of cutting-edge tools to build surrogate models with continuous, mixed and hierarchical variables. The main objective of this paper is to detail the new enhancements that have been added in this release compared to the original **SMT 0.2** release [9].

There are two new major capabilities in **SMT 2.0**: the ability to build surrogate models involving mixed variables and the support for hierarchical variables within Kriging models. To handle mixed variables in Kriging models, existing libraries such as BoTorch [3], Dakota [1], DiceKriging [70], LVGP [84], Parmoo [15], and Spearmint [30] implement simple mixed models by using either continuous relaxation (CR), also known as *one-hot encoding* [30], or a Gower distance (GD) based correlation kernel [33]. KerGP [71] (developed in R) implements more general kernels but there is no Python open-source toolbox that implements more general kernels to deal with mixed variables, such as the homoscedastic hypersphere (HH) [85] and exponential homoscedastic hypersphere (EHH) [77] kernels. Such kernels require the tuning of a large number of hyperparameters but lead to more accurate Kriging surrogates than simpler mixed kernels [63, 77]. **SMT 2.0** implements all these kernels (CR, GD, HH, and EHH) through a unified framework and implementation. To handle hierarchical variables, no library in the literature can build peculiar surrogate models except **SMT 2.0**, which implements two Kriging methods for these variables. Notwithstanding, most softwares are compatible with a naïve strategy called the imputation method [12] but this method lacks depth and depends on arbitrary choices. This is why Hutter and Osborne [37] proposed a first kernel, called **Arc-Kernel** which in turn was generalized by Horn et al. [35] with a new kernel called the **Wedge-Kernel** [36]. None of these kernels are available in any open-source modeling software. Furthermore, thanks to the framework introduced in Audet et al. [2], our proposed kernels are sufficiently general so that all existing hierarchical kernels are included within it. Section 4 describes the two kernels implemented in **SMT 2.0** that are referred as **SMT Arc-Kernel** and **SMT Alg-Kernel**. In particular, **Alg-Kernel** is a novel hierarchical kernel introduced in this paper. Table 1 outlines the main features of the state-of-the-art modeling software that can handle hierarchical and mixed variables.

Table 1: Comparison of software packages for hierarchical and mixed Kriging models. ✓ = implemented. * = user-defined.

Package	BOTorch	Dakota	DiceKriging	KerGP	LVGP	Parmoo	Spearmint	SMT 2.0
Reference	[3]	[1]	[70]	[71]	[84]	[15]	[30]	This paper
License	MIT	EPL	GPL	GPL	GPL	BSD	GNU	BSD
Language	Python	C	R	R	R	Python	Python	Python
Mixed var.	✓	✓	✓	✓	✓	✓	✓	✓
<i>GD kernel</i>	✓	✓	✓	*				✓
<i>CR kernel</i>					✓	✓	✓	✓
<i>HH kernel</i>				✓				✓
<i>EHH kernel</i>				*				✓
Hierarchical var.								✓

SMT 2.0 introduces other enhancements, such as additional sampling procedures, new surrogate models, new Kriging kernels (and their derivatives), Kriging variance derivatives, and an adaptive criterion for high-dimensional problems. SMT 2.0 adds applications of Bayesian optimization (BO) with hierarchical and mixed variables or noisy co-Kriging that have been successfully applied to aircraft design [76], data fusion [21], and structural design [74]. The SMT 2.0 interface is more user-friendly and offers an improved and more detailed documentation for users and developers ². SMT 2.0 is hosted publicly ³ and can be directly imported within Python scripts. It is released under the New BSD License and runs on Linux, MacOS, and Windows operating systems. Regression tests are run automatically for each operating system whenever a change is committed to the repository. In short, SMT 2.0 builds on the strengths of the original SMT package while adding new features. On one hand, the emphasis on derivatives (including prediction, training and output derivatives) is maintained and improved in SMT 2.0. On the other hand, this new release includes support for hierarchical and mixed variables Kriging based models. For the sake of reproducibility, an open-source notebook is available that gathers all the methods and results presented on this paper ⁴.

The remainder of the paper is organized as follows. First, we introduce the organization and the main implemented features of the release in Section 2. Then, we describe the mixed-variable Kriging model with an example in Section 3. Similarly, we describe and provide an example for a hierarchical-variable Kriging model in Section 4. The Bayesian optimization models and applications are described in Section 5. Finally, we describe the other relevant

²<http://smt.readthedocs.io/en/latest>

³<https://github.com/SMTorg/smt>

⁴https://github.com/SMTorg/smt/tree/master/tutorial/NotebookRunTestCases_Paper_SMT_v2.ipynb

contributions in Section 6 and conclude in Section 7.

2. SMT 2.0: an improved surrogate modeling toolbox

From a software point of view, SMT 2.0 maintains and improves the modularity and generality of the original SMT version [9]. In this section, we describe the software as follows. Section 2.1 describes the legacy of SMT 0.2. Then, Section 2.2 describes the organization of the repository. Finally, Section 2.3 shows the new capabilities implemented in the SMT 2.0 update.

2.1. Background on SMT former version: SMT 0.2

SMT [9] is an open-source collaborative work originally developed by ONERA, NASA Glenn, ISAE-SUPAERO/ICA and the University of Michigan. Now, both Polytechnique Montréal and the University of California San Diego are also contributors. SMT 2.0 updates and extends the original SMT repository capabilities among which the original publication [9] focuses on different types of derivatives for surrogate models detailed hereafter.

A Python surrogate modeling framework with derivatives. One of the original main motivations for SMT was derivative support. In fact, none of the existing packages for surrogate modeling such as Scikit-learn in Python [62], SUMO in Matlab [32] or GPML in Matlab and Octave [81] focuses on derivatives. Three types of derivatives are distinguished: prediction derivatives, training derivatives, and output derivatives. SMT also includes new models with derivatives such as Kriging with Partial Least Squares (KPLS) [8] and Regularized Minimal-energy Tensor-product Spline (RMTS) [38]. These developed derivatives were even used in a novel algorithm called Gradient-Enhanced Kriging with Partial Least Squares (GEKPLS) [6] to use with adjoint methods, for example [10].

Software architecture, documentation, and automatic testing. SMT is organized along three main sub-modules that implement a set of sampling techniques (**sampling methods**), benchmarking functions (**problems**), and surrogate modeling techniques (**surrogate models**). The toolbox documentation ⁵ is created using reStructuredText and Sphinx, a documentation generation package for Python, with custom extensions. Code snippets in the documentation pages are taken directly from actual tests in the source code and are automatically updated. The output from these code snippets and tables of options are generated dynamically by custom Sphinx extensions.

⁵<https://smt.readthedocs.org>

This leads to high-quality documentation with minimal effort. Along with user documentation, developer documentation is also provided to explain how to contribute to SMT. This includes a list of API methods for the **SurrogateModel**, **SamplingMethod**, and **Problem** classes, that must be implemented to create a new surrogate modeling method, sampling technique, or benchmarking problem. When a developer submits a pull request, it is merged only after passing the automated tests and receiving approval from at least one reviewer. The repository on GitHub ⁶ is linked to continuous integration tests (GitHub Actions) for Windows, Linux and MacOS, to a coverage test on coveralls.io and to a dependency version check for Python with Dependabot. Various parts of the source code have been accelerated using Numba [50], an LLVM-based just-in-time (JIT) compiler for numpy-heavy Python code. Numba is applied to conventional Python code using function decorators, thereby minimizing its impact on the development process and not requiring an additional build step. For a mixed Kriging surrogate with 150 training points, a speedup of up to 80% is observed, see Table 2. The JIT compilation step only needs to be done once when installing or upgrading SMT and adds an overhead of approximately 24 seconds on a typical workstation. In this paper, all results are obtained using an Intel® Xeon® CPU E5-2650 v4 @ 2.20 GHz core and 128 GB of memory with a Broadwell-generation processor front-end and a compute node of a peak power of 844 GFlops.

Table 2: Impact of using Numba on training time of the hierarchical Goldstein problem. Speedup is calculated excluding the JIT compilation table, as this step is only needed once after SMT installation.

Training set	Without Numba	Numba	Speedup	JIT overhead
15 points	1.3 s	1.1 s	15%	24 s
150 points	38 s	7.4 s	80%	23 s

2.2. Organization of SMT 2.0

The main features of the open-source repository SMT 2.0 are described in Figure 1. More precisely, **Sampling Methods**, **Problems** and **Surrogate models** are kept from SMT 0.2 and two new sections **Models applications** and **Interactive notebooks** have been added to the architecture of the code. These sections are highlighted in blue and detailed on Figure 1.

⁶<https://github.com/SMTorg/smt>

The new major features implemented in SMT 2.0 are highlighted in lavender whereas the legacy features that were already present in the original publication for SMT 0.2 [9] are in black.

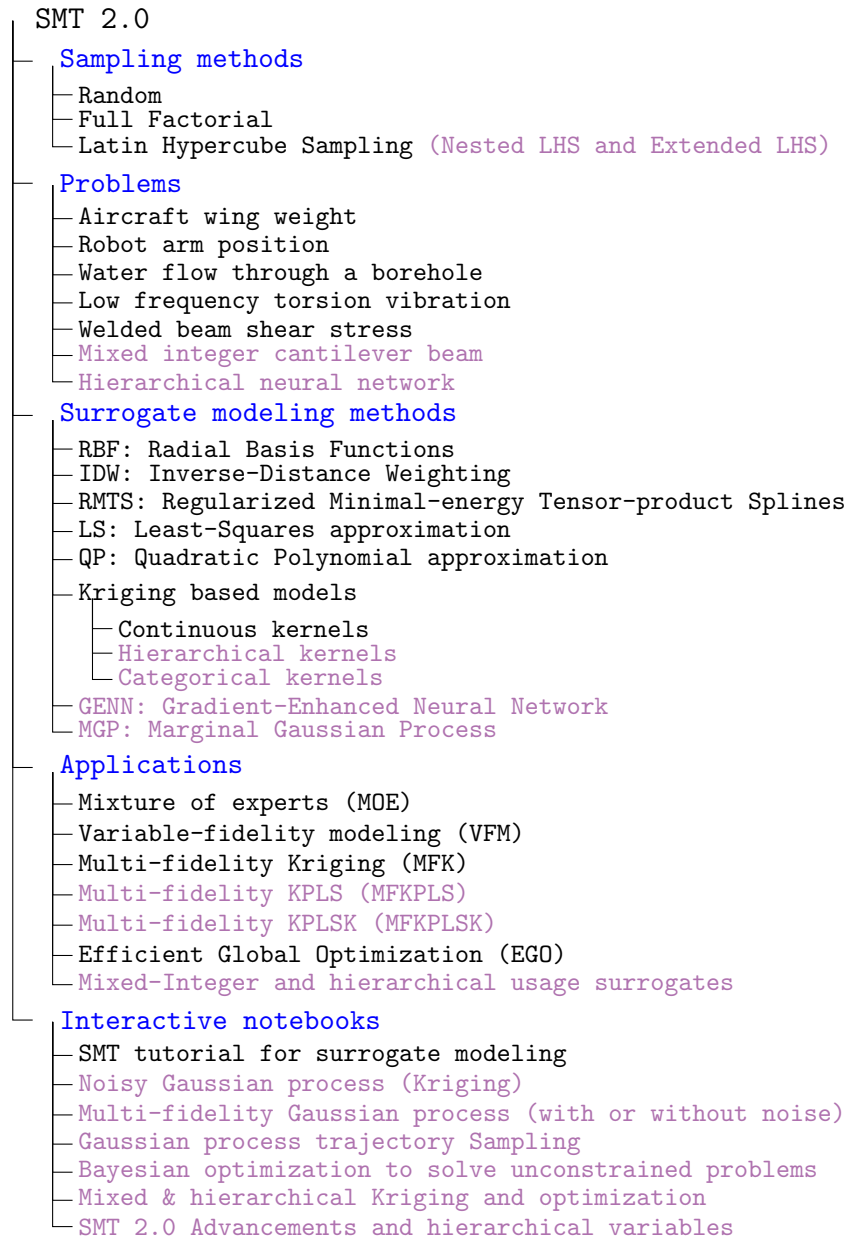


Figure 1: Functionalities of SMT 2.0. The new major features implemented in SMT 2.0 compared to SMT 0.2 are highlighted with the lavender color.

2.3. New features within SMT 2.0

The main objective of this new release is to enable Kriging surrogate models for use with both hierarchical and mixed variables. Moreover, for each of these five sub-modules described in Section 2.2, several improvements have been made between the original version and the SMT 2.0 release.

Hierarchical and mixed design space. A new design space definition class `DesignSpace` has been added that implements hierarchical and mixed functionalities. Design variables can either be continuous (`FloatVariable`), ordered (`OrdinalVariable`) or categorical (`CategoricalVariable`). The integer type (`IntegerVariable`) represents a special case of the ordered variable, specified by bounds (inclusive) rather than a list of possible values. The hierarchical structure of the design space can be defined using `declare_decreed_var`: this function declares that a variable is a decreed variable that is activated when the associated meta variable takes one of a set of specified values, see Section 4 for background. The `DesignSpace` class also implements mechanisms for sampling valid design vectors (i.e. design vectors that adhere to the hierarchical structure of the design space) using any of the below-mentioned samplers, for correcting and imputing design vectors, and for requesting which design variables are acting in a given design vector. Correction ensures that variables have valid values (e.g. integers for discrete variables) [12], and imputation replaces non-acting variables by some default value (0 for discrete variables, mid-way between the bounds for continuous variables in SMT 2.0) [83].

Sampling. SMT implements three methods for sampling. The first one is a naïve approach, called `Random` that draws uniformly points along every dimension. The second sampling method is called `Full Factorial` and draws a point for every cross combination of variables, to have an “exhaustive” design of experiments. The last one is the `Latin Hypercube Sampling (LHS)` [41] that draws a point in every Latin square parameterized by a certain criterion. For LHS, a new criterion to manage the randomness has been implemented and the sampling method was adapted for multi-fidelity and mixed or hierarchical variables. More details about the new sampling techniques are given in Section 6.1.

Problems. SMT implements two new engineering problems: a mixed variant of a cantilever beam described in Section 3 and a hierarchical neural network described in Section 4.

Surrogate models. In order to keep up with state-of-art, several releases done from the original version developed new options for the already existing surrogates. In particular, compared to the original publication [9], SMT 2.0 adds gradient-enhanced neural networks [10] and marginal Gaussian process [29] models to the list of available surrogates. More details about the new models are given in Section 6.2.

Applications. Several applications have been added to the toolbox to demonstrate the surrogate models capabilities. The most relevant application is efficient global optimization (EGO), a Bayesian optimization algorithm [42, 51]. EGO optimizes expensive-to-evaluate black-box problems with a chosen surrogate model and a chosen optimization criterion [43]. The usage of EGO with hierarchical and mixed variables is described in Section 5.

Interactive notebooks. These tutorials introduce and explain how to use the toolbox for different surrogate models and applications ⁷. Every tutorial is available both as a .ipynb file and directly on Google colab ⁸. In particular, a hierarchical and mixed variables dedicated notebook is available to reproduce the results presented on this paper ⁹.

In the following, Section 3 details the Kriging based surrogate models for mixed variables, and Section 4 presents our new Kriging surrogate for hierarchical variables. Section 5 details the EGO application and the other new relevant features aforementioned are described succinctly in Section 6.

3. Surrogate models with mixed variables in SMT 2.0

As mentioned in Section 1, design variables can be either of continuous or discrete type, and a problem with both types is a mixed-variable problem. Discrete variables can be ordinal or categorical. A discrete variable is *ordinal* if there is an order relation within the set of possible values. An example of an ordinal design variable is the number of engines in an aircraft. A possible set of values in this case could be 2, 4, 8. A discrete variable is *categorical* if no order relation is known between the possible choices the variable can take. One example of a categorical variable is the color of a surface. A possible example of a set of choices could be blue, red, green. The possible choices are called the *levels* of the variable.

⁷<https://github.com/SMTorg/smt/tree/master/tutorial>

⁸<https://colab.research.google.com/github/SMTorg/smt/>

⁹https://github.com/SMTorg/smt/tree/master/tutorial/NotebookRunTestCases_Paper_SMT_v2.ipynb

SPD. For an exponential kernel, Table 3 gives the parameterizations of Φ and κ that correspond to GD, CR, HH, and EHH kernels. The complexity of these different kernels depends on the number of hyperparameters that characterizes them. As defined by Saves et al. [77], for every categorical variable $i \in \{1, \dots, l\}$, the matrix $C(\Theta_i) \in \mathbb{R}^{L_i \times L_i}$ is lower triangular and built using a hypersphere decomposition [69, 68] from the symmetric matrix $\Theta_i \in \mathbb{R}^{L_i \times L_i}$ of hyperparameters. The variable ϵ is a small positive constant and the variable θ_i denotes the only positive hyperparameter that is used for the Gower distance kernel.

Table 3: Categorical kernels implemented in SMT 2.0.

Name	$\kappa(\phi)$	$\Phi(\Theta_i)$	# of hyperparam.
SMT GD	$\exp(-\phi)$	$[\Phi(\Theta_i)]_{j,j} := \frac{1}{2}\theta_i$; $[\Phi(\Theta_i)]_{j \neq j'} := 0$	1
SMT CR	$\exp(-\phi)$	$[\Phi(\Theta_i)]_{j,j} := [\Theta_i]_{j,j}$; $[\Phi(\Theta_i)]_{j \neq j'} := 0$	L_i
SMT EHH	$\exp(-\phi)$	$[\Phi(\Theta_i)]_{j,j} := 0$; $[\Phi(\Theta_i)]_{j \neq j'} := \frac{\log \epsilon}{2} ([C(\Theta_i)C(\Theta_i)^\top]_{j,j'} - 1)$	$\frac{1}{2}(L_i)(L_i - 1)$
SMT HH	ϕ	$[\Phi(\Theta_i)]_{j,j} := 1$; $[\Phi(\Theta_i)]_{j \neq j'} := [C(\Theta_i)C(\Theta_i)^\top]_{j,j'}$	$\frac{1}{2}(L_i)(L_i - 1)$

Another Kriging based model that can use mixed variables is Kriging with partial least squares (KPLS) [7]. KPLS adapts Kriging to high dimensional problems by using a reduced number of hyperparameters thanks to a projection into a smaller space. Also, for a general surrogate, not necessarily Kriging, SMT 2.0 uses continuous relaxation to allow whatever model to handle mixed variables. For example, we can use mixed variables with least squares (LS) or quadratic polynomial (QP) models. We now illustrate the abilities of the toolbox in terms of mixed modeling over an engineering test case.

3.2. An engineering design test-case

A classic engineering problem commonly used for model validation is the beam bending problem [71, 19]. This problem is illustrated on Figure 2a and consists of a cantilever beam in its linear range loaded at its free end with a force F . As in Cheng et al. [19], the Young modulus is $E = 200\text{GPa}$ and the chosen load is $F = 50\text{kN}$. Also, as in Roustant et al. [71], 12 possible cross-sections can be used. These 12 sections consist of 4 possible shapes that can be either hollow, thick or full as illustrated in Figure 2b.

To compare the mixed Kriging models of SMT 2.0, we draw a 98 point LHS as training set and the validation set is a grid of $12 \times 30 \times 30 = 10800$ points. For the four implemented methods, displacement error (computed with a root-mean-square error criterion), likelihood, number of hyperparameters and computational time for every model are shown in Table 4. For the

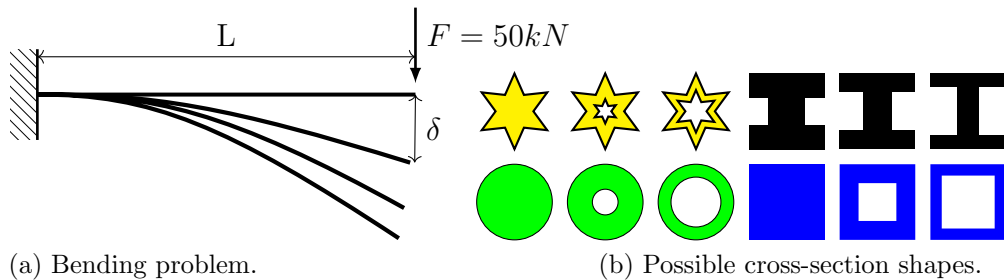


Figure 2: Cantilever beam problem [77, Figure 6].

continuous variables, we use the square exponential kernel. More details are found in [77]. As expected, the complex EHH and HH models lead to a lower displacement error and a higher likelihood value, but use more hyperparameters and increase the computational cost compared to GD and CR. On this test case, the kernel EHH is easier to optimize than HH but in general, they are similar in terms of performance. Also, by default `SMT 2.0` uses CR as it is known to be a good trade-off between complexity and performance [46].

Table 4: Results of the cantilever beam models [77, Table 4].

Categorical kernel	Displacement error (cm)	Likelihood	# of hyperparam.
SMT GD	1.3861	111.13	3
SMT CR	1.1671	155.32	14
SMT EHH	0.1613	236.25	68
SMT HH	0.2033	235.66	68

4. Surrogate models with hierarchical variables in `SMT 2.0`

To introduce the newly developed Kriging model for hierarchical variables implemented in `SMT 2.0`, we present the general mathematical framework for hierarchical and mixed variables established by Audet et al. [2]. In `SMT 2.0`, two variants of our new method are implemented, namely `SMT Alg-Kernel` and `SMT Arc-Kernel`. In particular, the `SMT Alg-Kernel` is a novel correlation kernel introduced in this paper.

4.1. The hierarchical variables framework

A problem structure is classified as hierarchical when the sets of active variables depend on architectural choices. This occurs frequently in industrial design problems. In hierarchical problems, we can classify variables as neutral, meta (also known as dimensional) or decreed (also known as conditionally active) as detailed in Audet et al. [2]. Neutral variables are the variables that are not affected by the hierarchy whereas the value assigned to

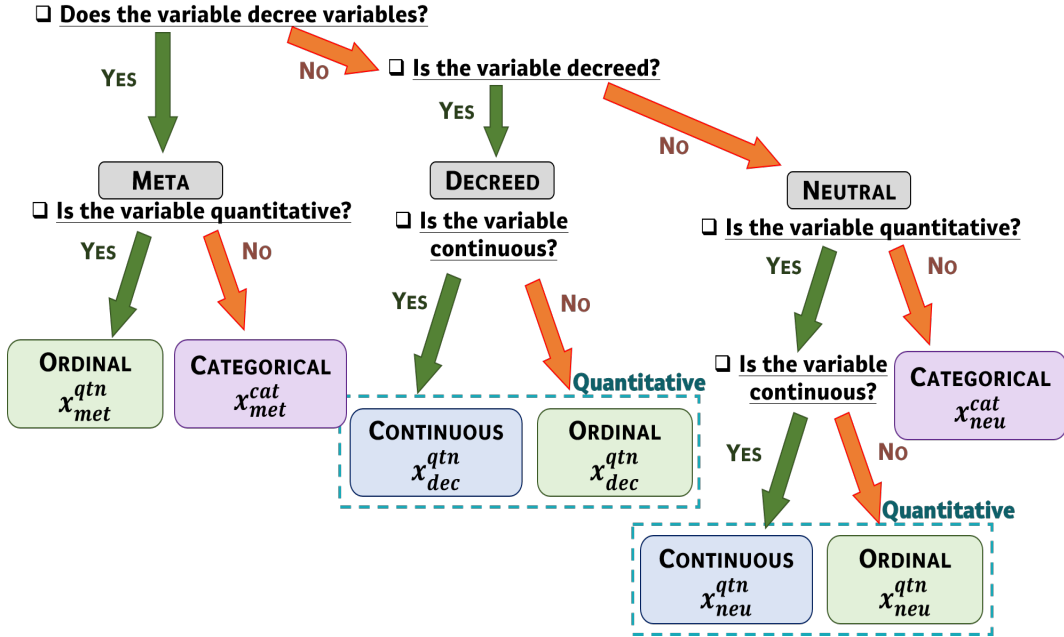


Figure 3: Variables classification as used in SMT 2.0.

meta variables determines which decreed variables are activated. For example, a meta variable could be the number of engines. If the number of engines changes, the number of decreed bypass ratios that every engine should specify also changes. However, the wing aspect ratio being neutral, it is not affected by this hierarchy.

Problems involving hierarchical variables are generally dependant on discrete architectures and as such involve mixed variables. Hence, in addition to their role (neutral, meta or decreed), each variable also has a variable type amongst categorical, ordinal or continuous. For the sake of simplicity and because both continuous and ordinal variables are treated similarly [77], we chose to regroup them as *quantitative variables*. For instance, the neutral variables x_{neu} may be partitioned into different variable types, such that $x_{neu} = (x_{neu}^{cat}, x_{neu}^{qnt})$ where x_{neu}^{cat} represents the categorical variables and x_{neu}^{qnt} are the quantitative ones. The variable classification scheme in SMT 2.0 is detailed in Figure 3.

To explain the framework and the new Kriging model, we illustrate the inputs variables of the model using a classical machine learning problem related to the hyperparameters optimization of a fully-connected Multi-Layer Perceptron (MLP) [2] on Figure 4. In Table 5, we detail the input variables of the model related to the MLP problem (i.e., the hyperparameters of the neural network, together with their types and roles). To keep things clear

and concise, the chosen problem is a simplification of the original problem developed by Audet et al. [2]. Regarding the MLP problem of Figure 4 and following the classification scheme of Figure 3, we start by separating the input variables according to their role. In fact,

1. changing the number of hidden layers modifies the number of inputs variables. Therefore, “# of hidden layers” is a meta variable.
2. The number of neurons in the hidden layer number k is either included or excluded. For example, the “# of neurons in the 3rd layer” would be excluded for an input that only has 2 hidden layers. Therefore, “# of neurons hidden layer k ” are decreed variables.
3. The “Learning rate”, “Momentum”, “Activation function” and “Batch size” are not affected by the hierarchy choice. Therefore, they are neutral variables.

According to their types, the MLP input variables can be classified as follows:

4. The meta variable “# of hidden layers” is an integer and, as such, is represented by the component $x_{\text{met}}^{\text{qnt}}$.
5. The decreed variables “# of neurons hidden layer k ” are integers and, as such, are represented by the component $x_{\text{dec}}^{\text{qnt}}$.
6. The “Learning rate”, “Momentum”, “Activation function” and “Batch size” are, respectively, continuous, for the first two (every value between two bounds), categorical (qualitative between three choices) and integer (quantitative between 6 choices). Therefore, the “Activation function” and the “Momentum” are represented by the component $x_{\text{neu}}^{\text{cat}}$. The “Learning rate” and the “Batch size” are represented by the component $x_{\text{neu}}^{\text{qnt}}$.

Table 5: A detailed description of the variables in the MLP problem.

MLP Hyperparameters	Variable	Domain	Type	Role
Learning rate	r	$[10^{-5}, 10^{-2}]$	FLOAT	NEUTRAL
Momentum	α	$[0, 1]$	FLOAT	NEUTRAL
Activation function	a	{ReLU, Sigmoid, Tanh}	ENUM	NEUTRAL
Batch size	b	{8, 16, ..., 128, 256}	ORD	NEUTRAL
# of hidden layers	l	{1, 2, 3}	ORD	META
# of neurons hidden layer k	n_k	{50, 51, ..., 55}	ORD	DECREED

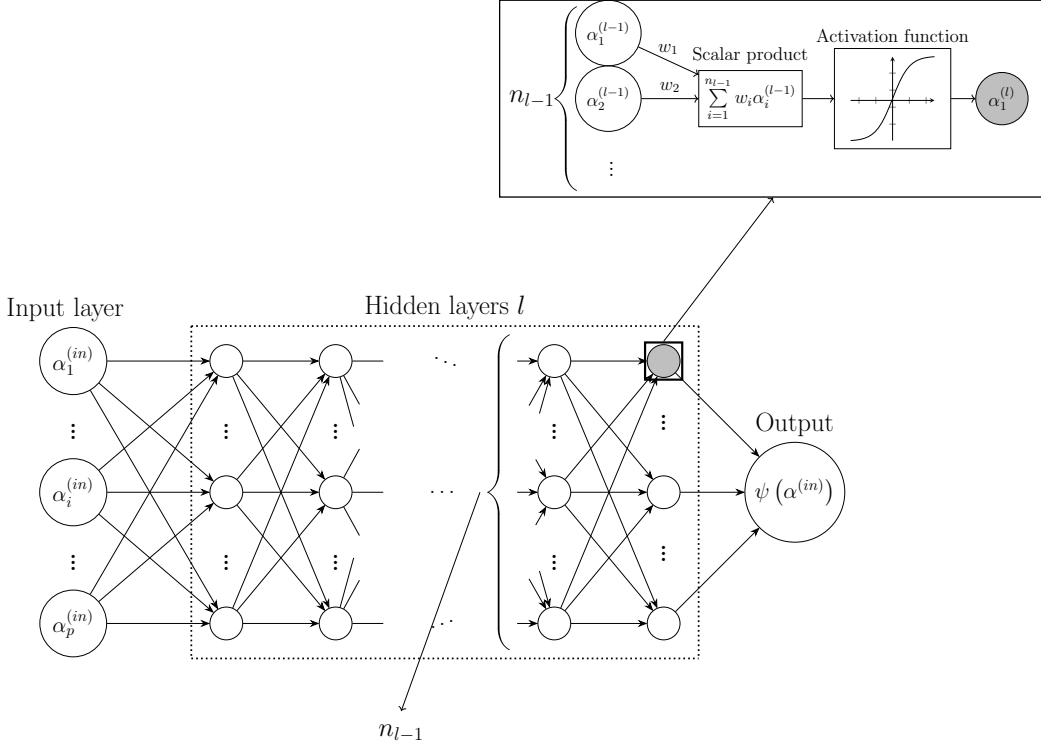


Figure 4: The Multi-Layer Perceptron (MLP) problem (figure adapted from [2, Figure 1]).

To model hierarchical variables, as proposed in [2], we separate the input space \mathcal{X} as $(\mathcal{X}_{\text{neu}}, \mathcal{X}_{\text{met}}, \mathcal{X}_{\text{dec}})$ where $\mathcal{X}_{\text{dec}} = \bigcup_{x_{\text{met}} \in \mathcal{X}_{\text{met}}} \mathcal{X}_{\text{inc}}(x_{\text{met}})$. Hence, for a given point $x \in \mathcal{X}$, one has $x = (x_{\text{neu}}, x_{\text{met}}, x_{\text{inc}}(x_{\text{met}}))$, where $x_{\text{neu}} \in \mathcal{X}_{\text{neu}}$, $x_{\text{met}} \in \mathcal{X}_{\text{met}}$ and $x_{\text{inc}}(x_{\text{met}}) \in \mathcal{X}_{\text{inc}}(x_{\text{met}})$ are defined as follows:

- The components $x_{\text{neu}} \in \mathcal{X}_{\text{neu}}$ gather all neutral variables that are not impacted by the meta variables but needed. For example, in the MLP problem, \mathcal{X}_{neu} gathers the possible learning rates, momentum, activation functions and batch sizes. Namely, from Table 5, $\mathcal{X}_{\text{neu}} = [10^{-5}, 10^{-2}] \times [0, 1] \times \{\text{ReLU}, \text{Sigmoid}, \text{Tanh}\} \times \{8, 16, \dots, 256\}$.
- The components x_{met} gather the meta (also known as dimensional) variables that determine the inclusion or exclusion of other variables. For example, in the MLP problem, \mathcal{X}_{met} represents the possible numbers of layers in the MLP. Namely, from Table 5, $\mathcal{X}_{\text{met}} = \{1, 2, 3\}$.
- The components $x_{\text{inc}}(x_{\text{met}})$, contain the decreed variables whose inclusion (decreed-included) or exclusion (decreed-excluded) is determined by the values of the meta components x_{met} . For example, in the MLP

problem, \mathcal{X}_{dec} represents the number of neurons in the decreed layers. Namely, from Table 5, $\mathcal{X}_{\text{inc}}(x_{\text{met}} = 3) = [50, 55]^3$, $\mathcal{X}_{\text{inc}}(x_{\text{met}} = 2) = [50, 55]^2$ and $\mathcal{X}_{\text{inc}}(x_{\text{met}} = 1) = [50, 55]$.

4.2. A Kriging model for hierarchical variables

In this section, a new method to build a Kriging model with hierarchical variables is introduced based on the framework aforementioned. The proposed methods are included in **SMT 2.0**.

4.2.1. Motivation and state-of-the-art

Assuming that the decreed variables are quantitative, Hutter and Osborne [37] proposed several kernels for the hierarchical context. A classic approach, called the imputation method (**Imp-Kernel**) leads to an efficient paradigm in practice that can be easily integrated into a more general framework as proposed by Bussemaker et al. [12]. However, this kernel lacks depth and depends on arbitrary choices. Therefore, Hutter and Osborne [37] also proposed a more general kernel, called **Arc-Kernel** and Horn et al. [35] generalized this kernel even more and proposed a new formulation called the **Wedge-Kernel** [36]. The drawbacks of these two methods are that they add some extra hyperparameters for every decreed dimension (respectively one extra hyperparameter for the **Arc-Kernel** and two hyperparameters for the **Wedge-Kernel**) and that they need a normalization according to the bounds. More recently, Pelamatti et al. [64] developed a hierarchical kernel for Bayesian optimization. However, our work is also more general thanks to the framework introduced earlier [2] that considers variable-wise formulation and is more flexible as we are allowing sub-problems to be intersecting.

In the following, we describe our new method to build a correlation kernel for hierarchical variables. In particular, we introduce a new algebraic kernel called **Alg-Kernel** that behaves like the **Arc-Kernel** whilst correcting most of its drawbacks. In particular, our kernel does not add any hyperparameters, and the normalization is handled in a natural way.

4.2.2. A new hierarchical correlation kernel

For modeling purposes, we assume that the decreed space is quantitative, i.e., $\mathcal{X}_{\text{dec}} = \mathcal{X}_{\text{dec}}^{\text{qnt}}$. Let $u \in \mathcal{X}$ be an input point partitioned as $u = (u_{\text{neu}}, u_{\text{met}}, u_{\text{inc}}(u_{\text{met}}))$ and, similarly, $v \in \mathcal{X}$ is another input such that $v = (v_{\text{neu}}, v_{\text{met}}, v_{\text{inc}}(v_{\text{met}}))$. The new kernel k that we propose for hierarchical variables is given by

$$k(u, v) = k_{\text{neu}}(u_{\text{neu}}, v_{\text{neu}}) \times k_{\text{met}}(u_{\text{met}}, v_{\text{met}}) \times k_{\text{met,dec}}([u_{\text{met}}, u_{\text{inc}}(u_{\text{met}})], [v_{\text{met}}, v_{\text{inc}}(v_{\text{met}})]), \quad (2)$$

where k_{neu} , k_{met} and $k_{\text{met,dec}}$ are as follows:

- k_{neu} represents the neutral kernel that encompasses both categorical and quantitative neutral variables, i.e., k_{neu} can be decomposed into two parts $k_{\text{neu}}(u_{\text{neu}}, v_{\text{neu}}) = k^{\text{cat}}(u_{\text{neu}}^{\text{cat}}, v_{\text{neu}}^{\text{cat}})k^{\text{qnt}}(u_{\text{neu}}^{\text{qnt}}, v_{\text{neu}}^{\text{qnt}})$. The categorical kernel, denoted k^{cat} , could be any Symmetric Positive Definite (SPD) [77] mixed kernel (see Section 3). For the quantitative (integer or continuous) variables, a distance-based kernel is used. The chosen quantitative kernel (Exponential, Matérn,...), always depends on a given distance d . For example, the n -dimensional exponential kernel gives

$$k^{\text{qnt}}(u^{\text{qnt}}, v^{\text{qnt}}) = \prod_{i=1}^n \exp(-d(u_i^{\text{qnt}}, v_i^{\text{qnt}})). \quad (3)$$

- k_{met} is the meta variables related kernel. It is also separated into two parts: $k_{\text{met}}(u_{\text{met}}, v_{\text{met}}) = k^{\text{cat}}(u_{\text{met}}^{\text{cat}}, v_{\text{met}}^{\text{cat}})k^{\text{qnt}}(u_{\text{met}}^{\text{qnt}}, v_{\text{met}}^{\text{qnt}})$ where the quantitative kernel is ordered and not continuous because meta variables take value in a finite set.
- $k_{\text{met,dec}}$ is an SPD kernel that models the correlations between the meta levels (all the possible subspaces) and the decreed variables. In what comes next, we detailed this kernel.

4.2.3. Towards an algebraic meta-decreed kernel

Meta-decreed kernels like the imputation kernel or the **Arc-Kernel** were first proposed in [83, 37] and the distance-based kernels such as **Arc-Kernel** or **Wedge-Kernel** [36] were proven to be SPD. Nevertheless, to guarantee this SPD property, the same hyperparameters are used to model the correlations between the meta levels and between the decreed variables [83]. For this reason, the **Arc-Kernel** includes additional continuous hyperparameters which makes the training of the GP models more expensive and introduces more numerical stability issues. In this context, we are proposing a new algebraic meta-decreed kernel (denoted as **Alg-Kernel**) that enjoys similar properties as **Arc-Kernel** but without using additional continuous hyperparameters nor rescaling. In the **SMT 2.0** release, we included **Alg-Kernel** and a simpler version of **Arc-Kernel** that do not relies on additional hyperparameters.

Our proposed **Alg-Kernel** kernel is given by

$$\begin{aligned} k_{\text{met,dec}}^{\text{alg}}([u_{\text{met}}, u_{\text{inc}}(u_{\text{met}})], [v_{\text{met}}, v_{\text{inc}}(v_{\text{met}})]) \\ = k_{\text{met}}^{\text{alg}}(u_{\text{met}}, v_{\text{met}}) \times k_{\text{dec}}^{\text{alg}}(u_{\text{inc}}(u_{\text{met}}), v_{\text{inc}}(v_{\text{met}})). \end{aligned} \quad (4)$$

Mathematically, we could consider that there is only one meta variable whose levels correspond to every possible included subspace. Let I_{sub} denotes the components indices of possible subspaces, the subspaces parameterized by the meta component u_{met} are defined as $\mathcal{X}_{\text{inc}}(u_{\text{met}} = l)$, $l \in I_{\text{sub}}$. It follows that the fully extended continuous decreed space writes as $\mathcal{X}_{\text{dec}} = \bigcup_{l \in I_{\text{sub}}} \mathcal{X}_{\text{inc}}(u_{\text{met}} = l)$ and I_{dec} is the set of the associated indices. Let $I_{u,v}^{\text{inter}}$ denotes the set of components related to the space $\mathcal{X}_{\text{inc}}(u_{\text{met}}, v_{\text{met}})$ containing the variables decreed-included in both $\mathcal{X}_{\text{inc}}(u_{\text{met}})$ and $\mathcal{X}_{\text{inc}}(v_{\text{met}})$.

Since the decreed variables are quantitative, one has

$$\begin{aligned} k_{\text{dec}}^{\text{alg}}(u_{\text{inc}}(u_{\text{met}}), v_{\text{inc}}(v_{\text{met}})) &= k^{\text{qnt}}(u_{\text{inc}}(u_{\text{met}}), v_{\text{inc}}(v_{\text{met}})) \\ &= \prod_{i \in I_{u,v}^{\text{inter}}} k^{\text{qnt}}([u_{\text{inc}}(u_{\text{met}})]_i, [v_{\text{inc}}(v_{\text{met}})]_i) \end{aligned} \quad (5)$$

The construction of the quantitative kernel k^{qnt} depends on a given distance denoted d^{alg} . The kernel $k_{\text{met}}^{\text{alg}}$ is an induced meta kernel that depends on the same distance d^{alg} to preserve the SPD property of $k_{\text{met,dec}}^{\text{alg}}$. For every $i \in I_{\text{dec}}$, if $i \in I_{u,v}^{\text{inter}}$, the new algebraic distance is given by

$$d^{\text{alg}}([u_{\text{inc}}(u_{\text{met}})]_i, [v_{\text{inc}}(v_{\text{met}})]_i) = \left(\frac{2|[u_{\text{inc}}(u_{\text{met}})]_i - [v_{\text{inc}}(v_{\text{met}})]_i|}{\sqrt{[u_{\text{inc}}(u_{\text{met}})]_i^2 + 1} \sqrt{[v_{\text{inc}}(v_{\text{met}})]_i^2 + 1}} \right) \theta_i, \quad (6)$$

where $\theta_i \in \mathbb{R}^+$ is a continuous hyperparameter. Otherwise, if $i \in I_{\text{dec}}$ but $i \notin I_{u,v}^{\text{inter}}$, there should be a non-zero residual distance between the two different subspaces $\mathcal{X}_{\text{inc}}(u_{\text{met}})$ and $\mathcal{X}_{\text{inc}}(v_{\text{met}})$ to ensure the kernel SPD property. To have a residual not depending on the decreed values, our model considers that there is a unit distance

$$d^{\text{alg}}([u_{\text{inc}}(u_{\text{met}})]_i, [v_{\text{inc}}(v_{\text{met}})]_i) = 1.0 \theta_i, \quad \forall i \in I_{\text{dec}} \setminus I_{u,v}^{\text{inter}}.$$

The induced meta kernel $k_{\text{met}}^{\text{alg}}(u_{\text{met}}, v_{\text{met}})$ to preserve the SPD property of k^{alg} is defined as:

$$k_{\text{met}}^{\text{alg}}(u_{\text{met}}, v_{\text{met}}) = \prod_{i \in I_{\text{met}}} k^{\text{qnt}}(1.0 \theta_i). \quad (7)$$

Not only our kernel of Eq. (2) uses less hyperparameters than the `Arc-Kernel` (as we cut off its extra parameters) but it is also a more flexible kernel as it allows different kernels for meta and decreed variables. Moreover, another advantage of our kernel is that it is numerically more stable thanks to the

new non-stationary [34] algebraic distance defined in Eq. (7) [80]. Our proposed distance also does not need any rescaling by the bounds to have values between 0 and 1.

In what comes next, we will refer to the implementation of the kernels `Arc-Kernel` and `Alg-Kernel` by `SMT Arc-Kernel` and `SMT Alg-Kernel`. We note also that the implementation of `SMT Arc-Kernel` differs slightly from the original `Arc-Kernel` as we fixed some hyperparameters to 1 in order to avoid adding extra hyperparameters and use the formulation of Eq. (2) and rescaling of the data.

4.2.4. Illustration on the MLP problem

In this section, we illustrate the hierarchical `Arc-Kernel` on the MLP example. For that sake, we consider two design variables u and v such that $u = (2.10^{-4}, 0.9, \text{ReLU}, 16, 2, 55, 51)$ and $v = (5.10^{-3}, 0.8, \text{Sigmoid}, 64, 3, 50, 54, 53)$. Since the value of u_{met} (i.e., the number of hidden layers) differs from one point to another (namely, 2 for u and 3 for v), the associated variables $u_{\text{inc}}(u_{\text{met}})$ have either 2 or 3 variables for the number of neurons in each layer (namely 55 and 51 for u , and 50, 54 and 53 for the point v). In our case, 8 hyperparameters ($[R_1]_{1,2}, \theta_1, \dots, \theta_7$) will have to be optimized where k is given by Eq. (2). These 7 hyperparameters can be described using our proposed framework as follows:

- For the neutral components, we have $u_{\text{neu}} = (2.10^{-4}, 0.9, \text{ReLU}, 16)$ and $v_{\text{neu}} = (5.10^{-3}, 0.8, \text{Sigmoid}, 64)$. Therefore, for a categorical matrix kernel R_1 and a square exponential quantitative kernel,

$$\begin{aligned} k_{\text{neu}}(u_{\text{neu}}, v_{\text{neu}}) &= k^{\text{cat}}(u_{\text{neu}}^{\text{cat}}, v_{\text{neu}}^{\text{cat}})k^{\text{qnt}}(u_{\text{neu}}^{\text{qnt}}, v_{\text{neu}}^{\text{qnt}}) \\ &= [R_1]_{1,2} \exp[-\theta_1(2.10^{-4} - 5.10^{-3})^2] \\ &\quad \exp[-\theta_2(0.9 - 0.8)^2] \exp[-\theta_3(16 - 64)^2]. \end{aligned}$$

The values $[R_1]_{1,2}$, θ_1 , θ_2 and θ_3 need to be optimized. Here, $[R_1]_{1,2}$ is the correlation between “ReLU” and “Sigmoid”.

- For the meta components, we have $u_{\text{met}} = 2$ and $v_{\text{met}} = 3$. Therefore, for a square exponential quantitative kernel,

$$\begin{aligned} k_{\text{met}}(u_{\text{met}}, v_{\text{met}}) &= k^{\text{cat}}(u_{\text{met}}^{\text{cat}}, v_{\text{met}}^{\text{cat}})k^{\text{qnt}}(u_{\text{met}}^{\text{qnt}}, v_{\text{met}}^{\text{qnt}}) \\ &= \exp[-\theta_4(3 - 2)^2]. \end{aligned}$$

The value θ_4 needs to be optimized.

- For the meta-decreed kernel, we have $[u_{\text{met}}, u_{\text{inc}}(u_{\text{met}})] = [2, (55, 51)]$ and $[v_{\text{met}}, v_{\text{inc}}(v_{\text{met}})] = [3, (50, 54, 53)]$ which gives

$$\begin{aligned} & k_{\text{met,dec}}^{\text{alg}}([u_{\text{met}}, u_{\text{inc}}(u_{\text{met}})], [v_{\text{met}}, v_{\text{inc}}(v_{\text{met}})]) \\ &= k_{\text{met}}^{\text{alg}}(2, 3) k_{\text{dec}}^{\text{alg}}((55, 51), (50, 54, 53)). \end{aligned}$$

The distance $d^{\text{alg}}(51, 54) = \left(\frac{2 \times |51-54|}{\sqrt{51^2+1}\sqrt{54^2+1}} \right) \theta_6 = 2.178 \cdot 10^{-3} \theta_6$. In general, for surrogate models, and in particular in **SMT 2.0**, the input data are normalized. With a unit normalization from $[50, 55]$ to $[0, 1]$, we would have $d^{\text{alg}}(0.2, 0.8) = \left(\frac{2 \times 0.6}{\sqrt{0.2^2+1}\sqrt{0.8^2+1}} \right) \theta_6 = 0.919 \theta_6$. Similarly, we have, between 55 and 50, $d^{\text{alg}}(0, 1) = 1.414 \theta_5$. Hence, for a square exponential quantitative kernel, one gets

$$\begin{aligned} & k_{\text{met,dec}}^{\text{alg}}([u_{\text{met}}, u_{\text{inc}}(u_{\text{met}})], [v_{\text{met}}, v_{\text{inc}}(v_{\text{met}})]) \\ &= \exp[-\theta_7] \times \exp[-1.414 \theta_5] \times \exp[-0.919 \theta_6], \end{aligned}$$

where the meta induced component is $k_{\text{met}}^{\text{alg}}(u_{\text{met}}, v_{\text{met}}) = \exp[-\theta_7]$ because the decreed value 53 in v has nothing to be compared with in u as in Eq. (7). The values θ_5 , θ_6 and θ_7 need to be optimized which complete the description of the hyperparameters.

We note that for the MLP problem, **Alg-Kernel** models use 10 hyperparameters whereas the **Arc-Kernel** would require 12 hyperparameters without the meta kernel (θ_4) but with 3 extra decreed hyperparameters and the **Wedge-Kernel** would require 15 hyperparameters. For deep learning applications, a more complex perceptron with up to 10 hidden layers would require 17 hyperparameters with **SMT 2.0** models against 26 for **Arc-Kernel** and 36 for **Wedge-Kernel**. The next section illustrates the interest of our method to build a surrogate model for this neural network engineering problem.

4.3. A neural network test-case using **SMT 2.0**

In this section, we apply our models to the hyperparameters optimization of a MLP problem aforementioned of Figure 4. Within **SMT 2.0** an example illustrates this MLP problem. For the sake of showing the Kriging surrogate abilities, we implemented a dummy function with no significance to replace the real black-box that would require training a whole Neural Network (NN) with big data. This function requires a number of variables that depends on the value of the meta variable, i.e the number of hidden layers. To simplify, we have chosen only 1, 2 or 3 hidden layers and therefore, we have 3 decreed variables but deep neural networks could also be investigated as our model

can tackle a few dozen variables. A test case (*test_hierarchical_variables_NN*) shows that our model `SMT Alg-Kernel` interpolates the data properly, checks that the data dimension is correct and also asserts that the inactive decreed variables have no influence over the prediction. In Figure 5 we illustrate the usage of Kriging surrogates with hierarchical and mixed variables based on the implementation of `SMT 2.0` for *test_hierarchical_variables_NN*.

To compare the hierarchical models of `SMT 2.0` (`SMT Alg-Kernel` and `SMT Arc-Kernel`) with the state-of-the-art imputation method previously used on industrial application (`Imp-Kernel`) [12], we draw a 99 point LHS (33 points by meta level) as a training set and the validation set is a LHS of $3 \times 1000 = 3000$ points. For the `Imp-Kernel`, the decreed-excluded values are replaced by 52 because the mean value 52.5 is not an integer (by default, `SMT` rounds to the floor value). For the three methods, the precision (computed with a root-mean-square error RMSE criterion), the likelihood and the computational time are shown in Table 6. For this problem, we can see that `SMT Alg-kernel` gives better performance than the imputation method or `SMT Arc-kernel`. Also, as all methods use the same number of hyperparameters, they have similar time performances. A direct application of our modeling method is Bayesian optimization to perform quickly the hyperparameter optimization of a neural network [20].

Table 6: Results on the neural network model.

Hierarchical method	Prediction error (RMSE)	Likelihood	# of hyperparam.
<code>SMT Alg-kernel</code>	3.7610	176.11	10
<code>SMT Arc-kernel</code>	4.9208	162.01	10
<code>Imp-Kernel</code>	4.5455	170.64	10

5. Bayesian optimization within `SMT 2.0`

Efficient global optimization (EGO) is a sequential Bayesian optimization algorithm designed to find the optimum of a black-box function that may be expensive to evaluate [43]. EGO starts by fitting a Kriging model to an initial DoE, and then uses an acquisition function to select the next point to evaluate. The most used acquisition function is the expected improvement. Once a new point has been evaluated, the Kriging model is updated. Successive updates increase the model accuracy over iterations. This enrichment process repeats until a stopping criterion is met.

Because `SMT 2.0` implements Kriging models that handle mixed and hierarchical variables, we can use EGO to solve problems involving such design variables. Other Bayesian optimization algorithms often adopt approaches

```

from smt.sampling_methods import LHS
from smt.surrogate_models import KRG, MixIntKernelType, \
    MixHrcKernelType, DesignSpace, FloatVariable, \
    IntegerVariable, OrdinalVariable, CategoricalVariable
from smt.applications.mixed_integer import \
    MixedIntegerSamplingMethod, MixedIntegerKrigingModel
import f1_NN, f2_NN, f3_NN #dummy example
def test_hierarchical_variables_NN(self):
    def dummy_f(x):
        if x[0] == 1:
            y=(f1_NN(x[1], x[2], x[3], 2**x[4], x[5]))
        elif x[0] == 2:
            y=(f2_NN(x[1], x[2], x[3], 2**x[4], x[5], x[6]))
        elif x[0] == 3:
            y=(f3_NN(x[1], x[2], x[3], 2**x[4], x[5], x[6], x[7]))
        return y
    # Define the mixed hierarchical design space
    ds = DesignSpace([
        IntegerVariable(1, 3), FloatVariable(1e-5, 1e-2),
        FloatVariable(0, 1),
        CategoricalVariable(["ReLU", "Sigmoid", "Tanh"]),
        IntegerVariable(3, 8), IntegerVariable(50, 55),
        IntegerVariable(50, 55), IntegerVariable(50, 55),
    ])
    # activate x5 when x0 in [2, 3]; x6 when x0 == 3
    ds.declare_decreed_var(
        decreed_var=6, meta_var=0, meta_value=[2, 3])
    ds.declare_decreed_var(7, meta_var=0, meta_value=3)
    #Perform the mixed integer sampling
    sampling = MixedIntegerSamplingMethod(
        LHS, ds, criterion="ese", random_state=42)
    Xt = sampling(100)
    Yt = dummy_f(Xt)
    #Build the surrogate
    sm = MixedIntegerKrigingModel(
        surrogate=KRG(design_space=ds, corr="abs_exp",
            categorical_kernel=MixIntKernelType.HOMO_HSPHERE,
            hierarchical_kernel=MixHrcKernelType.ALG_KERNEL)
        sm.set_training_values(Xt, Yt)
        sm.train()
    # Check prediction accuracy
    y_s = sm.predict_values(Xt)
    pred_RMSE = np.linalg.norm(y_s - Yt) / len(Yt)
    y_sv = sm.predict_variances(Xt)
    var_RMSE = np.linalg.norm(y_sv) / len(Yt)
    assert pred_RMSE < 1e-7
    assert var_RMSE < 1e-7

```

Figure 5: Example of usage of Hierarchical and Mixed Kriging surrogate.

based on solving subproblems with continuous or non-hierarchical Kriging. This subproblem approach is less efficient and scales poorly, but it can only solve simple problems. Several Bayesian optimization software packages can handle mixed or hierarchical variables with such a subproblem approach. The packages include BoTorch [3], SMAC [54], Trieste [65], HEBO [22], OpenBox [40], and Dragonfly [44].

5.1. A mixed optimization problem

Figure 6 compares the four EGO methods implemented in SMT 2.0: SMT GD, SMT CR, SMT EHH and SMT HH. The mixed test case that illustrates Bayesian optimization is a toy test case [86] detailed in Appendix A. This test case has two variables, one continuous and one categorical with 10 levels. To assess the performance of our algorithm, we performed 20 runs with different initial DoE sampled by LHS. Every DoE consists of 5 points and we chose a budget of 55 infill points. Figure 6a plots the convergence curves for the four methods. In particular, the median is the solid line, and the first and third quantiles are plotted in dotted lines. To visualize better the data dispersion, the boxplots of the 20 best solutions after 20 evaluations are plotted in Figure 6b. As expected, the more a method is complex, the better the optimization. Both SMT HH and SMT EHH converged in around 18 evaluations whereas SMT CR and SMT GD take around 26 iterations as shown on Figure 6a. Also, the more complex the model, the closer the optimum is to the real value as shown on Figure 6b.

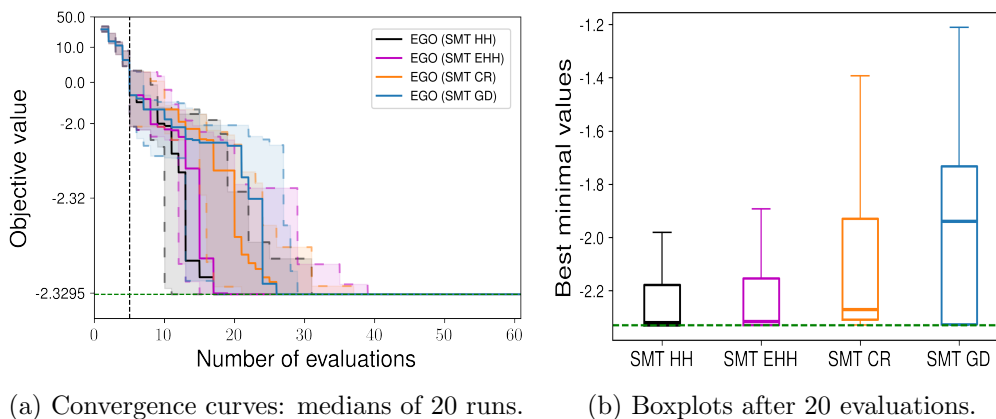


Figure 6: Optimization results for the Toy function [86].

In Figure 7 we illustrate how to use EGO with mixed variables based on the implementation of SMT 2.0. The illustrated problem is a mixed variant of the Branin function [73].


```

#Import the Mixed Integer API
from smt.surrogate_models import KRG,MixIntKernelType,\
    DesignSpace,FloatVariable,IntegerVariable
from smt.applications.mixed_integer import \
    MixedIntegerSamplingMethod as misamp
#Define the function
from smt.problems import Branin
fun = Branin(ndim=2)
#Define the mixed design space
design_space = DesignSpace([
    IntegerVariable(*fun.xlimits[0]),
    FloatVariable(*fun.xlimits[1]),
])
#Perform a mixed integer sampling with LHS
from smt.sampling_methods import LHS
smp = misamp(LHS,design_space,random_state=42)
xdoe = smp(10)
# Call the Bayesian optimizer
from smt.applications import EGO
criterion = "EI" #'EI' or 'SBO' or 'LCB'
ego = EGO(xdoe=xdoe,
    n_iter=20,
    criterion="EI",
    random_state=42,
    surrogate=KRG(design_space=design_space,
        categorical_kernel=MixIntKernelType.GOWER))
x_opt, y_opt, _, _, _ = ego.optimize(fun=fun)
# Check if the result is correct
self.assertAlmostEqual(0.494, float(y_opt), delta=1)

```

Figure 7: Example of usage of mixed surrogates for Bayesian optimization.

Note that a dedicated notebook is available to reproduce the results presented in this paper and the mixed integer notebook also includes an extra mechanical application with composite materials [74]¹⁰.

5.2. A hierarchical optimization problem

The hierarchical test case considered in this paper to illustrate Bayesian optimization is a modified Goldstein function [64] detailed in Appendix B.

¹⁰https://colab.research.google.com/github/SMTorg/smt/blob/master/tutorial/SMT_MixedInteger_application.ipynb

The resulting optimization problem involves 11 variables: 5 are continuous, 4 are integer (ordinal) and 2 are categorical. These variables consist in 6 neutral variables, 1 dimensional (or meta) variable and 4 decreed variables. Depending on the meta variable values, 4 different sub-problems can be identified. The optimization problem is given by:

$$\begin{aligned}
& \min f(x_{\text{neu}}^{\text{cat}}, x_{\text{neu}}^{\text{qnt}}, x_m^{\text{cat}}, x_{\text{dec}}^{\text{qnt}}) \\
& \text{w.r.t. } x_{\text{neu}}^{\text{cat}} = w_2 \in \{0, 1\} \\
& \quad x_{\text{neu}}^{\text{qnt}} = (x_1, x_2, x_5, z_3, z_4) \in \{0, 100\}^3 \times \{0, 1, 2\}^2 \\
& \quad x_m^{\text{cat}} = w_1 \in \{0, 1, 2, 3\} \\
& \quad x_{\text{dec}}^{\text{qnt}} = (x_3, x_4, z_1, z_2) \in \{0, 100\}^2 \times \{0, 1, 2\}^2
\end{aligned} \tag{8}$$

Compared to the model choice of Pelamatti et al. [64], we chose to model x_5 and w_2 as neutral variables even if f does not depend on x_5 when $w_2 = 0$. Other modeling choices are kept; for example, w_2 is a so-called “binary variable” and not a categorical one [61]. Similarly, we also keep the formulation of w_1 as a categorical variable but a better model would be to model it as a “cyclic variable” [78]. The resulting problem is described in Appendix B. To assess the performance of our algorithm, we performed 20 runs with different initial DoE sampled by LHS. Every DoE consists of $n + 1 = 12$ points and we chose a budget of $5n = 55$ infill points. To compare our method with a baseline, we also tested the random search method thanks to the `expand_lhs` new method [21] described in Section 6.1 and we also optimized the Goldstein function using EGO with a classic Kriging model based on imputation method (`Imp-Kernel`). This method replaces the decreed-excluded variables by their mean values: 50 or 1 respectively for (x_3, x_4) and (z_1, z_2) . Figure 8a plots the convergence curves for the four methods. In particular, the median is the solid line and the first and third quantiles are plotted in dotted lines. To visualize better the corresponding data dispersion, the boxplots of the 20 best solutions are plotted in Figure 8b. The results in Figure 8 show that the hierarchical Kriging models of **SMT 2.0** lead to better results than the imputation method or the random search both in terms of final objective value and variance over the 20 runs and in term of convergence rate. More precisely, **SMT Arc-Kernel** and **SMT Alg-Kernel** Kriging model gave the best EGO results and managed to converge correctly as shown in Figure 8b. More precisely, the 20 sampled DoEs led to similar performance and from one DoE, the method **SMT Alg-Kernel** managed to find the true minimum. However, this result has not been reproduced in other runs and is therefore not statistically significant. The variance between the runs is of similar magnitude regardless of the considered methods.

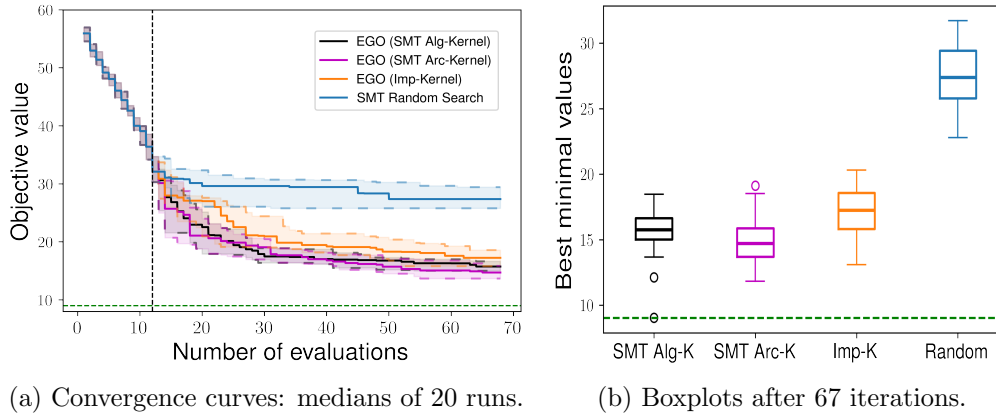


Figure 8: Optimization results for the hierarchical Goldstein function.

6. Other relevant contributions in SMT 2.0

The new release SMT 2.0 introduces several improvements besides Kriging for hierarchical and mixed variables. This section details the most important new contributions. Recall from Section 2.2 that five sub-modules are present in the code: `Sampling`, `Problems`, `Surrogate Models`, `Applications` and `Notebooks`.

6.1. Contributions to *Sampling*

Pseudo-random Sampling. The Latin Hypercube Sampling (LHS) is a stochastic sampling technique to generate quasi-random sampling distributions. It is among the most popular sampling method in computer experiments thanks to its simplicity and projection properties with high-dimensional problems. The LHS method uses the `pyDOE` package (Design Of Experiments for Python). Five criteria for the construction of LHS are implemented in SMT. The first four criteria (`center`, `maximin`, `centermaximin`, `correlation`) are the same as in `pyDOE`¹¹. The last criterion `ese`, is implemented by the authors of SMT [41]. In SMT 2.0 a new LHS method was developed for the Nested design of experiments (`NestedLHS`) [58] to use with multi-fidelity surrogates. A new mathematical method (`expand_lhs`) [21] was developed in SMT 2.0 to increase the size of a design of experiments while maintaining the `ese` property. Moreover, we proposed a sampling method for mixed variables, and the aforementioned LHS method was applied to hierarchical variables in Figure 8.

¹¹<https://pythonhosted.org/pyDOE/index.html>

6.2. Contributions to *Surrogate models*

New kernels and their derivatives for Kriging. Kriging surrogates are based on hyperparameters and on a correlation kernel. Four correlation kernels are now implemented in SMT 2.0 [53]. In SMT, these correlation functions are absolute exponential (`abs_exp`), Gaussian (`squar_exp`), Matern 5/2 (`matern52`) and Matern 3/2 (`matern32`). In addition, the implementation of gradient and Hessian for each kernel makes it possible to calculate both the first and second derivatives of the GP likelihood with respect to the hyperparameters [9].

Variance derivatives for Kriging. To perform uncertainty quantification for system analysis purposes, it could be interesting to know more about the variance derivatives of a model [55, 4, 13]. For that purpose and also to pursue the original publication about derivatives [9], SMT 2.0 extends the derivative support to Kriging variances and kernels.

Noisy Kriging. In engineering and in big data contexts with real experiments, surrogate models for noisy data are of significant interest. In particular, there is a growing need for techniques like noisy Kriging and noisy Multi-Fidelity Kriging (MFK) for data fusion [66]. For that purpose, SMT 2.0 has been designed to accommodate Kriging and MFK to noisy data including the option to incorporate heteroscedastic noise (using the `use_het_noise` option) and to account for different noise levels for each data source [21].

Kriging with partial least squares. Beside MGP, for high-dimensional problems, the toolbox implements Kriging with partial least squares (KPLS) [7] and its extension KPLSK [8]. The PLS information is computed by projecting the data into a smaller space spanned by the principal components. By integrating this PLS information into the Kriging correlation matrix, the number of inputs can be scaled down, thereby reducing the number of hyperparameters required. The resulting number of hyperparameters d_e is indeed much smaller than the original problem dimension d . Recently, in SMT 2.0, we extended the KPLS method for multi-fidelity Kriging (MFKPLS and MFKPLSK) [58, 16, 17]. We also proposed an automatic criterion to choose automatically the reduced dimension d_e based on Wold's R criterion [82]. This criterion has been applied to aircraft optimization with EGO where the number d_e (`n_comp` in the code) for the model is automatically selected at every iteration [76]. Special efforts have been made to accommodate KPLS for multi-fidelity and mixed integer data [16, 17].

Marginal Gaussian process. SMT 2.0 implements Marginal Gaussian Process (MGP) surrogate models for high dimensional problems [67]. MGP are Gaussian processes taking into account hyperparameters uncertainty defined as a density probability function. Especially we suppose that the function to model $f : \Omega \mapsto \mathbb{R}$, where $\Omega \subset \mathbb{R}^d$ and d is the number of design variables, lies in a linear embedding \mathcal{A} such as $\mathcal{A} = \{u = Ax, x \in \Omega\}$, $A \in \mathbb{R}^{d \times d_e}$ and $f(x) = f_{\mathcal{A}}(Ax)$ with $f_{\mathcal{A}} : \mathcal{A} \mapsto \mathbb{R}$ and $d_e \ll d$. Then, we must use a kernel $k(x, x') = k_{\mathcal{A}}(Ax, Ax')$ whose each component of the transfer matrix A is an hyperparameter. Thus we have $d_e \times d$ hyperparameters to find. Note that d_e is defined as `n_comp` in the code [29].

Gradient-enhanced neural network. The new release SMT 2.0 implements Gradient-Enhanced Neural Network (GENN) models [10]. Gradient-Enhanced Neural Networks (GENN) are fully connected multi-layer perceptrons whose training process was modified to account for gradient information. Specifically, the model is trained to minimize not only the prediction error of the response but also the prediction error of the partial derivatives: the chief benefit of gradient enhancement is better accuracy with fewer training points. Note that GENN applies to regression (single-output or multi-output), but not classification since there is no gradient in that case. The implementation is fully vectorized and uses ADAM optimization, mini-batch, and L2-norm regularization. For example, GENN can be used to learn airfoil geometries from a database. This usage is documented in SMT 2.0 ¹².

6.3. Contributions to Applications

Kriging trajectory and sampling. Sampling a GP with high resolution is usually expensive due to the large dimension of the associated covariance matrix. Several methods are proposed to draw samples of a GP on a given set of points. To sample a conditioned GP, the classic method consists in using a Cholesky decomposition (or eigendecomposition) of the conditioned covariance matrix of the process but some numerical computational errors can lead to non SPD matrix. A more recent approach based on Karhunen-Loève decomposition of the covariance kernel with the Nyström method has been proposed in [5] where the paths can be sampled by generating independent standard Normal distributed samples. The different methods are documented in the tutorial *Gaussian Process Trajectory Sampling* [59].

¹²https://smt.readthedocs.io/en/latest/_src_docs/examples/airfoil_parameters/learning_airfoil_parameters.html

Parallel Bayesian optimization. Due to the recent progress made in hardware configurations, it has been of high interest to perform parallel optimizations. A parallel criterion called qEI [31] was developed to perform Efficient Global Optimization (EGO): the goal is to be able to run batch optimization. At each iteration of the algorithm, multiple new sampling points are extracted from the known ones. These new sampling points are then evaluated using a parallel computing environment. Five criteria are implemented in SMT 2.0: Kriging Believer (KB), Kriging Believer Upper Bound (KBUB), Kriging Believer Lower Bound (KBLB), Kriging Believer Random Bound (KBRand) and Constant Liar (CLmin) [72].

7. Conclusion

SMT 2.0 introduces significant upgrades to the Surrogate Modeling Toolbox. This new release adds support for hierarchical and mixed variables and improves the surrogate models with a particular focus on Kriging (Gaussian process) models. SMT 2.0 is distributed through an open-source license and is freely available online¹³. We provide documentation that caters to both users and potential developers¹⁴. SMT 2.0 enables users and developers collaborating on the same project to have a common surrogate modeling tool that facilitates the exchange of methods and reproducibility of results.

SMT has been widely used in aerospace and mechanical modeling applications. Moreover, the toolbox is general and can be useful for anyone who needs to use or develop surrogate modeling techniques, regardless of the targeted applications. SMT is currently the only open-source toolbox that can build hierarchical and mixed surrogate models.

Data availability

Data will be made available on request. Results can be reproduced freely online at https://colab.research.google.com/github/SMTorg/smt/blob/master/tutorial/NotebookRunTestCases_Paper_SMT_v2.ipynb.

Supplementary material

Supplementary material related to this article can be found online at <https://doi.org/10.1016/j.advengsoft.2023.103571>.

¹³<https://github.com/SMTorg/SMT>

¹⁴<https://smt.readthedocs.io/en/latest/>

Acknowledgements

We want to thank all those who contribute to this release. Namely, M. A. Bouhlel, I. Cardoso, R. Carreira Rufato, R. Charayron, R. Conde Arenzana, S. Dubreuil, A. F. López-Lopera, M. Meliani, M. Menz, N. Moëlle, A. Thouvenot, R. Priem, E. Roux and F. Vergnes. This work is part of the activities of ONERA - ISAE - ENAC joint research group. We also acknowledge the partners institutions: ONERA, NASA Glenn, ISAE-SUPAERO, Institut Clément Ader (ICA), the University of Michigan, Polytechnique Montréal and the University of California San Diego.

The research presented in this paper has been performed in the framework of the AGILE 4.0 project (Towards cyber-physical collaborative aircraft development), funded by the European Union Horizon 2020 research and innovation framework programme under grant agreement n° 815122 and in the COLOSSUS project (Collaborative System of Systems Exploration of Aviation Products, Services and Business Models) funded by the European Union Horizon Europe research and innovation framework programme under grant agreement n° 101097120.

We also are grateful to E. Hallé-Hannan from Polytechnique Montréal for the hierarchical variables framework.

Appendix A. Toy test function

This Appendix gives the detail of the toy function of Section 5.1¹⁵. First, we recall the optimization problem:

$$\begin{aligned} \min f(x^{\text{cat}}, x^{\text{qnt}}) \\ \text{w.r.t. } x^{\text{cat}} = c_1 \in \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\} \\ x^{\text{qnt}} = x_1 \in [0, 1] \end{aligned} \quad (\text{A.1})$$

The toy function f is defined as

$$\begin{aligned} f(x, c_1) = & \mathbb{1}_{c_1=0} \cos(3.6\pi(x-2)) + x - 1 \\ & + \mathbb{1}_{c_1=1} 2 \cos(1.1\pi \exp(x)) - \frac{x}{2} + 2 \\ & + \mathbb{1}_{c_1=2} \cos(2\pi x) + \frac{1}{2}x \\ & + \mathbb{1}_{c_1=3} x(\cos(3.4\pi(x-1)) - \frac{x-1}{2}) \\ & + \mathbb{1}_{c_1=4} - \frac{x^2}{2} \\ & + \mathbb{1}_{c_1=5} 2 \cos(0.25\pi \exp(-x^4))^2 - \frac{x}{2} + 1 \\ & + \mathbb{1}_{c_1=6} x \cos(3.4\pi x) - \frac{x}{2} + 1 \\ & + \mathbb{1}_{c_1=7} - x(\cos(3.5\pi x) + \frac{x}{2}) + 2 \\ & + \mathbb{1}_{c_1=8} - \frac{x^5}{2} + 1 \\ & + \mathbb{1}_{c_1=9} - \cos(2.5\pi x)^2 \sqrt{x} - 0.5 \ln(x+0.5) - 1.3 \end{aligned} \quad (\text{A.2})$$

¹⁵<https://github.com/jbussemaker/SBArchOpt>

Appendix B. Hierarchical Goldstein test function

This Appendix gives the detail of the hierarchical Goldstein problem of Section 5.2¹⁶. First, we recall the optimization problem:

$$\begin{aligned}
& \min f(x_{\text{neu}}^{\text{cat}}, x_{\text{neu}}^{\text{qnt}}, x_m^{\text{cat}}, x_{\text{dec}}^{\text{qnt}}) \\
& \text{w.r.t. } x_{\text{neu}}^{\text{cat}} = w_2 \in \{0, 1\} \\
& \quad x_{\text{neu}}^{\text{qnt}} = (x_1, x_2, x_5, z_3, z_4) \in [0, 100]^3 \times \{0, 1, 2\}^2 \\
& \quad x_m^{\text{cat}} = w_1 \in \{0, 1, 2, 3\} \\
& \quad x_{\text{dec}}^{\text{qnt}} = (x_3, x_4, z_1, z_2) \in [0, 100]^2 \times \{0, 1, 2\}^2
\end{aligned} \tag{B.1}$$

The hierarchical and mixed function f is defined as a hierarchical function that depends on f_0 , f_1 , f_2 and $Gold_{\text{cont}}$ as describes in the following.

$$\begin{aligned}
& f(x_1, x_2, x_3, x_4, z_1, z_2, z_3, z_4, x_5, w_1, w_2) = \\
& \quad \mathbb{1}_{w_1=0} f_0(x_1, x_2, z_1, z_2, z_3, z_4, x_5, w_2) \\
& \quad + \mathbb{1}_{w_1=1} f_1(x_1, x_2, x_3, z_2, z_3, z_4, x_5, w_2) \\
& \quad + \mathbb{1}_{w_1=2} f_2(x_1, x_2, x_4, z_1, z_3, z_4, x_5, w_2) \\
& \quad + \mathbb{1}_{w_1=3} Gold_{\text{cont}}(x_1, x_2, x_3, x_4, z_3, z_4, x_5, w_2).
\end{aligned} \tag{B.2}$$

Then, the functions f_0 , f_1 and f_2 are defined as mixed variants of $Gold_{\text{cont}}$ as such

$$\begin{aligned}
& f_0(x_1, x_2, z_1, z_2, z_3, z_4, x_5, w_2) = \\
& \quad \mathbb{1}_{z_2=0} (\mathbb{1}_{z_1=0} Gold_{\text{cont}}(x_1, x_2, 20, 20, z_3, z_4, x_5, w_2) \\
& \quad \quad + \mathbb{1}_{z_1=1} Gold_{\text{cont}}(x_1, x_2, 50, 20, z_3, z_4, x_5, w_2) \\
& \quad \quad + \mathbb{1}_{z_1=2} Gold_{\text{cont}}(x_1, x_2, 80, 20, z_3, z_4, x_5, w_2)) \\
& \quad \mathbb{1}_{z_2=1} (\mathbb{1}_{z_1=0} Gold_{\text{cont}}(x_1, x_2, 20, 50, z_3, z_4, x_5, w_2) \\
& \quad \quad + \mathbb{1}_{z_1=1} Gold_{\text{cont}}(x_1, x_2, 50, 50, z_3, z_4, x_5, w_2) \\
& \quad \quad + \mathbb{1}_{z_1=2} Gold_{\text{cont}}(x_1, x_2, 80, 50, z_3, z_4, x_5, w_2)) \\
& \quad \mathbb{1}_{z_2=2} (\mathbb{1}_{z_1=0} Gold_{\text{cont}}(x_1, x_2, 20, 80, z_3, z_4, x_5, w_2) \\
& \quad \quad + \mathbb{1}_{z_1=1} Gold_{\text{cont}}(x_1, x_2, 50, 80, z_3, z_4, x_5, w_2) \\
& \quad \quad + \mathbb{1}_{z_1=2} Gold_{\text{cont}}(x_1, x_2, 80, 80, z_3, z_4, x_5, w_2)) \\
& f_1(x_1, x_2, x_3, z_2, z_3, z_4, x_5, w_2) = \\
& \quad \mathbb{1}_{z_2=0} Gold_{\text{cont}}(x_1, x_2, x_3, 20, z_3, z_4, x_5, w_2) \\
& \quad \quad + \mathbb{1}_{z_2=1} Gold_{\text{cont}}(x_1, x_2, x_3, 50, z_3, z_4, x_5, w_2) \\
& \quad \quad + \mathbb{1}_{z_2=2} Gold_{\text{cont}}(x_1, x_2, x_3, 80, z_3, z_4, x_5, w_2) \\
& f_2(x_1, x_2, x_4, z_1, z_3, z_4, x_5, w_2) = \\
& \quad \mathbb{1}_{z_1=0} Gold_{\text{cont}}(x_1, x_2, 20, x_4, z_3, z_4, x_5, w_2) \\
& \quad \quad + \mathbb{1}_{z_1=1} Gold_{\text{cont}}(x_1, 50, x_2, x_4, z_3, z_4, x_5, w_2) \\
& \quad \quad + \mathbb{1}_{z_1=2} Gold_{\text{cont}}(x_1, x_2, 80, x_4, z_3, z_4, x_5, w_2)
\end{aligned} \tag{B.3}$$

¹⁶<https://github.com/jbussemaker/SBArchOpt>

To finish with, the function $Gold_{\text{cont}}$ is given by

$$\begin{aligned}
Gold_{\text{cont}}(x_1, x_2, x_3, x_4, z_3, z_4, x_5, w_2) = & 53.3108 + 0.184901x_1 \\
& - 5.02914x_1^3 \cdot 10^{-6} + 7.72522x_1^{z_3} \cdot 10^{-8} - 0.0870775x_2 - 0.106959x_3 \\
& + 7.98772x_3^{z_4} \cdot 10^{-6} + 0.00242482x_4 + 1.32851x_4^3 \cdot 10^{-6} - 0.00146393x_1x_2 \\
& - 0.00301588x_1x_3 - 0.00272291x_1x_4 + 0.0017004x_2x_3 + 0.0038428x_2x_4 \\
& - 0.000198969x_3x_4 + 1.86025x_1x_2x_3 \cdot 10^{-5} - 1.88719x_1x_2x_4 \cdot 10^{-6} \\
& + 2.50923x_1x_3x_4 \cdot 10^{-5} - 5.62199x_2x_3x_4 \cdot 10^{-5} + w_2 \left(5 \cos \left(\frac{2\pi}{100}x_5 \right) - 2 \right).
\end{aligned} \tag{B.4}$$

References

- [1] B. Adams, W. Bohnhoff, K. Dalbey, M. Ebeida, J. Eddy, M. Eldred, R. Hooper, P. Patricia, K. Hu, J. Jakeman, and al. Dakota, a multilevel parallel object-oriented framework for design optimization, parameter estimation, uncertainty quantification, and sensitivity analysis: Version 6.13 user’s manual. Technical report, Sandia National Lab.(SNL-NM), Albuquerque, NM (United States), 2020.
- [2] C. Audet, E. Hallé-Hannan, and S. Le Digabel. A general mathematical framework for constrained mixed-variable blackbox optimization problems with meta and categorical variables. *Operations Research Forum*, 4:1–37, 2023.
- [3] M. Balandat, B. Karrer, D. Jiang, S. Daulton, B. Letham, A. Wilson, and E. Bakshy. Botorch: A framework for efficient monte-carlo bayesian optimization. *Advances in neural information processing systems*, 33:21524–21538, 2020.
- [4] G. Berthelin, S. Dubreuil, M. Salaün, N. Bartoli, and C. Gogu. Disciplinary proper orthogonal decomposition and interpolation for the resolution of parameterized multidisciplinary analysis. *International Journal for Numerical Methods in Engineering*, 123:3594–3626, 2022.
- [5] W. Betz, I. Papaioannou, and D. Straub. Numerical methods for the discretization of random fields by means of the karhunen–loève expansion. *Computer Methods in Applied Mechanics and Engineering*, 271:109–129, 2014.
- [6] M. A. Bouhleb and J. Martins. Gradient-enhanced kriging for high-dimensional problems. *Engineering with Computers*, 35:157–173, 2019.
- [7] M. A. Bouhleb, N. Bartoli, R. Regis, A. Otsmane, and J. Morlier. An improved approach for estimating the hyperparameters of the kriging model for high-dimensional problems through the partial least squares method. *Mathematical Problems in Engineering*, 2016:6723410, 2016.
- [8] M. A. Bouhleb, N. Bartoli, R. Regis, A. Otsmane, and J. Morlier. Efficient global optimization for high-dimensional constrained problems by using the kriging models combined with the partial least squares method. *Engineering Optimization*, 50:2038–2053, 2018.

- [9] M. A. Bouhlef, J. T. Hwang, N. Bartoli, R. Lafage, J. Morlier, and J. R. A. Martins. A python surrogate modeling framework with derivatives. *Advances in Engineering Software*, 135:102662, 2019.
- [10] M. A. Bouhlef, S. He, and J. Martins. Scalable gradient-enhanced artificial neural networks for airfoil shape design in the subsonic and transonic regimes. *Structural and Multidisciplinary Optimization*, 61:1363–1376, 2020.
- [11] J. H. Bussemaker, P. D. Ciampa, and B. Nagel. System architecture design space exploration: An approach to modeling and optimization. In *AIAA AVIATION 2020 FORUM*, 2020.
- [12] J. H. Bussemaker, N. Bartoli, T. Lefebvre, P. D. Ciampa, and B. Nagel. Effectiveness of surrogate-based optimization algorithms for system architecture optimization. In *AIAA AVIATION 2021 FORUM*, 2021.
- [13] I. Cardoso, S. Dubreuil, N. Bartoli, C. Gogu, M. Salaün, and R. Lafage. Disciplinary surrogates for gradient-based optimization of multidisciplinary systems. In *ECCOMAS Aerobest*, 2023.
- [14] A. Chan, A. F. Pires, and T. Polacsek. Trying to elicit and assign goals to the right actors. In *Conceptual Modeling: 41st International Conference, ER 2022*, 2022.
- [15] T. H. Chang and S. M. Wild. ParMOO: A Python library for parallel multiobjective simulation optimization. *Journal of Open Source Software*, 8:4468, 2023.
- [16] R. Charayron, T. Lefebvre, N. Bartoli, and J. Morlier. Multi-fidelity bayesian optimization strategy applied to overall drone design. In *AIAA SciTech 2023 Forum*, 2023.
- [17] R. Charayron, T. Lefebvre, N. Bartoli, and J. Morlier. Towards a multi-fidelity and multi-objective bayesian optimization efficient algorithm. *Aerospace Science and Technology*, 142:108673, 2023.
- [18] Y. Chen, F. Dababneh, B. Zhang, S. Kassaei, B. T. Smith, K. Liu, and A. M. Momen. Surrogate modeling for capacity planning of charging station equipped with photovoltaic panel and hydropneumatic energy storage. *Journal of Energy Resources Technology*, 142:050907, 2020.
- [19] G. H. Cheng, A. Younis, K. H. Hajikolaie, and G. G. Wang. Trust region based mode pursuing sampling method for global optimization of high dimensional design problems. *Journal of Mechanical Design*, 137:021407, 2015.
- [20] H. Cho, Y. Kim, E. Lee, D. Choi, Y. Lee, and W. Rhee. Basic enhancement strategies when using bayesian optimization for hyperparameter tuning of deep neural networks. *IEEE access*, 8:52588–52608, 2020.
- [21] R. Conde Arenzana, A. López-Lopera, S. Mouton, N. Bartoli, and T. Lefebvre. Multi-fidelity gaussian process model for CFD and wind tunnel data fusion. In *ECCOMAS Aerobest*, 2021.

- [22] A. I. Cowen-Rivers, W. Ly, Z. Wang, R. Tutunov, H. Jianye, J. Wang, and H. B. Ammar. HEBO: Heteroscedastic Evolutionary Bayesian Optimisation, 2020.
- [23] J. Cuesta-Ramirez, R. Le Riche, O. Roustant, G. Perrin, C. Durantin, and A. Gliere. A comparison of mixed-variables bayesian optimization approaches. *Advanced Modeling and Simulation in Engineering Sciences*, 9:1–29, 2021.
- [24] X. Deng, C. D. Lin, K. Liu, and R. K. Rowe. Additive gaussian process for computer models with qualitative and quantitative factors. *Technometrics*, 59:283–292, 2017.
- [25] V. Drouet, M. Balesdent, L. Brevault, S. Dubreuil, and J. Morio. Multi-fidelity algorithm for the sensitivity analysis of multidisciplinary problems. *Journal of Mechanical Design*, 145:1–22, 2023.
- [26] J. Eliáš, M. Vořechovský, and V. Sadílek. Periodic version of the minimax distance criterion for monte carlo integration. *Advances in Engineering Software*, 149:102900, 2020.
- [27] A. Faraci, P. Beaurepaire, and N. Gayton. Review on python toolboxes for kriging surrogate modelling. In *ESREL*, 2022.
- [28] M. E. A. Fouda, E. J. Adler, J. Bussemaker, J. R. R. A. Martins, D. F. Kurtulus, L. Boggero, and B. Nagel. Automated hybrid propulsion model construction for conceptual aircraft design and optimization. In *33rd Congress of the International Council of the Aeronautical Sciences, ICAS 2022*, 2022.
- [29] R. Garnett, M. Osborne, and P. Hennig. Active learning of linear embeddings for gaussian processes. In *Uncertainty in Artificial Intelligence - Proceedings of the 30th Conference*, 2013.
- [30] E. C. Garrido-Merchán and D. Hernández-Lobato. Dealing with categorical and integer-valued variables in bayesian optimization with gaussian processes. *Neurocomputing*, 380:20–35, 2020.
- [31] D. Ginsbourger, R. Le Riche, and L. Carraro. *Kriging Is Well-Suited to Parallelize Optimization*, pages 131–162. Springer Berlin Heidelberg, 2010.
- [32] D. Gorissen, K. Crombecq, I. Couckuyt, T. Dhaene, and P. Demeester. A surrogate modeling and adaptive sampling toolbox for computer based design. *Journal of Machine Learning Research*, 11:2051–2055, 2010.
- [33] M. Halstrup. *Black-Box Optimization of Mixed Discrete-Continuous Optimization Problems*. PhD thesis, TU Dortmund, 2016.
- [34] A. Hebbal, L. Brevault, M. Balesdent, E.-G. Talbi, and N. Melab. Bayesian optimization using deep gaussian processes with applications to aerospace system design. *Optimization and Engineering*, 22:321–361, 2021.
- [35] D. Horn, J. Stork, N.-J. Schüßler, and M. Zaefferer. Surrogates for hierarchical search spaces: The wedge-kernel and an automated analysis. In *Proceedings of the Genetic and Evolutionary Computation Conference*, 2019.

- [36] Y. Hung, V. R. Joseph, and S. N. Melkote. Design and analysis of computer experiments with branching and nested factors. *Technometrics*, 51:354–365, 2009.
- [37] F. Hutter and M. A. Osborne. A kernel for hierarchical parameter spaces, 2013.
- [38] J. T. Hwang and J. R. R. A. Martins. A fast-prediction surrogate model for large datasets. *Aerospace Science and Technology*, 75:74–87, 2018.
- [39] J. Jasa, P. Bortolotti, D. Zalkind, and G. Barter. Effectively using multifidelity optimization for wind turbine design. *Wind Energy Science*, 7:991–1006, 2022.
- [40] H. Jiang, Y. Shen, Y. Li, W. Zhang, C. Zhang, and B. Cui. Openbox: A python toolkit for generalized black-box optimization, 2023.
- [41] R. Jin, W. Chen, and A. Sudjianto. An efficient algorithm for constructing optimal design of computer experiments. *Journal of Statistical Planning and Inference*, 2: 545–554, 2005.
- [42] D. R. Jones. A taxonomy of global optimization methods based on response surfaces. *Journal of Global Optimization*, 21:345–383, 2001.
- [43] D. R. Jones, M. Schonlau, and W. J. Welch. Efficient global optimization of expensive black-box functions. *Journal of Global Optimization*, 13:455–492, 1998.
- [44] K. Kandasamy, K. R. Vysyaraju, W. Neiswanger, B. Paria, C. Collins, J. Schneider, B. Poczos, and E. Xing. Tuning hyperparameters without grad students: Scalable and robust bayesian optimisation with dragonfly. *Journal of Machine Learning Research*, 21:3098–3124, 2020.
- [45] P. Karban, D. Pánek, T. Orosz, I. Petrášová, and I. Doležel. Fem based robust design optimization with agros and artap. *Computers & Mathematics with Applications*, 81: 618–633, 2021.
- [46] R. Karlsson, L. Bliik, S. Verwer, and M. de Weerdt. Continuous surrogate-based optimization algorithms are well-suited for expensive discrete problems. In *Artificial Intelligence and Machine Learning*, 2021.
- [47] M. Kennedy and A. O’Hagan. Bayesian calibration of computer models. *J.R. Statist. Soc. B*, 63:425–464, 2001.
- [48] M. Krügener, J. Zapata Usandivaras, , M. Bauerheim, and A. Urbano. Coaxial-injector surrogate modeling based on reynolds-averaged navier–stokes simulations using deep learning. *Journal of Propulsion and Power*, 38:783–798, 2022.
- [49] J. Kudela and R. Matousek. Recent advances and applications of surrogate models for finite element method computations: a review. *Soft Computing*, 26:13709–13733, 2022.
- [50] L. S. Kwan, A. Pitrou, and S. Seibert. Numba: A llvm-based python jit compiler. In *Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC*, 2015.

- [51] R. Lafage. egobox, a Rust toolbox for efficient global optimization. *Journal of Open Source Software*, 7:4737, 2022.
- [52] C. Lataniotis, S. Marelli, and B. Sudret. Uqlab 2.0 and uqcloud: open-source vs. cloud-based uncertainty quantification. In *SIAM Conference on Uncertainty Quantification (SIAM UQ 2022)*, 2022.
- [53] H. Lee. *Gaussian Processes*, pages 575–577. Springer Berlin Heidelberg, 2011.
- [54] M. Lindauer, K. Eggensperger, M. Feurer, A. B., D. Deng, C. Benjamins, T. Ruhkopf, R. Sass, and F. Hutter. SMAC3: A versatile Bayesian optimization package for hyperparameter optimization. *Journal of Machine Learning Research*, 23:1–9, 2022.
- [55] A. F. López-Lopera, D. Idier, J. Rohmer, and F. Bachoc. Multioutput gaussian processes with functional data: A study on coastal flood hazard assessment. *Reliability Engineering & System Safety*, 218:108139, 2022.
- [56] C. A. Mader, J. R. R. A. Martins, J. J. Alonso, and E. van der Weide. ADjoint: An approach for the rapid development of discrete adjoint solvers. *AIAA Journal*, 46: 863–873, 2008.
- [57] J. R. R. A. Martins and A. Ning. *Engineering design optimization*. Cambridge University Press, 2021.
- [58] M. Meliani, N. Bartoli, T. Lefebvre, M. A. Bouhlef, J. R. R. A. Martins, and J. Morlier. Multi-fidelity efficient global optimization: Methodology and application to airfoil shape design. In *AIAA AVIATION 2019 FORUM*, 2019.
- [59] M. Menz, S. Dubreuil, J. Morio, C. Gogu, N. Bartoli, and M. Chiron. Variance based sensitivity analysis for monte carlo and importance sampling reliability assessment with gaussian processes. *Structural Safety*, 93:102116, 2021.
- [60] D. Ming, D. Williamson, and S. Guillas. Deep gaussian process emulation using stochastic imputation. *Technometrics*, 0:1–12, 2022.
- [61] J. Müller, C. A. Shoemaker, and R. Piché. SO-MI: A surrogate model algorithm for computationally expensive nonlinear mixed-integer black-box global optimization problems. *Computers & Operations Research*, 40:1383–1400, 2013.
- [62] F. Pedregosa, G. Varoquaux, A. Gramfort, V. M. B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.
- [63] J. Pelamatti, L. Brevault, M. Balesdent, E.-G. Talbi, and Y. Guerin. Efficient global optimization of constrained mixed variable problems. *Journal of Global Optimization*, 73:583–613, 2019.
- [64] J. Pelamatti, L. Brevault, M. Balesdent, E.-G. Talbi, and Y. Guerin. Bayesian optimization of variable-size design space problems. *Optimization and Engineering*, 22: 387–447, 2021.

- [65] V. Picheny, J. Berkeley, H. Moss, H. Stojic, U. Granta, S. Ober, A. Artemev, K. Ghani, A. Goodall, A. Paleyes, and al. Trieste: Efficiently exploring the depths of black-box functions with tensorflow, 2023.
- [66] J. Platt, S. Penny, T. Smith, T. Chen, and H. Abarbanel. A systematic exploration of reservoir computing for forecasting complex spatiotemporal dynamics. *Neural Networks*, 153:530–552, 2022.
- [67] R. Priem, Y. Diouane, N. Bartoli, S. Dubreuil, and P. Saves. High-dimensional efficient global optimization using both random and supervised embeddings. In *AIAA AVIATION 2023 Forum*, 2023.
- [68] F. Rapisarda, D. Brigo, and F. Mercurio. Parameterizing correlations: a geometric interpretation. *IMA Journal of Management Mathematics*, 18:55–73, 2007.
- [69] R. Rebonato and P. Jaeckel. The most general methodology to create a valid correlation matrix for risk management and option pricing purposes. *Journal of Risk*, 2: 17–27, 2001.
- [70] O. Roustant, D. Ginsbourger, and Y. Deville. Dicekriging, diceoptim: Two r packages for the analysis of computer experiments by kriging-based metamodeling and optimization. *Journal of statistical software*, 51:1–55, 2012.
- [71] O. Roustant, E. Padonou, Y. Deville, A. Clément, G. Perrin, J. Giorla, and H. Wynn. Group kernels for gaussian process metamodels with categorical inputs. *SIAM Journal on Uncertainty Quantification*, 8:775–806, 2020.
- [72] E. Roux, Y. Tillier, S. Kraria, and P.-O. Bouchard. An efficient parallel global optimization strategy based on kriging properties suitable for material parameters identification. *Archive of Mechanical Engineering*, 67, 2020.
- [73] S. Roy, W. A. Crossley, B. K. Stanford, K. T. Moore, and J. S. Gray. A mixed integer efficient global optimization algorithm with multiple infill strategy - applied to a wing topology optimization problem. In *AIAA SciTech 2019 Forum*, 2019.
- [74] R. C. Rufato, Y. Diouane, J. Henry, R. Ahlfeld, and J. Morlier. A mixed-categorical data-driven approach for prediction and optimization of hybrid discontinuous composites performance. In *AIAA AVIATION 2022 Forum*, 2022.
- [75] T. Savage, H. F. Almeida-Trasvina, E. A. del Río-Chanona, R. Smith, and D. Zhang. An adaptive data-driven modelling and optimization framework for complex chemical process design. *Computer Aided Chemical Engineering*, 48:73–78, 2020.
- [76] P. Saves, E. Nguyen Van, N. Bartoli, Y. Diouane, T. Lefebvre, C. David, S. Defoort, and J. Morlier. Bayesian optimization for mixed variables using an adaptive dimension reduction process: applications to aircraft design. In *AIAA SciTech 2022 Forum*, 2022.
- [77] P. Saves, Y. Diouane, N. Bartoli, T. Lefebvre, and J. Morlier. A mixed-categorical correlation kernel for gaussian process. *Neurocomputing*, 550:126472, 2023.

- [78] T. Tran, D. Sinoquet, S. Da Veiga, and M. Mongeau. Derivative-free mixed binary necklace optimization for cyclic-symmetry optimal design problems. *Optimization and Engineering*, 2021.
- [79] W. Wang, G. Tao, D. Ke, J. Luo, and J. Cui. Transpiration cooling of high pressure turbine vane with optimized porosity distribution. *Applied Thermal Engineering*, 223: 119831, 2023.
- [80] N. Wildberger. A rational approach to trigonometry. *Math Horizons*, 15:16–20, 2007.
- [81] C. K. Williams and C. E. Rasmussen. *Gaussian processes for machine learning*. MIT press Cambridge, MA, 2006.
- [82] H. Wold. Soft modelling by latent variables: The non-linear iterative partial least squares (nipals) approach. *Journal of Applied Probability*, 12:117–142, 1975.
- [83] M. Zaefferer and D. Horn. A first analysis of kernels for kriging-based optimization in hierarchical search spaces, 2018.
- [84] Y. Zhang, S. Tao, W. Chen, and D. Apley. A latent variable approach to gaussian process modeling with qualitative and quantitative factors. *Technometrics*, 62:291–302, 2020.
- [85] Q. Zhou, P. Z. G. Qian, and S. Zhou. A simple approach to emulation for computer models with qualitative and quantitative factors. *Technometrics*, 53:266–273, 2011.
- [86] M. M. Zuniga and D. Sinoquet. Global optimization for mixed categorical-continuous variables based on gaussian process models with a randomized categorical space exploration step. *INFOR: Information Systems and Operational Research*, 58:310–341, 2020.