Validation of Composite Systems by Discrepancy Propagation

David Reeb[1] **Kanil Patel**[1] **Karim Barsim**[1] **Martin Schiegg**[1] **Sebastian Gerwinn**[1]

[1]Bosch Center for Artificial Intelligence, Robert Bosch GmbH, 71272 Renningen, Germany

Abstract

Assessing the validity of a real-world system with respect to given quality criteria is a common yet costly task in industrial applications due to the vast number of required real-world tests. Validating such systems by means of simulation offers a promising and less expensive alternative, but requires an assessment of the simulation accuracy and therefore end-to-end measurements. Additionally, covariate shifts between simulations and actual usage can cause difficulties for estimating the reliability of such systems. In this work, we present a validation method that propagates bounds on distributional discrepancy measures through a composite system, thereby allowing us to derive an upper bound on the failure probability of the real system from potentially inaccurate simulations. Each propagation step entails an optimization problem, where – for measures such as maximum mean discrepancy (MMD) – we develop tight convex relaxations based on semidefinite programs. We demonstrate that our propagation method yields valid and useful bounds for composite systems exhibiting a variety of realistic effects. In particular, we show that the proposed method can successfully account for data shifts within the experimental design as well as model inaccuracies within the simulation.

1 INTRODUCTION

Industrial products cannot be released without a priori ensuring their validity, i.e. the product must be validated to work according to its specifications with high probability. Such validation is essential for safety-critical systems (e.g. autonomous cars, airplanes, medical machines) or systems with legal requirements (e.g. limits on output emissions or power consumption of new vehicle types), see e.g. [Kalra and Paddock, 2016, Koopman and Wagner, 2016, Belcastro and Belcastro, 2003]. When relying on real-world testing alone to validate system-wide requirements, one must perform enough test runs to guarantee an acceptable failure rate, e.g. at least $\sim 10^6$ runs for a guarantee below 10^{-6}. This is costly not only in terms of money but also in terms of time-to-release, especially when a failed system test necessitates further design iterations.

System validation is particularly difficult for complex systems which typically consist of multiple components, often developed and tested by different teams under varying operating conditions. For example, an advanced driver-assistance system is built from several sensors and controllers, which come from different suppliers but together must guarantee to keep the vehicle safely on the lane. Similarly, the powertrain system of a vehicle consists of the engine or battery, a controller and various catalysts or auxiliary components, but is legally required to produce low output emissions of various gases or energy consumption per distance as a whole. In both these examples, the validation of the system can also be viewed as the validation of its control component, when the other subsystems are considered fixed. To reduce the costs of real-world testing including system assembly and release delays, one can employ *simulations* of the composite system by combining models of the components, to perform *virtual validation* of the system [Wong et al., 2020].

However, it is difficult to assess how much such a composite virtual validation can be trusted, because the component models may be inaccurate w.r.t. the real-world components (*simulation model misfits*) or the simulation inputs may differ from the distribution of real-world inputs (*data-shift*). Incorporating these inaccuracies within the virtual validation analysis is particularly important for reliability analyses [Bect et al., 2012, Dubourg et al., 2013, Wang et al., 2016] in industrial applications with safety or legal relevance as those described above, where falsely judging a system to be reliable is much more expensive than false negatives. For this reason, we desire – if not an accurate estimate – then at least an upper bound on its true failure probability.

Proceedings of the 39th Conference on Uncertainty in Artificial Intelligence (UAI 2023), PMLR 216:1730–1740.

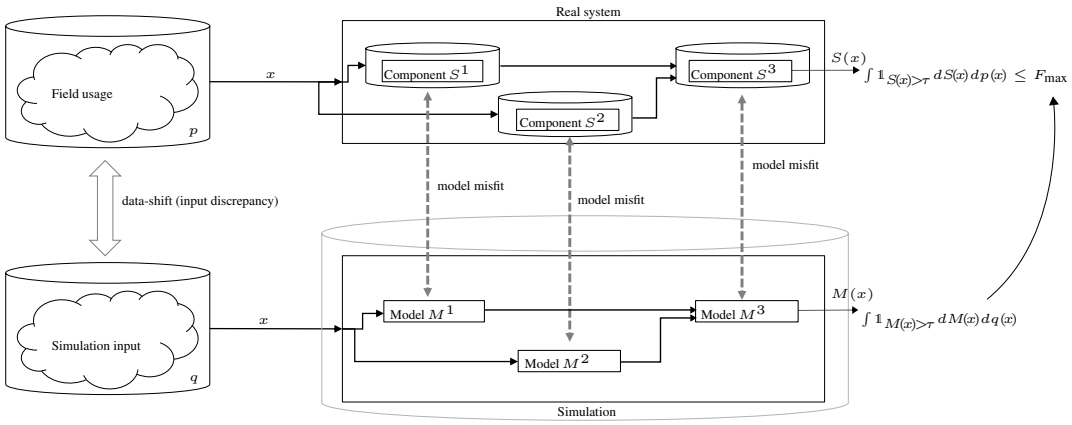


Figure 1: Illustration of our validation task: A real, composite system of interest (top) is modeled with corresponding simulation models (bottom). Measurements of the real system are available only for the individual components, while end-to-end simulation data can be generated from the models. The task of the virtual validation method is to estimate the real system performance S based on the simulations M, incorporating simulation model misfits w.r.t. the real-world components as well as any data-shift between the simulation input distribution and the field usage to be expected in the real system.

Existing validation methods are especially lacking the composite (multi-component) aspect, where measurement data are available only for each individual component (Sec. 2).

To state the problem mathematically, the goal of this work is to estimate an upper bound F_{max} on the failure probability $\Pr\big[S(x) > \tau\big]$ of a system over real-world inputs $x \sim p(x)$:

$$F_{\text{max}} \geq \Pr_{x,S}\big[S(x) > \tau\big] = \int \mathbb{1}_{S(x)>\tau} dS(x) dp(x), \quad (1)$$

where $S(x)$ measures the system performance upon input x, and τ is a critical performance threshold indicating a *system failure*. In the virtual validation setup, we assume that no end-to-end measurements from the full composite system S are available, and thus the upper bound F_{max} is to be estimated from the simulation M composed of models $M^1, M^2, \ldots$, which are assumed to be given. This estimate must take into account *model misfits* and *data-shift* in the simulation input distribution (see Fig. 1). To assess the model misfits, we assume validation measurements from the individual components $S^1, S^2, \ldots$ to be given, e.g. from component-wise development (for details see Sec. 3.1).

In this paper, we develop a method to estimate F_{max} from simulation runs by propagating bounds on distributional distances between simulation models and real-world components through the composite system. This propagation method incorporates model misfits and data-shifts in a pessimistic fashion by iteratively maximizing for the worst-case output distribution that is consistent with previously computed constraints on the input. Importantly, our method requires models and validation data from the individual components only, not from the full system.

Our main contributions can be summarized as follows:

1. We propose a novel, distribution-free bound on the distance between simulation-based and real-world distributions, without the need to have end-to-end measurements from the real world (Sec. 3.2).
2. We justify the method theoretically (Prop. 1) and show its practicality in reliability benchmarks (Sec. 4.2).
3. We demonstrate that – in contrast to alternative methods – the proposed method can account for data-shifts as well as model inaccuracies (Sec. 4).

2 RELATED WORK

Estimating the failure probability of a system is a core task in reliability engineering. In the reliability literature, one focus is on making this estimation more efficient compared to naive Monte Carlo sampling by reducing the variance on the estimator of the failure probability. Such classical methods include importance sampling [Rubinstein and Kroese, 2004], subset sampling [Au and Beck, 2001], line sampling [Pradlwarter et al., 2007], and first-order [Hohenbichler et al., 1987, Du and Hu, 2012, Zhang et al., 2015] or second-order [Kiureghian and Stefano, 1991, Lee et al., 2012] Taylor expansions. While being more efficient, they still require a large number of end-to-end function evaluations and cannot incorporate more detailed simulations.

Another line of research investigates how to reduce real-world function evaluations through virtualization of this performance estimation task [Xu and Saleh, 2021]. The failure probability is estimated based on a surrogate model and hence cannot account for mismatches between the system and its surrogate. Dubourg et al. [2013] proposed a hybrid approach, where the proposal distribution of the importance sampling depends on the learned surrogate model. While this approach accounts, to some extent, for model mismatches, the proposal distribution might still be biased by a

poor surrogate model. In summary, none of the approaches that are based on surrogate models provide a reliable bound on the true failure probability. Furthermore, all these approaches require end-to-end measurements from the real system, ignoring the composite structure of the system.

In practice, however, the system output $S(\cdot)$ in Eq. (1) refers to a complex system that often has a *composite structure*. That is, global inputs x propagate through an arrangement, oftentimes termed a *function network*, of subsystems or components, see Fig. 1. Exploiting such a structure is expected to have a notable impact on the target task, be it experimental design [Marque-Pucheu et al., 2019], calibration and optimization [Astudillo and Frazier, 2019, 2021, Kusakawa et al., 2022, Xiao et al., 2022], uncertainty quantification [Sanson et al., 2019], or system validation as presented here.

In the context of Bayesian Optimization (BO), for example, Astudillo and Frazier [2021] construct a surrogate system of Gaussian Processes (GP) that mirrors the compositional structure of the system. Similarly, Sanson et al. [2019] discuss similarities of such structured surrogate models to Deep GPs [Damianou and Lawrence, 2013], and extend this framework to local evaluations of constituent components. However, learning (probabilistic) models of inaccuracies [Sanson et al., 2019, Riedmaier et al., 2021] introduces further modeling assumptions and cannot account for datashifts. Instead, we aim at model-free worst-case statements.

Marque-Pucheu et al. [2019] showed that a composite function can be efficiently modeled from local evaluations of constituent components in a sequential design approach. Friedman et al. [2021] extend this framework to cyclic structures of composite systems for adaptive experimental design. They derive bounds on the simulation error in composite systems, although assuming knowledge of Lipschitz constants as well as uniformly bounded component-wise errors.

Stitching different datasets covering the different parts of a larger mechanism without loosing the causal relation was analyzed by Chau et al. [2021] and corresponding models were constructed, but the quality with which statements about the real mechanism can be made was not analyzed.

Bounding the test error of models under input-datashift was analyzed empirically in Jiang et al. [2022] by investigating the disagreement between different models. Although they find a correlation between disagreement and test error, the authors do not provide a bound on the test error (Sec. 3.3)

3 METHOD

3.1 SETUP: COMPOSITE SYSTEM VALIDATION

We consider a *(real) system* or *system under test* S that is composed of subsystems S^c ($c = 1, 2, \ldots, C$), over which we have only limited information. The validation task is to determine whether S conforms to a given specification, such as whether the system output $y = S(x)$ stays below a given threshold τ for typical inputs x – or whether the system's *probability of failure*, defined as violating the threshold, is sufficiently low, see Eq. (1). Our approach to this task is built on a *model* M (typically a simulation, with no analytic form) of S that is similarly composed of corresponding submodels M^c. The main challenge in assessing the system's failure probability lies in determining how closely M approximates S, in the case where the system data originate from disparate component measurements, which cannot be combined to consistent end-to-end data.

Components and signals. Mathematically, each component of S – and similarly for M – is a (potentially stochastic) map S^c, which upon input of a signal x^c produces an output signal (sample) $y^c \sim S^c(\cdot|x^c)$ according to the conditional distribution S^c. The stochasticity allows for aleatoric system behavior or unmodeled influences. We consider the case where all signals are tuples $x^c = (x^c_1, \ldots, x^c_{d^c_{in}})$, such as real vectors. The allowed "compositions" of the subsystems S^c must be such that upon input of any signal (stimulus) x, an output sample $y \sim S(\cdot|x)$ can be produced by iterating through the components S^c in order $c = 1, 2, \ldots, C$. More precisely, we assume that the input signal x^c into S^c is a concatenation of some entries $x|_{0\to c}$ of the overall input tuple x and entries $y^{c'}|_{c'\to c}$ of some *preceding* outputs $y^{c'}$ (with $c' = 1, \ldots, c-1$); thus, S^c is ready to be queried right after S^{c-1}. We assume the overall system output $y = y^C \in \mathbb{R}$ to be real-valued as multiple technical performance indicators (TPIs) could be considered separately or concatenated by weighted mean, etc. The simplest example of such a composite system is a *linear chain* $S = S^C \circ \ldots \circ S^2 \circ S^1$, where $x \equiv x^1$ is the input into S^1 and the output of each component is fed into the next, i.e. $x^{c+1} \equiv y^c$. Another example is shown in Fig. 1, where x^3 is concatenated from both outputs y^1 and y^2. We assume the identical compositional structure for the model M with components M^c.

Validation data. An essential characteristic of our setup is that neither S nor the subsystem maps S^c are known explicitly, and that "end-to-end" measurements (x, y) from the full system S are unavailable (see Sec. 1). Rather, we assume that *validation data* are available only for every subsystem S^c, i.e. pairs (x^c_v, y^c_v) of inputs x^c_v and corresponding output samples $y^c_v \sim S^c(\cdot|x^c_v)$ ($v = 1, \ldots, V^c$). Such validation data may have been obtained by measuring subsystem S^c in isolation on some inputs x^c_v, without needing the full system S; note, the inputs x^c_v do not necessarily follow the distribution from previous components. In the same spirit, the models M^c may also have been trained from such "local" system data; we assume M^c, M to be given from the start.

Probability distributions. We aim at probabilistic validation statements, namely that the system fails or violates its requirements only with low probability. For this, we assume that S is repeatedly operated in a situation where its inputs

come from a distribution $x \sim p_x$, in an i.i.d. fashion. For the example where S is a car, the input x might be a route that typical drivers take in a given city. Importantly, we do not assume much knowledge about p_x: merely a number of samples $x_v \sim p_x$ may be given, or alternatively its distance to the simulation input distribution $q_x = \frac{1}{n_M}\sum_{n=1}^{n_M} \delta_{x_n^M}$; here, $\delta_{x_n^M}$ are point measures on the input signals x_n^M on which M is being simulated. The input distribution $x \sim p_x$ will induce a (joint) distribution p of all intermediate signals x^c, y^c of the composite system S and importantly the TPI output $y = y^C \sim S(x)$. Similarly, M generates a joint model distribution q by starting from q_x and sampling through all M^c; via this simulation, we assume q and all its marginals on intermediate signals x^c, y^c to be available in sample-based form. The *(true) failure probability* is given by $p_{\text{fail}} = \int \mathbb{1}_{S>\tau} dS(x)dp(x)$, where in this paper we identify a system failure as the TPI exceeding the given threshold τ. The model failure probability is $q_{\text{fail}} = \int \mathbb{1}_{M>\tau} dM(x)dq(x) \simeq \frac{1}{n_M}\sum_n \mathbb{1}_{y_n^M>\tau}$, where y_n^M denote sampled model TPI outputs for the inputs x_n^M. It is often useful in our setting to think of a distribution as a set of sample points, and vice versa.

Discrepancies. To track how far the simulation model M diverges from the true system behavior S in our probabilistic setting, we employ discrepancy measures D between probability distributions. Such a measure D maps two probability distributions p, q over the same space to a real number, often having some interpretation of distance. We consider MMD distances $D = \text{MMD}_k$ [Gretton et al., 2012], defined as the RKHS norm $\text{MMD}_k(p,q) = \|p-q\|_k = \left[\int_{x,x'}(p(x) - q(x))k(x,x')(p(x') - q(x'))dxdx'\right]^{1/2}$ w.r.t. a kernel k on the underlying space (e.g. a squared-exponential or IMQ kernel [Gorham and Mackey, 2017]). Further possibilities include the cosine similarity $\text{COS}_k(p,q) = \langle p,q\rangle_k / \|p\|_k\|q\|_k$ w.r.t. a kernel k, a Wasserstein distance $D = W_p$ w.r.t. a metric on the space, and the total variation norm $D = TV$ [Sriperumbudur et al., 2009, 2010]; however, the latter cannot be estimated reliably from samples.

Specifically, we assume a discrepancy measure $D^{c'\to c}$ to be given[1] for those pairs $0 \le c' < c \le C+1$ for which (a sub-tuple of) the output signal $y^{c'}$ is fed into the input x^c (cf. the compositional structure above, and where we define $y^{c'=0} \equiv x$ and $x^{c=C+1} \equiv y := y^C$). This $D^{c'\to c}$ acts on probability distributions over the space of such sub-tuples like $y^{c'}|_{c'\to c}$ (or synonymously, $x^c|_{c'\to c}$), which is defined as the signal entries running from $y^{c'}$ to x^c. We denote the marginal of p on these signal entries by $p|_{c'\to c}$, and similar $q|_{c'\to c}$ for q. In the simplest case of a linear chain, $D^{c'\to c}$ with $c' = c-1$ acts on probability distributions such as $p|_{c'\to c}$ over the space of the (full) vectors $y^{c'} = x^c$. We omit superscripts $D^{c'\to c} \equiv D$ when clear from the context.

[1] We will later address how to choose D from a parameterized family D_ℓ, e.g. with different lengthscales ℓ.

Our method requires (upper bounds on) the discrepancies $D(p|_{0\to c}, q|_{0\to c})$ between marginals of the system and model input distributions p_x, q_x; specifically between the marginal distributions $p|_{0\to c}$ and $q|_{0\to c}$ over those sub-tuples $x|_{0\to c}$ which are input to subsequent components c. These discrepancies can either be estimated from samples x_v, x_n^M of p_x, q_x, see the biased and unbiased estimates for MMD in Gretton et al. [2012][App. A.2, A.3], which are accurate up to at most $\sim \sqrt{(1/n_{min})\log(1/\delta)}$ at confidence level $1-\delta$ (where n_{min} denotes the size of the smaller of both sample sets); alternatively, these discrepancies may be directly given or upper bounded. These upper bounds are the quantities $B^{0\to c}$ below in Eq. (2). No further knowledge of the real-world input distribution p_x is required.

3.2 DISCREPANCY PROPAGATION METHOD

We now describe the key step in our method to quantify how closely the model's TPI output distribution, which we denote by $q_y \equiv q|_{C\to C+1}$, approximates the actual (but unknown) system output distribution $p_y \equiv p|_{C\to C+1}$. We do this by iteratively propagating worst-case discrepancy values through the (directed and acyclic) graph of components S^c/M^c, using only the available information, in particular the given validation data (x_v^c, y_v^c) on a per-subsystem basis.

Discrepancy bound propagation. The basic idea is to go through the components $c = 1, 2, \ldots, C$ one-by-one. At each step S^c, we consider the "input discrepancies" $D(p|_{c'\to c}, q|_{c'\to c})$ (for $c' < c$), about which we already have information, and propagate this to gain information about the "output discrepancies" $D(p_{c\to c''}, q|_{c\to c''})$ (for $c'' > c$). Here, we consider "information" in the form of inequalities $D(p|_{c'\to c}, q|_{c'\to c}) \le B^{c'\to c}$, i.e. the information is the value of the (upper) bound $B^{c'\to c}$. Given bounds $B^{c'\to c}$ on the input signal of S^c, an upper bound on $D(p|_{c\to c''}, q|_{c\to c''})$ for each fixed $c'' > c$ can be found by maximizing the latter discrepancy over all (unknown) distributions p that satisfy all the input discrepancy bounds:

$$B^{c\to c''} = \text{maximize}_p\ D(p|_{c\to c''}, q|_{c\to c''}) \qquad (2)$$
$$\text{subject to}\ \ D(p|_{c'\to c}, q|_{c'\to c}) \le B^{c'\to c}\ \forall c' < c.$$

Note that the (sample-based) model distribution q and its marginals in (2) are known and fixed after the simulation M has been run on the input samples x_n^M which constitute q_x (see above). In contrast, as the actual p is not known, we maximize over all possible system distributions p in (2) according to the bounds from the previous components c'.

It remains to optimize over all possible sets of marginals $p|_{c\to c''}, p|_{c'\to c}$ (for all $c' < c$) occurring in (2). Ideally, one would consider all distributions $p(x^c)$ over input signals x^c, apply S^c to each x^c to obtain all possible joint distributions $p(x^c, y^c) = p(x^c)S^c(y^c|x^c)$ of in- and outputs, and compute from this all possible sets of marginals $p|_{c\to c''}, p|_{c'\to c}$.

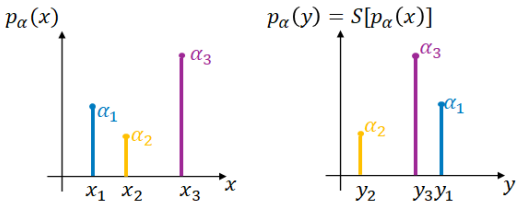


Figure 2: Illustration of the (marginals of the) joint input-output distribution p_α (3), parameterized by weights α_v. Corresponding in-/outputs x_v, y_v have the same weight α_v.

However, this is impossible as we do not know the action of S^c on every possible input x^c. Rather, we merely know about the action of S^c on the validation inputs x_v^c, namely that $y_v^c \sim S^c(x_v^c)$ is a corresponding output sample. We thus consider only the joint distributions $p(x^c, y^c) = p_\alpha$ that can be formed from the given validation data (Fig. 2)[2]:

$$p_\alpha = \sum_{v=1}^{V^c} \alpha_v \delta_{x_v^c} \delta_{y_v^c}, \tag{3}$$

such that the optimization variable becomes now a probability vector $\alpha \in \mathbb{R}^{V^c}$, i.e. with nonnegative entries $\alpha_v \geq 0$ summing to $\sum_v \alpha_v = 1$. By restricting to this (potentially skewed) set of joint distributions p_α, the exact bound $B^{c\to c''}$ turns into an estimate; for further discussion see Prop. 1, which also proposes another possible parametrization p_α.

Using ansatz (3), the exact bound propagation (2) becomes:

$$\begin{aligned} B^{c\to c''} = \max_\alpha\ & D(p_\alpha|_{c\to c''}, q|_{c\to c''}) \\ \text{s.t. }\ & D(p_\alpha|_{c'\to c}, q|_{c'\to c}) \leq B^{c'\to c}\ \ \forall c' < c, \\ & \alpha \geq 0,\ \ \sum_v \alpha_v = 1. \end{aligned} \tag{4}$$

Note that for sample-based distributions like p_α in (3) or q, the marginals in this optimization have a similar form, e.g. $p_\alpha|_{c\to c''} = \sum_v \alpha_v \delta_{y_v^c|_{c\to c''}}$ or $p_\alpha|_{c'\to c} = \sum_v \alpha_v \delta_{x_v^c|_{c'\to c}}$.

As the discrepancy measures D in (4) are usually convex, this optimization problem is "almost" convex: All its constraints are convex, however, we aim to *maximize* a convex objective. For MMD measures we derive convex (semidefinite) relaxations of (4) by rewriting it with squared MMDs $D(p_\alpha, q)^2$, which are quadratic in α and thus *linear* in a new matrix variable $A = \alpha\alpha^T$; this last equality is then relaxed to the semidefinite inequality $A \geq \alpha\alpha^T$ (App. A). While the relaxation is tight in most instances (App. E.4), the number of variables increases from V^c to $\sim (V^c)^2/2$, restricting the method to $V^c \lesssim 10^3$ validation samples per component. In our implementation, we solve these SDPs using the CVXPY package [Diamond and Boyd, 2016].

[2] δ_{z_0} denotes a Dirac point mass at $z = z_0$.

Bounding the failure probability. The final step of the preceding *bound propagation* yields an upper bound $B^y := B^{C\to C+1}$ on the discrepancy $D(p_y, q_y)$ between the (unknown) system TPI output distribution p_y and its model counterpart q_y, which is given by samples y_n^M. We now apply an idea similar to (3) to obtain (a bound on) the system failure probability $p_{\text{fail}} := \int_{y>\tau} p_y(y)dy$: Rather than maximizing p_{fail} over all distributions p_y on $\mathbb{R} \ni y$ subject to the constraint B^y, we make the optimization finite-dimensional by selecting grid-points $g_1 < g_2 < \ldots < g_V \in \mathbb{R}$ and parameterizing $p_y \equiv p_\alpha = \sum_{v=1}^V \alpha_v \delta_{g_v}$, such that $p_{\text{fail}} = \sum_{v:g_v>\tau} \alpha_v$. In practice, we choose an equally-spaced grid in an interval $[g_{\min}, g_{\max}] \subset \mathbb{R}$ that covers the "interesting" or "plausible" TPI range, such as the support of q_y as well as sufficient ranges below and above the threshold τ. The size of the optimization problem corresponds to the number of grid-points V, so $V \simeq 10^3$ is easily possible here.

With this, our final upper bound $p_{\text{fail}} \leq F_{\max}$ on the failure probability becomes the following convex program:

$$\begin{aligned} F_{\max} = \max_\alpha\ & \sum_{v:g_v>\tau} \alpha_v \\ \text{s.t. }\ & D(p_\alpha, q_y) \leq B^y,\ \ \alpha \geq 0,\ \ \sum_v \alpha_v = 1. \end{aligned} \tag{5}$$

One can obtain better (i.e. smaller) bounds $F_{\max}$ by restricting p_α further by plausible assumptions: (a) *Monotonicity:* When bounding a tail probability, i.e. p_{fail} is expected to be small, it may be reasonable to assume that p_y is monotonically decreasing beyond some tail threshold τ'. For an equally-spaced grid this adds constraints $\alpha_v \leq \alpha_{v-1}$ for all v with $g_v \geq \tau'$ to (5); we always assume this with $\tau' := \tau$. (b) *Lipschitz condition:* To avoid that p_α becomes too "spiky", we pose a Lipschitz condition $|\alpha_{v+1} - \alpha_v| \leq \Lambda_{\max}|g_{v+1} - g_v|$ with a constant $\Lambda_{\max}$ estimated from the set of outputs y_n^M. See also App. B.

Note that our final bound $F_{\max}$ is a probability, whose interpretation is independent of the chosen discrepancy measures, kernels, or lengthscales. We can thus select these "parameters" by minimizing the finally obtained $F_{\max}$ over them. We do this using Bayesian optimization [Fröhlich et al., 2020].

We summarize our full discrepancy propagation method to obtain a bound $F_{\max}$ on the system's failure probability in Algorithm 1, which we refer to as `DPBound`.

Upper bound property. We replaced the optimization over all possible system distributions p in (2) by the distributions p_α from (3) due to the limited system validation data and to make the optimization tractable. This restricted and possibly skewed p_α can potentially cause $B^{c\to c''}$ and ultimately $F_{\max}$ from (4),(5) to not be true upper bounds on D or even the system's (unknown) failure probability p_{fail}, although the worst-case tendency of the maximizations alleviates the issue. We investigate this in the experiments (Sec. 4.2), and in the following proposition we state conditions under which

Algorithm 1 `DPBound`

Input: compositional structure of S; composite simulation model M; discrepancy measures D; validation data (x_v^c, y_v^c); simulation input samples $\{x_n^M\} \equiv q_x$; either *(a)* upper bounds $B^{0\to c}$ on input discrepancies or *(b)* samples $x_v \sim p_x$ from the real-world input distribution.
Run M on all x_n^M; collect all intermediate signals x_n^c, y_n^c to build the sample-based marginals $q|_{c'\to c}$ of q.
In case *(b)*, estimate $B^{0\to c}$ from $p_x \simeq \{x_v\}$ and q_x.
for $c = 1, \ldots, C$ **do**
  for every $c'' = c+1, \ldots, C+1$ connected to c **do**
    Compute $B^{c\to c''}$ via Eq. (4) (or via App. A).
Using the thus obtained $B^y := B^{C\to C+1}$, compute the final bound F_{max} via Eq. (5) (or via App. B).

(4),(5) are upper bounds:

Proposition 1. *Suppose that for each component $c = 1, \ldots, C$: (i) the validation inputs x_v^c cover the space of occurring inputs into S^c; (ii) (necessary only for components S^c having stochastic output) the $\delta_{y_v^c}$ in the defining equation of p_α (Eq. (3)) is replaced by the system output distribution $S^c(x_v^c)$ (represented e.g. by samples or its kernel-mean embedding); (iii) the grid $\{g_v\}$ covers the occurring TPI values (e.g. discrete and bounded). Then $p_{fail} \le F_{max}$, where F_{max} is defined by the computations in Eqs. (4) and (5) (or alternatively, by the convex relaxations and forms in Apps. A and B).*

App. C gives a proof as well as an additional limit statement about the $B^{c\to c''}$ and F_{max} in the more realistic setting of increasingly dense inputs x_v^c and approximations of $S^c(x_v^c)$.

3.3 FAILURE BOUND VIA SURROGATE MODEL

As an alternative, more heuristic but simpler, baseline method to estimate an upper bound F_{max} on the failure probability p_{fail}, we introduce a sampling-based method that also operates on the available data only. The underlying concept in quantifying model accuracy is similar to the one from [Jiang et al., 2022] and can also be thought of one particular form of error modeling [Riedmaier et al., 2021].

The general idea is to train an additional "surrogate" model M'^c for each system component S^c, and employ the resulting composite M' to estimate how far M deviates from the real S in terms of TPI outputs. This is possible because for training M'^c we use the available validation data (x_v^c, y_v^c) measured from S^c. We take Gaussian processes (GPs) for M'^c, but other probabilistic models like normalizing flows or deterministic ones like neural networks are possible as well. Choosing M'^c from a different model class than M^c will generally lead to a more conservative estimate F_{max}.

After training the M'^c, we run the resulting composite surrogate model M' on the simulation input samples x_n^M (which make up q_x) to obtain TPI output samples $y_{n,i}'^M$ (with k repetitions $i = 1, \ldots, k$), in the same way the given model M can generate outputs $y_{n,i}^M$ from given x_n^M. By comparing the $y_{n,i}'^M$ to the $y_{n,i}^M$ we obtain a heuristic estimate of the error of M in simulating the actual TPI of system S, see [Jiang et al., 2022]. Concretely in this paper, we use the averaged outputs $y_n^M := (1/k)\sum_i y_{n,i}^M$ and similarly $y_n'^M$, taking as the simulation error Δ a high quantile (here, 95%) of the (signed or absolute) deviations:

$$\Delta = \text{quantile}_{0.95}[\{y_n'^M - y_n^M\}]. \tag{6}$$

The 100%-quantile $\max_n(y_n'^M - y_n^M)$ would lead to more conservative bounds F_{max}. For an illustration see App. E.2.

Finally, we estimate an upper bound on p_{fail} by including a safety margin Δ before the threshold τ:

$$F_{\text{max}} := \int_{\tau-\Delta}^{\infty} q(y)dy = \frac{1}{n_M \cdot k}\sum_{n,i} \mathbb{1}_{y_{n,i}^M > \tau-\Delta}. \tag{7}$$

Note that this method *cannot* account for discrepancies between the (unknown) system input distribution p_x w.r.t. which we would like to bound the failure probability p_{fail}, and the given simulation inputs x_n^M that make up q_x, see Sec. 4. Rather, the method can be expected to work well only when the sample-based q_x is close to the actual p_x. Furthermore, if simulation model and surrogate model share unjustified modeling assumptions, an agreement between the two models might mask differences to the actual system.

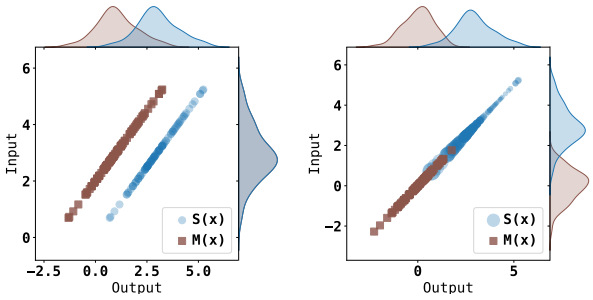
4 EXPERIMENTS

We evaluate the proposed method on 8 benchmark systems in Sec. 4.2.[3] For each system, we create 4 configurations where the simulation models and/or the simulation input distributions differ. These configurations are illustrated on an artificial example in Sec. 4.1.

4.1 ILLUSTRATION: GAUSSIAN MODELS

As an exemplary validation problem, consider the following one-component, one-dimensional setup: For a linear system $S : \mathbb{R} \to \mathbb{R}$ with $S(x) = w_S x + b_S$, we want to assess the failure probability in (1) by means of a linear model $M(x) = w_M x + b_M$. Both S and M are stimulated with samples from Gaussian distributions $p_x = \mathcal{N}(\mu_p, \sigma_p^2)$ and $q_x = \mathcal{N}(\mu_q, \sigma_q^2)$, respectively. We can control the accuracy of M and its input distribution q_x separately, thereby investigating their impacts on the estimated failure probability.

[3] Our organization is carbon neutral. Therefore, all its activities including research activities (e.g., compute clusters) no longer leave a carbon footprint.



(a) (b)

Figure 3: Illustration of DPBound for a linear mapping between (samples from) Gaussian signals. **(a)** There is model mismatch $M \neq S$, but the input distribution $q_x = p_x$ is perfect. **(b)** $M = S$ is a perfect model, but the model input distribution $q_x \neq p_x$ is biased w.r.t. the real world. The computed weights α_v from Eqs. (3),(4) are depicted by the size of the blue $S(x)$-markers (α is uniform in case (a)).

Specifically, we analyze the following two configurations: (a) M and S differ in $b_M \neq b_S$, but they receive identical inputs $q_x = p_x$ (*Misfit Model & Perfect Input*); and (b) the model M is identical to S, but their respective input distributions $q_x \neq p_x$ differ (*Perfect Model & Biased Input*). Under these two configurations, DPBound is illustrated for a single propagation step in Fig. 3, where the marginal output distributions of M and S differ (top row marginals in brown resp. blue). However, the output discrepancy bound $B^{1\to 2}$ from Eq. (4) originates differently in the two configurations: In Fig. 3(a), the input discrepancy $B^{0\to 1}$ vanishes due to identical inputs $p_x = q_x$. The output bound (4) thus directly reflects the difference between the output marginals $S(q_x)$ and $M(q_x)$, without optimization since the constraint forces the weights α to be uniform so that $p_\alpha = q_x$ (here, we assumed as validation inputs the q_x-samples for simplicity).

In Fig. 3(b), however, the bound on the output discrepancy stems solely from the non-zero input discrepancy $B^{0\to 1} = MMD(p_x, q_x) > 0$, rather than from any difference between M and S: DPBound finds in Eq. (4) the worst-case weighted output distribution p_α consistent with the input discrepancy $B^{0\to 1}$. The output bound is then the nonzero difference between this p_α and $M(q_x)$, even though both p_α and $M(q_x)$ were built with outputs from $S = M$.

In this way, our method DPBound can account for both model misfits and biased inputs. The latter is not true of the SurrModel method from Sec. 3.3, as it ignores the real-world distribution p_x and can thus fail for biased inputs $q_x \neq p_x$. For further details and illustrations, see App. E.

4.2 RELIABILITY BENCHMARK EVALUATION

We now demonstrate the feasibility of our discrepancy propagation method in a reliability benchmark.

Compared benchmark systems. The performance of DPBound is evaluated on 3 single-component and 5 multi-component problems from the reliability and uncertainty propagation literature. These problems are briefly summarized in Tab. 2 below (see App. D for more details).

For the evaluation, we set the threshold τ on the scalar output for each of those problems such that the ground-truth failure probability $\Pr_{x\sim p_x}[S(x) > \tau] = 1\%$ (see Eq. (1)).

We evaluate each system under four different simulation configurations (cf. Sec. 4.1): As simulation models, we take either *Perfect Models* $M^c = S^c$ or GP-based *Misfit Models* $M^c \neq S^c$; as simulation input distribution, we take either *Perfect Input* $q_x = p_x$ or *Biased Input* $q_x \neq p_x$ (App. D).

Compared virtual validation methods. We compare our failure probability bound with two alternative methods:
DPBound (ours): Failure bound $F_{\max}$ calculated by propagating MMD-based bounds according to Algorithm 1.
MCCP: 95%-confidence Clopper-Pearson bound on the failure probability, calculated on binary Monte-Carlo samples obtained by thresholding the output of the simulation model.
SurrModel: Bound from Eq. (7), obtained by accounting for the difference between the simulation and a GP-based surrogate model learned on the validation data (Sec. 3.3).

Experimental results. The obtained results are summarized in Tab. 1. We first focus on the validity of the methods (bold numbers in Tab. 1): In this regard one can see that, in the "Perfect Input–Perfect Model" setting the methods produce generally valid bounds, with at most 17.5% invalidness for SurrModel. In the more challenging and realistic settings with misfit and/or input bias, however, the invalidness ratios for MCCP and SurrModel increase beyond acceptable levels, especially for *Biased Input* reaching invalidness up to 75%. DPBound on the other hand remains perfectly valid under both *Misfit Model* and *Biased Input* (as in Sec. 4.1).

To understand why MCCP and SurrModel have challenges with the *Biased Input* setting, notice that these methods disregard the actual system inputs p_x, instead relying solely on the simulation inputs q_x without any means of dealing with a potential discrepancy between both distributions. When the simulation input distribution is biased towards significantly lower simulated TPI values, the delivered bounds can then be invalid (for an illustration see App. E.2). DPBound on the other hand natively accounts for this input discrepancy through the initial bounds $B^{0\to c}$, which in our experiments are estimated via samples from p_x, q_x (see end of Sec. 3.1). In addition to this shortcoming of ignoring input discrepancies, MCCP remains unaware of any potential *Misfit Model*, as it completely ignores the system S and its validation data.

Table 1: Bounds on the failure probability (in %, with standard deviations from 5 repetitions) delivered by the three compared methods for each benchmark problem under the four simulation configurations *Perfect vs. Misfit Model* and *Perfect vs. Biased Input*. Each problem has been normalized such that the ground-truth failure probability is 1%. Also shown (in bold) is the ratio of invalid bounds (i.e. bounds below 1%) delivered by each method among the 40 runs per configuration.

		Perfect Model			**Misfit Model**		
	Problem	DPBound	MCCP	SurrModel	DPBound	MCCP	SurrModel
Perfect Input	Single Component						
	Borehole	1.10 ± 0.2	1.87 ± 0.3	3.04 ± 1.3	1.75 ± 1.7	1.06 ± 0.4	3.48 ± 1.6
	Branin	19.98 ± 3.3	2.86 ± 0.6	1.84 ± 0.5	18.87 ± 2.8	2.76 ± 0.8	1.80 ± 0.7
	Four Branch	23.1 ± 1.0	2.08 ± 0.4	1.48 ± 0.6	22.07 ± 0.7	1.07 ± 0.2	0.84 ± 0.6
	Multiple Components						
	Chained Solvers	14.92 ± 2.7	2.08 ± 0.4	1.28 ± 0.6	15.21 ± 2.2	2.18 ± 0.6	1.60 ± 0.4
	Borehole	8.94 ± 6.1	2.03 ± 0.5	3.12 ± 1.9	7.51 ± 2.6	2.06 ± 0.9	3.52 ± 1.6
	Branin	17.91 ± 6.8	2.71 ± 0.5	1.56 ± 0.4	16.6 ± 6.3	2.82 ± 0.2	1.88 ± 0.4
	Four Branch	36.4 ± 3.1	2.18 ± 0.8	1.52 ± 0.7	35.82 ± 3.2	0.81 ± 0.2	0.56 ± 0.3
	Controlled Solvers	10.39 ± 4.6	1.97 ± 0.7	0.92 ± 0.5	11.01 ± 4.3	1.91 ± 0.7	0.88 ± 0.5
	% Invalid Bounds	**5.0**	**0.0**	**17.5**	**0.0**	**22.5**	**27.5**
Biased Input	Single Component						
	Borehole	16.57 ± 8.3	0.80 ± 0.3	0.88 ± 0.8	15.19 ± 10.5	0.60 ± 0.0	0.76 ± 0.8
	Branin	92.7 ± 3.3	0.73 ± 0.3	0.08 ± 0.2	93.35 ± 2.9	2.01 ± 0.8	1.00 ± 0.6
	Four Branch	22.69 ± 0.9	1.98 ± 0.5	1.28 ± 0.5	21.96 ± 1.0	0.88 ± 0.2	0.60 ± 0.3
	Multiple Components						
	Chained Solvers	26.39 ± 1.7	0.74 ± 0.2	0.08 ± 0.1	26.7 ± 1.7	1.39 ± 0.7	0.56 ± 0.5
	Borehole	20.96 ± 9.0	0.93 ± 0.4	0.76 ± 0.8	15.89 ± 7.5	0.60 ± 0.0	0.44 ± 0.3
	Branin	89.52 ± 6.1	0.81 ± 0.2	0.12 ± 0.1	89.52 ± 6.1	0.91 ± 0.5	0.24 ± 0.3
	Four Branch	35.83 ± 3.2	1.96 ± 0.9	1.28 ± 0.7	35.42 ± 2.9	0.74 ± 0.2	0.44 ± 0.2
	Controlled Solvers	13.36 ± 4.1	0.60 ± 0.0	0.00 ± 0.0	13.51 ± 5.9	0.67 ± 0.2	0.04 ± 0.1
	% Invalid Bounds	**0.0**	**67.5**	**70.0**	**0.0**	**67.5**	**75.0**

This explains the jump in invalidness from 0.0% to 22.5% when isolating this effect in the *Perfect Input* setting. Note, while `MCCP` is not expected to produce upper bounds in 100% of cases due to its 95%-confidence specification, it stays significantly below the 95% promise (App. E.3).

Although `DPBound` provides valid upper bounds $F_{\max}$ in almost all cases, it can sometimes still underestimate, as happened in two validation runs for Borehole(Single) under *Perfect Input–Perfect Model* with bounds close to the 1% ground-truth. To explain how this can happen, note that only under dense sampling conditions is `DPBound` expected to be perfectly valid (Prop. 1 and App. C). In any case, `DPBound` shows high validity overall, while the two competing methods' likelihood for falsely positive validation is certainly too high for trustable statements.

Summarizing these validity results, of the three compared methods, only `DPBound` should be further considered to be viable at all as a reliable validation method.

Despite its high validity, `DPBound` delivers bounds $F_{\max}$ that are mostly far from trivial (i.e. much below 100%), also in the challenging misfit and/or biased settings. These bounds in conjunction with their high validity thus yield useful information, which can serve as a basis for extended validation approaches (see Conclusion). The fact that `MCCP` and `SurrModel` often produce much smaller or "tighter"[4] bounds is no advantage per se, as these bounds are often invalid and thus misleading in validation (see above). We did not focus on the thightness evaluation because our main evaluation criterion was the rate of falsely positive validations, which already excluded both competing methods in safety-relevant situations.

While it remains for future work to combine `DPBound`'s non-trivial and reliable bounds with other validation approaches, our investigation here constitutes the first one into the validity of validation methods in the component-wise setting, establishing `DPBound` as a viable candidate.

5 CONCLUSION

Validating complex composite systems is a notoriously difficult task, e.g. validating the performance of autonomously driving vehicles. Instead of expensively testing the system

[4]Note, the "tightness" of the bounds can be read off from Tab. 1 by subtracting from each bound the ground-truth value of 1%.

Table 2: Compared benchmark systems.

Benchmark System	Input Dim.	Components
Controlled Solvers [Sanson et al., 2019]	16	4
Chained Solvers [Sanson et al., 2019]	1	2
Borehole [Surjanovic and Bingham, 2023]	8	1 / 5
Branin [Surjanovic and Bingham, 2023]	2	1 / 3
Four Branch [UQWorld, 2019]	2	1 / 4

in the real world, simulations can reduce the validation effort [Wong et al., 2020]. However, it is hard to assess the effect of simulation model inaccuracies or input data shifts on the validation target, especially for composite systems. We have developed a method to estimate an upper bound on the system failure probability, as underestimation of failure rates is typically much more costly than overestimation. Our method assumes that a simulation model as well as measurement data for each subsystem are available. Our evaluations show that the obtained bounds are useful and valid in general, with theoretical guarantees in the large-data limit (Prop. 1).

Due to its individual-component nature, our method is especially fit to use when only one component in an already deployed system changes, e.g. a sensor or the software controller in an autonomous driving system. Although the values computed for the bounds by our propagation method are larger than what would typically be required for safety-relevant applications, they still yield useful information, for example by acting as a safe-guard before entering an expensive real-world testing phase. Continuing the proposed avenue of research may ultimately spawn validation with immensely reduced number of real-world test runs. For the future, it remains to explore the method in higher-dimensional situations, possibly by extending our parameterization of the joint distribution p_α. While the presented method was derived for static signals, it can be extended to dynamic systems and models by replacing the static signals with embeddings of time-series signals [Morrill et al., 2020]. Exploring the proposed method in these regimes, which often include feedback loops, is subject to future research.

Acknowledgements

This research was supported by the German Federal Ministry for Economic Affairs and Climate Action under the joint project "KI-Embedded" (grant no. 19I21043A).

References

Raul Astudillo and Peter Frazier. Bayesian optimization of composite functions. In *International Conference on Machine Learning*, pages 354–363. PMLR, 2019.

Raul Astudillo and Peter Frazier. Bayesian optimization of function networks. *Advances in Neural Information Processing Systems*, 34:14463–14475, 2021.

Siu-Kui Au and James L Beck. Estimation of small failure probabilities in high dimensions by subset simulation. *Probabilistic engineering mechanics*, 16(4):263–277, 2001.

Julien Bect, David Ginsbourger, Ling Li, Victor Picheny, and Emmanuel Vazquez. Sequential design of computer experiments for the estimation of a probability of failure. *Statistics and Computing*, 22(3):773–793, 2012.

Christine Belcastro and Celeste Belcastro. On the validation of safety critical aircraft systems, part I: An overview of analytical & simulation methods. In *AIAA Guidance, Navigation, and Control Conference and Exhibit*, page 5559, 2003.

Siu Lun Chau, Jean-François Ton, Javier González, Yee Teh, and Dino Sejdinovic. BayesIMP: Uncertainty quantification for causal data fusion. *Advances in Neural Information Processing Systems*, 34:3466–3477, 2021.

Andreas Damianou and Neil D Lawrence. Deep Gaussian processes. In *Artificial intelligence and statistics*, pages 207–215. PMLR, 2013.

Steven Diamond and Stephen Boyd. CVXPY: A Python-embedded modeling language for convex optimization. *Journal of Machine Learning Research*, 17(83):1–5, 2016.

Xiaoping Du and Zhen Hu. First order reliability method with truncated random variables. *Journal of Mechanical Design*, 134(9), 2012.

Vincent Dubourg, Bruno Sudret, and Franois Deheeger. Metamodel-based importance sampling for structural reliability analysis. *Probabilistic Engineering Mechanics*, 33:47–57, 2013.

Sam Friedman, John Davis Jakeman, Michael S Eldred, Lorenzo Tamellini, Alex Gorodestky, and Doug Allaire. Adaptive resource allocation for surrogate modeling of systems comprised of multiple disciplines with varying

fidelity. Technical report, Sandia National Lab (SNL-NM), Albuquerque, NM (United States), 2021.

Lukas Fröhlich, Edgar Klenske, Julia Vinogradska, Christian Daniel, and Melanie Zeilinger. Noisy-input entropy search for efficient robust Bayesian optimization. In *International Conference on Artificial Intelligence and Statistics*, pages 2262–2272. PMLR, 2020.

Jackson Gorham and Lester Mackey. Measuring sample quality with kernels. In *International Conference on Machine Learning*, pages 1292–1301. PMLR, 2017.

Arthur Gretton, Karsten M Borgwardt, Malte J Rasch, Bernhard Schölkopf, and Alexander Smola. A kernel two-sample test. *The Journal of Machine Learning Research*, 13(1):723–773, 2012.

Michael Hohenbichler, Stephan Gollwitzer, Wolfgang Kruse, and Rüdiger Rackwitz. New light on first- and second-order reliability methods. *Structural safety*, 4(4): 267–284, 1987.

Yiding Jiang, Vaishnavh Nagarajan, Christina Baek, and J Zico Kolter. Assessing generalization of SGD via disagreement. In *10th International Conference on Learning Representations, ICLR 2022*, 2022.

Nidhi Kalra and Susan M Paddock. Driving to safety: How many miles of driving would it take to demonstrate autonomous vehicle reliability? *Transportation Research Part A: Policy and Practice*, 94:182–193, 2016.

Armen Der Kiureghian and Mario De Stefano. Efficient algorithm for second-order reliability analysis. *Journal of engineering mechanics*, 117(12):2904–2923, 1991.

Philip Koopman and Michael Wagner. Challenges in autonomous vehicle testing and validation. *SAE International Journal of Transportation Safety*, 4(1):15–24, 2016.

Shunya Kusakawa, Shion Takeno, Yu Inatsu, Kentaro Kutsukake, Shogo Iwazaki, Takashi Nakano, Toru Ujihara, Masayuki Karasuyama, and Ichiro Takeuchi. Bayesian optimization for cascade-type multistage processes. *Neural Computation*, 34(12):2408–2431, 2022.

Ikjin Lee, Yoojeong Noh, and David Yoo. A novel second-order reliability method (SORM) using noncentral or generalized chi-squared distributions. *Journal of Mechanical Design*, 134(10), 2012.

Sophie Marque-Pucheu, Guillaume Perrin, and Josselin Garnier. Efficient sequential experimental design for surrogate modeling of nested codes. *ESAIM: Probability and Statistics*, 23:245–270, 2019.

James Morrill, Adeline Fermanian, Patrick Kidger, and Terry Lyons. A generalised signature method for multivariate time series feature extraction. *arXiv preprint arXiv:2006.00873*, 2020.

HJ Pradlwarter, GI Schueller, Phaedon-Stelios Koutsourelakis, and Dimos C Charmpis. Application of line sampling simulation method to reliability benchmark problems. *Structural Safety*, 29(3):208–221, 2007.

Stefan Riedmaier, Benedikt Danquah, Bernhard Schick, and Frank Diermeyer. Unified framework and survey for model verification, validation and uncertainty quantification. *Archives of Computational Methods in Engineering*, 28(4):2655–2688, 2021.

Reuven Y Rubinstein and Dirk P Kroese. *The Cross-Entropy Method: A Unified Approach to Combinatorial Optimization, Monte-Carlo Simulation, and Machine Learning*, volume 133. Springer, 2004.

Francois Sanson, Olivier Le Maitre, and Pietro Marco Congedo. Systems of Gaussian process models for directed chains of solvers. *Computer Methods in Applied Mechanics and Engineering*, 352:32–55, 2019.

Bharath K. Sriperumbudur, Kenji Fukumizu, Arthur Gretton, Bernhard Schölkopf, and Gert R.G. Lanckriet. On integral probability metrics, ϕ-divergences and binary classification. *arXiv preprint arXiv:0901.2698*, 2009.

Bharath K Sriperumbudur, Kenji Fukumizu, Arthur Gretton, Bernhard Schölkopf, and Gert RG Lanckriet. Non-parametric estimation of integral probability metrics. In *2010 IEEE International Symposium on Information Theory*, pages 1428–1432, 2010.

Sonja Surjanovic and Derek Bingham. Virtual library of simulation experiments: Test functions and datasets. Retrieved February 13, 2023, from `http://www.sfu.ca/~ssurjano`, 2023.

UQWorld. Four-Branch Function. Retrieved February 13, 2023, from `https://uqworld.org/t/four-branch-function/59`, 2019.

Hongqiao Wang, Guang Lin, and Jinglai Li. Gaussian process surrogates for failure detection: A Bayesian experimental design approach. *Journal of Computational Physics*, 313:247–259, 2016.

Kelvin Wong, Qiang Zhang, Ming Liang, Bin Yang, Renjie Liao, Abbas Sadat, and Raquel Urtasun. Testing the safety of self-driving vehicles by simulating perception and prediction. In *European Conference on Computer Vision*, pages 312–329. Springer, 2020.

Tesi Xiao, Krishnakumar Balasubramanian, and Saeed Ghadimi. A projection-free algorithm for constrained

stochastic multi-level composition optimization. In *Advances in Neural Information Processing Systems*, volume 35, 2022.

Zhaoyi Xu and Joseph Homer Saleh. Machine learning for reliability engineering and safety applications: Review of current status and future opportunities. *Reliability Engineering & System Safety*, 211:107530, 2021.

Zhe Zhang, Chao Jiang, GG Wang, and Xue Han. First and second order approximate reliability analysis methods using evidence theory. *Reliability Engineering & System Safety*, 137:40–49, 2015.