

Learning hidden constraints using a Stepwise Uncertainty Reduction strategy based on Gaussian Process Classifiers

Morgane Menz, Miguel Munoz-Zuniga, Delphine Sinoquet

▶ **To cite this version:**

Morgane Menz, Miguel Munoz-Zuniga, Delphine Sinoquet. Learning hidden constraints using a Stepwise Uncertainty Reduction strategy based on Gaussian Process Classifiers. 2022. hal-03848238v1

HAL Id: hal-03848238

https://hal.science/hal-03848238v1

Preprint submitted on 10 Nov 2022 (v1), last revised 8 Sep 2023 (v2)

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.

Learning hidden constraints using a Stepwise Uncertainty Reduction strategy based on Gaussian Process Classifiers

Morgane Menz[1*], Miguel Munoz-Zuniga[1†] and Delphine Sinoquet[1†]

[1*]IFP Energies nouvelles, 1 et 4 avenue de Bois-Préau, Rueil-Malmaison, 92852, France.

*Corresponding author(s). E-mail(s): morgane.menz@ifpen.fr;
Contributing authors: miguel.munoz-zuniga@ifpen.fr; delphine.sinoquet@ifpen.fr;
†These authors contributed equally to this work.

Abstract

The robust optimization of engineering systems generally involves the use of complex numerical models that take as input both design variables and random variables that model the uncertainties. The numerous simulations required by such analyses can become computationally very expensive. Moreover, some input conditions can lead to simulation failures or instabilities, due, for instance, to convergence issues of the numerical scheme of complex partial derivative equations. Most of the time, the set of inputs corresponding to failures is not known a priori and thus may be associated to a hidden constraint. Since the observation of a simulation failure regarding this hidden constraint may be as costly as a feasible simulation, we seek to learn the feasible set of inputs and thus target areas without simulation failure before further analysis. In this context, we propose an adaptive Gaussian Process Classifier method to learn the feasible domain. The proposed methodology is based on a Stepwise Uncertainty Reduction strategy on random sets in the classification setting with Gaussian Process Classifiers. The performance of this strategy on different hidden-constrained problems will be presented in particular on an application in wind turbine reliability analysis.

Keywords: Gaussian Process Classifier, Stepwise Uncertainty Reduction, Excursion Set Inversion, Hidden Constraints

1 Introduction

Nowadays, the design of engineering systems implies the use of numerical models, involving complex physical systems with non-linear behaviours and numerous design and uncertain variables. Taking into account a large number of uncertainties, for instance, in the context of uncertainty propagation (Sudret, 2007) or robust design (Moustapha and Sudret, 2019) require numerous simulations of these numerical models and can become computationally very expensive.

Moreover, certain combinations of values of the input variables can lead to simulation instabilities or failures. Indeed, the computer codes can fail to converge due for instance to instabilities in the numerical scheme of complex partial derivative equations (e.g. time step smaller than the critical step), to an inadequate mesh (e.g. not fine enough in areas of high stress concentration) or computed values that tend to infinity (e.g. species concentration outputs in chemical reaction simulations (See (Poirette et al, 2017) for instance)).

In these cases, it is difficult and most often impossible for the expert to determine a priori the

feasible set of inputs i.e. inputs corresponding to converged simulation. Belonging to this feasible set corresponds to a hidden constraint (Digabel and Wild, 2015).

Since the observation of a simulation failure might be as costly as a feasible simulation, it is useful to assess the convergence domain likeliness.

Hence, we seek to learn the feasible set of inputs in order to target areas without simulation failure during processes such as optimization, active or sequential metamodeling, reliability assessment or more basically to study code performances.

No hypothesis (smoothness or regularity) between the failure and feasible areas is assumed and neither is any level of robustness (percentage of failure) of the simulator. Therefore, in order to learn the hidden constraint, we consider in this study that only binary observations corresponding to failure or non-failure status are available.

It is actually a binary classification problem and thus classification modeling techniques can be used to approximate the failure likelihood. Moreover, adaptive approaches are one promising way to reduce the number of expensive simulations for the classifier construction. There is a wide literature on adaptive approaches for regression in particular for Gaussian process (GP) (Bect et al, 2012; Chevalier, 2013; Echard et al, 2011; Bichon et al, 2012; El Amri et al, 2020), Support Vector Machines (Basudhar and Missoum, 2013; Bourinet, 2018) and polynomial-chaos-based Kriging (Schöbi R. et al, 2017) surrogates. In the context of hidden constraints, strategies using GPC (Bachoc et al, 2020; Nickisch and Rasmussen, 2008) during an optimization process, in particular Bayesian optimization, have been proposed (Bachoc et al, 2020) but enrichment points are chosen to improve the optimization and not the classifier in these approaches. In a more global classification setting, active learning strategies have also been proposed for Support Vector Classifier (Bourinet, 2018) and GPC (Zhao et al, 2021b,a).

In this paper, we propose a Gaussian Process Classifier (GPC) based active learning to assess the feasible domain. Indeed, Gaussian process based models have the advantage to provide a measure of uncertainty on the (class) prediction and thus allowing the definition of enrichment criteria naturally derived from this uncertainty measure. The proposed methodology is an adaptation of Stepwise Uncertainty Reduction strategies, usually used in inversion with Gaussian Process Regression, in the classification setting with GPC models.

The rest of the article is organized as follows. First, we provide a presentation of the GPC model and Stepwise Uncertainty Reduction strategies. Then, we present our Stepwise Uncertainty Reduction strategy for classification with GPC models. Finally, we present the performance of our methodology on three application examples. The two first applications are analytical cases and the third one is an industrial case targeting the simulation-based estimation of the accumulated damage of a wind turbine subject to environmental conditions.

2 General settings of hidden constrained problem

Let $f : \Omega \subset \mathbb{R}^d \to \mathbb{R}$ represent a numerical simulator output. Let x be the parameters that are input to this simulator. For some sets of inputs, simulations fail to provide an output. Hence, we assume that, for each simulated input condition x, we get an observation $y(x)$ equal to 0 when the simulator failed to provide a converged output and to 1 when the simulation succeeded.

In the context of hidden constraint learning, the quantity of interest is the feasible domain $\Gamma^* \subset \Omega$ given by:

$$\Gamma^* = \{x \in \Omega \ / \ y(x) = 1\} \tag{1}$$

where $y : \Omega \to \{0, 1\}$ is a priori unknown. Hence, the purpose is to get an estimation of the feasible domain Γ^*.

In the next section, we will focus on GPCs based strategies.

3 The Gaussian process classification model

In probabilistic binary classification, only binary values $y_i \in \{0, 1\}$ are observed. Given these observations, one would like to predict the membership probability of a new point to the class of interest, here class "1" indicating feasibility (convergence

of the simulation). GP based classification models propose to use a latent random process that describes the class membership.

The classical GPC model is described in (Rasmussen and Williams, 2006; Nickisch and Rasmussen, 2008). In this model, the latent distribution posterior is non-gaussian and several approaches have been proposed to build a Gaussian approximation of the posterior (such as the Laplace approximation, Expectation-Propagation, the Kullback-Leibler method and variational method). The article (Nickisch and Rasmussen, 2008) gives an overview of all these methods. The authors conclude that the Expectation-Propagation approximation, although numerically expensive, is the best one in terms of accuracy.

More recently, another Gaussian process (GP) based classification model has been proposed in (Bachoc et al, 2020) that consists in conditioning the latent GP on the signs observations characterizing the belonging to a class. This model has the advantage to come with theoretical guarantees, unlike the first model that implies Gaussian approximation of the posterior. In the following, we will use this latter to build our active learning strategy.

Let Z be a GP characterized by a constant mean function $\mu_Z \in \mathbb{R}$ and a stationary kernel $k_\theta^Z(\cdot,\cdot)$ on Ω^2 with hyperparameters $\theta \in \Theta \subset \mathbb{R}^d$. The posterior distribution of Z knowing the observations $\{\mathcal{X} = (x_1,\dots,x_n), \mathbf{z_n} = (z_1,\dots,z_n)\}$ is Gaussian with posterior mean $m_n(\cdot,\mathbf{z_n})$ and covariance $k_n(\cdot,\cdot)$ given for $x,x' \in \Omega^2$ by

$$m_n(x,\mathbf{z_n}) = \mu_Z + k_\theta^Z(x)K^{-1}(z_n - \mu_Z) \quad (2)$$

$$k_n(x,x') = k_\theta^Z(x,x') + k_\theta^Z(x)K^{-1}k_\theta^Z(x) \quad (3)$$

with $K = k_\theta^Z(\mathcal{X},\mathcal{X})$ the covariance matrix between the observations and $k_\theta^Z(x) = (k(x,x_1),\dots,k(x,x_n))$. In addition, we denote by $\sigma_n^2(x) = k_n(x,x)$ the posterior variance of the GP.

In probabilistic binary classification, we don't have access to the realizations $\mathbf{z_n}$ of the GP Z but only to binary observations $\mathcal{Y} = (y_1,\dots,y_n) \in \{0,1\}^n$ of an indicator of the feasibility domain $\mathbf{Y_n} = (\mathbb{1}_{x_1\in\Gamma^*},\dots,\mathbb{1}_{x_n\in\Gamma^*})$.

The GPC modeling of the feasibility (or non-failure) probability proposed in (Bachoc et al, 2020) consists in defining a latent GP Z and predicting for any $x \in \Omega$, $Y(x) = \mathbb{1}_{Z(x)>0}$ given $\mathbf{Y_n} = (\mathbb{1}_{Z(x_1)>0},\dots,\mathbb{1}_{Z(x_n)>0})$: the 0-1 encoding of the sign of Z(x) knowing the sign of $\mathbf{Z_n} = (Z(x_1),\dots,Z(x_n))$. The feasibility probability is then defined in (Bachoc et al, 2020) as $p_n(x) = \mathbb{P}[Z(x) > 0|\mathcal{X},\mathbf{Y_n} = \mathcal{Y}]$ which is equivalent to the full Bernoulli formulation:

$$p_n(x) = \mathbb{P}[Y(x) = 1|\mathcal{X},\mathbf{Y_n} = \mathcal{Y}]. \quad (4)$$

The feasibility probability can actually be expressed as follows

$$p_n(x) = \int_{\mathbb{R}^n} \bar{\Phi}\left(\frac{-m_n(x,\mathbf{z_n})}{\sigma_n(x)}\right) \phi_{\mathbf{s_n}}^{\mathbf{Z_n}}(\mathbf{z_n})\,\mathrm{d}\mathbf{z_n}, \quad (5)$$

where $\phi_{\mathbf{s_n}}^{\mathbf{Z_n}}(\mathbf{z_n})$ is the Gaussian truncated conditioned probability density function of $\mathbf{z_n}$ restricted to respect $\mathbf{Y_n} = \mathcal{Y}$ and $\bar{\Phi}$ is defined as:

$$\bar{\Phi}(\frac{a}{b}) = \begin{cases} 1 - \Phi(\frac{a}{b}) & \text{if } b \neq 0 \\ \mathbb{1}_{-a>0} & \text{if } b = 0 \end{cases} \quad (6)$$

with Φ is the cumulative density function of the standard normal distribution.

The value of $p_n(x)$ can be approximated by a Monte Carlo method as

$$\widehat{p_n}(x) = \frac{1}{N}\sum_{i=1}^{N} \bar{\Phi}\left(\frac{-m_n(x,\mathbf{z_n^i})}{\sigma_n(x)}\right) \quad (7)$$

where $(\mathbf{z_n^i})_{i=[1,N]}$ are N realizations of the latent random vector $\mathbf{Z_n}$ sampled following the truncated distribution $\phi_{\mathbf{s_n}}^{\mathbf{Z_n}}(\mathbf{z_n})$. As the sampling of the realizations $(\mathbf{z_n^i})_{i=[1,N]}$ is independent of x, it can thus be generated only once (Botev, 2017; Pakman and Paninski, 2014).

The hyperparameters μ_Z and θ of the latent GP must be estimated to sample realizations at observation points and approximate the probability of failure at any point. They are estimated by maximizing the likelihood leading to the estimators

$$(\hat{\mu_Z},\hat{\theta}) = \underset{\mu_Z,\theta}{\arg\max}\, \mathbb{P}_{\mu_Z,\theta}\left[\mathbf{Y_n} = \mathcal{Y}\right]. \quad (8)$$

This likelihood is a Gaussian orthant probability that can be estimated by Monte Carlo based methods (Genz and Bretz, 2009; Botev, 2017; Azzimonti and Ginsbourger, 2018).

Note that the GPC model does not provide a "deterministic" posterior GP mean in the sense that the latent GP posterior mean $m_n(x, \mathbf{Z_n})$ actually depends on the random vector $\mathbf{Z_n}$ of the latent GP at the observation points $\mathcal{X}$.

Hence, we can not make use of existing learning functions in regression that depends directly on the posterior GP mean and covariance function. In the classification setting, we are then restricted to learning strategies that rely on the feasibility probability $p_n(x)$ and the Stepwise Uncertainty Reduction (Bect et al, 2012) paradigm is well suited for this type of problem.

4 Stepwise Uncertainty Reduction strategy for hidden constraints learning

4.1 Stepwise Uncertainty Reduction strategies

A Stepwise Uncertainty Reduction (SUR) strategy aims to sequentially choose a sequence of learning points in order to reduce the future uncertainty given a quantity of interest. In the classification problem, our quantity of interest is the feasible set Γ^*.

Let us consider the random set:

$$\Gamma = \{x \in \Omega / Y_n(x) = 1\} \tag{9}$$

with $Y_n(x)$ the conditional Bernoulli (Dai et al, 2013) random variable $Y(x)|\mathcal{X}, \mathbf{Y_n} = \mathcal{Y}$.

Let $\mathcal{U}_n$ be a measure of uncertainty on the random set Γ knowing observations at points $\mathcal{X}$. A SUR strategy for this uncertainty measure consists in finding at each step the point x^*_{n+1} such that

$$x^*_{n+1} = \underset{x_{n+1} \in \Omega}{\arg\min}\, J_n(x_{n+1}) \tag{10}$$

with

$$J_n(x_{n+1}) = \mathbb{E}_n\left[\mathcal{U}_{n+1}(x_{n+1}, Z(x_{n+1}))\right], \tag{11}$$

where $\mathbb{E}_n[\cdot]$ corresponds to a conditional expectation on the stochastic process used to model the quantity of interest.

In regression, these strategies seek to minimize the expectation with respect to the posterior GP $Z(x_{n+1})$, i.e.

$$\mathbb{E}_n[\cdot] = \mathbb{E}_{Z(x_{n+1})}\left[\cdot|\mathcal{X}, Z(\mathcal{X})\right]. \tag{12}$$

However computing the criterion (11), using this latter formulation in our classification setting, would involve, as in the regression one, a double integration: one on the domain Ω and a second on the random response $Z(x_{n+1})$ (Chevalier, 2013). Moreover, closed-form expressions allowing to reduce the computational cost of the SUR criteria have been proposed in (Chevalier, 2013) for the regression setting but are not applicable here due to the form of the coverage probability $p_n(x)$ in the GPC context.

For this reason, we propose to minimize the expectation of the uncertainty using the expectation on the Bernoulli process $\mathbf{Y_n}$

$$\mathbb{E}_n[\cdot] = \mathbb{E}_{Y_n(x_{n+1})}\left[\cdot|\mathcal{X}, \mathbf{Y_n} = \mathcal{Y}\right]. \tag{13}$$

As $Y_n(x_{n+1})$ is a Bernoulli random variable with probability $p_n(x_{n+1})$, the expression of $J_n(x_{n+1})$ is then given by

$$\begin{aligned} J_n(x_{n+1}) = &(1 - p_n(x_{n+1}))\, \mathcal{U}_{n+1}(x_{n+1}, Y_n(x_{n+1}) = 0) \\ &+ p_n(x_{n+1})\, \mathcal{U}_{n+1}(x_{n+1}, Y_n(x_{n+1}) = 1). \end{aligned} \tag{14}$$

This expression of the criterion J_n allows to get rid of an integration over the realizations of the latent GP $Z(x_{n+1})$.

This criterion can actually be extended to a parallel version that allows to evaluate simultaneously several learning points at each step n. Indeed, let $x^q = (x_{n+1}, \ldots, x_{n+q}) \in \Omega^q$ be a batch of $q \geq 1$ candidate points at step n. $Y_n(x^q) = (Y_n(x_{n+i}))_{i=1,\ldots,q}$ is a multivariate Bernoulli random vector with $Y_n(x_{n+i})$ a Bernoulli random variable with probability $p_n(x_{n+i})$. The expression of the parallel criterion is then given by:

$$J_n(x^q) = \sum_{y^q \in \{0,1\}^q} \mathcal{U}_{n+q}(x^q, y^q) p(y^q) \tag{15}$$

where $y^q = (y_{n+1}, \ldots, y_{n+q}) \in \{0,1\}^q$ are the possible outcomes of $Y_n(x^q)$ and

$$p(y^q) = \left(\prod_{i=1}^{q-1} \mathbb{P}[Y(x_{n+i}) = y_{n+i} | \mathcal{X}, \mathcal{Y}]\right) \times \mathbb{P}[Y(x_{n+q}) = y_{n+q} | \mathcal{X}, \mathcal{Y}, x^{q-1}, y^{q-1}] \quad (16)$$

with $\mathbb{P}[Y(x_{n+i}) = y_{n+i} | \mathcal{X}, \mathcal{Y}]$ equals to $p_n(x_{n+i})^{y_{n+i}}(1 - p_n(x_{n+i}))^{1-y_{n+i}}$ and $\mathbb{P}[Y(x_{n+q}) = y_{n+q} | \mathbf{x}, \mathcal{Y}, x^{q-1}, y^{q-1}]$ equals to $p_{n+q-1}(x_{n+q})^{y_{n+q}}(1 - p_{n+q-1}(x_{n+q}))^{1-y_{n+q}}$.

Note that the number of terms of the sum in (15) increases exponentially with q. Accordingly, the computation time of the criterion (15) - and even more its optimization (10) - is growing explosively with the increase of q.

In the next section, we provide some notions from random set theory (Molchanov, 2005; Vorobyev and Lukyanova, 2013) that allow the definition of a measure of uncertainty on the random set Γ. These notions have already been used to define SUR strategies for inversion (Chevalier, 2013; El Amri et al, 2020).

4.2 Vorob'ev expectation and deviation

Let us define Q_α the α-percentiles of Γ, that are actually the level sets of $p_n(x)$, as

$$Q_\alpha = \{x \in \Omega : p_n(x) \geq \alpha\}, \alpha \in]0,1]. \quad (17)$$

The Vorob'ev expectation (Vorobyev and Lukyanova, 2013) is defined as the α^*-percentile of Γ, where α^* is chosen so that the volume of the defined set equals the expected volume of Γ

$$\mathbb{E}[\mu(\Gamma)] = \mu(Q_{\alpha^*}), \quad (18)$$

with μ the Lebesgue measure. In practice, α^* can be obtained by a simple dichotomy.

The Vorob'ev expectation is actually a global minimiser (see proof in (Molchanov, 2005)), among closed sets of volume equal to the mean volume of Γ, of the Vorob'ev deviation, given by

$$Var_n(\Gamma) = \mathbb{E}[\mu(Q_\alpha \Delta \Gamma) | \mathcal{X}, \mathcal{Y}], \quad (19)$$

with $\Gamma \Delta Q_\alpha = (\Gamma \setminus Q_\alpha) \cup (Q_\alpha \setminus \Gamma)$ the symmetric difference between the two sets Γ and Q_α.

4.3 ARCHISSUR: An Active Recovery of a Constrained and Hidden Subset by SUR method based on the Vorob'ev deviation

For the classification problems that we wish to tackle, we propose an Active Recovery of a Constrained and Hidden Subset by SUR (ARCHISSUR) method. This SUR strategy boils down to iteratively solving the optimization problem (14) with uncertainty measure defined by the Vorob'ev deviation Eq. (19) such that

$$\begin{aligned} &\mathcal{U}_{n+1}(x_{n+1}, Y_n(x_{n+1}) = y_{n+1}) \\ &= Var_{n+1}(\Gamma) \\ &= \int (1 - p_{n+1}(x)) \mathbb{1}_{p_{n+1}(x) \geq \alpha^*} \mu(dx) \\ &+ \int p_{n+1}(x) \mathbb{1}_{p_{n+1}(x) < \alpha^*} \mu(dx) \end{aligned} \quad (20)$$

and more generally problem (15) for $q \geq 1$ with uncertainty measure defined by

$$\begin{aligned} &\mathcal{U}_{n+q}(x^q, Y_n(x^q) = y^q) \\ &= Var_{n+q}(\Gamma) \\ &= \int (1 - p_{n+q}(x)) \mathbb{1}_{p_{n+q}(x) \geq \alpha^*} \mu(dx) \\ &+ \int p_{n+q}(x) \mathbb{1}_{p_{n+q}(x) < \alpha^*} \mu(dx), \end{aligned} \quad (21)$$

where $p_{n+q}(x)$ is the updated feasibility probability estimated with the GPC updated for the considered outcome $\mathbf{y^q} = (y_{n+1}, \ldots, y_{n+q}) \in \{0,1\}^q$ of $Y_n(x^q)$, i.e.

$$p_{n+q}(x) = \mathbb{P}[Y(x) = 1 | \mathcal{X}, \mathcal{Y}, x^q, Y_n(x^q) = \mathbf{y^q}]. \quad (22)$$

Given the estimator of the conditional feasibility probability expression in Eq. (7), the future feasibility probability can be estimated by

$$p_{n+q}(x) = \frac{1}{N} \sum_{i=1}^{N} \bar{\Phi}\left(\frac{-(m_{n+q}(x, \mathbf{z^i_{n+q}})}{\sigma_{n+q}(x)}\right), \quad (23)$$

where $(\mathbf{z^i_{n+q}})_{i=[1,N]}$ are N realizations of the latent random vector $\mathbf{Z_{n+q}} = (Z(x_1), \ldots, Z(x_n), Z(x_{n+1}), \ldots, Z(x_{n+q}))$ sampled from the truncated distribution $\phi_{\mathcal{Y}}^{\mathbf{Z_{n+q}}}(\mathbf{z_{n+q}})$ and $m_{n+q}(x, \mathbf{z^i_{n+q}})$, $k_{n+q}(x, x')$ are respectively the updated mean and covariance function.

For each q-batch of points x^q, the evaluation of the criterion given by (15) (resp. (14) if $q = 1$)

involves 2^q distinct updates of the GPC $Y_n(x)$ for the 2^q possible outcomes of $Y_n(x^q)$. The use of a crude GPC update with $((\mathcal{X}, x^q), (\mathcal{Y}, Y_n(x^q)))$ would require generating 2^q times realizations of a truncated (n+q)-dimensional normal distribution, leading to an untractable criterion. Hence, to allow fast computation of the criterion we propose to use the GP update formulae provided in (Chevalier, 2013) so that we avoid sampling the whole random vector $\mathbf{Z_{n+q}}$.

The updated mean m_{n+q} and covariance function k_{n+q} of the GPC are then given by

$$\begin{aligned} m_{n+q}(x, \mathbf{z^i_{n+q}}) &= m_n(x, \mathbf{z^i_n}) \\ &+ \lambda_{new}(x)^T \left(\mathbf{z^{q,i}} - m_n(x^q, \mathbf{z^i_n})\right), \end{aligned} \tag{24}$$

$$k_{n+q}(x, x') = k_n(x, x') - k_n(x, x^q)^T K_{new}^{-1} k_n(x', x^q) \tag{25}$$

with $\lambda_{new}(x) = K_{new}^{-T} k_n(x, x^q) \in \mathbb{R}^q$, $K_{new} = k_n(x^q, x^q) \in \mathbb{R}^{q \times q}$ and $\mathbf{z^{q,i}} = (z^i_{n+1}, \ldots, z^i_{n+q})$.

Moreover, $\mathbf{z^{q,i}}$ follows the truncated multivariate normal distribution given $\mathbf{z^i_n}$

$$\mathbf{z^{q,i}} \sim \mathcal{N}\left(m_n(x^q, \mathbf{z^i_n}), K_{new}\right) \tag{26}$$

such that $(\mathbb{1}_{z^i_{n+1}>0}, \ldots, \mathbb{1}_{z^i_{n+q}>0}) = \mathbf{y^q}$.

Thus, only one realization of the random vector $\mathbf{z^{q,i}}$ knowing the $\mathbf{z^i_n}$ has to be sampled to compute $m_{n+q}(x, \mathbf{z^i_{n+q}})$. This can be simply done by rejection while not being too numerically expensive if the batch size q is reasonable.

4.4 ARCHISSUR as an extension of another GPC active learning method

Some other GPC based active learning methods have already been proposed. In particular, the Mean Objective Cost of Uncertainty (MOCU) learning function (Yoon et al, 2013), which is defined as the increase of the classification error due to the model uncertainty, is actually related to the SUR strategy even if the link is not done in the paper (Yoon et al, 2013). Indeed, it can be shown that the MOCU function expression (A) can be rewritten as

$$U^{MOCU}(x_{n+1}) = \mathbb{E}_n[Var_{n+1}(\Gamma)] - Var_n(\Gamma) \tag{27}$$

where $Var_n(\Gamma)$ is given by (19) with $\alpha = 1/2$.

In that respect, it can be interpreted as a particular case of the Vorob'ev based SUR strategy for $q = 1$ with a fixed level set α equals to $1/2$, i.e. with the Vorob'ev median instead of the value α^* solving (18).

Note that more recently, a smooth concave approximation of the MOCU function Soft-MOCU has been proposed in (Zhao et al, 2021b) in order to improve efficiency in the long run. In fact, this function is obtained by using a smooth approximation of the maximum function (see the original expression of MOCU in (A1)) by the nested log of the sum of exponentials, generally named LogSumExp (or RealSoftMax). We will compare the results obtained with ARCHISSUR to the ones achieved using the SMOCU learning function in the numerical applications.

5 Applications

5.1 Methodology settings and comparison measures

The GPC active learning method actually provides a random set whose uncertainty has been reduced as a result. Then, a deterministic classifier must be chosen to characterize the feasible set. This can be done by choosing a statistical moment of the random set. A natural choice is then the Vorob'ev expectation which minimizes the symmetric difference to the random set Γ.

The performances of the different learning strategies can be assessed by using different error measures on the built classifiers when we have access to the real feasible set (for analytic examples) by means of a grid of validation point

- relative error on the feasible set:

$$\begin{aligned} crit_F &= \frac{\mu(\Gamma^* \Delta Q_{\alpha^*})}{\mu(\Gamma^*)} \\ &= \frac{FN+FP}{TP+FN} \end{aligned} \tag{28}$$

- the true positives and negatives rates:

$$crit_P = \frac{TP}{TP + FN} \tag{29}$$

$$crit_N = \frac{TN}{TN + FP}. \tag{30}$$

where TP, TN hold respectively for the number of true positives and negatives, i.e. the number of validation points correctly predicted in each class,

and FP, FN for the number of false positives and negatives, i.e. the number of validation points mispredicted in each class.

In practice, the stopping criterion proposed in (El Amri et al, 2020) for SUR strategies can be adopted for ARCHISSUR. Indeed, this criterion is based on the Vorob'ev deviation that is available at each step of the algorithm. This stopping criterion is given by

$$\forall \quad 0 \leq j \leq l, \quad \Delta^j \leq \epsilon \tag{31}$$

with $\Delta^j = |Var_{j-1}(\Gamma) - Var_j(\Gamma)|$ the absolute error on Vorob'ev deviations between two consecutive iterations, ϵ a tolerance value and l the number of steps during which Δ^j must be smaller than the ϵ.

The application of GP based active learning methods is usually done by updating hyperparameters by MLE at each iteration. However, we noticed that ARCHISSUR has better exploration properties when the hyperparameters are constant for a certain number of iterations. Moreover, the Vorob'ev deviation evolution is smoother when fixing hyperparameters, which allows the use of the stopping criterion presented above. Hence, we recommend fixing the hyperparameters for a certain number of iterations and updating them periodically by MLE considering the new observations.

We benchmark the proposed ARCHISSUR strategy with other active learning methods including

- the Soft-MOCU (Zhao et al, 2021b) method using its implementation available at https://github.com/QianLab/NR_SMOCU_SGD_GPC,
- a naive approach for GPC that consists in enriching simultaneously the model with two points corresponding to the one maximising the variance of the latent process and the one which is the closest to a feasibility probability of 0.5,
- and with simple GPC built on Maximin (Pronzato and Müller, 2012) optimal design of experiments (DoE).

Note that the Soft-MOCU method (Zhao et al, 2021b) uses the classical GPC model (Nickisch and Rasmussen, 2008) with the Expectation-Propagation approximation.

5.2 Analytic example in two dimensions

The performances of our proposed method and different methods were assessed on a two-dimensional classification problem based on the Branin function f_{branin} defined as follows

$$y(x) = \begin{cases} 1 \text{ if } f(x) \leq 10 \\ 0 \text{ else} \end{cases} \tag{32}$$

where

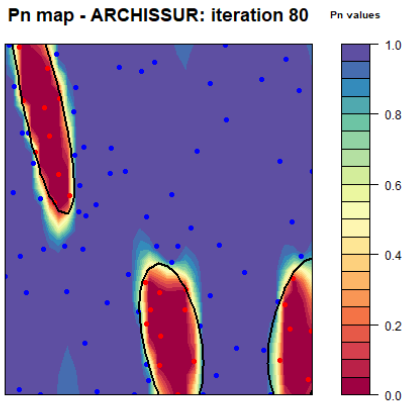
$$f_{branin}(x) = a(x_2 - bx_1^2 + cx_1 - r)^2 + s(1-t)cos(x_1) + s \tag{33}$$

with $a = 1$, $b = {}^{5.1}/_{4\pi^2}$, $c = {}^{5}/_{\pi}$, $r = 6$, $s = 10$ and $t = {}^{1}/_{(8\pi)}$.

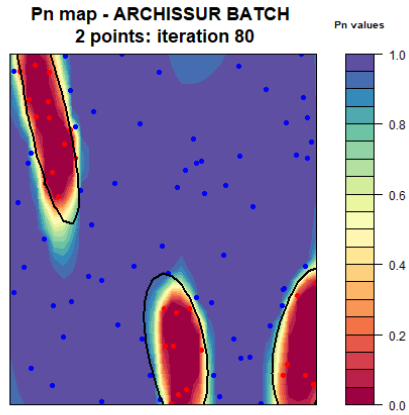
We have applied the naive strategy, the SMOCU method as well as the ARCHISSUR method with one point and with a batch of two points on the example (32). All methods were run for the same 80 initial DoEs of 12 samples, a budget of 80 enrichment points and 1000 integration points (used in the integration-based criterion estimation) for ARCHISSUR and SMOCU. Moreover, the GPC hyperparameters were optimized every 10 iterations for ARCHISSUR and every 5 iterations for ARCHISSUR with a batch of two points.

The final feasibility probability map and DoE resulting from a run of ARCHISSUR, ARCHISSUR BATCH 2 points and the SMOCU method for the same initial DoE are illustrated on Figures 1. On these Figures, we can see that all methods manage to learn a good approximation of the real feasible set (in black lines on the Figures). It can also be observed that the classical GPC model with EP approximation that is used for the SMOCU algorithm is way more smooth than the GPC model based on signs used in ARCHISSUR, as already highlighted in the article (Bachoc et al, 2020).

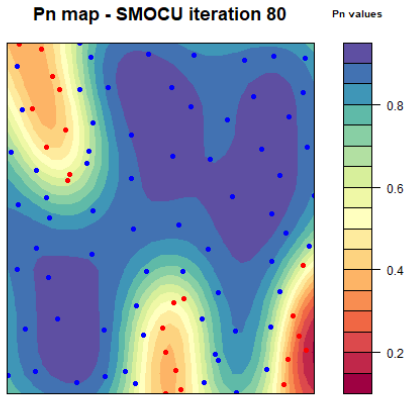
The results obtained, on average, by 80 runs on GPC model with a Maximin DoE of 92 points and the mean results of all active learning methods are presented in Tab. 2. Moreover, the evolution of the relative error through the run of each active learning method is presented on Figure 2. These results show that ARCHISSUR achieves a better



(a) ARCHISSUR



(b) ARCHISSUR BATCH



(c) SMOCU with GPC model (Nickisch and Rasmussen, 2008)

Fig. 1: Probability of feasibility $p_n(x)$ maps obtained with ARCHISSUR, ARCHISSUR BATCH with 2 points and SMOCU active learning methods

estimation of the feasible set on this example and more precisely the 2 points batch version gives a smaller error mean. In fact, the significant difference of performance between ARCHISSUR and SMOCU algorithms is mainly due to the type of GPC model chosen in the original SMOCU algorithm implementation. Indeed, using the classical GPC model in ARCHISSUR implementation is greatly reducing its efficiency.

In order to compare the criteria and their implementations in terms of numerical cost, the computational times of ARCHISSUR, ARCHISSUR for a batch of two points, ARCHISSUR formulation using usual SUR definition applied to classification Eq. (12) and SMOCU criteria were assessed on this test case and are given in Table 2. Note that the proposed ARCHISSUR formulation allows reducing on average the computational time by almost 4

Method	$\mathbb{E}[crit_F]$	$CoV(crit_F)$
GPC (Bachoc et al, 2020)	4.97×10^{-2}	0.23
ARCHISSUR	2.86×10^{-2}	0.30
ARCHISSUR BATCH	$\mathbf{3.08 \times 10^{-2}}$	0.23
SMOCU	1.32×10^{-1}	0.25
MIX	4.82×10^{-2}	0.28

Table 1: Branin-Hoo based analytical hidden constraint — mean and coefficient of variation (CoV) of the relative error $crit_F$ (28) for each method

in comparison to the usual definition adapted to classification.

Method	CPU time (s)
ARCHISSUR	**0.079**
ARCHISSUR BATCH (2 points)	0.412
ARCHISSUR with formulation 12	0.311
SMOCU	0.849

Table 2: Mean of system and user CPU times sum (in seconds) on 20 repetitions for each criterion evaluation with 2000 integration points x

The Figures 3 to 6 show the evolution of the true negative and positives rates $crit_N$ and $crit_P$ during the runs of each algorithm. From these Figures, it can be noted that the approximation obtained by SMOCU is more conservative, in the sense that the unfeasible domain is overestimated (Rates of true positive around 20%), compared to the other methods (Rates of true positives around 98%). This is as well a consequence of the smoothness of the model.

5.3 Analytic example in ten dimensions

The performances of the different algorithms were tested in higher dimension on an analytical example with 10 inputs $x \in [0,1]^{10}$, given by:

$$y(x) = \begin{cases} 0 \text{ if } x \in \{x/f_1(x) \geq 0\} \cup \{x/f_2(x) \geq 0\} \\ 1 \text{ else} \end{cases} \tag{34}$$

where:

$$\begin{aligned} f_1(x) &= x_2 + x_1 - 1.6 + 0.1x_3 \\ f_2(x) &= 0.15(0.1 + x_4) - (x_5^2 + (x_6 + 0.1)^2) \end{aligned} \tag{35}$$

The feasible set represents 86% of the domain for this example and the unfeasible sets are located at borders of the domain with different shapes.

On this example, only the results obtained with the proposed algorithm ARCHISSUR and the naive said Mix, method are presented on Figure 7.

Indeed, the optimization of the acquisition function in SMOCU method is done by random optimization (see (Zhao et al, 2021b)) that weakens the chances of coming across good exploration or exploiting points in higher dimensions and thus leads to poor results on this test case.

The naive strategy and the ARCHISSUR method with one point were applied with initial DoEs of 60 points, a budget of 500 enrichment points and 10000 integration points for ARCHISSUR. Moreover, the GPC hyperparameters were optimized every 10 iterations for the first 100th iterations and on an interval of 20 iterations then.

The evolution of the relative error $crit_F$ for the two learning criteria is given on Figure 7. It can be observed that the relative error is decreasing a little faster for the naive method during the 250th first iterations but with notable outliers that reach higher error values in comparison to the evolution for ARCHISSUR which is more regular. After the 250th iteration, the relative error continues to decrease and achieves lower values with ARCHISSUR while the error interval for the naive method is wider with higher values. Table 4 gives the mean and coefficient of variation of $crit_F$ obtained at the end of each algorithm as well as the values for a GPC built on a Maximin DoE with the equivalent number of points as for the algorithms, i.e. 560 points. On the basis of these results, we can note that ARCHISSUR continues to perform well in higher dimensions with a final mean relative error 2.1 times lower than a naive method with far more regular results and 4.2 times lower than a crude Maximin DoE.

The Figures 8 and 9 show the evolution of the true negative and positives rates $crit_N$ and $crit_P$ during the runs of both naive and ARCHISSUR algorithms. From these Figures, it can be noted that ARCHISSUR achieves better and more regular predictions of the feasible set.

Independently from the results, the computational times of the different criteria were assessed on this test case and are given in Table 4.

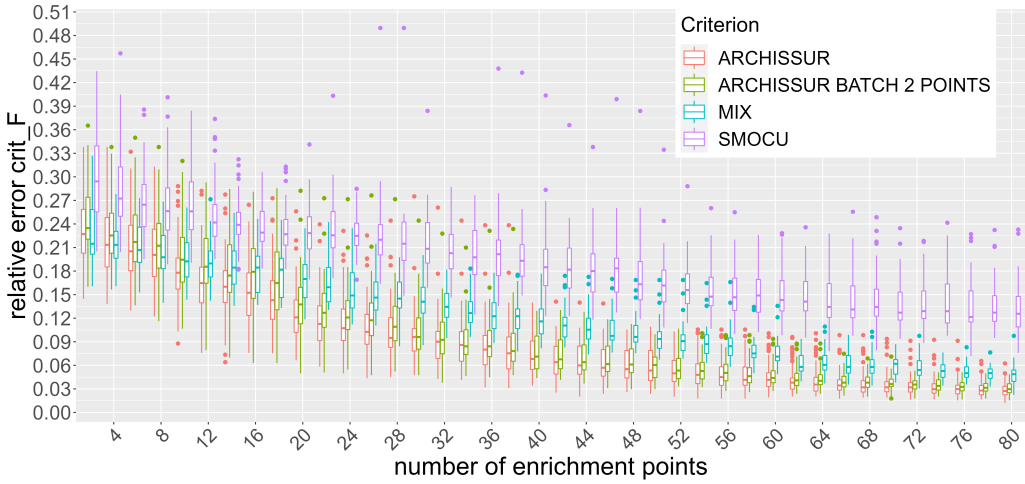


Fig. 2: Evolution of the relative error on the feasible set $crit_F$ as a function of the number of enrichment points for the Branin based example

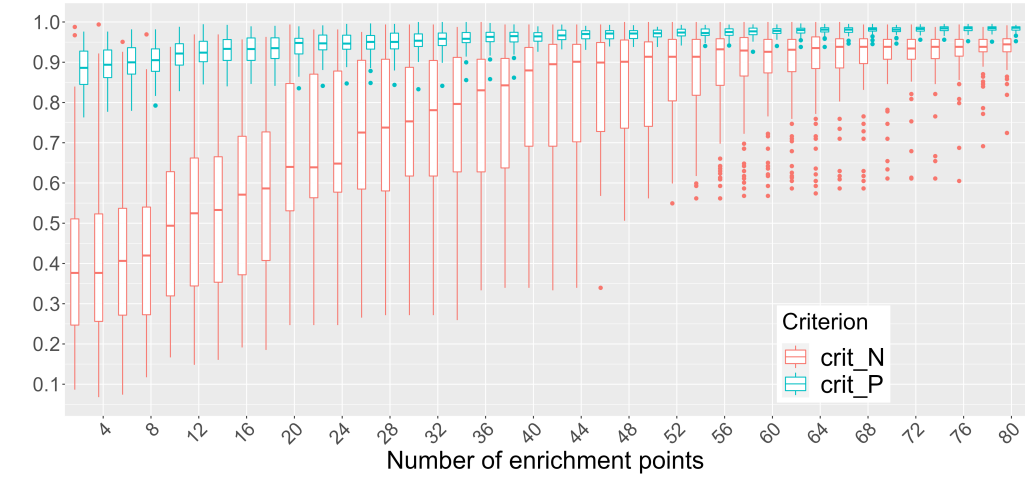


Fig. 3: Evolution of the true positives $crit_P$ and true negatives $crit_N$ rates as a function of the number of enrichment points throughout a run of ARCHISSUR on the Branin based example

Method	$\mathbb{E}[crit_F]$	$CoV(crit_F)$
GPC (Bachoc et al, 2020)	2.24×10^{-1}	0.86
ARCHISSUR	$\mathbf{5.3 \times 10^{-2}}$	0.38
MIX	1.12×10^{-1}	1.11

Table 3: Ten inputs example — mean and coefficient of variation (CoV) of the relative error $crit_F$ (28) for each method applied on the ten-dimensional example

Method	CPU time (s)
ARCHISSUR	**0.363**
ARCHISSUR BATCH (2 points)	4.529
ARCHISSUR with formulation 12	1.670
SMOCU	7.58

Table 4: Mean System and User CPU time sum (in seconds) on 10 repetitions for each criterion evaluation with 10000 integration points x on the ten-dimensional test case

We can note that the numerical cost of ARCHISSUR with a batch of two points goes up significantly in comparison to the times measured

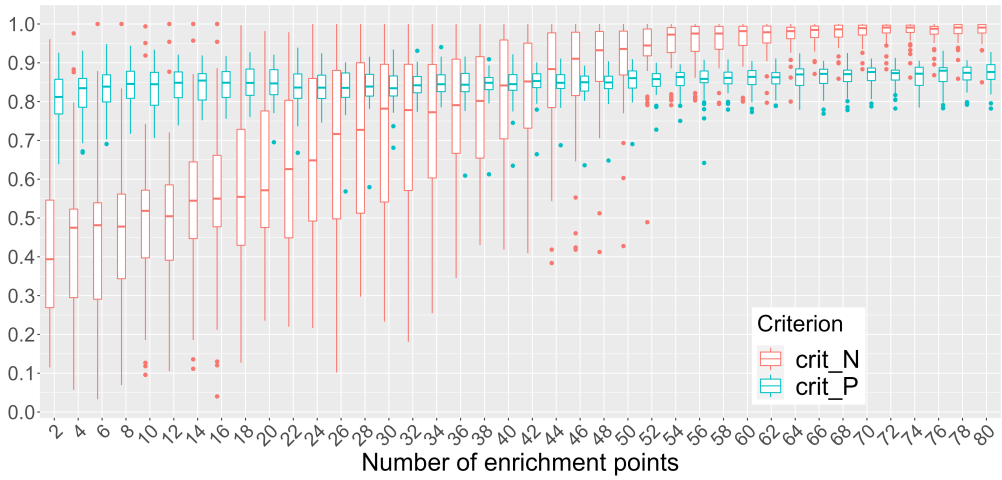


Fig. 4: Evolution of the true positives $crit_P$ and true negatives $crit_N$ rates as a function of the number of enrichment points throughout a run of SMOCU on the Branin based example

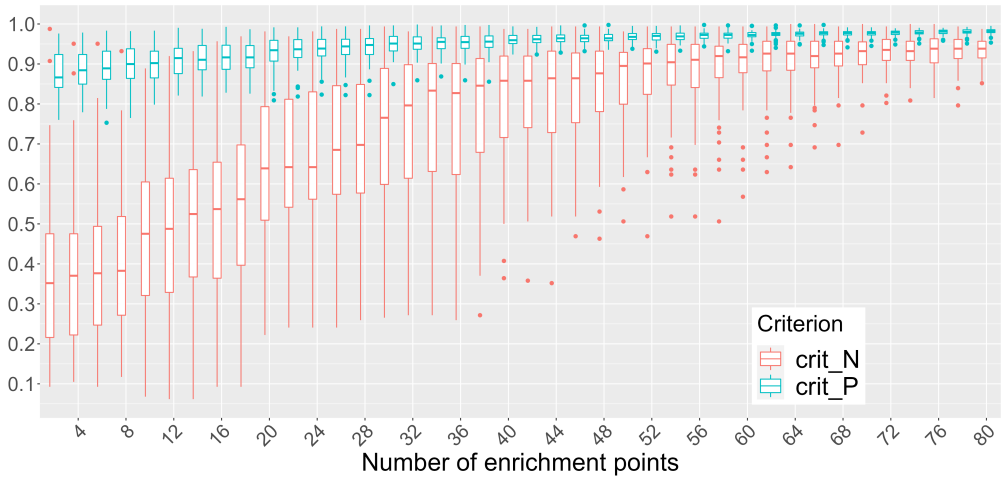


Fig. 5: Evolution of the true positives $crit_P$ and true negatives $crit_N$ rates as a function of the number of enrichment points throughout a run of ARCHISSUR with a batch of two points on the Branin based example

in two dimensions. Combined with the cost of optimization in ten dimensions, the use of a batch strategy becomes quickly intractable.

5.4 Hidden constraint study of a wind turbine damage computation code

5.4.1 Description of the problem

In this section, the damage prediction of an onshore wind turbine NREL 5MW is studied. As illustrated on Figure 10, the wind turbine is subject to several wind loads that cause damage at the base of the tower. The computation of the damage involves the use of the open-source multi-physics simulator FAST (Jonkman and Buhl Jr., 2005) taking as inputs four environmental variables that concerns the wind loads. The simulator provides an estimation of one hour damage for 36 regularly spaced impact points that represent the damages on the whole wind turbine base of the tower circumference.

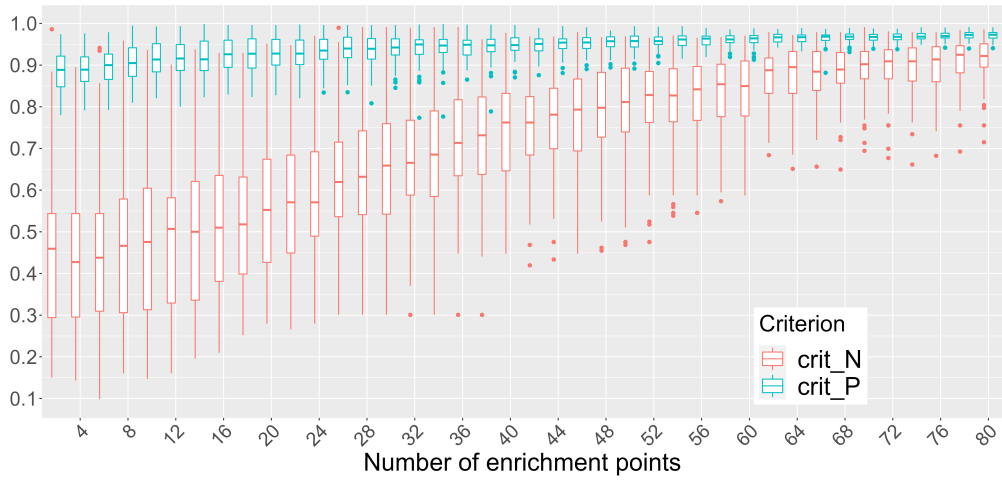


Fig. 6: Evolution of the true positives $crit_P$ and true negatives $crit_N$ rates as a function of the number of enrichment points throughout a run of the naive algorithm on the Branin based example

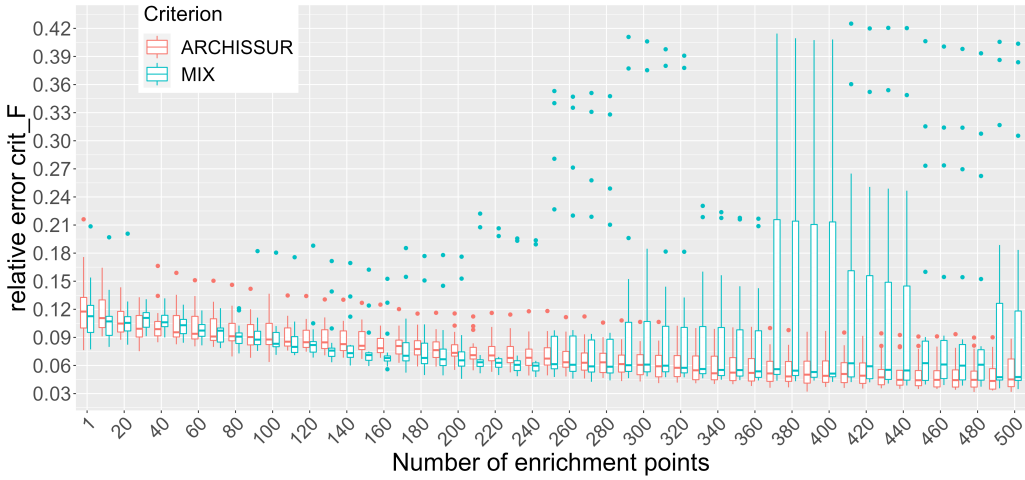


Fig. 7: Evolution of the relative error on the feasible set $crit_F$ as a function of the number of enrichment points for the ten dimensional example

The wind direction θ_{wind} influence on the damage is considered negligible and this parameter is fixed to 30°. Hence, the damage d computation depends on the three following parameters: the mean wind speed in the 10-minute interval: $\bar{U}$, the turbulence intensity: TI and the misalignment angle of the wind turbine blades: $NacYaw$. These parameters are supposed to be uncertain and are modeled by independent random variables following probability distributions given in Table 5.

Input	$\bar{U}$	TI	$NacYaw$
Unit	m/s	%	°
Probability law	Uniform	Uniform	Uniform
minimum	10	2.5	-20
maximum	22	25	20

Table 5: Probability distributions of the wind turbine problem parameters.

Moreover, the wind is modeled by a stochastic process and for each set of inputs $(\bar{U}, TI, NacYaw)$ realizations of the wind are simulated using the open source software Turbsim

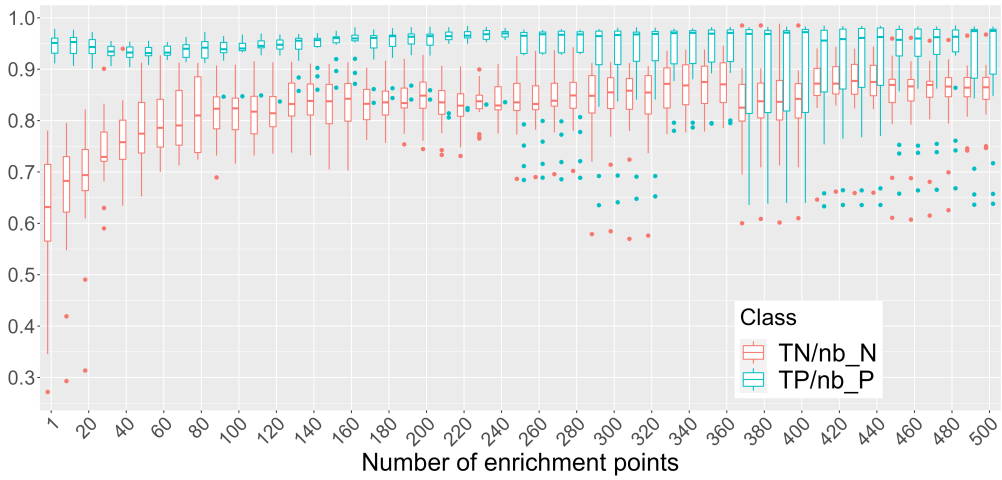


Fig. 8: Evolution of the true positives $crit_P$ and true negatives $crit_N$ rates as a function of the number of enrichment points throughout a run of the naive algorithm on the ten-dimensional example

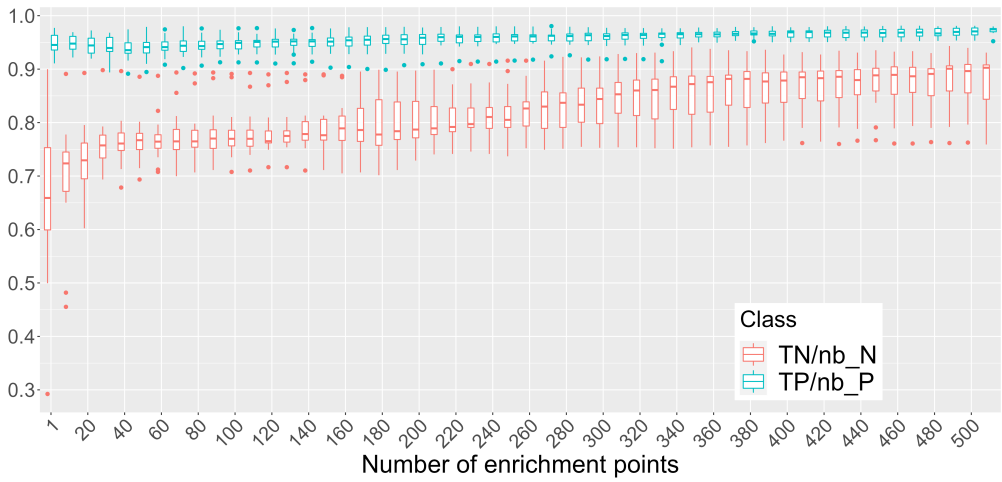


Fig. 9: Evolution of the true positives $crit_P$ and true negatives $crit_N$ rates as a function of the number of enrichment points throughout a run of ARCHISSUR algorithm on the ten-dimensional example

(Jonkman and Buhl Jr., 2006). In the following, we chose arbitrarily to fix the number of wind realizations to 18 per input set.

However, FAST and/or the Turbsim simulators encounter simulation crash due to poor convergence for some values of the inputs. The feasible domain can thus be estimated in this test case in order to avoid simulation failures when predicting damage values. We consider that a simulation fails for an input set when a failure happens for at least one wind realization.

This test case is interesting as it allows to test the algorithms on a real physical problem on one side and is numerically affordable enough to simulate test points to have a picture of the feasible domain on the other. Indeed, short-term wind simulation and a rather small number of uncertain variables are considered, leading to a few minutes-long simulation with parallelization use. Thus, we have simulated 3000 test points whose classification in regards to simulation failure is plotted on

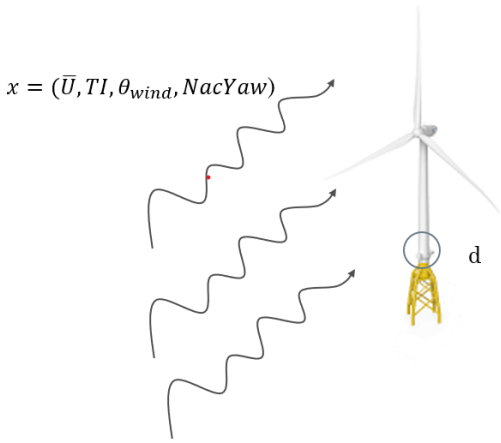


Fig. 10: Diagram of an onshore wind turbine subject to wind loads described by 4 parameters: wind speed $\bar{U}$, wind direction θ_{wind}, turbulence intensity TI and misalignment angle $NacYaw$.

Figure 11a. On this Figure, it can be observed that the feasible and unfeasible domains can be rather well differentiated. But we can also note that the frontiers are blurred and detect the presence of some outliers. Moreover, the feasible domain coverage, estimated from these test points, is about 58.8% of the domain.

5.4.2 Results

Contrary to the previous example, the hidden constraint is non-deterministic due to the stochasticity of the wind. As indicated in the previous section, the learning algorithm might encounter outliers and the GPC model is challenged by a blurred limit.

Thus, in order to get a more realistic modelling of the problem we consider a noisy latent GP. Considering independent homoscedastic noise σ_n, the prior $k_\theta^Z(\mathbf{x},\mathbf{x})$ becomes:

$$k_{\theta,\sigma_n}^Z(\mathbf{x},\mathbf{x}) = k_\theta^Z(\mathbf{x},\mathbf{x}) + \sigma_n I_n \qquad (36)$$

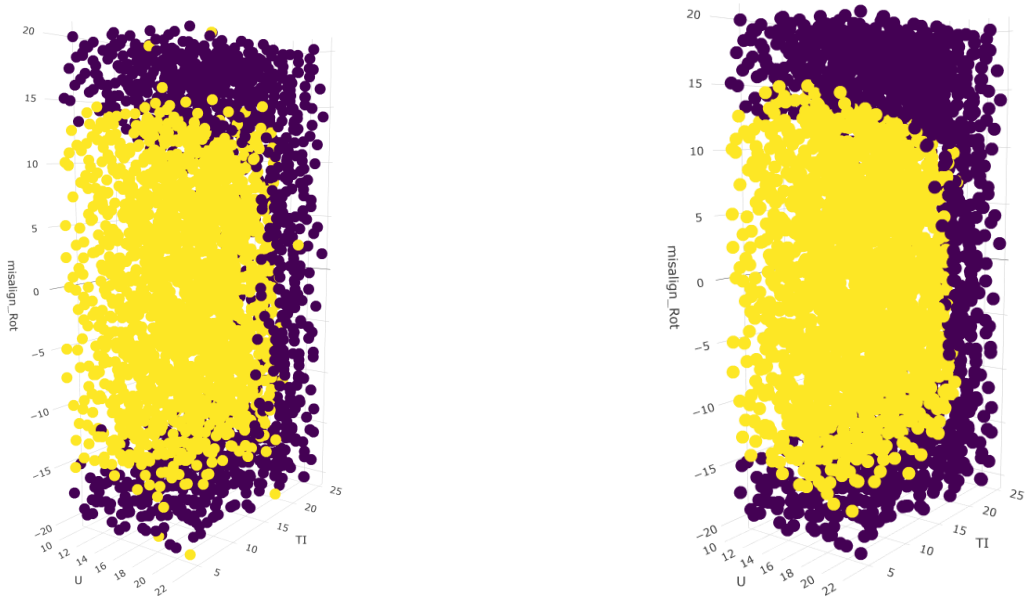
In this model, the noise is an additional hyperparameter to be optimized by maximisation of the likelihood (MLE) as presented in Section 3.

The ARCHISSUR method with and without noise and the SMOCU algorithms have been applied on this example. All methods were run for the same 20 initial DoEs of 20 samples, a budget of 200 enrichment points and 5000 integration points. The error indicators were computed on the basis of the same 3000 test points.

The evolution of the relative error through the run of each active learning method are presented on Figure 12. This Figure shows that the use of a noisy model greatly improves the results of ARCHISSUR. Moreover, it can be observed that the relative error decreases faster using ARCHISSUR with noisy GPC than SMOCU. Both methods achieve an almost constant relative error value between 0.10 and 0.12, which is mainly due to the non-regularity of the test points at the frontiers of the feasible set.

The Figures 13 and 14 show the evolution of $crit_N$ and $crit_P$ during the runs of respectively ARCHISSUR with noisy GPC and SMOCU methods. We can note that both methods assess a true positive rate of 0.95 and a true positive rate of 0.925, which confirms the observation made with the relative error $crit_F$.

Finally, we provide on Figure 11b an example of the classification of the test points (see Figure 11a to observe the true class of each point) obtained at the end of a run of ARCHISSUR with a noisy GPC. This Figure allows to verify the conclusion drawn from the observation of the error reduction limit, i.e. the classification of the points at the border is more regular than the reality and the model obviously does not predict outliers.



(a) Wind turbine damage simulated points classification: purple dots account for failed simulations and yellow dots for converged points

(b) Example of the classification obtained at the end of a run of ARCHISSUR with noisy GPC on the wind turbine test case

Fig. 11: Wind turbine damage validation points

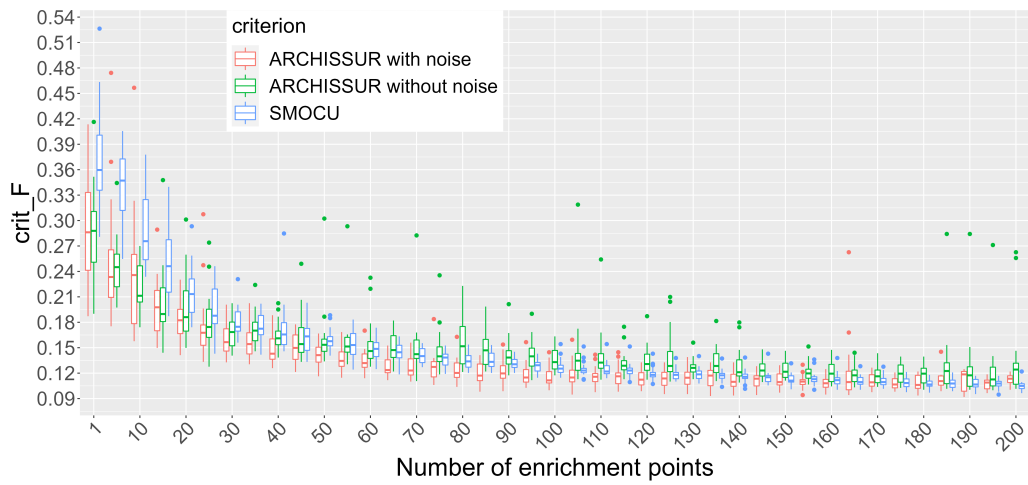


Fig. 12: Evolution of the relative error on the feasible set $crit_F$ as a function of the number of enrichment points for the wind turbine test case

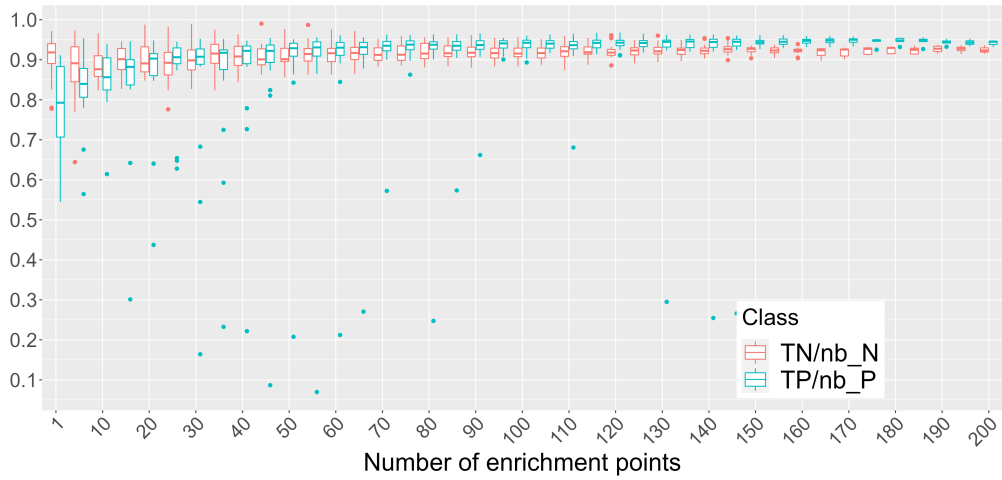


Fig. 13: Evolution of the true positives $crit_P$ and true negatives $crit_N$ rates as a function of the number of enrichment points throughout a run of ARCHISSUR with noisy GPC on the wind turbine test case

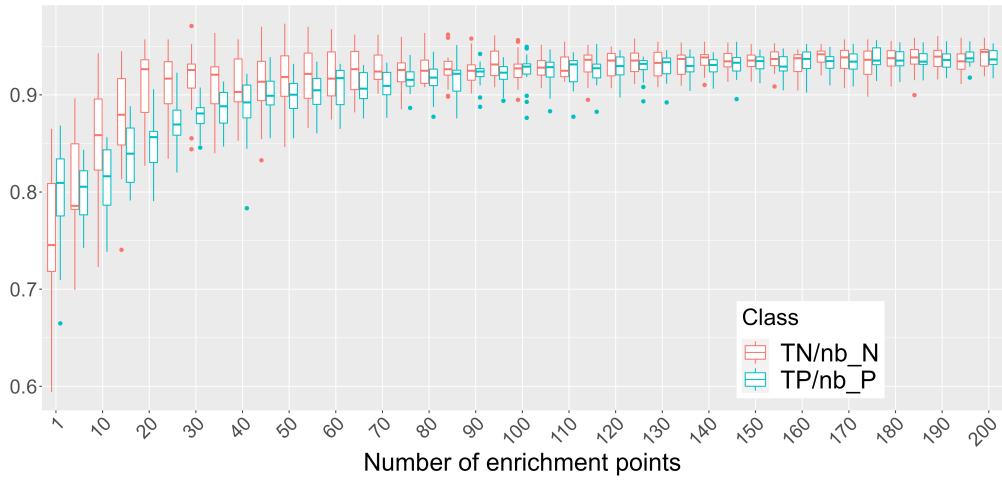


Fig. 14: Evolution of the true positives $crit_P$ and true negatives $crit_N$ rates as a function of the number of enrichment points throughout a run of SMOCU on the wind turbine test case

5.5 Discussion and conclusions

In this paper, we proposed a GPC active learning method based on Stepwise Uncertainty Reduction strategies to assess hidden constraints prediction. We provided a formulation of the enrichment criteria suited for classification that is less numerically expensive and that allows the selection of a batch of multiple points at a time in order to improve the classification model.

The proposed algorithm was benchmarked with other methods on three different applications: two analytical examples and an industrial case. In the different applications, we have highlighted the importance of the choice of the GPC model for the performances of the algorithm. Indeed, we have noticed that the GPC model (Bachoc et al, 2020) shows improved performances on completely deterministic hidden constraints. In the non-determistic case, meaning outliers exist or frontiers between feasible and unfeasible domains are blurred, a smoother model performs better. Hence, more accurate predictions were achieved by using a noisy GPC model in this context.

Moreover, we have observed that the use of the batch strategy becomes quickly untractable when the problem dimension increases. Future work should involve an improvement of the learning function to allow the use of the batch strategy.

Another perspective of this work is its use in constrained optimization. Indeed, hidden crash constraints are a well-known problem in design optimization. However, the learning of crash constraints is usually done using the points proposed by the optimizer. Hence, an interesting perspective could be to combine it wisely with the learning of crash constraints themselves.

In order to avoid the costly MC probability estimation (7), a final opening for further work could be to consider the problem without the introduction of a latent space by directly expressing the feasibility probability p_n as the result of a conditional Bernoulli model (Dai et al, 2013).

Acknowledgments. This work was supported by the French National Research Agency (ANR) through the SAMOURAI project under grant ANR20-CE46-0013.

Appendix A Mean Objective Cost of Uncertainty

The Mean Objective Cost of Uncertainty (MOCU) function as defined in (Yoon et al, 2013) is written with our notations as follows

$$\begin{aligned} &U^{MOCU}(x_{n+1}) \\ &= \mathbb{E}_{Y_n(x_{n+1})}[\mathbb{E}_X\left[1-\max(p_{n+1}(X),1-p_{n+1}(X))|\mathcal{X},\mathcal{Y}\right]] \\ &-\mathbb{E}_X\left[1-\max\left(p_n(X),1-p_n(X)\right)\right]. \end{aligned} \tag{A1}$$

In fact:

$$\begin{aligned} &\mathbb{E}_X\left[1-\max\left(p_n(X),1-p_n(X)\right)\right] \\ &= \int_\Omega 1-\max\left(p_n(x),1-p_n(x)\right)dx \\ &= \int_\Omega (1-p_n(x))\mathbb{1}_{p_n(x)\geq 1/2}dx + \int_\Omega p_n(x)\mathbb{1}_{p_n(x)<1/2}dx \\ &= \int_{Q_{1/2}} p_n(x)dx + \int_{Q_{1/2}^C} 1-p_n(x)dx \\ &= Var_n(\Gamma) \end{aligned} \tag{A2}$$

with

$$Q_{1/2} = \{x\in\Omega : p_n(x)\geq 1/2\}. \tag{A3}$$

Similarly, we have

$$\mathbb{E}_X\left[1-\max\left(p_{n+1}(X),1-p_{n+1}(X)\right)\right] = Var_{n+1}(\Gamma). \tag{A4}$$

Hence, the MOCU function can also be expressed as

$$U^{MOCU}(x_{n+1}) = \mathbb{E}_n[Var_{n+1}(\Gamma)] - Var_n(\Gamma) \tag{A5}$$

where $Var_n(\Gamma)$ is given by (19) with $\alpha = 1/2$.

References

Azzimonti D, Ginsbourger D (2018) Estimating orthant probabilities of high-dimensional gaussian vectors with an application to set estimation. Journal of Computational and Graphical Statistics 27(2):255–267. https://doi.org/10.1080/10618600.2017.1360781, URL https://doi.org/10.1080/10618600.2017.1360781, https://arxiv.org/abs/https://doi.org/10.1080/10618600.2017.1360781

Bachoc F, Helbert C, Picheny V (2020) Gaussian process optimization with failures: classification and convergence proof. Journal of Global Optimization 78(3):483–506. https://doi.org/10.1007/s10898-020-00920-0, URL https://link.springer.com/10.1007/s10898-020-00920-0

Basudhar A, Missoum S (2013) Reliability assessment using probabilistic support vector machines. International Journal of Reliability and Safety 7(2):156–173. https://doi.org/10.1504/IJRS.2013.056378, URL https://www.inderscienceonline.com/doi/abs/10.1504/IJRS.2013.056378

Bect J, Ginsbourger D, Li L, et al (2012) Sequential design of computer experiments for the estimation of a probability of failure. Statistics and Computing 22(3):773–793. https://doi.org/10.1007/s11222-011-9241-4, URL https://link.springer.com/article/10.1007/s11222-011-9241-4

Bichon BJ, Eldred MS, Mahadevan S, et al (2012) Efficient Global Surrogate Modeling for Reliability-Based Design Optimization. Journal of Mechanical Design 135(1):011,009–011,009–13. https://doi.org/10.1115/1.4022999, URL

http://dx.doi.org/10.1115/1.4022999

Botev ZI (2017) The normal law under linear restrictions: simulation and estimation via minimax tilting. Journal of the Royal Statistical Society: Series B (Statistical Methodology) 79(1):125–148

Bourinet JM (2018) Reliability analysis and optimal design under uncertainty-Focus on adaptive surrogate-based approaches. PhD Thesis, Université Clermont Auvergne

Chevalier C (2013) Fast uncertainty reduction strategies relying on Gaussian process models. PhD Thesis

Dai B, Ding S, Wahba G (2013) Multivariate Bernoulli distribution. Bernoulli 19(4):1465 – 1483. https://doi.org/10.3150/12-BEJSP10, URL https://doi.org/10.3150/12-BEJSP10

Digabel SL, Wild SM (2015) A taxonomy of constraints in simulation-based optimization. arXiv preprint arXiv:150507881

Echard B, Gayton N, Lemaire M (2011) AK-MCS: An active learning reliability method combining Kriging and Monte Carlo Simulation. Structural Safety 33(2):145–154. https://doi.org/10.1016/j.strusafe.2011.01.002, URL http://www.sciencedirect.com/science/article/pii/S0167473011000038

El Amri MR, Helbert C, Lepreux O, et al (2020) Data-driven stochastic inversion via functional quantization. Statistics and Computing 30(3):525–541

Genz A, Bretz F (2009) Computation of multivariate normal and t probabilities, Lecture Notes in Statistics, vol 195. Springer Science & Business Media

Jonkman JM, Buhl Jr. ML (2005) "FAST User's Guide" NREL/EL-500-29798. National Renewable Energy Laboratory, Golden, CO., URL https://www.nrel.gov/wind/nwtc/openfast.html

Jonkman JM, Buhl Jr. ML (2006) "TurbSim User's Guide " NREL/TP-500-39797. National Renewable Energy Laboratory, Golden, CO., URL https://www.nrel.gov/wind/nwtc/turbsim.html

Molchanov I (2005) Theory of random sets, vol 19. Springer

Moustapha M, Sudret B (2019) Surrogate-assisted reliability-based design optimization: a survey and a new general framework. arXiv:190103311 [stat] URL http://arxiv.org/abs/1901.03311

Nickisch H, Rasmussen CE (2008) Approximations for binary gaussian process classification. Journal of Machine Learning Research 9(67):2035–2078. URL http://jmlr.org/papers/v9/nickisch08a.html

Pakman A, Paninski L (2014) Exact hamiltonian monte carlo for truncated multivariate gaussians. Journal of Computational and Graphical Statistics 23(2):518–542. URL http://www.jstor.org/stable/43305741

Poirette Y, Guiton M, Huwart G, et al (2017) An optimization method for the configuration of inter array cables for floating offshore wind farm. In: International Conference on Offshore Mechanics and Arctic Engineering, American Society of Mechanical Engineers, p V010T09A072

Pronzato L, Müller WG (2012) Design of computer experiments: space filling and beyond. Statistics and Computing 22(3):681–701

Rasmussen CE, Williams CKI (2006) Gaussian processes for machine learning. Adaptive computation and machine learning, MIT Press, Cambridge, Mass

Schöbi R., Sudret B., Marelli S. (2017) Rare Event Estimation U²sing Polynomial-Chaos Kriging. ASCE-ASME Journal of Risk and Uncertainty in Engineering Systems, Part A: Civil Engineering 3(2):D4016,002. https://doi.org/10.1061/AJRUA6.0000870, URL https://ascelibrary.org/doi/abs/10.1061/AJRUA6.0000870

Sudret B (2007) Uncertainty propagation and sensitivity analysis in mechanical models–Contributions to structural reliability and

stochastic spectral methods. Habilitationa diriger des recherches, Université Blaise Pascal, Clermont-Ferrand, France 147

Vorobyev OY, Lukyanova NA (2013) A Mean Probability Event for a Set of Events. Journal of Siberian Federal University Mathematics and Physics p 128–136

Yoon BJ, Qian X, Dougherty ER (2013) Quantifying the objective cost of uncertainty in complex dynamical systems. IEEE Transactions on Signal Processing 61(9):2256–2266

Zhao G, Dougherty E, Yoon BJ, et al (2021a) Efficient active learning for gaussian process classification by error reduction. Advances in Neural Information Processing Systems 34:9734–9746

Zhao G, Dougherty E, Yoon BJ, et al (2021b) Bayesian active learning by soft mean objective cost of uncertainty. In: International Conference on Artificial Intelligence and Statistics, PMLR, pp 3970–3978