# Model-based sequential designs of computer experiments in GPs

Ruizi Yan (s2265990@ed.ac.uk)
Supervised by Professor Finn Lindgren
University of Edinburgh

19th August 2022

## 1 Introduction

Computer experiments have been widely used to emulate complex processes in many fields; for example, estimating the endangered wildlife population in ecology[14], simulating the chemical capture process [9] and forecasting the energy in power systems[6]. These problems usually correspond to the estimation of output given multiple inputs. It happens quite often that only limited information is available about the underlying process, or it is complicated to replicate. For example, when trying to model the distribution of sheep in Scotland, it is impossible to explore everywhere across Scotland. In such cases, a surrogate model (or meta-model) can be constructed by taking observations at certain locations to mimic the original process at an acceptable discrepancy level. While computer simulations can overcome the difficulties which may occur in field studies like experimental resources shortage or long-term results collection, it can be computationally intensive and time-consuming for modelling complex systems. In such cases, fewer operations are desirable to reach the goal. Therefore, optimal experimental designs are needed to make as much use as possible of available information and to avoid the waste of the budget. Classic one-shot sampling policy takes fixed number of observations all at once in advance of experiments, which can easily cause either lack of accuracy or waste of time as in practice one does not know how many observations are needed. Sequential strategy can deal with this problem by choosing the follow-up sample points iteratively based on previous observations, so that one can stop the simulation process when the objective is satisfied or after exhausting the maximum budgets. This kind of methods work efficiently in many applications, such as global fitting, optimization and classification.

In this project, we consider particularly the problem of global fitting, and investigate how the sequential design strategies work with Gaussian Processes. The GP surrogate model is fitted with designs iteratively determined by sampling criteria. Among various criteria serving for different objectives, we will detail in four criteria, Maximum Mean Squared Error (MMSE) criterion, Maximum Entropy (ME) criterion, Integrated Mean Squared Error (IMSE) criterion and Integrated Mean Dawid-Sebastiani (IMDS) criterion. It can be shown that ME criteria in fact is equivalent to MMSE criteria in the case of GPs. Squared Error (SE) and Dawid-Sebastiani (DS) scores will be used to assess models constructed by different criteria.

The report is structured as follows: section 2 gives background information about Bayesian framework, GPs and computer experimental design, which are needed for model constructions. In section 3, four criteria, MMSE, ME, IMSE and IMDS, are presented. Implementations and illustrations are provided in sections 3.6 and 3.7 3.8. Finally, we have a short conclusion in section 4.

# 2 Background

## 2.1 Bayesian framework in linear regression

Bayesian framework comes a lot in modern statistics modeling where the problems of parameter estimation and model prediction are concerned. Unlike traditional frequentist statistics where inference is made by fixed model parameters, Bayesian statistics treats parameters as random variables with unknown distributions. Instead of identifying the best fitted parameters, Bayesian statistics gives a posterior distribution over parameters which allows us to make predictions at new locations and quantify uncertainty in model estimates. For a comprehensive review, see [12].

Let $Y$ be a random variable from a distribution with unknown parameters $\theta$. Assuming a prior distribution over $\theta$ is given, the posterior distribution conditioned on observed data $Y = \mathbf{y}$ can be calculated as

$$P(\theta|\mathbf{y}) = \frac{p(\theta)p(\mathbf{y}|\theta)}{\int_{\theta^\star} p(\theta^\star)p(\mathbf{y}|\theta^\star)d\theta^\star}. \tag{1}$$

Then the posterior predictive distribution of $\mathbf{y}^\star$ at new test locations $x^\star$ is given by

$$p(\mathbf{y}^\star|\mathbf{x}^\star, \mathbf{y}) = \int_\theta p(\mathbf{y}^\star|\mathbf{x}^\star, \theta)p(\theta|\mathbf{y})d\theta. \tag{2}$$

In general, since the integral involved in (1) (2) are difficult to compute, methods like Monte Carlo integration are resorted to approximate the original exact integrals. However, things become tractable under Gaussian settings, as we will show in section 2.2.

## 2.2 Gaussian processes

Gaussian Process (GP) has been a canonical surrogate model due to its uncertainty quantification competency and nonlinear flexibility. Typically, it describes a distribution over functions characterised by a mean function $m(x)$ and a covariance function (which is also called kernel) $k(x, x^{'})$, denoted by

$$f(.) \sim \mathcal{GP}(m(.), k(.,.)). \tag{3}$$

The mean function $m(x)$ gives the expectations of the output and the kernel is used to define the covariance of two inputs $x_i$ and $x_j$. We will see how this functional distribution can be fitted by a set of training samples and how predictions at new testing locations can be made under the Bayesian framework.

Let $f : \mathbb{R}^d \to \mathbb{R}$ be the objective function which is unknown or expensive to evaluate. Let $\mathbf{S} = \{(x^{(i)}, y^{(i)}\}_{i=1}^m$ denotes a set of i.i.d. samples drawn from the objective function $f$. Then a Gaussian Process regression model can be fitted with $\mathbf{S}$, which can be used to make prediction at new locations in lieu of expensive evaluations.

In the Gaussian process regression model,

$$y^{(i)} = f(x^{(i)}) + \epsilon^{(i)}, i = 1, ..., m \tag{4}$$

where $\epsilon^{(i)}$ are observation "noises" with independent $\mathcal{N}(0, \sigma_\epsilon^2)$ distributions. Note that in Bayesian inference a prior knowledge about the distribution is required, meaning that we need to specify a mean function $m(.)$ and a kernel $k(.,.)$ in (3). In general, any functions can be used for $m(x)$, but only functions which result in a positive semi-definite covariance matrix $K$ for any inputs can be valid for kernels $k(.,.)$. Usually a zero-mean Gaussian process prior

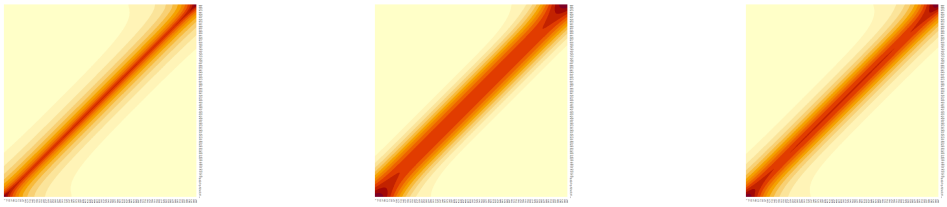
$$f(.) \sim \mathcal{GP}(0, k(.,.)) \tag{5}$$

with a valid covariance function $k(.,.)$ is assumed. Table (1) summarise some commonly used kernels for GPs.

| | |
|---|---|
| Exponential kernel | $k_{\mathrm{E}}(x_i, x_j) = s^2 \exp\left(-\frac{\|x_i - x_j\|}{l}\right)$ |
| Squared exponential kernel | $k_{\mathrm{SE}}(x_i, x_j) = s^2 \exp\left(-\frac{1}{2}\left(\frac{\|x_i - x_j\|}{l}\right)^2\right)$ |
| Matérn class of kernels | $k_{\mathrm{Mat}}(x_i, x_j) = \frac{2^{1-\nu}}{\Gamma(\nu)}\left(\frac{\sqrt{2\nu}\|x_i - x_j\|}{l}\right)^{\nu} K_\nu\left(\frac{\sqrt{2\nu}\|x_i - x_j\|}{l}\right)$ |

Table 1: Examples of kernel functions. $\Gamma(.)$ is Gamma function and $K_\nu$ is the modified Bessel function of the second kind. $s$, $l$ and $\nu$ are parameters.

All the kernels listed above are examples of stationary and monotony kernels. These kernels are functions of the distance $\|x_i - x_j\|$ instead of directly $x$, and hence will result in stationary covariance matrices. Furthermore, they are monotonically decreasing with respect to the distances, which means larger covariances are obtained for closer pairs of points while smaller covariances are achieved for pairs far apart. Figure (1) show examples of stationary and monotony kernels and figure (2) is the plot of covariances against distances.



(a) Exponential kernel with length parameter $l = 1$.

(b) Square exponential kernel with length parameter $l = 1$.

(c) 3/2 Matérn kernel with length parameter $l = 1$

Figure 1: Examples of stationary kernels

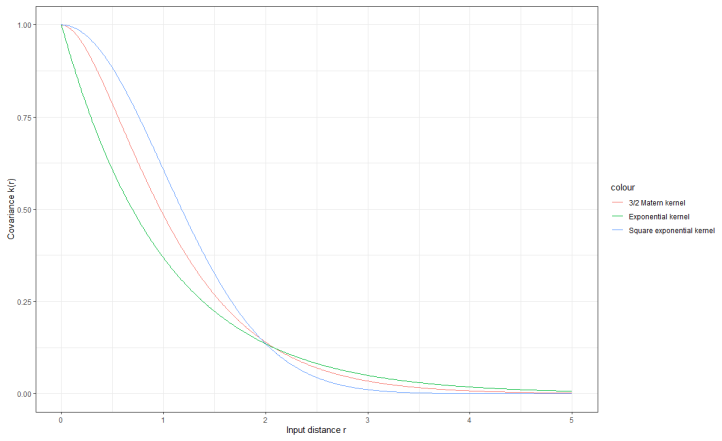



Figure 2: Covariances at different distances. (All the length parameters are set to $l = 1$)

Note that the exponential kernels are continuous but not differentiable with respect to the input $x$, so they are usually used to deal with rough objective functions. On the contrary, since square exponential kernels are infinitely differentiable, smooth function models are obtained. For Matérn family of kernels, one can control the smoothness of functions by tuning the parameter $\nu$. A small value of $\nu$ will lead to sharp jumps in functions while large $\nu$ will give smooth results. It turns out that the Matérn kernel becomes square exponential kernel as $\nu$ goes to infinity, and is a exponential kernel when $\nu = 1/2$. Figure (3) illustrates some random functions drawn from a zero-mean GP with different kernels.

Such stationary kernels are restrict to encode input-independent function dynamics, such as

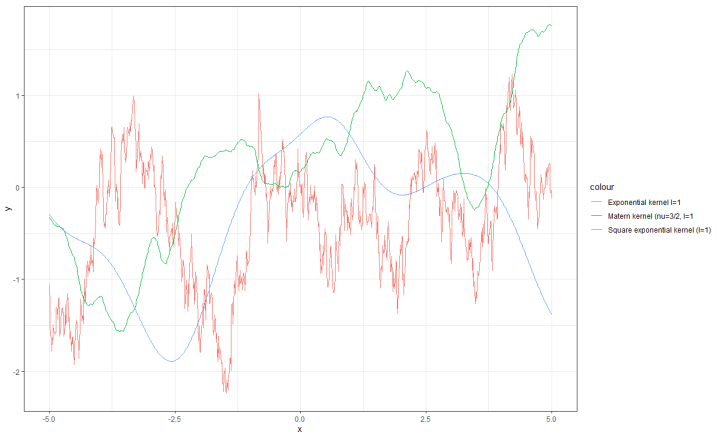



Figure 3: Random functions drawn from a zero-mean GP with different kernels

in credit risk evaluation [5] and chemical energy surface modelling [13]. In more complex cases where the covariances between points are not only depend on the distances but also spatial locations and time, non-stationary kernels can be applied to better capture the information. See [10] for example.

Since our focus is on computer experimental design, we assume that a proper kernel is available for model construction. In particular, we will use Matérn kernel with $\nu = 3/2$ and $l = 1$, namely

$$k_{\text{Mat}}^{3/2}(x_i, x_j) = \left(1 + \sqrt{3}\|x_i - x_j\|\right) \exp\left(-\sqrt{3}\|x_i - x_j\|\right) \tag{6}$$

We interpret a zero-mean Gaussian Process with 3/2 Matérn kernel as a prior, and use the notation $K(.,.)$ for $k_{\text{Mat}}^{3/2}(x_i, x_j)$ in the following context. That is,

$$f \sim \mathcal{GP}(0, K(.,.)). \tag{7}$$

Now let $T = \{(x_*^{(i)}, y_*^{(i)}\}_{i=1}^m$ be a set of i.i.d. testing points drawn from the same distribution as $\mathbf{S}$. Since any finite number of points from a GP form a multivariate normal distribution under we have that

$$\left[\begin{array}{c} f \\ f_* \end{array}\right] \Bigg| X, X_* \sim \mathcal{N}\left(0, \left[\begin{array}{cc} K(X,X) & K(X,X_*) \\ K(X_*,X) & K(X_*,X_*). \end{array}\right]\right). \tag{8}$$

Recall that we have the assumption for observation noises to be stationary. i.e. $\epsilon \sim \mathcal{N}(0, \sigma_\epsilon^2)$. By the property that the sum of the two independent Gaussian distributions is also a Gaussian, we can get

$$\left[\begin{array}{c} y \\ y_* \end{array}\right] \Bigg| X, X_* = \left[\begin{array}{c} f \\ f_* \end{array}\right] + \left[\begin{array}{c} \epsilon \\ 0 \end{array}\right] \sim \mathcal{N}\left(0, \left[\begin{array}{cc} K(X,X) + \sigma_\epsilon^2 I & K(X,X_*) \\ K(X_*,X) & K(X_*,X_*) \end{array}\right]\right). \tag{9}$$

Now applying the Bayes' theorem, the posterior distribution of $y_*$ conditioned on $y$ can be formulated as

$$y_* | y, X, X_* \sim \mathcal{N}(\mu^*, \Sigma^*)$$

where

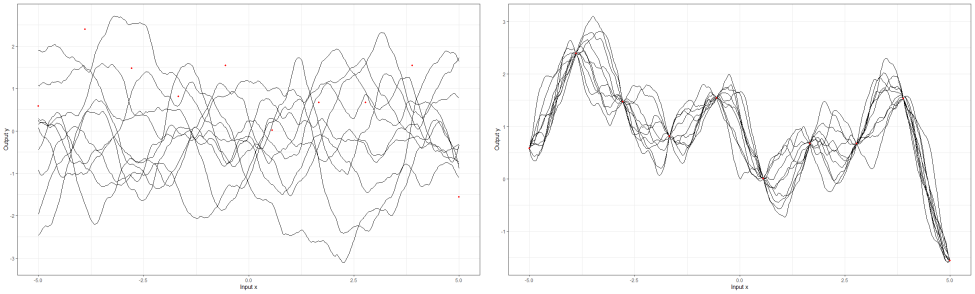
$$\mu^* = K(X_*, X)(K(X,X) + \sigma_\epsilon^2 I)^{-1} \vec{y} \tag{10}$$

and

$$\Sigma^* = K(X_*, X_*) - K(X_*, X)(K(X,X) + \sigma_\epsilon^2 I)^{-1} K(X, X_*). \tag{11}$$

Details of the derivation can be found in appendix. To visualise this procedure, we start with a prior in 7 and update to a posterior distribution by taking observations at 10 equally-spaced

locations with noises $\epsilon \sim \mathcal{N}(0, 0.01)$. Figure (4) shows 10 functions drawn from the prior and posterior distribution respectively. We can see that after 10 observations of the objective function, the uncertainties of the random field are reduced.



(a) 10 functions drawn from a prior distribution. The red dots denotes 10 observations of the objective function.

(b) 10 functions drawn from a posterior distribution. The red dots denotes 10 observations of the objective function.

Figure 4: Example of fitting a GP.

### 2.3 Computer experimental design

Since GPs are trained by samples obtained by taking evaluations of the objective functions which are usually computational intensive, it is crucial to properly decide the sample locations (also referred as designs of experiments). Traditionally, designs are predetermined in advance of the experiments. While little information is available, this kind of strategies will easily cause either lack of accuracy or computational burdens. Sequential designs have been developed to deal with this problem by iteratively choosing the follow-up designs. In general, sequential designs can be divided into space-filling sequential designs and model-based sequential designs. Hybrid sequential strategies are also available which make them to a combination [3]. Space-filling designs are originally one-shot with the aim of making points to best cover the input space, but in recent years various sequential versions using geometric criteria have been proposed, such as augmented space-filling designs [7] and sequential nested Latin hypercubes [2]. While space-filling designs only consider the geometric features, model-based sequential designs leverage available information of the pre-assumed model. For example when GPs are concerned, the prediction mean and variance can be quantified without much effort. Some criteria associated with prediction error then be proposed to choose the follow-up sample points to reduce the model uncertainty. A straightforward criterion is MMSE [8], aiming to take the next observation where the prediction variance is the largest. Another variance based criterion is IMSE [4] which instead considers the integrated variance over the entire input space. Apart from prediction variances, entropy is also a popular mean of quantifying the uncertainty. The so-called Maximum Entropy Sampling have been first applied to spatial designs by [11]. We use the idea to make it a sequential criterion. It can be shown that the ME criterion ends up with the same expression as MMSE, which in fact consolidates MMSE criterion. Besides those popular SE based criteria, one can also base criteria on other proper scores such as Dawid-Sebastiani. Since we use prediction mean as the point estimate, the expectation of DS scores is proportional to the logarithm of the variance, and as a consequence Maximum Mean Dawid-Sebastiani (MMDS) criterion is exactly the same as MMSE. Similar to IMSE, the Integrated Mean Dawid-Sebastiani (IMDS) criterion can be formulated by considering the integral of expected (or mean) DS scores. However, one should note that the model-based sequential designs can be inefficient if the prior knowledge about the model is inauthentic. Particularly in the case of GPs, the model performances using model-based sequential designs depend highly on the choice of kernels.

# 3 Sequential design of experiment under GPs

In this section, we lay out four commonly used sequential sampling criteria in global fitting, MMSE, ME, IMSE and IMDS. We will show that ME criterion is in fact equivalent to MMSE. The derivations and implementation details are presented in the following.

## 3.1 Setup

Consider a function $f : \mathbb{R}^d \to \mathbb{R}$ which is expensive to evaluate. In the following work, we assume the function $f$ is stationary. Recall also the assumption of stationary observation noises, $\epsilon \sim \mathcal{N}(0, \sigma_\epsilon^2)$. We aim to develop a GP model $\tilde{f} : \mathbb{R}^d \to \mathbb{R} \sim \mathcal{N}(m(.), \Sigma(.,.))$ that approximates $f$ based on a set of sequentially well-designed observations. We introduce the notations in table (2).

| | |
|---|---|
| $n$ | the current stage |
| $n+1$ | the next stage |
| $\mathcal{X}$ | the input space |
| $X$ | observed locations until the current stage (i.e.$x_1, x_2, ..., x_n$) |
| $Y$ | observed values at $X$ |
| $S$ | the observed set $\{X, Y\}$ |
| $x_*$ | candidate point of the new design |
| $y_*$ | observation at $x_*$ |
| $f(.)$ | the true objective function |
| $\mu_n(.)$ | the prediction expectation at stage $n$ (conditioned on $y_{\leq n}$) |
| $k_n(.,.)$ | covariance at stage $n$ (conditioned on $y_{\leq n}$) |
| $\Delta_n^f(.)$ | difference between $\mu_n$ and $f$ which has the distribution $\mathcal{N}(0, k_n(.,.))$ |
| $\epsilon_n$ | the observation noise at stage $n$ |
| $\sigma_n^2$ | the prediction variance at stage $n$ |
| $\sigma_\epsilon^2$ | the observation variance |

Table 2: Notation summary

## 3.2 Maximum Mean Squared Error (MMSE)

This criterion is motivated by the aim of reducing the maximum mean squared error (MSE). Recall that the squared Error score (SE) for a prediction $F$ is defined as

$$S_{\text{SE}}(F, y) = (y - \hat{y}_F)^2, \tag{12}$$

where $\hat{y}_F$ is the point estimate under $F$ of the actual value $y$. In the case of GPs, we let the conditional mean be the point estimate, namely $\hat{y}_F = \mu_n$. Therefore, the expectation of SE scores given $y_1, ...y_n$ observed is

$$\begin{aligned}\mathbb{E}[S_{\text{SE}}(\{\mu_n(x), \sigma_n(x)\}, f(x))|y_1, ...y_n] &= \mathbb{E}[(\mu_n(x) - f(x))^2|y_1, ...y_n] \\ &= \sigma_n^2(x).\end{aligned}$$

We would like to see the expected SE scores reduce the most, and therefore the next design is chosen by the criterion:

$$x_{\text{new}} = \arg\max_{x \in \mathcal{X}} \sigma_n^2(x) \tag{13}$$

### 3.3 Maximum Entropy

The Shannon entropy of a random variable $X$ with probability density function $p(.)$ is defined as

$$\text{Ent}(x) = \mathbb{E}_X[-\log(p(X)]. \tag{14}$$

It can be a measure of the unpredictability of a random variable, meaning that the lower the entropy the lower the uncertainty. This motivates us to explore the point which has the maximum entropy, so that the total uncertainty can be reduced.

Let $Y$ be the observed values and $y_*$ be a candidate point. Since $Y$ and $y_*$ form a joint multivariate Gaussian distribution with mean $\mu$ and covariance matrix $\Sigma$ as in equation (9), by definition we have that

$$\begin{aligned}
\text{Ent}\left(\left[\begin{array}{c} Y \\ y_* \end{array}\right] \middle| X, x_*\right) &= -\mathbb{E}[\log((2\pi)^{-n/2}|\Sigma|^{-1/2}\exp(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)))] \\
&= \frac{n}{2}\log(2\pi) + \frac{1}{2}\log|\Sigma| + \frac{1}{2}\mathbb{E}[x-\mu)^T\Sigma^{-1}(x-\mu)] \\
&= \frac{n}{2}\log(2\pi) + \frac{1}{2}\log|\Sigma| + \frac{1}{2}\text{tr}(\Sigma^{-1}\mathbb{E}[(x-\mu)(x-\mu)^T]) \\
&= \frac{n}{2}\log(2\pi) + \frac{1}{2}\log|\Sigma| + \frac{1}{2}\text{tr}(\Sigma^{-1}\Sigma) \\
&= \frac{n}{2}(\log(2\pi)+1) + \frac{1}{2}\log|\Sigma|.
\end{aligned}$$

Observe that the entropy of the joint distribution conditioned on $X$ and $x_*$ is just $\frac{1}{2}\log|\Sigma|$ plus a constant. Applying the rule of calculating the determinant of block matrices, it follows that

$$\begin{aligned}
\log|\Sigma| &= \log\left(\left|\left[\begin{array}{cc} K(X,X)+\sigma_\epsilon^2 I & K(X,x_*) \\ K(x_*,X) & K(x_*,x_*) \end{array}\right]\right|\right) \\
&= \log\left(|K(X,X)+\sigma_\epsilon^2 I| \cdot |K(x_*,x_*) - K(x_*,X)(K(X,X)+\sigma_\epsilon^2 I)^{-1}K(X,x_*)|\right) \\
&= \log|K(X,X)+\sigma_\epsilon^2 I| + \log|K(x_*,x_*) - K(x_*,X)(K(X,X)+\sigma_\epsilon^2 I)^{-1}K(X,x_*)| \\
&= \log|K(X,X)+\sigma_\epsilon^2 I| + \log(\sigma_n^2(x_*)).
\end{aligned}$$

Since $\log|K(X,X)+\sigma_\epsilon^2 I|$ is deterministic, the problem becomes choosing the next point to reduce the maximum of $\log(\sigma_n^2)$. Hence the ME criterion simplifies to

$$x_{\text{new}} = \arg\max_{x\in\mathcal{X}} \sigma_n^2(x), \tag{15}$$

which is indeed equivalent to MMSE criterion (13).

### 3.4 Integrated Mean Squared Error (IMSE)

MMSE criterion chooses the next point where the prediction variance is the largest, which however dose not guarantee that the total prediction variance will be reduced the most. IMSE achieves it by taking the integrated expectation of the predictive variance over the entire space.

Consider the SE scores of $f$ conditioned on $y_1, ...y_n, y_{n+1}$,

$$\begin{aligned}
(\mu_{n+1}(x) - f(x))^2 &= (\mu_n(x) + \gamma_n(f(x_*) - \mu_n(x_*)) + \gamma_n\epsilon_{n+1} - f(x))^2 \\
&= (-\Delta_n^f(x) + \gamma\Delta_n^f(x_*) + \gamma\epsilon_{n+1})^2 \\
&= \Delta_n^f(x)^2 + \gamma_n^2\Delta_n^f(x_*)^2 + \gamma_n^2\epsilon_{n+1}^2 \\
&\quad - 2\Delta_n^f(x)\gamma_n\Delta_n^f(x_*) - 2\Delta_n^f(x)\gamma_n\epsilon_{n+1} + 2\gamma_n^2\Delta_n^f(x_*)\epsilon_{n+1},
\end{aligned}$$

where we use $\gamma_n$ to denote $k_{x_*X}/(k_{XX}+\sigma_\epsilon^2)$ for brevity. Taking the expectation conditioned on the previous observations $y_1, ..., y_n$, we get

$$\begin{aligned}
\mathbb{E}[(\mu_{n+1}(x) - f(x))^2 | y_1, ...y_n] &= \mathbb{E}[\Delta_n^f(x)^2 + \gamma_n^2 \Delta_n^f(x_*)^2 + \gamma_n^2 \epsilon_{n+1}^2 \\
&\quad - 2\Delta_n^f(x)\gamma_n \Delta_n^f(x_*) - 2\Delta_n^f(x)\gamma_n \epsilon_{n+1} + 2\gamma_n^2 \Delta_n^f(x_*)\epsilon_{n+1} | y_1, ..., y_n] \\
&= \sigma_n^2(x) + \gamma_n^2 \sigma_n(x_*)^2 + \gamma_n^2 \sigma_\epsilon^2 - 2k_n(x, x_*)\gamma_n \\
&= \sigma_n^2(x) + \frac{k_n(x, x_*)^2}{(k_n(x_*, x_*) + \sigma_\epsilon^2)^2} \cdot (k_n(x_*, x_*) + \sigma_\epsilon^2) - 2 \cdot \frac{k_n(x, x_*)^2}{k_n(x_*, x_*) + \sigma_\epsilon^2} \\
&= \sigma_n(x) - \frac{k_n(x, x_*)^2}{k_n(x_*, x_*) + \sigma_\epsilon^2}.
\end{aligned}$$

Then take the integral on both sides over the input space.

$$\int_{\mathcal{X}} \mathbb{E}[(\mu_{n+1}(x) - f(x))^2 | y_1, ..., y_n] dx = \int_{\mathcal{X}} \sigma_n(x) dx - \int_{\mathcal{X}} \frac{k_n(x, x_*)^2}{k_n(x_*, x_*) + \sigma_\epsilon^2} dx.$$

Since $\int_{\mathcal{X}} \sigma_n^2(x) dx$ is deterministic, minimizing the integrated expected squared error is equivalent to maximizing $\int_{\mathcal{X}} \frac{k_n(x, x_*)^2}{k_n(x_*, x_*) + \sigma_\epsilon^2} dx$. Recognising that this quantity is the integral of difference between $\sigma_{n+1}^2$ and $\sigma_n^2$, we can write the criterion as

$$x_{\text{new}} = \arg\max_{x_* \in \mathcal{X}} \left( \int_{\mathcal{X}} (\sigma_n^2(x) - \sigma_{n+1}^2(x)) dx \right). \tag{16}$$

The integral involved in (16) can be approximated in discrete case.

$$\begin{aligned}
\int_{\mathcal{X}} (\sigma_n^2 - \sigma_{n+1}^2) dx &= \int_{\mathcal{X}} \frac{k_n(x, x_*)^2}{k_n(x_*, x_*) + \sigma_\epsilon^2} dx \\
&= \frac{\|\Sigma_{x_*,.}\|^2}{\Sigma_{x_*,x_*} + \sigma_\epsilon^2}.
\end{aligned}$$

Then in the case of GPs, IMSE criterion (16) reduces to

$$x_{\text{new}} = \arg\max_{x_* \in X} \frac{\|\Sigma_{x_*,.}\|^2}{\Sigma_{x_*,x_*} + \sigma_\epsilon^2}. \tag{17}$$

Compared with MMSE which only requires comparisons of prediction variances at each point, IMSE involves the computations of vector norm and divisions which make it more expensive.

### 3.5 Integrated Mean Dawid-Sebastiani (IMDS)

Instead of basing the criteria on SE, one can base it on Dawid-Sebastiani (DS). IMDS criterion is motivated by the aim of reducing the integrated expectation of DS scores over the input space.

The Dawid-Sebastiani (DS) proper score of a predictor $F$ is defined as

$$S_{\text{DS}}(F, y) = \frac{(y - \mu_F)^2}{\sigma_F} + \log(\sigma_F^2), \tag{18}$$

where $y$ is the actual value. Using the definition, the expected DS scores at the next stage $n+1$

conditioned on $y_1, ..., y_n$ can be calculated as

$$\begin{aligned}\mathbb{E}[S_{\mathrm{DS}}[\{\mu_{n+1}(x), \sigma_{n+1}(x)\}, f(x)]|y_1, ...y_n] &= \mathbb{E}\left[\frac{(f(x)-\mu_{n+1}(x))^2}{\sigma^2_{n+1}(x)} + \log(\sigma^2_{n+1}(x))\middle| y_1, ..., y_n\right] \\ &= \frac{\mathbb{E}[(f(x)-\mu_{n+1}(x))^2|y_1, ..., y_n]}{\sigma^2_{n+1}(x)} + \log(\sigma^2_{n+1}(x)) \\ &= \frac{\sigma^2_{n+1}(x)}{\sigma^2_{n+1}(x)} + \log(\sigma^2_{n+1}(x)) \\ &= 1 + \log(\sigma^2_{n+1}(x))\end{aligned}$$

Taking the integral over the input space, we have that

$$\begin{aligned}\int_{\mathcal{X}} \mathbb{E}[S_{\mathrm{DS}}[\{\mu_{n+1}(x), \sigma_{n+1}(x)\}, f(x)]|y_1, ...y_n]dx &= \int_{\mathcal{X}} (1 + \log(\sigma^2_{n+1}(x)))dx \\ &= \int_{\mathcal{X}} \log(\sigma^2_{n+1}(x))dx + \text{constant}\end{aligned}$$

Since DS score is negatively oriented, we have the objective of taking the next observation to minimize the integrated expectation of it, and hence the IMDS criterion can be defined as

$$x_{\mathrm{new}} = \arg\min_{x_* \in \mathcal{X}} \int_{\mathcal{X}} \log(\sigma^2_{n+1}(x))dx. \tag{19}$$

For implementation, we take advantage of the expression for prediction variance and get the numerical approximation

$$x_{\mathrm{new}} = \arg\min_{x_* \in \mathcal{X}} (\log(k_n(x_*, x_*) - \frac{\|k_n(x_*, .)\|^2}{k_n(x_*, x_*) + \sigma^2_\epsilon})). \tag{20}$$

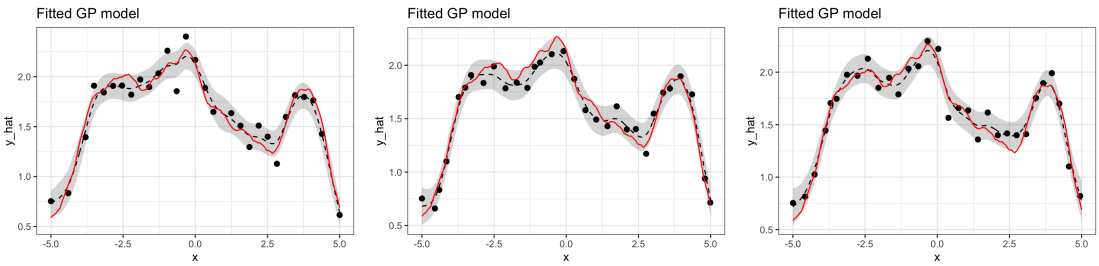
### 3.6 Implementations and illustrations

A function `seq_DoE` has been defined in the function.R file to fit a GP by sequentially taking evaluations. As mentioned, the kernel is the key to a successful model and in the example we assume a suitable kernel is available. This can be done by simulating the values of the objective function by the kernel function we concerned. In the example, we consider the interval $[-5, 5]$, and simulate the objective function values by the following steps.

1. Compute the initial covariance matrix $\Sigma$ by $k^{3/2}_{\mathrm{Mat}}(., .)$.
2. Take the Choleskey decomposition of $\Sigma$ to get $R$ such that $\Sigma = R^T R$.
3. Simulate a random vector $z$ of the same length as $x$ from $\mathcal{N}(0, 1)$.
4. Compute $y = R^T z$.

In such way, $y$ has the property that $Cov(y, y) = \mathbb{E}(R^T z z^T R) = R^T I R = \Sigma$ which is exactly what we want. Figure (5) show the 1d examples of fitted GPs by 30 sequential designs and an initial observation at the left endpoint.

Clearly the three criteria have different favours of locations. We are interested in how they make their decisions to reduce variances. Figure (6) is the plot of the prediction variance after each update. MMSE chooses the next design which has the largest variance. Starting with one initial observation, the variance at the neighbours of the observed point is lower than that at the points far apart from it. In the 1d example, if there are two observations the maximum variance will either be attained at the edges or at the middle of the two observed locations, depending on the distances. If the maximum distance between one observed point and one of




(a) MMSE (b) IMSE (c) IMDS

Figure 5: Fitted GPs with 30 samples sequentially selected by three different criteria. The red solid line represents the objective function while the dash line represents the prediction mean. The observation noises are from $\mathcal{N}(0, 0.01)$.

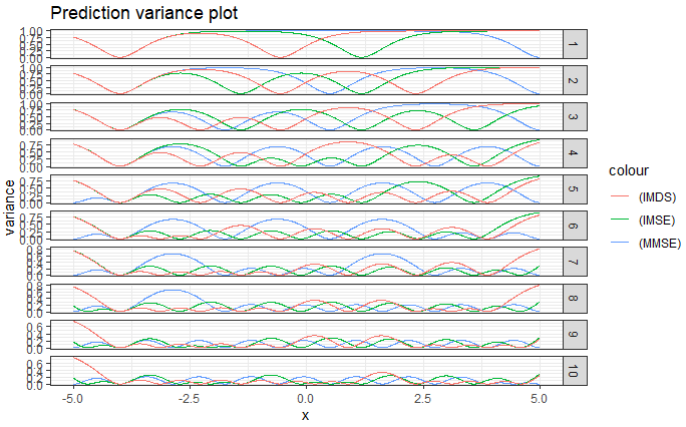



Figure 6: Variance evolution over iterations. Integers in grey boxes indicate the iteration state. The initial designs are fixed to $x = -4$ for all three methods. The variance of observation noise is 0.01.

the edges is greater than two times the distance of the two observed points themselves, that endpoint will be chosen for next observation. Otherwise it chooses the midpoint of the two observed points. We can see that in 1d case with assumptions specified in previous sections, the MMSE criterion shares the idea of covering the input space with space-filling strategy. It tends to spread points out, dragging down the "peaks" one by one until all the "peaks" are in the same level, and repeating. Due to this nature, the endpoints are more likely to be chosen at an early stage. For the above example, the right endpoint is chosen just at the first iteration. On the contrary, IMSE aims to reduce the overall variance, namely "the area under the curve". Quite differently, dragging the edges down will never be the optimal choice for the purpose of reducing "the area under the curve". In our example, we see that the endpoints have never be chosen in the first 10 iterations. Even worse, in each "cycle of reduction", points near the edges are the last to be considered. In this example where local variances are mostly symmetric about observed points, both MMSE and IMSE tend to choose points around the middle of a gap. However, that is not the case for IMDS due to the logarithm involved which de-emphasizes large variances. Intuitively, the D-S based score IMDS sometimes allows local variance to be a bit larger if it can instead reduce variance in larger regions.

As mentioned, IMSE seems to dislike the endpoints of the input interval, which motivates us to investigate the case where we force the two endpoints to be observed at first. Figure (7) shows the prediction variances with initial design to be endpoints. For the first few iterations, MMSE and IMSE make identical decisions. Although later on they choose different locations, the expectations of reduced variances over the entire space are roughly the same. Table (3)

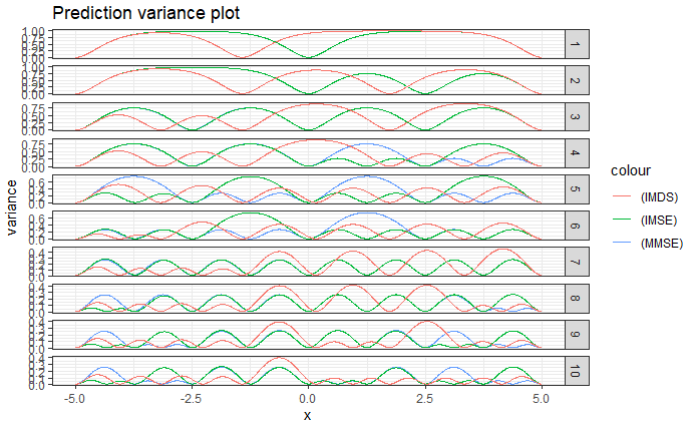



Figure 7: Variance evolution over iterations. Integers in grey boxes indicate the iteration state. The initial designs are fixed to two endpoints. The variance of observation noise is 0.01.

summarises the total prediction variance after 5, 10, 20 and 30 iterations, where we can see that the two SE based criteria indeed result in the same total variance at two significant figures.

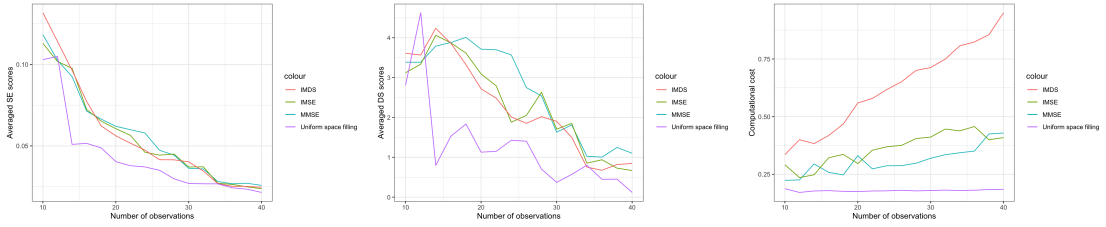
| Max iterations | MMSE | IMSE | IMDS |
|---|---|---|---|
| 5 | 2.9281655 | 2.9280354 | 2.5279668 |
| 10 | 1.0115613 | 1.0058992 | 0.8893801 |
| 20 | 0.2569073 | 0.2540502 | 0.2480816 |
| 30 | 0.1225132 | 0.1219731 | 0.1297792 |

Table 3: Total variance after different iterations.

### 3.7 Model assessment

In this part of work, we fit GPs using different sampling methods respectively, and calculate the averaged SE and DS scores for unobserved data to quantify how well the models make predictions at new locations. Apart from the three sequential sampling methods, uniform space-filling method is also tested for comparisons.

Figure(8) shows the plots of averaged SE, averaged DS and computational costs of four models. Looking at the scores plots of sequential methods, we can see that in early stages




(a) Averaged SE scores (b) Averaged SE scores (c) Computational costs

Figure 8: Scores and computational costs of different models

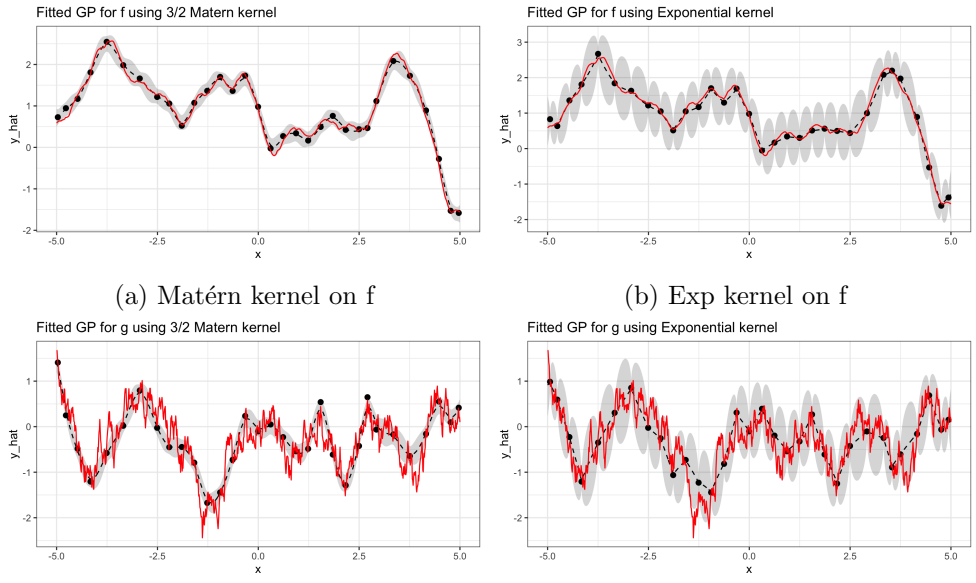
IMDS give the highest averaged SE scores, but that decreases at a high rate as the number of observations increases. After around 18 observations, IMDS has generally the lowest averaged SE scores and MMSE has the largest among sequential methods. As the design size is further increased, all will have similar results. DS score mostly agrees with SE. As expected, IMDS have the lowest averaged DS score in general. It is unclear which sequential methods will have

lower scores when large sample size available due to the fluctuation of the plot. Figure (8c) shows the computational costs of each method. Clearly, one-shot design costs the least. For sequential methods, IMDS is much expensive than the other two SE based sequential methods, and IMSE basically costs more than MMSE.

We should note that the uniform space filling method gives both lower SE and DS scores than sequential methods and also costs less. One reason may be that the sequential approaches choose the next observation as it were the last chance to explore, and try to optimise the next choice given the previous designs fixed. One-shot uniform strategy knows the total budgets in advance so that it can arrange all future points properly. However, in practice one can hardly know the design size and therefore the sequential strategies can be applied to overcome the problem with little loss of accuracy. There are various batch sequential strategies proposed in recent years, trying to look not only one step further but n-step ahead to improve the model performance and efficiency. See ([15]) for example. Besides, our 1-d test case is rather a toy example that even a simple sampling methods can lead to a well fitted model. In more complex situations such as when input space has higher dimensions or observation noises are significant and non-stationary, uniform space filling method is usually abandoned. In latter case, sequential methods may suggest to produce replications at some observed locations just to reduce the uncertainty, whereas this cannot be detected by the uniform sampling method. ([1]) provide a in-depth discussion on the merits of sequential methods in the context of Gaussian process regression with replication.

### 3.8 Tests on different kernels

Note that all the previous results are based on the assumption that we have the covariance function properly chosen, which is nearly unrealistic. To see how the covariance function matters and how much it can influence the outcomes, we construct another function $g$ following the same procedure as $f$ but instead use the exponential kernel to test our models.




(a) Matérn kernel on f

(b) Exp kernel on f

(c) Matérn kernel on g

(d) Exp kernel on g

Figure 9: Predictions of different combinations of models. Criterion IMSE is used for example.

Remember that $f$ is constructed by Matérn kernel and $g$ is constructed by exponential kernel. We apply our models with these two kernels respectively on $f$ and $g$. Table (4) summarises the averaged SE and DS scores over all the unobserved locations for each combination. For function $f$, the model using Matérn kernel leads to lower scores, whereas the exponential kernel

gives mostly lower scores for $g$. We see that a kernel that matches the target function will make the surrogate model more reliable. Hence in practice we should make use of the prior knowledge as much as possible to choose the kernel. This becomes the main challenge, choosing a proper covariance function. In practice, we often have a guess about the graphical features of the target function. If the function tends to be rough, for example some signal functions in electronic engineering one can choose exponential kernel or Matérn kernel with small $\nu$. For smooth functions, the square exponential kernel may be applied.

| Function | Kernel | Criterion | Average SE | Average DS |
|---|---|---|---|---|
| f(Mat) | Matérn | MMSE | 0.011918353 | -3.620203 |
| f(Mat) | Exp | MMSE | 0.024771335 | -2.070489 |
| f(Mat) | Matérn | IMSE | 0.009873302 | -3.584582 |
| f(Mat) | Exp | IMSE | 0.012079000 | -2.129863 |
| f(Mat) | Matérn | IMDS | 0.012038961 | -3.476370 |
| f(Mat) | Exp | IMDS | 0.016398915 | -2.105377 |
| g(Exp) | Matérn | MMSE | 0.155038038 | 8.256491 |
| g(Exp) | Exp | MMSE | 0.130495044 | -1.117635 |
| g(Exp) | Matérn | IMSE | 0.111687697 | 4.629791 |
| g(Exp) | Exp | IMSE | 0.127607804 | -1.185927 |
| g(Exp) | Matérn | IMDS | 0.117111791 | 5.638083 |
| g(Exp) | Exp | IMDS | 0.117111791 | -1.162802 |

Table 4: Average scores display.

# 4 Conclusion

This project studied the model-based sequential designs of computer experiments. We derived four design criteria, MMSE, ME, IMSE and IMDS, in the context of Gaussian Processes modeling. 1-d simulation tests on different sequential criteria and uniform sampling strategy were conducted and compared. It turns out that uniform sampling is the most efficient given the sample size known. However, sequential strategies can benefit experiments in a flexible way of "sampling as you go", and may therefore be favoured in practice. Among the criteria, IMSE and IMDS seem to have higher accuracy at the out-of-sample locations with higher costs than MMSE especially when the sample size is small. All the criteria give similar model performances as the sample size goes large, but IMDS costs much more than the others and hence is not preferred. While it may not be worth the computational cost for the sequential design, the DS score itself does provide value in being much better at detecting model differences; e.g. see the last lines of table (4), where the average SE is the same for two two kernel choices, but DS clearly reveals the difference between the two kernels, showing that the one matching the true data is a better match. There are also limitations of our work, one of which is the assumption that kernels are available for modeling. We have shown that using improper kernels could lead to unreliable models.

# Acknowledgement

I would like to express my deep gratitude to my supervisor Prof. Finn Lindgren, for his generous help and valuable suggestions throughout the project. With his guidance, I have the opportunity to learn about interesting and advanced topics, which motivates me to explore more about statistics in the future. This project could not have been possible without the support from

College of Science and Engineering and School of Mathematics. Thanks also to University of Edinburgh for providing library resources and high power computers.

# A Appendix

## A.1 Derivation of posterior distribution

Suppose $X_A$ and $X_B$ form a joint multivariate Gaussian distribution.

$$\begin{bmatrix} X_A \\ X_B \end{bmatrix} \sim \mathcal{N}\left( \begin{bmatrix} \mu_A \\ \mu_B \end{bmatrix}, \begin{bmatrix} \Sigma_{AA} & \Sigma_{AB} \\ \Sigma_{BA} & \Sigma_{BB} \end{bmatrix} \right).$$

By Bayes' theorem, the posterior distribution of $X_A$ given $X_B$ can be evaluated as

$$\begin{aligned} P(X_A|X_B) &= \frac{P(X_A, X_B)}{\int_{X_A} P(X_A, X_B; \mu, \Sigma) dx_A} \propto P(X_A, X_B) \\ &\propto \exp\left\{ -\frac{1}{2} \left( \begin{bmatrix} X_A \\ X_B \end{bmatrix} - \begin{bmatrix} \mu_A \\ \mu_B \end{bmatrix} \right)^T \begin{bmatrix} V_{AA} & V_{VB} \\ V_{BA} & V_{BB} \end{bmatrix} \left( \begin{bmatrix} X_A \\ X_B \end{bmatrix} - \begin{bmatrix} \mu_A \\ \mu_B \end{bmatrix} \right) \right\} \\ &\propto \exp\left( -\frac{1}{2} (x_A^T V_{AA} X_A - 2X_A^T V_{AA} \mu_A + 2X_A^T V_{AB}(X_B - \mu_B)) \right) \\ &= \exp\left( -\frac{1}{2} \left( X_A - (\mu_A - V_{AA} V_{AB}(X_B - \mu_B) \right)^T \cdot V_{AA} \cdot \left( X_A - (\mu_A - V_{AA}^{-1} V_{AB}(X_B - \mu_B)) \right) \right) \end{aligned}$$

, where we use $V = \begin{bmatrix} V_{AA} & V_{AB} \\ V_{BA} & V_{BB} \end{bmatrix}$ to denote $\Sigma^{-1} = \begin{bmatrix} \Sigma_{AA} & \Sigma_{AB} \\ \Sigma_{BA} & \Sigma_{BB} \end{bmatrix}^{-1}$. Recognising the form of Gaussian distribution, we have

$$X_A|X_B \sim \mathcal{N}(\mu_A - V_{AA}^{-1} V_{AB}(X_B - \mu_B), V_{AA}^{-1}).$$

By calculating the inverse of block matrix, it follows that

$$X_A|X_B \sim \mathcal{N}(\mu_*, \Sigma_*),$$

where $\mu_* = \mu_A + \Sigma_{AB}\Sigma_{BB}^{-1}(X_B - \mu_B)$ and $\Sigma_* = \Sigma_{AA} - \Sigma_{AB}\Sigma_{BB}^{-1}\Sigma_{BA}$.

## A.2 Functions code

```
1
2  ##' The objective function f constructed by 3/2 Matern
3  ##'
4  ##' @param x
5  ##'
6  ##' @return f(x)
7  f <- function(x){
8    l <- length(x)
9    Sigma <- Matern_k(x,x)
10   R <- chol(Sigma)
11   return( t(R)%*%c(rnorm(n=l,mean=0,sd = 1)) )
12 }
13
14
15 ##' The objective function g constructed by Exp
16 ##'
17 ##' @param x
18 ##'
19 ##' @return g(x)
```

```r
g <- function(x){
  l <- length(x)
  Sigma <- Exp_k(x,x)
  R <- chol(Sigma)
  return( t(R)%*%c(rnorm(n=l,mean=0,sd = 1))  )
}


##' 3/2 Matern kernel
##'
##' @param X1 numeric vector
##' @param X2 numeric vector
##' @param l length parameter
##'
##' @return the covariance matrix of X1 and X2
Matern_k <- function(X1,X2,l=1){
  Sigma <- matrix(rep(0, length(X1)*length(X2)), nrow=length(X1))
  for (i in 1:nrow(Sigma)) {
    for (j in 1:ncol(Sigma)) {
      r <- abs(X1[i]-X2[j])
      Sigma[i,j] <- (1+sqrt(3)*r/l)*exp(-sqrt(3)*r/l)
    }
  }
  return (Sigma)
}


##' Exponential kernel
##' @param X1 numeric vector
##' @param X2 numeric vector
##' @param l length parameter
##'
##' @return the covariance matrix of X1 and X2
Exp_k <- function(X1,X2,l=1){
  Sigma <- matrix(rep(0, length(X1)*length(X2)), nrow=length(X1))
  for (i in 1:nrow(Sigma)) {
    for (j in 1:ncol(Sigma)) {
      r <- abs(X1[i]-X2[j])
      Sigma[i,j] <- exp(-r/l)
    }
  }
  return (Sigma)
}

##' Squared Exponential kernel
##' @param X1 numeric vector
##' @param X2 numeric vector
##' @param l length parameter
##'
##' @return the covariance matrix of X1 and X2
SE_k <- function(X1,X2,l=1){
  Sigma <- matrix(rep(0, length(X1)*length(X2)), nrow=length(X1))
  for (i in 1:nrow(Sigma)) {
    for (j in 1:ncol(Sigma)) {
      r <- abs(X1[i]-X2[j])
      Sigma[i,j] <- exp(-1/2*(r/l)^2)
    }
  }
  return (Sigma)
}

##' Criterion 1--MMSE
##' Choosing the next point where the variance is the largest
```

```r
##'
##' @param Sigma the covariance matrix
##' @param sigma2_noise variance of observation noise
##'
##' @return the index of the next point to be chosen
criterion_MMSE <- function(Sigma,sigma2_noise=0){
  return (which.max(diag(Sigma)))
}

##' Criterion 2--IMSE
##' Choosing the next point at which the overall
##'  variance can be reduced the most
##'
##' @param Sigma the covariance matrix
##' @param sigma2_noise variance of observation noise
##'
##' @return the index of the next point to be chosen
criterion_IMSE <- function(Sigma,sigma2_noise=0){
  int_var <- rowSums(Sigma^2)/(diag(Sigma) + sigma2_noise)
  return (which.max(int_var))
}

##' Criterion 3--Dawid-Sebastiani (IMDS = Integrated mean D-S)
##' Choosing the next point at which the overall
##'  expected Dawid-Sebastiani score is minimised
##'
##' @param Sigma the covariance matrix
##' @param sigma2_noise variance of observation noise
##'
##' @return the index of the next point to be chosen
criterion_IMDS <- function(Sigma,sigma2_noise=0){
  # Try to handle the zero noise case:
  sigma2_noise <- max(1e-6, sigma2_noise)
  Var <- diag(Sigma)
  ok <- is.finite(Var)
  ok[ok] <- Var[ok] > 0
  n <- sum(ok)
  V <- matrix(Var[ok], n, n)
  int_ds <- colSums(log(V - Sigma[ok, ok]^2 / (t(V) + sigma2_noise)))
  return (which(ok)[which.min(int_ds)])
}


#' Compute an observation update
#'
#' @param mu Previous expectation vector
#' @param Sigma Previous covariance matrix
#' @param sigma2_noise variance of observation noise
#' @param y_new Vector of the new observation(s)
#' @param obs logical vector or index vector (into mu and Sigma) for
#'   the new observation(s)
update_estimate <- function(mu, Sigma, sigma2_noise,
                            y, obs) {
  if (sigma2_noise == 0) {
    # Only involve elements with nonzero variance
    obs_prev <- diag(Sigma) == 0
    if (any(obs_prev[obs])) {
      warning("Element already observed, and noise variance is zero.
              Expect numerical problems.")
    }
    obs_all <- obs_prev
    obs_all[obs] <- TRUE
    # No observation noise
```

```
    Sigma_obs <- Sigma[obs, obs, drop=FALSE]
    # Update unobserved covariances and expectations:
    Sigma_new <- Sigma
    Sigma_new[!obs_all,!obs_all] <- Sigma[!obs_all,!obs_all] -
      Sigma[!obs_all, obs, drop=FALSE] %*%
      solve(Sigma_obs, Sigma[obs, !obs_all,drop=FALSE])
    Sigma_new[obs, ] <- 0
    Sigma_new[, obs] <- 0
    # Symmetrise:
    Sigma_new <- (Sigma_new + t(Sigma_new)) / 2

    mu_new <- mu
    mu_new[obs] <- y
    mu_new[!obs_all] <- mu[!obs_all] +
      Sigma[!obs_all, obs, drop=FALSE] %*%
      solve(Sigma_obs, y - mu[obs])
  } else {
    # Find out how many new observations there are:
    if (is.logical(obs)) {
      n <- sum(obs)
    } else {
      n <- length(obs)
    }
    # Add observation noise to the observation submatrix:
    Sigma_obs <- (Sigma[obs, obs, drop=FALSE] +
                    diag(x = sigma2_noise, nrow = n, ncol = n))

    # Update unobserved covariances and expectations:
    Sigma_new <- Sigma -
      Sigma[, obs, drop=FALSE] %*%
      solve(Sigma_obs, Sigma[obs, ,drop=FALSE])
    # Symmetrise:
    Sigma_new <- (Sigma_new + t(Sigma_new)) / 2

    mu_new <- mu +
      Sigma[, obs,drop=FALSE] %*%
      solve(Sigma_obs, y - mu[obs])
  }
  list(mu = mu_new, Sigma = Sigma_new)
}



##' Fit a GP model with sequentially chosen points
##'
##' @param x a vector
##' @param y a vector of observation values
##' @param initial_design vector indices, indicating the initial designs
##' @param criterion criterion function MMSE, IMSE or IMDS
##' @param kernel function to calculate initial covariance matrix
##' @param sigma2_noise scalar, variance of observation noises
##' @param max_iterations the number of total iterations
##'
##' @return a dataframe of 5 veriables,
##' including x, y, prediction expectation, prediction variance,
##' and set of all the observations.
seq_DoE <- function(x,y,initial_design, criterion,
                    kernel=Matern_k, sigma2_noise=0, max_iterations=30){

  y_hat <- rep(NA, length(y))

  # Initial mean
```

```r
  mu <- rep(0,length(y))
  mu_new <-mu

  # Initial covariance:
  Sigma <- kernel(x,x)
  Sigma_new <- Sigma

  # Initial observation
  obs_new <- rep(FALSE, nrow(Sigma))
  x_idx_new <- initial_design
  obs_new[x_idx_new] <- TRUE

  new_values <- update_estimate(mu, Sigma, sigma2_noise = sigma2_noise,
                                y = y[x_idx_new], obs = x_idx_new)
  mu <- new_values$mu
  Sigma <- new_values$Sigma
  obs <- obs_new

  # Update current "best estimate"
  y_hat <- mu

  for (i in 1:max_iterations){
    # Choose the next point
    x_idx_new <- criterion(Sigma, sigma2_noise = sigma2_noise)

    # Now condition on the new point, at index x_idx_new
    obs_new <- rep(FALSE, nrow(Sigma))
    obs_new[x_idx_new] <- TRUE

    new_values <- update_estimate(mu, Sigma, sigma2_noise = sigma2_noise,
                                  y = y[x_idx_new], obs = x_idx_new)
    mu <- new_values$mu
    Sigma <- new_values$Sigma

    # Update current "best estimate"
    y_hat <- mu

    # Update:
    obs <- obs | obs_new

  }
  data.frame(x=x, y=y, y_hat=y_hat, var=pmax(0,diag(Sigma)), obs=obs)

}

##' Plot prediction
##' Plot prediction mean and confidence interval, together with
##' observations and true function values.
##'
##' @param df dataframe of at least variables x, y, y_hat, var and obs
##'    usually the output of seq_DoE
##' @param fx vector of true function values (without observation noises)
##'
##' @return ggplot of prediction mean and confidence interval
plt_pred <- function(df,fx){
  ggplot(df)+
    geom_ribbon(aes(x = x, ymin = y_hat-1.97*sqrt(var), ymax = y_hat+1.97*sqrt(
    var)),
                alpha = 0.2) +
    geom_line(aes(x = x, y = y_hat), linetype = "dashed") +
    geom_point(data = data.frame(x=df$x[df$obs],y=df$y[df$obs]),
               aes(x = x, y = y),size = 2) +
    theme_bw() +
```

```
    geom_line(aes(x=x,y=fx), colour= "red")+
    ggtitle("Fitted GP model")
}



##' Calculate DS scores for unobserved x
##'
##' @param df data frame containing at least y_hat, y and var
##' @param obs logic
##'
##' @return vector of DS scores
Cal_DS <- function(df,obs=NULL){
  if (is.null(obs)){
    obs <- df$obs
  }
  y_pred <- df$y_hat
  y_true <- df$f
  var <- df$var
  score<-numeric(nrow(df))
  score[obs] <- NA
  score[!obs] <- (y_true[!obs]-y_pred[!obs])^2/var[!obs]+log(var[!obs])
  score
}

##' Calculate SE scores for unobserved x
##'
##' @param df data frame containing at least y_hat and y
##' @param obs logic, specifying the indices of observations when
##' the parameter df doesn't have information about observed locations
##' @return vector of SE scores
Cal_SE <- function(df,obs=NULL){
  if (is.null(obs)){
    obs <- df$obs
  }
  y_pred <- df$y_hat
  y_true <- df$f
  score<-numeric(nrow(df))
  score[!obs] <- (y_true[!obs]-y_pred[!obs])^2
  score[obs] <- NA
  score
}

#' Test statistics (for permutation tests)
#'
#' Compute all the test statistics for each
#' function-criterion combination for a given dataset
#'
#' @param data the dataset to be considered
#'
#' @return a dataframe, indicating the test statistics for different
    combinations
#' in the given dataset
t_stat <- function(data){
  data %>%
    summarise(SE = abs(mean(SE[kernel=="Matern"],na.rm=TRUE)-
                         mean(SE[kernel=="Exp"],na.rm=TRUE)),
              DS = abs(mean(DS[kernel=="Matern"],na.rm=TRUE)-
                         mean(DS[kernel=="Exp"],na.rm=TRUE)),
              .groups = "drop")
}
```

## A.3 Analysis code

```r
# Load function definitions
source("functions.R")

# Set the random seed to a fixed value to allow reproducible results.
set.seed(12345L)

########################################
#####initializations####################
########################################
x <- seq(-5,5,len=1000)
dx <- (x[length(x)]-x[1])/length(x)
fx <- f(x)
y <- fx+rnorm(length(x),sd=0.1)
y_hat <- rep(NA, length(x))

# Initial mean
mu <- rep(0,length(x))
mu_new <-mu

# Initial covariance:
Sigma <- Matern_k(x,x)
Sigma_new <- Sigma

save(x,dx,fx,y,y_hat,mu,mu_new,Sigma,Sigma_new, file="initializations.RData")

########################################
##Scores and computational costs########
########################################
initial_design <- c(1,length(x))
iterations <- seq(10,40,len=16)


data <- matrix(NA,length(iterations),13)
data[,1] <- iterations
for (i in  1:length(iterations)){

  obs_sf <- rep(FALSE, length(x))
  obs_sf[seq(1,length(x),len=iterations[i])] <- TRUE
  t0 <- Sys.time()
  c0 <- update_estimate(mu= rep(0,length(y)), Sigma=Matern_k(x,x), sigma2_noise
    = 0.01,
                        y = y[obs_sf], obs = obs_sf)
  data[i,11] <- Sys.time()-t0
  data[i,12] <- mean(Cal_SE(c0,obs=obs_sf))
  data[i,13] <- mean(Cal_DS(c0,obs=obs_sf))


  t0 <- Sys.time()
  c1 <- seq_DoE(x=x,initial_design=initial_design, y=y,  sigma2_noise = 0.01,
                criterion=criterion_MMSE, max_iterations=iterations[i]-2)
  data[i,8] <- Sys.time()-t0
  data[i,2] <- mean(Cal_SE(c1))
  data[i,5] <- mean(Cal_DS(c1))

  t0 <- Sys.time()
  c2 <- seq_DoE(x=x,initial_design=initial_design, y=y, sigma2_noise = 0.01,
                criterion=criterion_IMSE, max_iterations=iterations[i]-2)
  data[i,9] <- Sys.time()-t0
  data[i,3] <- mean(Cal_SE(c2))
  data[i,6] <- mean(Cal_DS(c2))

  t0 <- Sys.time()
  c3 <- seq_DoE(x=x,initial_design=initial_design,y=y, sigma2_noise = 0.01,
```

```
                criterion=criterion_IMDS, max_iterations=iterations[i]-2)
  data[i,10] <- Sys.time()-t0
  data[i,4] <- mean(Cal_SE(c3))
  data[i,7] <- mean(Cal_DS(c3))
}

data <- data.frame(max_iterations=data[,1],
                   SE_MMSE=data[,2],
                   SE_IMSE=data[,3],
                   SE_IMDS=data[,4],
                   SE_USF=data[,12],
                   DS_MMSE=data[,5],
                   DS_IMSE=data[,6],
                   DS_IMDS=data[,7],
                   DS_USF=data[,13],
                   time_cost_MMSE=data[,8],
                   time_cost_IMSE=data[,9],
                   time_cost_IMDS=data[,10],
                   time_cost_USF=data[,11])

saveRDS(data, file = "data/table1.rds")

########################################
##overall variance (variance integral)##
########################################

initial_design <- c(1,length(x))
iterations <- c(5,10,20,30)

data <- matrix(NA,4,4)
data[,1] <- iterations
for (i in  1:length(iterations)){
  c1 <- seq_DoE(x=x,initial_design=initial_design, y=y,
                criterion=criterion_MMSE,sigma2_noise=0.1^2,
                max_iterations=iterations[i])
  data[i,2] <- sum(c1$var)

  c2 <- seq_DoE(x=x,initial_design=initial_design, y=y,
                criterion=criterion_IMSE, sigma2_noise=0.1^2,
                max_iterations=iterations[i])
  data[i,3] <- sum(c2$var)

  c3 <- seq_DoE(x=x,initial_design=initial_design, y=y,
                criterion=criterion_IMDS, sigma2_noise=0.1^2,
                max_iterations=iterations[i])
  data[i,4] <- sum(c3$var)
}

data <- data.frame(max_iterations=data[,1],
                   MMSE=dx*data[,2],
                   IMSE=dx*data[,3],
                   IMDS=dx*data[,4])

saveRDS(data, file = "data/table2.rds")



################################################
#variance evolution (with initial design x=-4)#
################################################
initial_design <- c(100)
var_dat_MMSE <- data.frame(x=x)
var_dat_IMSE <- data.frame(x=x)
```

```r
var_dat_IMDS <- data.frame(x=x)
for (i in 1:10){
  df1 <- seq_DoE(x,y,initial_design,criterion_MMSE,sigma2_noise=0.1^2,
                 max_iterations = i)
  var_dat_MMSE <- cbind(var_dat_MMSE, df1$var)

  df2 <- seq_DoE(x,y,initial_design,criterion_IMSE,sigma2_noise=0.1^2,
                 max_iterations = i)
  var_dat_IMSE <- cbind(var_dat_IMSE, df2$var)

  df3 <- seq_DoE(x,y,initial_design,criterion_IMDS,sigma2_noise=0.1^2,
                 max_iterations = i)
  var_dat_IMDS <- cbind(var_dat_IMDS, df3$var)
}

colnames(var_dat_MMSE) <- c("x",1:10)
var_dat_MMSE <- pivot_longer(var_dat_MMSE, cols = 2:11,
                             names_to="iterations",values_to='var')
colnames(var_dat_IMSE) <- c("x",1:10)
var_dat_IMSE <- pivot_longer(var_dat_IMSE, cols = 2:11 ,
                             names_to="iterations",values_to='var')
colnames(var_dat_IMDS) <- c("x",1:10)
var_dat_IMDS <- pivot_longer(var_dat_IMDS, cols = 2:11,
                             names_to="iterations",values_to='var')

saveRDS(var_dat_MMSE, file = "data/var_dat_MMSE.rds")
saveRDS(var_dat_IMSE, file = "data/var_dat_IMSE.rds")
saveRDS(var_dat_IMDS, file = "data/var_dat_IMDS.rds")

################################################
##variance evolution#############################
#(with initial designs being the two endpoints)#
################################################
initial_design <- c(1,length(x))
var_dat_MMSE1 <- data.frame(x=x)
var_dat_IMSE1 <- data.frame(x=x)
var_dat_IMDS1 <- data.frame(x=x)
for (i in 1:10){
  df1 <- seq_DoE(x,y,initial_design,criterion_MMSE,sigma2_noise=0.1^2,
                 max_iterations = i)
  var_dat_MMSE1 <- cbind(var_dat_MMSE1, df1$var)

  df2 <-seq_DoE(x,y,initial_design,criterion_IMSE,sigma2_noise=0.1^2,
                max_iterations = i)
  var_dat_IMSE1 <- cbind(var_dat_IMSE1, df2$var)

  df3 <- seq_DoE(x,y,initial_design,criterion_IMDS,sigma2_noise=0.1^2,
                 max_iterations = i)
  var_dat_IMDS1 <- cbind(var_dat_IMDS1, df3$var)

}
colnames(var_dat_MMSE1) <- c("x",1:10)
var_dat_MMSE1 <- pivot_longer(var_dat_MMSE1, cols = 2:11,
                              names_to="iterations",values_to='var')
colnames(var_dat_IMSE1) <- c("x",1:10)
var_dat_IMSE1 <- pivot_longer(var_dat_IMSE1, cols = 2:11 ,
                              names_to="iterations",values_to='var')
colnames(var_dat_IMDS1) <- c("x",1:10)
var_dat_IMDS1 <- pivot_longer(var_dat_IMDS1, cols = 2:11,
                              names_to="iterations",values_to='var')

saveRDS(var_dat_MMSE1, file = "data/var_dat_MMSE1.rds")
saveRDS(var_dat_IMSE1, file = "data/var_dat_IMSE1.rds")
```

```r
saveRDS(var_dat_IMDS1, file = "data/var_dat_IMDS1.rds")

################################
##Tests on different kernels#####
################################
f0 <- f(x)
g0 <- g(x)
y1 <- f0+rnorm(length(x),sd=0.1)
y2 <- g0+rnorm(length(x),sd=0.1)

initial_design <- c(250,750)
f_km_MMSE <- cbind(seq_DoE(x,y1,initial_design, criterion_MMSE,
                           Matern_k,sigma2_noise = 0.01),f=f0) # Mat Mat MMSE
f_ke_MMSE <- cbind(seq_DoE(x,y1,initial_design, criterion_MMSE,
                           Exp_k,sigma2_noise = 0.01),f=f0) # Mat Exp MMSE

f_km_IMSE <- cbind(seq_DoE(x,y1,initial_design, criterion_IMSE,
                           Matern_k,sigma2_noise = 0.01),f=f0) # Mat Mat IMSE
f_ke_IMSE <- cbind(seq_DoE(x,y1,initial_design, criterion_IMSE,
                           Exp_k,sigma2_noise = 0.01),f=f0) # Mat Exp IMSE

f_km_IMDS <- cbind(seq_DoE(x,y1,initial_design, criterion_IMDS,
                           Matern_k,sigma2_noise = 0.01),f=f0) # Mat Mat IMDS
f_ke_IMDS <- cbind(seq_DoE(x,y1,initial_design, criterion_IMDS,
                           Exp_k,sigma2_noise = 0.01),f=f0) # Mat Exp IMDS

g_km_MMSE <- cbind(seq_DoE(x,y2,initial_design, criterion_MMSE,
                           Matern_k,sigma2_noise = 0.01),f=g0) #Exp Mat MMSE
g_ke_MMSE <- cbind(seq_DoE(x,y2,initial_design, criterion_MMSE,
                           Exp_k,sigma2_noise = 0.01),f=g0) # Exp Exp MMSE

g_km_IMSE <- cbind(seq_DoE(x,y2,initial_design, criterion_IMSE,
                           Matern_k,sigma2_noise = 0.01),f=g0) #Exp Mat IMSE
g_ke_IMSE <- cbind(seq_DoE(x,y2,initial_design, criterion_IMSE,
                           Exp_k,sigma2_noise = 0.01),f=g0) # Exp Exp IMSE

g_km_IMDS <- cbind(seq_DoE(x,y2,initial_design, criterion_IMDS,
                           Matern_k,sigma2_noise = 0.01),f=g0) #Exp Mat IMDS
g_ke_IMDS <- cbind(seq_DoE(x,y2,initial_design, criterion_IMDS,
                           Exp_k,sigma2_noise = 0.01),f=g0) # Exp Exp IMDS
table4 <- data.frame(
  MSE=c(mean(Cal_SE(f_km_MMSE),na.rm=TRUE),
        mean(Cal_SE(f_ke_MMSE),na.rm=TRUE),
        mean(Cal_SE(f_km_IMSE),na.rm=TRUE),
        mean(Cal_SE(f_ke_IMSE),na.rm=TRUE),
        mean(Cal_SE(f_km_IMDS),na.rm=TRUE),
        mean(Cal_SE(f_ke_IMDS),na.rm=TRUE),
        mean(Cal_SE(g_km_MMSE),na.rm=TRUE),
        mean(Cal_SE(g_ke_MMSE),na.rm=TRUE),
        mean(Cal_SE(g_km_IMSE),na.rm=TRUE),
        mean(Cal_SE(g_ke_IMSE),na.rm=TRUE),
        mean(Cal_SE(g_km_IMDS),na.rm=TRUE),
        mean(Cal_SE(g_km_IMDS),na.rm=TRUE)),
  MDS=c(mean(Cal_DS(f_km_MMSE),na.rm=TRUE),
        mean(Cal_DS(f_ke_MMSE),na.rm=TRUE),
        mean(Cal_DS(f_km_IMSE),na.rm=TRUE),
        mean(Cal_DS(f_ke_IMSE),na.rm=TRUE),
        mean(Cal_DS(f_km_IMDS),na.rm=TRUE),
        mean(Cal_DS(f_ke_IMDS),na.rm=TRUE),
        mean(Cal_DS(g_km_MMSE),na.rm=TRUE),
        mean(Cal_DS(g_ke_MMSE),na.rm=TRUE),
        mean(Cal_DS(g_km_IMSE),na.rm=TRUE),
        mean(Cal_DS(g_ke_IMSE),na.rm=TRUE),
```

```
        mean(Cal_DS(g_km_IMDS),na.rm=TRUE),
        mean(Cal_DS(g_ke_IMDS),na.rm=TRUE))
)

table4 <-
  arrange(expand.grid(
    func=c("f(Mat)","g(Exp)"),kernel=c("Matern","Exp"),
    Criterion=c("MMSE","IMSE","IMDS")),func,Criterion,kernel) %>%
  cbind(table4)

#####################################################################
##############Monte Carlo permutation tests#####################
#test on H_0: Exp kernel and 3/2 Matern kernel are exchangeable#
#####################################################################
fg_scores_all <- fg_scores_all %>%
  group_by(func,Criterion)
h1_t <- t_stat(fg_scores_all)
# Monte Carlo permutation tests
N = 10000
h0_t_se <- matrix(NA,N,6) # Used to store Tj for SE
h0_t_ds <- matrix(NA,N,6) # Used to store Tj for DS
for (i in 1:N){
  ts <- fg_scores_all %>%
    mutate(kernel=sample(kernel,length(kernel),replace=FALSE)) %>%
    t_stat
  h0_t_se[i,] <- ts$SE
  h0_t_ds[i,] <- ts$DS
}

saveRDS(h0_t_se, file = "data/h0_t_se.rds")
saveRDS(h0_t_ds, file = "data/h0_t_ds.rds")
saveRDS(h1_t, file = "data/h1_t.rds")

p_se <- numeric(6)
p_ds <- numeric(6)
MCsd_se <- numeric(6)
MCsd_ds <- numeric(6)
for (i in 1:6){

  p_se[i] <- mean(h0_t_se[,i]>=h1_t$SE[i])
  p_ds[i] <- mean(h0_t_ds[,i]>=h1_t$DS[i])

  MCsd_se[i] <- sqrt(p_se[i]*(1-p_se[i])/N)
  MCsd_ds[i] <- sqrt(p_ds[i]*(1-p_ds[i])/N)
}

h1_t %>%
  mutate(SE=NULL,DS=NULL,p_SE=p_se,p_DS=p_ds,
         sd_SE=MCsd_se,sd_DS=MCsd_ds)
```

# References


[1] Mickaël Binois et al. 'Replication or Exploration? Sequential Design for Stochastic Simulation Experiments'. In: *Technometrics* 61.1 (2019), pp. 7–23. DOI: 10.1080/00401706.2018.1469433. eprint: https://doi.org/10.1080/00401706.2018.1469433. URL: https://doi.org/10.1080/00401706.2018.1469433.

[2] K. Crombecq, E. Laermans and T. Dhaene. 'Efficient space-filling and non-collapsing sequential design strategies for simulation-based modeling'. In: *European Journal of Operational Research* 214.3 (2011), pp. 683–696. ISSN: 0377-2217. DOI: https://doi.org/10.1016/j.ejor.2011.05.032. URL: https://www.sciencedirect.com/science/article/pii/S0377221711004577.

[3] Karel Crombecq et al. 'A Novel Hybrid Sequential Design Strategy for Global Surrogate Modeling of Computer Experiments'. In: *SIAM Journal on Scientific Computing* 33.4 (2011), pp. 1948–1974. DOI: 10.1137/090761811. eprint: https://doi.org/10.1137/090761811. URL: https://doi.org/10.1137/090761811.

[4] Robert G. Easterling. '[Design and Analysis of Computer Experiments]: Comment'. In: *Statistical Science* 4.4 (1989), pp. 425–427. DOI: 10.1214/ss/1177012415. URL: https://doi.org/10.1214/ss/1177012415.

[5] Hao Jiang et al. 'Stationary Mahalanobis kernel SVM for credit risk evaluation'. In: *Applied Soft Computing* 71 (2018), pp. 407–417. ISSN: 1568-4946. DOI: https://doi.org/10.1016/j.asoc.2018.07.005. URL: https://www.sciencedirect.com/science/article/pii/S1568494618303892.

[6] Hai Lan et al. 'Electrical Energy Forecasting and Optimal Allocation of ESS in a Hybrid Wind-Diesel Power System'. In: *Applied Sciences* 7.2 (2017). ISSN: 2076-3417. DOI: 10.3390/app7020155. URL: https://www.mdpi.com/2076-3417/7/2/155.

[7] Lu Lu and Christine M. Anderson-Cook. 'Strategies for sequential design of experiments and augmentation'. In: *Quality and Reliability Engineering International* 37.5 (2021), pp. 1740–1757. DOI: https://doi.org/10.1002/qre.2823. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/qre.2823. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/qre.2823.

[8] David J. C. MacKay. 'Information-Based Objective Functions for Active Data Selection'. In: *Neural Computation* 4.4 (1992), pp. 590–604. DOI: 10.1162/neco.1992.4.4.590.

[9] Aroonsri Nuchitprasittichai. 'Simulation-optimization of carbon dioxide capture process and the impact of uncertainty on the solution'. English. Copyright - Database copyright ProQuest LLC; ProQuest does not claim copyright in the individual underlying works; Last updated - 2021-05-20. PhD thesis. 2013, p. 221. ISBN: 978-1-303-07991-7. URL: https://www.proquest.com/dissertations-theses/simulation-optimization-carbon-dioxide-capture/docview/1367156357/se-2.

[10] Antti Piironen, Juho Piironen and Toni Laaksonen. 'Predicting spatio-temporal distributions of migratory populations using Gaussian process modelling'. In: *Journal of Applied Ecology* 59.4 (2022), pp. 1146–1156. DOI: https://doi.org/10.1111/1365-2664.14127. eprint: https://besjournals.onlinelibrary.wiley.com/doi/pdf/10.1111/1365-2664.14127. URL: https://besjournals.onlinelibrary.wiley.com/doi/abs/10.1111/1365-2664.14127.

[11] M. C. Shewry and H. P. Wynn. 'Maximum entropy sampling'. In: *Journal of Applied Statistics* 14.2 (1987), pp. 165–170. DOI: 10.1080/02664768700000020. eprint: https://doi.org/10.1080/02664768700000020. URL: https://doi.org/10.1080/02664768700000020.

[12] Rens van de Schoot et al. 'Bayesian statistics and modelling'. English. In: *Nature Reviews Methods Primers* 1 (Jan. 2021). ISSN: 2662-8449. DOI: 10.1038/s43586-020-00003-0.

[13] Avery E. Wiens, Andreas V. Copan and Henry F. Schaefer. 'Multi-fidelity Gaussian process modeling for chemical energy surfaces'. In: *Chemical Physics Letters* 737 (2019). Articles initially published in Chemical Physics Letters: X 1-4, 2019, p. 100022. ISSN: 0009-2614. DOI: https://doi.org/10.1016/j.cpletx.2019.100022. URL: https://www.sciencedirect.com/science/article/pii/S2590141919300315.

[14] Yuan Yuan et al. 'Point process models for spatio-temporal distance sampling data from a large-scale survey of blue whales'. English. In: *Annals of Applied Statistics* 11.4 (Dec. 2017), pp. 2270–2297. ISSN: 1932-6157. DOI: 10.1214/17-AOAS1078.

[15] Boya Zhang et al. 'Batch-sequential design and heteroskedastic surrogate modeling for delta smelt conservation'. In: *The Annals of Applied Statistics* 16.2 (2022), pp. 816–842. DOI: 10.1214/21-AOAS1521. URL: https://doi.org/10.1214/21-AOAS1521.