Eur. Phys. J. Spec. Top. (2021) 230:2253–2263
https://doi.org/10.1140/epjs/s11734-021-00204-y

THE EUROPEAN PHYSICAL JOURNAL SPECIAL TOPICS

Regular Article

Statistical Analysis of Complex Computer Models in Astronomy

Joshua Lukemire[1], Qian Xiao[2], Abhyuday Mandal[2,a], and Weng Kee Wong[3]

[1] Department of Biostatistics and Bioinformatics, Emory University, Atlanta, GA, USA
[2] Department of Statistics, University of Georgia, Athens, GA, USA
[3] Department of Biostatistics, University of California, Los Angeles, CA, USA

Received 12 February 2021 / Accepted 22 June 2021 / Published online 9 August 2021

Abstract We introduce statistical techniques required to handle complex computer models with potential applications to astronomy. Computer experiments play a critical role in almost all fields of scientific research and engineering. These computer experiments, or simulators, are often computationally expensive, leading to the use of emulators for rapidly approximating the outcome of the experiment. Gaussian process models, also known as Kriging, are the most common choice of emulator. While emulators offer significant improvements in computation over computer simulators, they require a selection of inputs along with the corresponding outputs of the computer experiment to function well. Thus, it is important to select inputs judiciously for the full computer simulation to construct an accurate emulator. Space-filling designs are efficient when the general response surface of the outcome is unknown, and thus they are a popular choice when selecting simulator inputs for building an emulator. In this tutorial we discuss how to construct these space filling designs, perform the subsequent fitting of the Gaussian process surrogates, and briefly indicate their potential applications to astronomy research.

1 Introduction

Computer experiments, or simulators, are an increasingly important tool in many scientific fields. In these experiments, a computer model is defined relating a set of inputs to an output. Instead of conducting a traditional experiment, a researcher will provide a set of inputs to the computer model and obtain the model output. This approach is very appealing in fields such as physics, where the computer experiment model can be setup using a series of known relationships/equations and different inputs may consist of unknown constants in those equations or other properties such as mass or chemical compositions. These experiments can be effective alternatives to experiments which may be too expensive or otherwise impossible to perform in a traditional setting. They differ from standard experiments in several key ways. Most importantly, computer experiments are generally deterministic; for a set of input settings the experiment will return the same result every time it is conducted. Second, the experiments will generally not have an easily described response surface; for example a standard linear regression model will not generally describe the outcome accurately.

Many research areas in astronomy do not easily permit conducting traditional experiments. For example, researchers may be interested in the formation of binary black holes. Clearly the researchers will not be able to create multiple black holes and observe their dynamics over time. Computer experiments make it possible to study such phenomena by creating computer models based on the theorized properties of these binary systems and then comparing the output to what is observed in Nature (Fig. 1). For example, Compact Object Mergers: Population Astrophysics and Statistics (COMPAS) is used to investigate binary population synthesis. The computer experiment takes input as initial conditions and simulates the lifespan of stars [50] [56]. Similarly, binary population synthesis code ComBinE has been used to perform binary population syntheses [32], and the tool UniverseMachine [4] allows researchers to study galaxy formation.

Computer experiments for many complex systems can be very expensive to perform (see, for example, [62]). This computational expense can be a significant problem, especially if a researcher hopes to conduct the experiment for many sets of inputs. An alternative to directly performing these computer experiments is to instead create a surrogate or emulator [18]. Surrogates are popular for computer experiments when it is not realistic to evaluate a fine grid over the entire input space. Instead, a (relatively) small number of points are chosen to evaluate under the original computer simulation. Then, a model is fit to the output from these limited runs. Predictions under this model for new inputs, as well as uncertainty quantification,

[a] e-mail: amandal@stat.uga.edu (corresponding author)

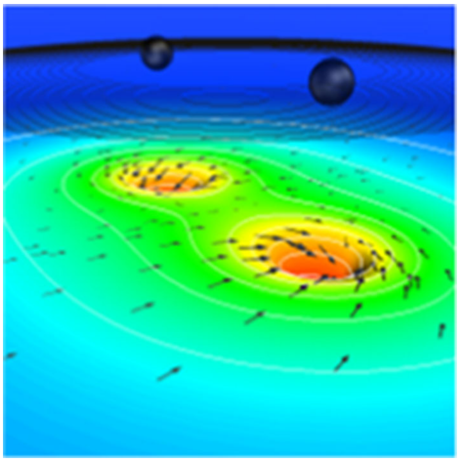

Fig. 1 An example from a simulation examining whether two black holes merge. Source: https://www.black-holes.org/code/SpEC.html

can be obtained from the surrogate without the need to re-run the expensive computer simulation at the new points. If the model fits well, then the predicted values will be close to the true values that would have been obtained if the full computer experiment was used.

The most common tool used to fit the data points and create the surrogate model is the Gaussian process (GP) [48] [18]. The GP is appealing for creating surrogates because it interpolates known data to evaluate new data points. This is especially important when the outcome for a fixed set of inputs is deterministic, which is frequently the case in computer experiments. This approach is becoming more popular in the astronomy literature. Some recent work includes [22], who proposed using GP emulation to obtain confidence intervals for the parameter vector of a phase-space distribution function for dwarf spheroidal galaxies.

Section 2 of this tutorial paper introduces GP models and discusses their applications to computer experiments. We provide codes and examples throughout in the R programming language [45]. Section 3 of this paper focuses on determining what inputs to use to generate the responses used to fit the GP model to obtain an accurate surrogate. We draw upon the design of experiments statistical literature to discuss design of computer experiments. In particular we focus on Latin hypercube designs and discuss several techniques for finding them.

2 Surrogates for Computer Models

Simpler surrogates or *emulators* are often preferred for complex deterministic computer models. GP models are a popular choice for this purpose [48]. Consider an n-run computer experiment with d-dimensional input vectors $\mathbf{x}_i = (x_{i1}, \ldots, x_{id})^{\mathrm{T}}$ and deterministic outputs $y(\mathbf{x}_i)$, for $i = 1, 2, \ldots, n$. To fix ideas, assume that we are interested in a 2-dimensional input for a computer experiment with output given by the Branin function as defined by [6], see also [14].

$$y(x_1, x_2) = \left(x_2 - \frac{5.1}{4\pi^2} x_1^2 + \frac{5}{\pi} x_1 - 6 \right)^2 + 10 \left(1 - \frac{1}{8\pi} \right) \cos(x_1) + 10, \qquad (1)$$

where the design space is given by values of $x_1 \in [-5, 10]$ and $x_2 \in [0, 15]$. The R code below can be used to evaluate this function.

```
# x is a vector of inputs (length 2)
branin <- function(x){
  a <- 1
  b <- 5.1  / (4 * pi^2)
  c <- 5 / pi
  r <- 6
  s <- 10
  t <- 1 / (8 * pi)
  return( a*(x[2] - b*x[1]^2 + c*x[1] -r)^2 + s*(1-t)*cos(x[1]) + s )
}
```

The left panel in Figure 2 displays the output for this function over the entire design space.

2.1 Stationary Gaussian Process - Krigging

The simplest possible GP model, known as ordinary GP or *krigging*, is given by

$$y(\mathbf{x}_i) = \mu + Z(\mathbf{x}_i), \qquad (2)$$

where μ is the mean and $Z(\mathbf{x})$ is a GP, denoted by $Z(\mathbf{x}) \sim GP(0, \sigma^2 R)$. This notation implies that the GP has zero-mean, and the covariance function $Cov\,(Z(\mathbf{x}_i), Z(\mathbf{x}_j)) = \sigma^2 R(.|\boldsymbol{\theta})$, where $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_d)^{\mathrm{T}}$ is the vector of unknown correlation parameters with all $\theta_s > 0$ $(s = 1, \ldots, d)$. The correlation between outputs is determined by a stationary correlation function R with parameter $\boldsymbol{\theta}$. Two of the more commonly-used correlation functions are the power-exponential and the Gaussian functions. Under a power-exponential correlation structure the $(i, j)^{\text{th}}$ term is defined as:

$$R(\mathbf{x}_i, \mathbf{x}_j | \boldsymbol{\theta}) = \prod_{s=1}^{d} \exp\left\{ -\theta_s \mid x_{is} - x_{js} \mid^{p_s} \right\} \quad \text{for all } i, j, \qquad (3)$$

where smoothness parameters $p_1, \ldots, p_s$ are all between 0 and 2. Of special importance is $p_s = 2$, for all $s = 1, \ldots, d$, which corresponds to the popular Gaussian correlation function:

$$R(\mathbf{x}_i, \mathbf{x}_j | \boldsymbol{\theta}) = \exp\left\{ -\sum_{s=1}^{d} \theta_s (x_{is} - x_{js})^2 \right\} \quad \text{for all } i, j. \qquad (4)$$

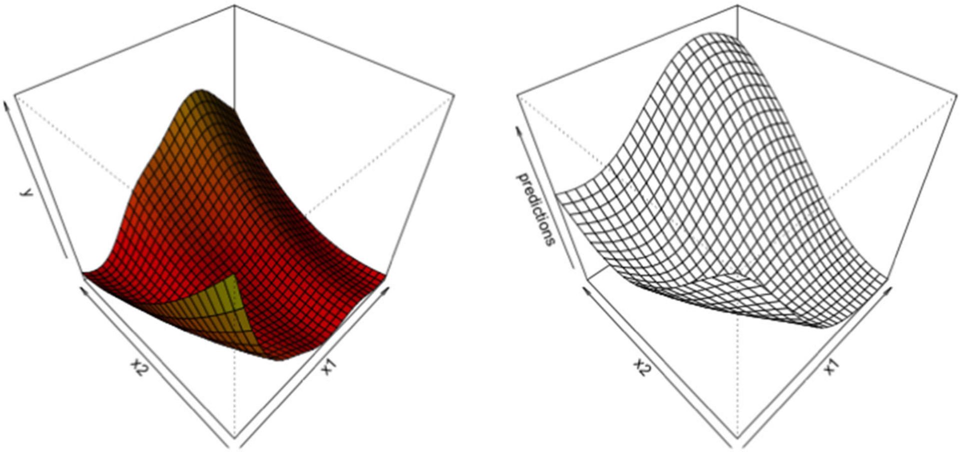

Fig. 2 Left: the true response under the Branin function. Right: The estimated response using the surrogate model

The flexibility of the correlation structure is what makes the GP model a popular surrogate for complex computer models. For any given input $\mathbf{x}^*$ in the design space, the fitted GP surrogate gives the predicted computer model response as,

$$\hat{y}(\mathbf{x}^*) = \mu + \mathbf{r}^{\mathrm{T}}(\mathbf{x}^*)\mathbf{R}^{-1}(\mathbf{y} - \mu\mathbf{1}_n), \tag{5}$$

where

$$\mathbf{r}(\mathbf{x}^*) = \Big[\mathrm{corr}\Big(Z(\mathbf{x}^*), Z(\mathbf{x}_1)\Big), \mathrm{corr}\Big(Z(\mathbf{x}^*), Z(\mathbf{x}_2)\Big), \ldots, \mathrm{corr}\Big(Z(\mathbf{x}^*), Z(\mathbf{x}_n)\Big)\Big]^{\mathrm{T}}, \tag{6}$$

$\mathbf{1}_n$ is a vector of ones of length n, $\mathbf{R}$ is the $n \times n$ correlation matrix for $(Z(\mathbf{x}_1), ..., Z(\mathbf{x}_n))^{\mathrm{T}}$, $\mathbf{y}$ is the response vector $(y(\mathbf{x}_1), \ldots, y(\mathbf{x}_n))^{\mathrm{T}}$, and the associated uncertainty estimate is

$$s^2(\mathbf{x}^*) = \sigma^2\Big(1 - \mathbf{r}(\mathbf{x}^*)^{\mathrm{T}}\mathbf{R}^{-1}\mathbf{r}(\mathbf{x}^*)\Big). \tag{7}$$

In practice, the parameters μ, σ^2 and $\boldsymbol{\theta}$ in Equations (5) and (7) are unknown and need to be estimated from the data. The parameters can be estimated using the `mlegp` function in R. Assume that we already have a design with 10 points (details for obtaining this design will be presented in Section 3). Then we can fit a GP with a Gaussian correlation function as,

```
library(mlegp)
# Obtaining this design is discussed in Section 3
design <- matrix(c(-4.25, -1.25, 4.75, 7.75, -2.75, 1.75,
    6.25, 3.25, 0.25, 9.25, 8.25, 6.75, 11.25, 12.75, 0.75,
    2.25, 9.75, 5.25, 14.25, 3.75), ncol=2)
# Obtain the output at the set points in our design
Yx <- apply(design, 1, branin)
# Use the observed outputs to construct a surrogate
branin_surrogate_1 <- mlegp(design, Yx)
```

Similarly, estimates across the entire design space can be obtained by using the surrogate model by specifying the inputs on a grid:

```
# Construct a grid of points to obtain predictions at
x1 <- seq(from = -5, to =10, length.out = 25)
x2 <- seq(from = 0, to =15, length.out = 25)
test_points <- expand.grid(x1, x2)
# Get predictions from the gaussian process
yhat <- predict(branin_surrogate_1, test_points)
predictions <- matrix( yhat, nrow = length(x1) )
# plot the predictions, theta and phi control viewing angle
persp(x1, x2, predictions, theta = -45, phi=45)
```

The right panel in Figure 2 displays a plot of the surrogate output. Comparing this output to the true values in the left panel, it is clear that the surrogate model is able to obtain a very close approximation to the true process.
The formulation in equation (2) can be extended to incorporate a global trend function for the mean μ [57]. This is known as *Universal Kriging*:

$$y(\mathbf{x}) = \mu(\mathbf{x}) + Z(\mathbf{x}), \tag{8}$$

with $\mu(\mathbf{x}) = g(\mathbf{x})^{\mathrm{T}}\beta = \sum_{i=1}^{m} \beta_i g_i(\mathbf{x})$, where g is a m-dimensional known function and $\beta = (\beta_1, \ldots, \beta_m)^{\mathrm{T}}$ is a vector of unknown parameters. If we assume $g_1(\mathbf{x}) = 1$ and let $\mathbf{G} = (g_1(\mathbf{x}), \ldots, g_m(\mathbf{x}))^{\mathrm{T}}$, then the optimal predictor under model (8) is given by

$$\hat{y}(\mathbf{x}^*) = g^{\mathrm{T}}(\mathbf{x}^*)\hat{\beta} + \mathbf{r}^{\mathrm{T}}(\mathbf{x}^*)\mathbf{R}^{-1}(\mathbf{y} - \mathbf{G}\hat{\beta}), \tag{9}$$

where $\hat{\beta} = (\mathbf{G}^{\mathrm{T}}\mathbf{R}^{-1}\mathbf{G})^{-1}(\mathbf{G}^{\mathrm{T}}\mathbf{R}^{-1}\mathbf{y})$. If the assumed $\mu(\mathbf{x})$ is close to the truth, this formulation will lead to a better prediction than ordinary krigging. Note that this universal kriging formulation uses $\mu(\mathbf{x})$ to capture the known trends, but in most real applications, these trends are not known, and hence ordinary kriging is commonly used [61].

2.2 Non-stationarity

Note that Equation (3) refers to a stationary GP, that is

$$Cov\Big(Z(\mathbf{x}+\mathbf{h}), Z(\mathbf{x})\Big) = \sigma^2 R(\mathbf{h}), \tag{10}$$

where the correlation function $R(\mathbf{h})$ is a positive semidefinite function with $R(\mathbf{0}) = 1$ and $R(-\mathbf{h}) = R(\mathbf{h})$. These stationary GPs are popular surrogates for complex computer models, since it can be shown that the corresponding predictor of μ in equation (2)

$$\hat{\mu} = (\mathbf{1}_n^{\mathrm{T}}\mathbf{R}^{-1}\mathbf{1}_n)^{-1}\mathbf{1}_n^{\mathrm{T}}\mathbf{R}^{-1}\mathbf{y} \tag{11}$$

is the best linear unbiased predictor (BLUP) in the sense that it minimizes the mean squared prediction error. In reality this assumption of stationarity may not hold. Under these circumstances, the above predictor is no longer optimal. Some literature is available to deal with non-stationary GPs for emulating computationally expensive functions. For example, [67] introduced the idea of nonlinear mapping based on a parameterized density function, and [20] proposed a Bayesian tree structure by dividing the design space into subregions.

[2] used composite Gaussian process (CGP) models to address the nonstationarity problem. In their formulation, the model takes the following form:

$$\begin{aligned} y(\mathbf{x}) &= Z_{\text{global}}\,(\mathbf{x}) + Z_{\text{local}}\,(\mathbf{x}), \\ Z_{\text{global}}\,(\mathbf{x}) &\sim \text{GP}\left(\mu, \tau^2 R_1(\cdot)\right), \\ Z_{\text{local}}\,(\mathbf{x}) &\sim \text{GP}\left(0, \sigma^2 R_2(\cdot)\right). \end{aligned} \tag{12}$$

Here $Z_{\text{global}}\,(\mathbf{x})$ and $Z_{\text{local}}\,(\mathbf{x})$ are two stationary GPs that are independent of each other. Just as the universal kriging generalizes the ordinary kriging by adding a trend function $\mu(\mathbf{x})$, the composite GP model given in equation (12) is a further extension which adds a more flexible global trend component. The model was extended to incorporate the non-constant variance assumption as follows:

$$\begin{aligned} y(\mathbf{x}) &= Z_{\text{global}}\,(\mathbf{x}) + \sigma(\mathbf{x}) Z_{\text{local}}\,(\mathbf{x}), \\ Z_{\text{global}}\,(\mathbf{x}) &\sim \text{GP}\left(\mu, \tau^2 R_1(\cdot)\right), \\ Z_{\text{local}}\,(\mathbf{x}) &\sim \text{GP}\left(0, R_2(\cdot)\right). \end{aligned} \tag{13}$$

The model can be further extended for noisy data by adding a third GP (with zero correlation) to the model (13).

2.3 Numeric Considerations - Local GP

Note that the prediction involves the inversion of the $n \times n$ correlation matrix $\mathbf{R}$, where n is the number of data points (see equation (5) or (11), for example). This is a big hurdle in implementing GPs. To overcome this problem, [19] introduced the idea of local GP approximation for large computer models. They provided a family of local sequential design schemes that dynamically define the support points of a GP predictor based on a local subset of the data. Their approach is different from that of k-nearest neighbours. The basic idea is simple, under the standard choices of the covariance structures the correlation between points is dependent on the distance between those points, with data points far from $\mathbf{x}^*$ having very little effect on its prediction. Hence it is not a good use of computational resources to invert the full covariance matrix, as the elements corresponding to "far away" points will contribute little to the prediction of $y(\mathbf{x}^*)$. An interested reader should refer to [19] for the formulas of the GP predictor based on a local subset of data. The end result is a global predictor that takes advantage of modern multicore parallel computing tools.

2.4 Extension to Qualitative Inputs

The conventional GP models consider quantitative predictor variables only, but many computer experiments may have both quantitative and qualitative inputs. In order to construct an emulator with qualitative factors, a naive approach would be to create distinct GP models for data collected at the different level combinations of the qualitative factors. Clearly this approach has many limitations, particularly when there are several qualitative factors. There are some more advanced techniques to deal with such cases. To fix ideas, for an n-run computer model, denote the k^{th} $(k = 1, \ldots, n)$ data input as $\mathbf{w}_k = (\mathbf{x}_k^{\mathrm{T}}, \mathbf{z}_k^{\mathrm{T}})^{\mathrm{T}}$ where $\mathbf{x}_k = (x_{k1}, \ldots, x_{kp})^{\mathrm{T}} \in \mathbb{R}^p$ is the quantitative part and $\mathbf{z}_k = (z_{k1}, \ldots, z_{kq})^{\mathrm{T}} \in \mathbb{N}^q$ is the qualitative part (coded in levels) of the input. Note here that previously $\mathbf{x}$ denoted the input, which was entirely continuous. However, now $\mathbf{w}$ denotes the entire input, with $\mathbf{x}$ referring to the continuous part. For these kind of problems, a popular GP based model was introduced by [44], among many others [23], [72], [53], [70] and [71]. Specifically, an ordinary GP model with a multiplicative covariance function is considered (for any two inputs $\mathbf{w}_1$ and $\mathbf{w}_2$):

$$\text{Cov}(Z(\mathbf{w}_1), Z(\mathbf{w}_2)) = \sigma^2 \prod_{j=1}^{q} \tau^{(j)}_{z_{1j}z_{2j}} R(\mathbf{x}_1, \mathbf{x}_2|\boldsymbol{\theta}), \tag{14}$$

where the parameter $\tau^{(j)}_{z_{1j}z_{2j}}$ represents the correlation between two levels (z_{1j} and z_{2j}) in the j^{th} qualitative factor $z^{(j)}$, and $R(\mathbf{x}_1, \mathbf{x}_2|\boldsymbol{\theta})$ is given before in equation (4). Different choices of $\tau^{(j)}_{z_{1j}z_{2j}}$ lead to different types of correlation functions. For example, an exchangable correlation function is obtained when $\tau^{(j)}_{z_{1j}z_{2j}}$ is some constant between 0 and 1. Alternatively, an additive GP model was proposed in [11], which adopts the following covariance function:

$$\text{Cov}(Z(\mathbf{w}_1), Z(\mathbf{w}_2)) = \sum_{j=1}^{q} \sigma_j^2 \tau^{(j)}_{z_{1j}z_{2j}} R(\mathbf{x}_1, \mathbf{x}_2|\boldsymbol{\theta}^{(j)}), \tag{15}$$

where σ_j^2 and $\boldsymbol{\theta}^{(j)}$ $(j = 1, \dots, q)$ are the process variance and correlation parameters, respectively, corresponding to $z^{(j)}$.

The methods above do not have good physical interpretation of the correlation structures. Motivated by this, [64] proposed an EzGP method inspired from the ANOVA (Analysis of Variance) idea to jointly model the quantitative and qualitative inputs:

$$Y(\mathbf{w}) = \mu + G_{\mathbf{z}}(\mathbf{x}), \tag{16}$$

which implies that for any level combination of $\mathbf{z}$, $Y(\mathbf{w})$ is a GP. In particular, they considered

$$G_{\mathbf{z}}(\mathbf{x}) = G_0(\mathbf{x}) + G_{z^{(1)}}(\mathbf{x}) + \cdots + G_{z^{(q)}}(\mathbf{x}), \tag{17}$$

where G_0 and $G_{z^{(h)}}$ $(h = 1, \dots q)$ are independent GPs with mean zero and some covariance functions. Here, G_0 is a standard GP taking only quantitative inputs $\mathbf{x}$, which can be viewed as the base GP reflecting the intrinsic relation between y and $\mathbf{x}$. On the other hand, $G_{z^{(h)}}$'s can be viewed as an adjustment to the base GP by the impact of the qualitative factor $z^{(h)}$ $(h = 1, \dots q)$. This EzGP technique enjoys some nice theoretical properties and is able to flexibly address heterogeneity in computer models involving multiple qualitative factors. [64] also developed two variants of the EzGP model to achieve computational efficiency for data with high dimensionality and large sizes.

2.5 Calibration

The notion of calibration and sensitivity analysis is important in the context of physical and computer experiments. Instead of observing the real physical process, y^{Real}, we are only able to observe a process y^{Field} as:

$$y^{Field}(\mathbf{x}) = y^{Real}(\mathbf{x}) + \epsilon, \tag{18}$$

where ϵ is the usual normal error. This y^{Real} is approximated by a computer model y^{Model}. Note that the computer model y^{Model} not only has the input variables $\mathbf{x}$, but also some unknown parameters θ, called calibration parameters which are used to fine tune the model. Note that these calibration parameters can be, for example, the correlation parameters discussed above. The field data y^{Field} is used mainly to learn more about the real phenomenon y^{Real}. [30] proposed a Bayesian framework to address this as follows:

$$\begin{aligned} y^{Real}(\mathbf{x}) &= y^{Model}(\mathbf{x}, \theta) + b(\mathbf{x}) \\ y^{Field}(\mathbf{x}) &= y^{Model}(\mathbf{x}, \theta) + b(\mathbf{x}) + \epsilon, \end{aligned} \tag{19}$$

where $b(\mathbf{x})$ is a functional discrepancy, called bias. [30] used Bayesian methods to estimate the bias correction function and unknown calibration parameter θ under a GP prior. An alternative to this Bayesian approach is an iterative history matching algorithm such as the one proposed by [55] for calibrating a galaxy formation model called GALFORM. This is actually a hands-on process, which intelligently eliminates the implausible points from the input (or parameter) space and returns a set of plausible candidates for the parameters θ. Recently, [5] used this algorithm for calibrating hydrological time-series models and [46] further extended this method with a more systematic approach, in which they discretize the target response series on a few time points, and then iteratively apply the history matching algorithm with respect to the discretized targets.

3 Design of Computer Experiments

The computer experiments under consideration have deterministic outputs, and thus replicates at a given set of input settings should be avoided, as they do not provide any further information about the response. Good designs for computer experiments are then designs that are "space-filling" in some sense, which make it easier to fit accurate surrogate models. We will next discuss a few types of space-filling designs and examine techniques which can be used to construct them.

3.1 LHD: Efficient Experimental Designs

Latin hypercube designs (LHDs) are $n \times d$ matrices whose columns are permutations of numbers 1 to n (or 0 to $n-1$) [39]. They have unique point projections on every dimension and avoid replications, making them ideal for determining which inputs to use for computer experiments [13]. For a given number of runs and input size, an LHD can easily be generated in R:

```r
# Load an R library for finding LHDs
library(LHD)
# Generate a Latin Hypercube design with 10 runs and 2 factors
lhd1 <- rLHD(10, 2)
```

While it is intuitive to favor a design that is space-filling, in practice it is difficult to identify such designs for experiments with different number of runs and factors. One of the more common approaches is to use orthogonal or nearly-orthogonal LHDs (OLHDs). An OLHD minimizes the correlations among the input settings of the design [15] [52]. They can be obtained by minimizing a correlation-based design criteria. For example, two of the most commonly used criteria for OLHDs are the average absolute correlation (ave($|r|$)) and the maximum absolute correlation (max$|r|$):

$$\begin{aligned} \text{ave}(|r|) &= \frac{2\sum_{s=1}^{d-1}\sum_{s'=s+1}^{d}|r_{ss'}|}{d(d-1)}, \\ \max|r| &= \max_{s,s'}|r_{ss'}|, \end{aligned} \tag{20}$$

where $r_{ss'}$ is the correlation between the sth and s'th columns of the design. If the design is a orthogonal LHD, then ave$(|r|)$ = 0 and max$|r|$ = 0. For example, to generate an OLHD in R with 8 factors and 32 runs we can write:

```
# Obtain an orthogonal latin hypercube design
# Need n_factor = r * 2^(C+1)
OLHD <- OLHD.S2010(C = 3, r = 2, type = ‘‘even’’)
```

The design can easily be verified to be orthogonal by examining:

```
# All off diagonal elements are 0
t(OLHD) %*% OLHD
```

However, for many combinations of run size and number of inputs an orthogonal LHD does not exist, and thus a good design will be one with small ave$(|r|)$ and max$|r|$ values. Many algebraic construction methods have been proposed for finding OLHDs, and they can also be found via searching algorithms using ave$(|r|)$ or max$|r|$ as objective functions. Some specific results include [69], who proposed techniques for constructing orthogonal LHDs with run-size $n = 2^m$ or $n = 2^m + 1$ where m is an integer. [3] proposed to rotate the 2^d factorial designs for constructing d-factor orthogonal LHDs where d must be some power of 2 and the run-size is $n = 2^d$. For further examples, please refer to [7], [49], [9], [35], [51] and [68]; see [59] for a survey.

While OLHDs are very commonly used, they are not guaranteed to be space-filling; see design (a) in Figure 3 for an example [63]. In light of this, various design optimality criteria have been developed related to measures of space-filling.

3.1.1 Centered L_2-Discrepancy Criteria

[24] defined several discrepancy–based criteria among which the centered L_2-discrepancy (CD) is the most popular. The intuition behind the CD criteria is that a space-filling design should have points spread out uniformly in the whole design space or any sub-space of the design space. If this is the case, for any rectangular region of the design space we examine, the number of design points in that space should be proportional to the volume of that space. The CD criteria is defined as,

$$CD(D_n)^2 = \sum_{v\neq\varnothing}\int_{C^v}\left|\frac{\#(D_{n_v}, J_{x_v})}{n} - \text{Volume}(J_{x_v})\right|^2 \mathrm{d}x, \tag{21}$$

where D_n is the n-run, d-factor, q-level design, v is some non empty subset of $1, 2, \ldots, q$, C^v is the subspace defined by the coordinate indexes selected by v, D_{n_v} is the projection of D_n onto the subspace C^u, x_v is the projection of vector $x = (x_1, x_2, \ldots, x_q)$ on to the subspace C^v, J_x is the chosen rectangle space defined by x, J_{x_u} is the projection of J_x onto the subspace defined by C^u, $\#(D_{n_u}, J_{x_u})$ is the total number of designs points in D_{n_u} within the chosen area defined by J_{x_u}, and $Volume(J_{x_u})$ is the volume of J_{x_u}. For more details on the rationale of the CD criteria, see the Chapter 3 in [13] for a survey.

3.1.2 Multi-objective Criteria

Another commonly-used metric for evaluating designs' space-filling properties is the maximin distance criterion [27]. This criteria favors designs with maximum pairwise distances between inputs. Maximin designs are popular due to their robustness, since the design criteria focuses on optimizing the worst case scenario – the closest pairwise distance between any two points. [43] defined a computationally efficient scalar value for evaluating the maximin distance criterion:

$$\phi_p = \left(\sum_{i=2}^{n}\sum_{j=1}^{i-1}\frac{1}{u_{i,j}^p}\right)^{\frac{1}{p}}, \tag{22}$$

where $u_{i,j}$ is the distance between the ith and jth design points. Designs with smaller ϕ_p values are more space-filling. For sufficiently large p (e.g. $p > 15$), the ϕ_p criterion is asymptotically identical to the true maximin distance criterion.

Due to the desirability of both the orthogonality and maximin properties, [28] proposed a multi-objective criterion (denoted OMmcri) to generate orthogonal-maximin LHDs (OMm LHDs), which acts as a compromise between orthogonal and maximin designs. The OMmcri criteria is given by,

$$\begin{aligned}&\text{OMmcri}(x,\omega)\\&\quad= \omega\rho^2 + (1-\omega)\frac{(\phi_p - \phi_{p,lowerbound})}{(\phi_{p,upperbound} - \phi_{p,lowerbound})}.\end{aligned} \tag{23}$$

Here, ϕ_p is the maximin criteria value from Equation (22), ρ is the $ave(|r|)$ criteria value as defined in Equation (20), ω is a weight value reflecting the trade-off between the orthogonality and maximin criteria, and $\phi_{p,lowerbound}$ and $\phi_{p,upperbound}$ are given by,

$$\phi_{p,lowerbound} = \left\{\binom{n}{2}\left(\frac{\lceil\overline{u}\rceil - \overline{u}}{\lfloor\overline{u}\rfloor^p} - \frac{\overline{u} - \lfloor\overline{u}\rfloor}{\lceil\overline{u}\rceil^p}\right)\right\}^{\frac{1}{p}}, \text{ and}$$

$$\phi_{p,upperbound} = \left(\sum_{i=1}^{n-1}\frac{n-i}{(id)^p}\right)^{\frac{1}{p}},$$

respectively. Here $\overline{u}$ is the average distance between the design points and $\lfloor\overline{u}\rfloor$ and $\lceil\overline{u}\rceil$ are the largest integer smaller than $\overline{u}$ and the smallest integer larger than $\overline{u}$.

Another popular class of efficient LHDs is the orthogonal array based LHDs (OALHDs) by [54], where the levels in randomized orthogonal arrays (OAs) are expanded to form LHDs. The OALHDs have desirable sampling and projection properties, but they are not

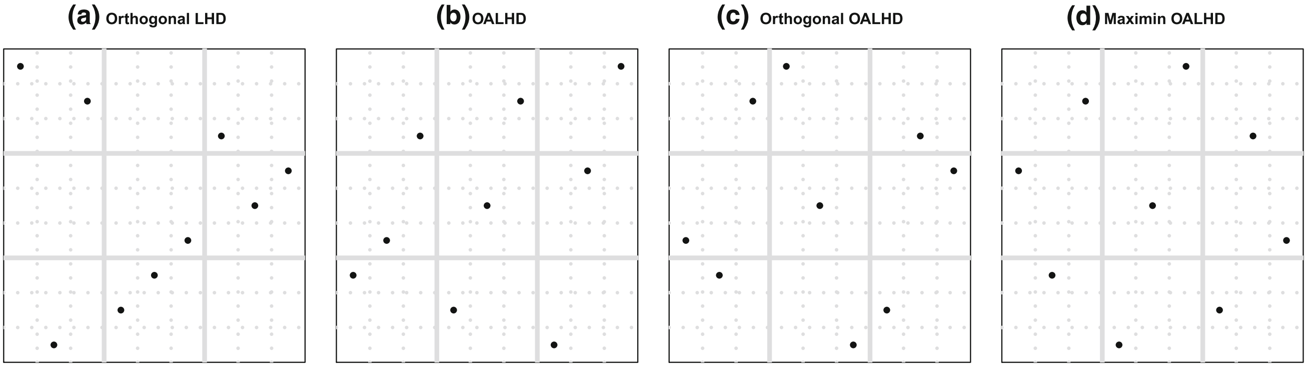

Fig. 3 Some examples of 9-run 2-factor LHDs

necessarily space-filling [63]; see designs (b) and (c) in Figure 3 for some examples. [33] proposed to use a simulated annealing algorithm to search for space-filling OALHDs, and [66] further proposed to consider both level permutation and level expansion for generating OALHDs. See design (d) in Figure 3 for an example. Some algebraic construction methods are also available for constructing maximin LHDs for certain design sizes [65] [60].

3.1.3 Maxpro: Maximum projection designs

Space-filling LHDs, including CD and and maximin distance LHDs, focus on the design's properties in the full dimensional spaces. Yet, their space-filling properties in some sub-spaces (projections) may not be adequate. [29] proposed the maximum projection LHDs (Maxpro LHDs) that guarantee designs have space-filling properties in all projections. The maximum projection criterion is defined as

$$\min_{\mathbf{X}} \psi(\mathbf{X}) = \left\{ \frac{1}{\binom{n}{2}} \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{1}{\prod_{s=1}^{d} (x_{is} - x_{js})^2} \right\}^{1/d}. \tag{24}$$

Here, $\mathbf{X}$ is a $n \times d$ matrix where each row is an input to the computer experiment, and the minimization is over all pairs of rows in $\mathbf{X}$. Clearly a design minimizing ψ will have every pair of design points apart from each other in all projections, justifying the name "Maxpro."

3.2 Searching Algorithms for Generating Efficient LHDs with Flexible Sizes

The metrics discussed above for evaluating designs such as the minimax criteria provide a way of quantifying how "good" a design is in some sense. It remains to determine how to actually construct designs that have a good value of the criterion, which is a challenging problem in many situations. For many such design problems, it is popular to use metaheuristic optimization algorithms to find designs. There are many problems in astronomy that metaheuristic optimization algorithms are applicable to. For example, large scale clustering problems can be approached using metaheuristic algorithms [12] [26]. In recent years, these metaheuristic algorithms have seen widespread use, see, for example, [8], [16], [41], [40], and [42].

These algorithms are preferred due to their flexibility - in general they will work with any objective function. For a more detailed review of metaheuristic algorithms for finding designs, see [37]. Here we will focus on two of the more commonly used approaches: Simulated Annealing (SA) and Genetic Algorithms (GA).

3.2.1 Simulated Annealing Algorithms

SA is one of the most widely used general probabilistic optimization techniques [31]. The algorithm follows the annealing process in metallurgy, in which materials are heated to a high temperature where their properties change, and then are allowed to slowly cool. [43] adapted the classic SA algorithm for finding maximin distance LHDs, and the approach can easily be modified to search for other types of designs by using the other optimality criteria defined in Section 3.1.

SA starts with a random LHD and then improves it via an element exchange method, where two random elements from a random column in the design are exchanged. If this exchange results in a more efficient design, then the change is kept. If the exchange does not result in any improvement, the change is kept with probability controlled by the current temperature (tuning parameter). Allowing changes that do not improve the design helps the search algorithm to escape local optima. The SA algorithm will iteratively repeat this exchange procedure. After a certain number of rounds, the temperature would be annealed to decrease (cool down) the probability of updating the current design following the annealing schedule. We summarize a general SA framework in Algorithm 1, where the target function Φ to be minimized can be the optimality criterion defined in (20), (21), (22), (23) and (24) for the orthogonal, CD, maximin, OMm and Maxpro LHDs, respectively.

In the Algorithm 1, the maximum number of iterations N is recommended to be around 500 according to the convergence analysis in [59]. The decreasing rate for the current temperature T is another important tuning parameter. A larger rate will make T decline faster,

Algorithm 1 Simulated Annealing

```
1: Choose values for the tuning parameters: the starting temperature $T$, the number of attempts before lowering the temperature $S$, and the maximum number of iterations $N$.
2: Set the counter index $C = 1$.
3: Construct a random starting LHD $\mathbf{X}$.
4: Select a column from $\mathbf{X}$ at random.
5: Exchange two randomly selected elements within this chosen column. Denote the new design by $\mathbf{X}_{new}$.
6: If $\Phi(\mathbf{X}_{new}) < \Phi(\mathbf{X})$, then $\mathbf{X} = \mathbf{X}_{new}$ (accept the new design). Otherwise, let $\mathbf{X} = \mathbf{X}_{new}$ with probability $\exp\left\{-\frac{\Phi(\mathbf{X}_{new})-\Phi(\mathbf{X})}{T}\right\}$.
7: If $S$ attempts have passed since the last improvement, decrease the temperature $T$ and repeat Steps 4–6.
8: If $C < N$, increment $C$ and repeat Steps 4–7; Otherwise, terminate and return $\mathbf{X}$.
```

and thus lead to a faster stop of the algorithm. Yet, it may also result in larger probability of missing the true global optimum. Considering this trade-off, it is recommended to set T between 0.05 to 0.15. The tuning parameter S indicates the maximum consecutive attempts the algorithm will try without improvements before temperature reduces, and [43] recommends it to be around 5, depending on how expensive the objective function is to evaluate.

It is straightforward to use Simulated Annealing to find designs in R. For example:

```
# 10 Runs, 2 inputs, 25 iterations of Simulated Annealing algorithm
LHD_SA <- SA(n = 10, k = 2, N = 25)
```

Similarly, designs satisfying the multi-objective approach can be found by:

```
# 10 Runs, 2 inputs, 25 iterations of Simulated Annealing algorithm
# using multi-objective
multi_obj_design <- SA2008(n = 10, k = 2, N = 25)
```

3.2.2 Genetic Algorithms

The GA is a metaheuristic algorithm inspired by the process of natural selection [25] [17]. The GA starts from a population of randomly generated candidate solutions (designs), called chromosomes. The population of chromosomes in each iteration is called a generation. For each generation the objective function will be evaluated for each chromosome, with the corresponding value being known as the fitness. The more fit chromosomes will be allowed to survive to the next generation, while the less fit chromosomes will be replaced by new offspring. These offspring are obtained by selecting several chromosomes (called parents) and recombining their settings using crossover and mutation techniques to produce offspring with potentially better fitness.

[34] adapted the general GA framework for searching for maximin LHDs. Their approach begins with random LHDs as the initial population. They then perform a selection step in which the best half of the LHDs are allowed to survive to the next generation. Then, a crossover step is performed in which random columns in these survivors are exchanged with other survivors. Additionally, to encourage diversity in the solutions and prevent the algorithm becoming stuck in a local optima, a mutation step is performed in which two random elements in a column are exchanged. Note that the current best chromosome is excluded from this mutation in order to preserve the best current solution. Finally, the fitness of the new population of LHDs is calculated, and the process is repeated until the stopping criteria is satisfied. We include a detailed description of the GA, along with the tuning parameters, in Algorithm 2.

Algorithm 2 Genetic Algorithm for LHD

```
1: Set the probability of mutation, $p_{mut}$. Suggested setting is $p_{mut} = 1/(d-1)$ [34]. Set the maximum number of iterations $N$ and the counter index $C = 1$.
2: Generate $m$ random $n \times d$ LHDs, denoted by $X_1, \ldots, X_m$, where $m$ is the population size (number of chromosomes). Here, $m$ must be an even number.
3: Evaluate the objective function, $\Phi(X_i)$, for $i = 1, \ldots, m$.
4: Select *survivors*: order the $X_i$ by their objective function values and select the best $\frac{m}{2}$ $X_i$ (with the smallest $\frac{m}{2}$ $\Phi$ values), denoted by $X_i^s$ for $i = 1, \ldots, \frac{m}{2}$, WLOG.
5: Let $X_b^s = \underset{i}{\operatorname{argmin}}\, \Phi(X_i^s)$ (i.e. $X_b^s$ is the best *survivor*)
6: **for** each $X_i^s$, excluding $X_b^s$, **do**
7:     Randomly choose a column $j$ from $X_b^s$, and replace it with the $j^{th}$ column from $X_i^s$.
8: **end for**
9: **for** each $X_i^s$, excluding $X_b^s$, **do**
10:     Randomly choose a column $j$ from $X_i^s$, and replace it with the $j^{th}$ column from $X_b^s$.
11: **end for**
12: Update $X_i$: let $X_1 = X_b^s$ and the $X_2, \ldots, X_{m/2}$ be the design matrices obtained by steps 6–8. Let $X_{m/2+1} = X_b^s$ and $X_{m/2+2}, \ldots, X_m$ be the design matrices generated by Steps 9–11.
13: **for** each $X_i$ (except $X_1$) **do**
14:     **for** each column $j$ of $X_i$ **do**
15:         **if** $z < p_{mut}$ where $z \sim \text{Uniform}(0,1)$ **then**
16:             Exchange two randomly selected elements in $j$.
17:         **end if**
18:     **end for**
19: **end for**
20: Calculate $\Phi(X_i)$ for all $i$.
21: if $C \le N$, set $C = C + 1$ and repeat Steps 4-21; otherwise, stop the algorithm.
```

It is also straightforward to use the GA to find space-filling designs in R. For example:

```
# 10 Runs, 2 inputs, 25 iterations of Genetic algorithm
# OC is optimality criteria - phi_p is the maximin distance
LHD_GA <- GA(n = 10, k = 2, N = 25, OC = ‘‘phi_p’’)
```

4 Summary and Conclusions

Sophisticated computer simulators allow scientists to test complex systems which would be too expensive or completely impossible to assess otherwise. These simulations are usually very time-consuming, and computationally cheap surrogates are called for to facilitate the analysis and optimization of the underlying system. GPs are popular choices for such surrogates (or emulators). In order to effectively reap the benefits of utilizing the surrogate, the simulator should be evaluated on a set of points chosen efficiently. Latin hypercube designs have proven efficient for that purpose.

In this tutorial paper we discussed design criteria and subsequent metaheuristic optimization strategies for finding designs that allow astronomy researchers to extract the maximum benefit offered by GP surrogate modeling. We provided an overview of model fitting using GPs and identification of optimal Latin hypercube designs. Relevant R codes have been used for illustration. Apart from the libraries discussed in the paper, there are many other packages in R that can be used. Interested readers may want to consider the laGP (Local Approximate Gaussian Process Regression [21]), DiceKriging (Kriging Methods for Computer Experiments [47]), GPfit (Gaussian Processes Modeling [38]) and SLHD (Maximin-Distance (Sliced) Latin Hypercube Designs [1]) packages.

One consideration not covered in this tutorial paper is how to best utilize GP models when the data sets are astronomically large. Such "big data" may cause the estimation techniques to become quite slow, requiring advanced techniques to speed up the estimation. This is a topic of active research. For further details, see [36].

References

1. S. Ba, SLHD: maximin-distance (sliced) Latin hypercube designs, https://cran.r-project.org/web/packages/SLHD/index.html, R package version 2.1-1 (2015)
2. S. Ba, V.R. Joseph, Composite Gaussian process models for emulating expensive functions. Ann. Appl. Stat. **6**(4), 1838–1860 (2012)
3. S.D. Beattie, D.K.J. Lin, A new class of Latin hypercube for computer experiments, in Contemporary multivariate analysis and designs of experiments, in Celebration of Prof. Kai-Tai Fang's 65th Birthday. Singapore: World Scientific, pp. 205–226 (2005)
4. P. Behroozi, R. Wechsler, A. Hearin, C. Conroy, UniverseMachine: the correlation between galaxy growth and dark matter halo assembly from Z= 0–10. Mon. Not. R. Astron. Soc. **488**(3), 3143–3194 (2019)
5. N. Bhattacharjeea, P. Ranjan, A. Mandal, E.W. Tollner, A history matching approach for calibrating hydrological models. Environ. Ecol. Stat. **26**(1), 87–105 (2019)
6. D. Bingham, Branin function. Virtual library of simulation experiments, https://www.sfu.ca/~ssurjano/branin.html
7. D. Bursztyn, D.M. Steinberg, Rotation designs: orthogonal first-order designs with higher order projectivity. Appl. Stoch. Models Bus. Ind. **18**(3), 197–206 (2002)
8. P. Charbonneau, Genetic algorithms in astronomy and astrophysics. Astrophys. J. Suppl. Ser. **101**, 309–334 (1995)
9. T.M. Cioppa, T.W. Lucas, Efficient nearly orthogonal and space-filling Latin hypercubes. Technometrics **49**(1), 45–55 (2007)
10. G.M. Dancik, mlegp: maximum likelihood estimates of Gaussian processes,https://cran.r-project.org/web/packages/mlegp/index.html, R package version 3.1.8 (2020)
11. X. Deng, C.D. Lin, K.-W. Liu, R.K. Rowe, Additive Gaussian process for computer models with qualitative and quantitative factors. Technometrics **59**(3), 283–292 (2017)
12. S.G. Djorgovski, R. Brunner, A. Mahabal, R. Williams, R. Granat, P. Stolorz, Challenges for cluster analysis in a virtual observatory, in *Statistical Challenges in Astronomy* (Springer, 2003), pp. 127–141
13. K.T. Fang, R. Li, A. Sudjianto, *Design and modeling for computer experiments* (CRC Press, Boca Raton, 2006)
14. A. Forrester, A. Sobester, A. Keane, *Engineering design via surrogate modelling: a practical guide* (John Wiley & Sons, Hoboken, 2008)
15. S.D. Georgiou, Orthogonal Latin hypercube designs from generalized orthogonal designs. J. Stat. Plan. Inference **139**(4), 1530–1540 (2009)
16. M. Giuliano, M. Johnston, Multi-objective evolutionary algorithms for scheduling the James webb space telescope. ICAPS, pp. 107–115 (2008)
17. D.E. Goldberg, Genetic algorithms in search, in *Optimization and Machine Learning* (Addison Wesley Publishing Co. Inc, 1989)
18. R.B. Gramacy, *Surrogates: Gaussian process modeling, design, and optimization for the applied sciences* (CRC Press, Boca Raton, 2020)
19. R.B. Gramacy, D.W. Apley, Local Gaussian process approximation for large computer experiments. J. Comput. Graph. Stat. **24**, 2 (2015)
20. R.B. Gramacy, H.K.H. Lee, Bayesian treed Gaussian process models with an application to computer modeling. J. Am. Stat. Assoc. **103**, 1119–1130 (2008)
21. R.B. Gramacy, F. Sun, laGP: local approximate gaussian process regression, https://cran.r-project.org/web/packages/laGP/index.html, R package version 1.5-5 (2019)
22. A. Gration, M. Wilkinson, Dynamical modelling of Dwarf spheroidal galaxies using Gaussian-process emulation. Mon. Not. R. Astron. Soc. **485**(4), 4878–4892 (2019)
23. G. Han, T.J. Santner, W.I. Notz, D.L. Bartel, Prediction for computer experiments having quantitative and qualitative input variables. Technometrics **51**(3), 278–288 (2009)
24. F. Hickernell, A generalized discrepancy and quadrature error bound. Math. Comput. Am. Math. Soc. **67**(221), 299–322 (1998)
25. J.H. Holland et al., *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence* (MIT press, Cambridge, 1992)

26. E.R. Hruschka, R.J.G.B. Campello, A.A. Freitas et al., A survey of evolutionary algorithms for clustering. IEEE Trans. Syst. Man Cybern. Part C Appl. Rev. **39**(3), 133–155 (2009)
27. M.E. Johnson, L.M. Moore, D. Ylvisaker, Minimax and maximin distance designs. J. Stat. Plan. Inference **26**(2), 131–148 (1990)
28. V.R. Joseph, Y. Hung, Orthogonal-maximin Latin hypercube designs. Stat. Sin. **18**, 171–186 (2008)
29. V.R. Joseph, E. Gul, S. Ba, Maximum projection designs for computer experiments. Biometrika **102**(2), 371–380 (2015)
30. M. Kennedy, A. O'Hagan, Bayesian calibration of computer models. J. R. Stat. Soc. Ser. B Stat. Methodol. **63**(3), 425–464 (2002)
31. S. Kirkpatrick, C.D. Gelatt, M.P. Vecchi, Optimization by simulated annealing. Science **220**(4598), 671–680 (1983)
32. M. Kruckow, T. Tauris, N. Langer, M. Kramer, Robert G. Izzard, Progenitors of gravitational wave mergers: binary evolution with the stellar grid-based code COMBINE. Mon. Not. R. Astron. Soc. **481**(2), 1908–1949 (2018)
33. S. Leary, A. Bhaskar, A. Keane, Optimal orthogonal-array-based Latin hypercubes. J. Appl. Stat. **30**(5), 585–598 (2003)
34. M. Liefvendahl, R. Stocki, A study on algorithms for optimization of Latin hypercubes. J. Stat. Plan. Inference **136**(9), 3231–3247 (2006)
35. C.D. Lin, R. Mukerjee, B. Tang, Construction of orthogonal and nearly orthogonal Latin hyper-cubes. Biometrika **96**(1), 243–247 (2009)
36. H. Liu, Y.-S. Ong, X. Shen, J. Cai, When Gaussian process meets big data: a review of scalable GPs. IEEE Trans. Neural Netw. Learn. Syst. **31**(11), 4405–4423 (2020)
37. A. Mandal, W.K. Wong, Y. Yu, Algorithmic searches for optimal designs, in *Handbook of Design and Analysis of Experiments* (CRC Press, Boca Raton, 2015), pp. 755–783
38. B. MacDoanld, H. Chipman, C. Campbell, P. Ranjan, GPfit: Gaussian processes modeling, https://cran.r-project.org/web/packages/GPfit/index.html, R package version 1.0-8 (2019)
39. M.D. McKay, R.J. Beckman, W.J. Conover, Comparison of three methods for selecting values of input variables in the analysis of output from a computer code. Technometrics **21**(2), 239–245 (1979)
40. M. Misiak et al., Evolutionary algorithms in astrodynamics. Int. J. Astron. Astrophys. **6**(4), 435–439 (2016)
41. S. Mohanty, Particle swarm optimization and regression analysis-I. Astron. Rev. **7**(2), 29–35 (2012)
42. S. Mohanty, E. Fahnestock, Adaptive spline fitting with particle swarm optimization. Comput. Stat. **36**, 155–191 (2020)
43. M.D. Morris, T.J. Mitchell, Exploratory designs for computational experiments. J. Stat. Plan. Inference **43**(3), 381–402 (1995)
44. P.Z.G. Qian, H. Wu, C.F.J. Wu, Gaussian process models for computer experiments with qualitative and quantitative factors. Technometrics **50**(3), 383–396 (2008)
45. R Core Team, R: a language and environment for statistical computing, R Foundation for Statistical Computing, Vienna, Austria, https://www.R-project.org/ (2019)
46. J. Resch, A. Mandal, P. Ranjan, Inverse problem for dynamic computer simulators via multiple scalar-valued contour estimation, https://arxiv.org/abs/2010.08941 (2021)
47. O. Roustant, D. Ginsbourger, Y. Deville, C. Clement, Y. Richet, DiceKriging: kriging methods for computer experiments, https://cran.r-project.org/web/packages/DiceKriging/index.html, R package version 1.5.8 (2020)
48. J. Sacks, W.J. Welch, T.J. Mitchell, H.P. Wynn, Design and analysis of computer experiments. Stat. Sci. **4**(4), 409–423 (1989)
49. D.M. Steinberg, D.K.J. Lin, A construction method for orthogonal Latin hypercube designs. Biometrika **93**(2), 279–288 (2006)
50. S. Stevenson, A. Vigna-Gómez, I. Mandel, J.W. Barrett, C.J. Neijssel, D. Perkins, S.E. De Mink, Formation of the first three gravitational-wave observations through isolated binary evolution. Nat. Commun. **8**(1), 1–7 (2017)
51. F. Sun, M.-Q. Liu, D.K.J. Lin, Construction of orthogonal Latin hypercube designs with flexible run sizes. J. Stat. Plan. Inference **140**(11), 3236–3242 (2010)
52. F. Sun, B. Tang, A general rotation method for orthogonal Latin hypercubes. Biometrika **104**(2), 465–472 (2017)
53. L.P. Swiler, P.D. Hough, P.Z.G. Qian, X. Xu, C. Storlie, H. Lee, Surrogate models for mixed discrete-continuous variables, *Constraint Programming and Decision Making* (Springer, 2014), pp. 181–202
54. B. Tang, Orthogonal array-based Latin hypercubes. J. Am. Stat. Assoc. **88**(424), 1392–1397 (1993)
55. I. Vernon, M. Goldstein, R.G. Bower, Galaxy formation: a Bayesian uncertainty analysis. Bayesian Anal **5**(4), 619–669 (2010)
56. A. Vigna-Gómez, C.J. Neijssel, S. Stevenson, J.W. Barrett, K. Belczynski, S. Justham, S.E. de Mink, B. Müller, P. Podsiadlowski, M. Renzo, D. Szécsi, On the formation history of Galactic double neutron stars. Mon. Not. R. Astron. Soc. **481**(3), 4009–4029 (2018)
57. H. Wackernagel, *Multivariate geostatistics* (Springer, Berlin, 2002)
58. H. Wang, Q. Xiao, A. Mandal, LHD: Latin hypercube designs (LHDs), https://CRAN.R-project.org/package=LHD, R package version 1.3.1 (2020)
59. H. Wang, Q. Xiao, A. Mandal, Musings about constructions of efficient Latin hypercube designs with flexible run-sizes, arXiv preprint arXiv:2010.09154v2 (2020)
60. L. Wang, Q. Xiao, H. Xu, Optimal maximin L_1-distance Latin hypercube designs based on good lattice point designs. Ann. Stat. **46**(6B), 3741–3766 (2018)
61. W.J. Welch, R.J. Buck, J. Sacks, H.P. Wynn, T.J. Mitchell, M.D. Morris, Screening, predicting, and computer experiments. Technometrics **34**, 15–25 (1992)
62. D. Williams, I.S. Heng, J. Gair, J.A. Clark, B. Khamesra, A precessing numerical relativity waveform surrogate model for binary black holes: a Gaussian process regression approach, arXiv preprint arXiv:1903.09204 (2019)
63. Q. Xiao, Constructions and applications of space-filling designs, Ph.D. Dissertation, University of California Los Angeles (2017)

64. Q. Xiao, A. Mandal, C.D. Lin, X. Deng, EzGP: Easy-to-interpret Gaussian Process models for computer experiments with both quantitative and qualitative factors. Under revision for SIAM/ASA J. Uncertain. Quantif. (2021)
65. Q. Xiao, H. Xu, Construction of maximin distance Latin squares and related Latin hypercube designs. Biometrika **104**(2), 455–464 (2017)
66. Q. Xiao, H. Xu, Construction of maximin distance designs via level permutation and expansion. Stat. Sin. **28**(3), 1395–1414 (2018)
67. Y. Xiong, W. Chen, D.W. Apley, X. Ding, A non-stationary covariance-based kriging method for metamodelling in engineering design. Int. J. Numer. Methods Eng. **71**, 733–756 (2007)
68. J. Yang, M. Liu, Construction of orthogonal and nearly orthogonal Latin hypercube designs from orthogonal designs. Stat. Sin. **22**, 433–442 (2012)
69. K.Q. Ye, Orthogonal column Latin hypercubes and their application in computer experiments. J. Am. Stat. Assoc. **93**(444), 1430–1439 (1998)
70. Y. Zhang, W.I. Notz, Computer experiments with qualitative and quantitative variables: a review and reexamination. Qual. Eng. **27**(1), 2–13 (2015)
71. Y. Zhang, S. Tao, W. Chen, D.W. Apley, A latent variable approach to Gaussian process modeling with qualitative and quantitative factors. Technometrics **62**(3), 291–302 (2020)
72. Q. Zhou, P.Z.G. Qian, S. Zhou, A simple approach to emulation for computer models with qualitative and quantitative factors. Technometrics **53**(3), 266–273 (2011)