Research Article **Open Access**
Special Issue: Recent Developments in Quantitative Risk Management

Guojun Gan and Emiliano A. Valdez*

An empirical comparison of some experimental designs for the valuation of large variable annuity portfolios

DOI 10.1515/demo-2016-0022
Received August 24, 2016; accepted November 26, 2016

Abstract: Variable annuities contain complex guarantees, whose fair market value cannot be calculated in closed form. To value the guarantees, insurance companies rely heavily on Monte Carlo simulation, which is extremely computationally demanding for large portfolios of variable annuity policies. Metamodeling approaches have been proposed to address these computational issues. An important step of metamodeling approaches is the experimental design that selects a small number of representative variable annuity policies for building metamodels. In this paper, we compare empirically several multivariate experimental design methods for the GB2 regression model, which has been recently discovered to be an attractive model to estimate the fair market value of variable annuity guarantees. Among the experimental design methods examined, we found that the data clustering method and the conditional Latin hypercube sampling method produce the most accurate results.

Keywords: Variable annuity, Portfolio valuation, Metamodeling, Generalized beta of the second kind (GB2), Multivariate experimental design, Data clustering, Latin hypercube

MSC: 62K99

1 Introduction

A variable annuity (VA) is a popular life insurance product that offers dynamic investment opportunities with guarantees [21, 33]. There are two main classes of guarantees provided by variable annuities [28]: guaranteed minimum death benefit (GMDB) and guaranteed minimum living benefit (GMLB). The second class can be further divided into the following three subclasses: guaranteed minimum accumulation benefit (GMAB), guaranteed minimum income benefit (GMIB), and guaranteed minimum withdrawal benefit (GMWB). The whole set of guarantees are often referred to as GMxB, where the letter x indicates the class of benefits involved. Figure 1 shows the hierarchy of the set of guarantees mentioned above.

A basic GMDB guarantees a return of the principal upon the death of the policyholder regardless of the performance of the investment account. One variation to the basic form is a roll-up GMDB, which guarantees the principal accumulated at a specified roll-up rate. A ratchet GMDB is another variation that locks the gains in the account balance on each of the dates specified. A GMAB is similar to a GMDB except that the guarantees are not contingent on the death of the policyholder. The guarantees of a GMAB are typically triggered on policy anniversaries. Variations of a GMAB include roll-up rates, ratchets, and resets. The reset feature enables a policyholder to start a new GMAB by locking a higher benefit. A GMIB guarantees a minimum in-

Guojun Gan: Department of Mathematics, University of Connecticut, E-mail: guojun.gan@uconn.edu
***Corresponding Author: Emiliano A. Valdez:** Department of Mathematics, University of Connecticut, E-mail: emiliano.valdez@uconn.edu

come stream from a specified future point in time. A GMWB guarantees that a policyholder can withdraw a specified amount for a specified number of years.

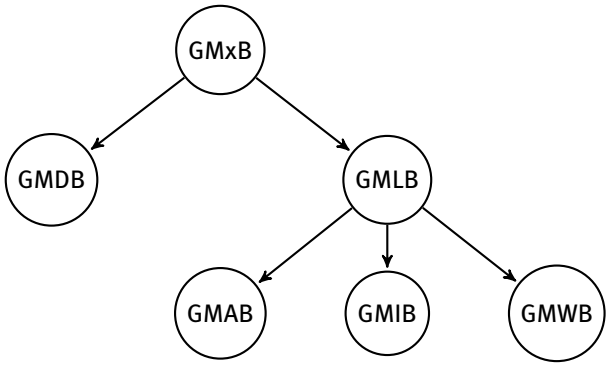


Figure 1: Some common guarantees provided by variable annuities.

Many insurance companies that have a VA business adopt dynamic hedging to mitigate the financial risks associated with the guarantees. Dynamic hedging requires calculating the sensitivities (i.e., Greeks) of the VA liabilities to key risk drivers [3]. Due to the complexity of the guarantees, insurance companies rely on Monte Carlo simulation to calculate the Greeks, which are quantities that measure sensitivities of the guarantee values to major market indices. For example, the Delta measures the sensitivity of the guarantee value to the underlying asset and the Rho measures the sensitivity of the guarantee value to interest rates. Under Monte Carlo simulation, the cash flows of the guarantee are projected along many scenarios for a long time horizon (e.g., 30 years) at some time steps (e.g., monthly). As a result, using Monte Carlo simulation to calculate the Greeks of a large portfolio of VA policies is extremely time-consuming. Insurance companies that have a VA business usually have a large portfolio of these products. To implement a dynamic hedging strategy, it is essential to be able to calculate the Greeks of the whole portfolio in order to adjust the hedge position promptly. In practice, insurance companies have used parallel computing to calculate the Greeks [29].

Using parallel computing to address the computational issue has some limitations. First, this method is costly. An insurance company with a large VA portfolio requires many computers in order to complete the calculation within a reasonable time frame. Second, this method is not scalable. When an insurance company grows its VA business, it needs to use more computers in order to complete the calculation within the existing time interval.

In the past few years, the metamodeling approach [1, 8] has been used to address the computational issue from the software perspective. The metamodeling method involves four major steps:

Step 1 Select a small number of representative VA policies.

Step 2 Use Monte Carlo simulation to calculate the quantities (e.g., Greeks) of interest for the representative VA policies.

Step 3 Build a metamodel.

Step 4 Use the metamodel to estimate the quantities of interest for the whole VA portfolio.

Here a metamodel is referred to as a model of the Monte Carlo simulation model. Since Monte Carlo simulation is applied to only a small number of representative VA policies, the metamodeling approach is able to reduce the runtime significantly while producing accurate estimates.

A metamodeling method contains two important components: an experimental design method for selecting representative VA policies and a metamodel. In [10], a kriging metamodel was proposed to estimate the fair market value, dollar delta, and dollar rho of a large VA portfolio. In [14], a functional kriging was used to estimate the dollar delta of a large VA portfolio under nested simulation. In these metamodeling methods, a clustering method was used to select representative VA policies. In [11], a Latin hypercube sampling method

was used to select representative VA policies. In [15], a two-level metamodeling method was proposed to estimate the dollar deltas efficiently for dynamic hedging purpose. In [30], a metamodeling approach was used to analyze mortality-linked contracts in stochastic mortality models. In [17], a neural network method was proposed to calculate the Greeks of a large VA portfolio.

Recently, we developed a GB2 (generalized beta of the second kind) regression model as a metamodel in order to capture the positive skewness typically observed in the distribution of the fair market value of guarantees [16]. In this metamodeling approach, the conditional Latin hypercube sampling method [25, 31] was used to select representative VA policies. Previous studies have shown that Latin hypercube sampling works well with kriging [34]. In this paper, we investigate some experimental design methods for the GB2 regression model, which is similar in concept to generalized linear models (GLM) [6, 23]. A major contribution of this paper is to provide guidance for choosing an experimental design method for the GB2 regression model.

GLMs have been traditionally used in actuarial science for pricing, reserving, and solvency testing [7]. In these traditional applications, we generally work with observational data, which are readily available at little or no cost. For example, a P&C insurance company has the information of policyholders and their claim data, which can be used to build a GLM. Building a GLM as a metamodel for predicting the Greeks of variable annuity guarantees is quite different in that we only know the policy information but do not know the Greeks. In order to build a GLM as a metamodel, we need to use the Monte Carlo model to calculate the Greeks of a small set of VA policies. The small set of VA policies is referred to as an experimental design or a design. The choice of design is an important task in the development of an adequate GLM.

The remaining part of this paper is organized as follows. In Section 2, we give a brief description of the GB2 regression model. In Section 3, we introduce five experimental design methods for selecting representative VA policies: random sampling, low-discrepancy sequence, data clustering, Latin hypercube sampling, and conditional Latin hypercube sampling. In Section 4, we present some numerical results to illustrate the performance of the experimental design methods in terms of accuracy and speed. Section 5 concludes the paper with some remarks.

2 The Dataset and the GB2 Regression Model

In this section, we give a brief description of the dataset, the GB2 regression model proposed in [16], and how the parameters are estimated.

2.1 The Dataset

The dataset contains 10,000 synthetic VA policies, each of which is described by 50 variables. Since many of variables have identical value, these variables are excluded in the regression model. The policyholders of these VA policies are allowed to choose from ten investment funds. The explanatory variables used to build the regression model include:

- `gender` - Gender of the policyholder,
- `prodType` - Product type of the VA policy,
- `gmdbAmt` - GMDB amount,
- `gmwbAmt` - GMWB amount,
- `gmwbBalance` - GMWB balance,
- `gmmbAmt` - GMMB amount,
- `withdrawal` - Total withdrawal,
- `FundValue`i - Account value of the ith fund, for $i = 1, 2, \ldots, 10$,
- `age` - Age of the policyholder, and
- `ttm` - Time to maturity in years.

Table 1: Summary statistics of the explanatory variables and the dependent variable for the GB2 regression model.

(a)

Variable	Category:Count				
gender	F:4071	M:5929			
prodType	DBRP:2028	DBRU:2018	MB:1959	WB:1991	WBSU:2004

(b)

Variable	Min	1st Q	Mean	3rd Q	Max
gmdbAmt	0	0	135116.88	256528.88	986536.04
gmwbAmt	0	0	7888.8	15041.76	69403.72
gmwbBalance	0	0	94151.68	149780.98	991481.79
gmmbAmt	0	0	54715.12	0	499925.4
withdrawal	0	0	26348.89	0	418565.23
FundValue1	0	0	33325.04	49291.86	1030517.37
FundValue2	0	0	43224.12	60462.73	1094839.83
FundValue3	0	0	28623.53	41359.13	672927
FundValue4	0	0	27479.09	41113.74	547874.38
FundValue5	0	0	24225.22	36497.56	477843.32
FundValue6	0	0	35305.36	53520.34	819144.24
FundValue7	0	0	28903.78	44193.65	794470.82
FundValue8	0	0	28745.1	44971.75	726031.63
FundValue9	0	0	27191.4	41633.23	808213.6
FundValue10	0	0	26666.22	41283.93	709232.82
age	34.36	42.08	49.38	56.82	64.37
ttm	0.68	10.41	14.65	18.85	28.68

(c)

Variable	Min	1st Q	Median	Mean	3rd Q	Max
fmv	-30214.7	-2593.15	7010.66	18210.98	31362.78	285182.42

The summary statistics of the explanatory variables are given in Table 1. Table 1(a) shows the categories and their frequencies of the two categorical variables. From the table, we see that about 40% of the policyholders are female. There are five types of guarantees and the percentage of each type is about 20% of the portfolio.

Table 1(b) shows the summary statistics of the continuous explanatory variables. All these variables have nonnegative values. Except for the variables `age` and `ttm`, all variables have many zero values. In particular, more than 75% of the values of `gmmbAmt` and `withdrawal` are zero. The reason is that only VA policies with a GMMB may have a positive GMMB amount and only VA policies with a GMWB may have a positive withdrawal amount.

The fair market values of the guarantees are calculated by a simple Monte Carlo simulation model [12]. Table 1(c) shows the summary statistics of the fair market values. From the table, we see that there are negative fair market values. Since the fair market value is equal to the present value of benefits minus that of the fees, it is negative when the present value of benefits is less than that of the fees. We also see that the median is much lower than the mean, indicating that the distribution is skewed to the right. The positive skewness is confirmed in Figure 2, which shows a histogram of the fair market values.

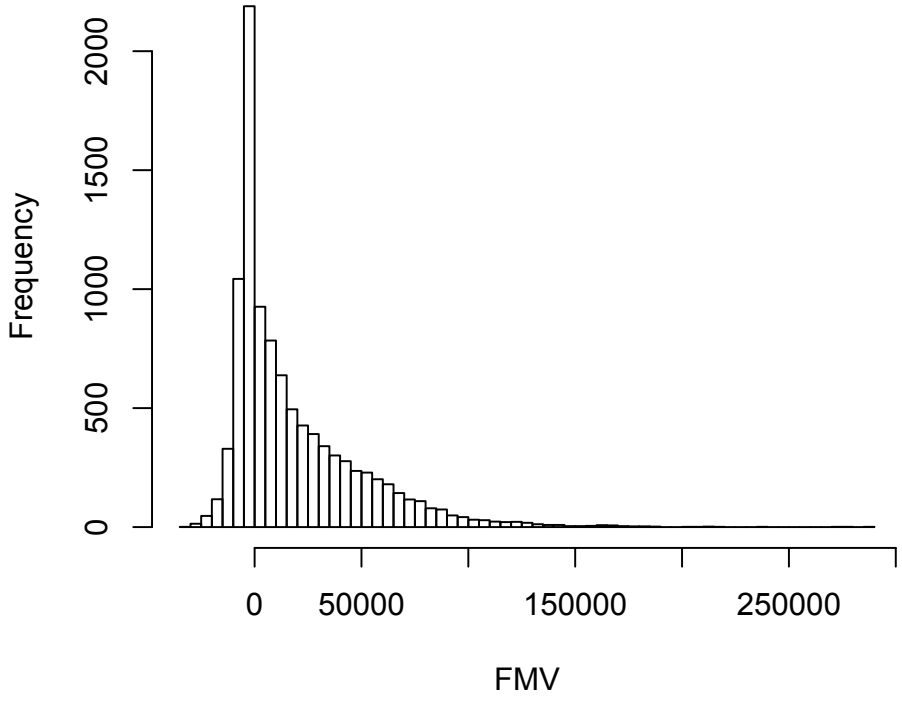


Figure 2: A histogram of the fair market values.

2.2 The Model

The GB2 is a four-parameter distribution for positive continuous data and can be used to model distributions characterized by fat tails [5]. With four parameters, the GB2 provides an extremely flexible functional form and include many other distributions as special or limiting cases. For example, the gamma distribution, the lognormal distribution, the Weibull distribution, and the exponential distribution are special or limiting cases of the GB2 distribution. Due to its flexibility and ability to model skewed data, the GB2 was proposed to model the fair market values of VA guarantees that are typically observed to be positively skewed [16].

The probability density function of a GB2 random variable X is defined as [5, 19]:

$$f(x) = \frac{|a|}{bB(p,q)} \left(\frac{x}{b}\right)^{ap-1} \left[1 + \left(\frac{x}{b}\right)^{a}\right]^{-p-q}, \quad x > 0, \tag{1}$$

where $a \neq 0$, $p > 0$, $q > 0$, $b > 0$, and $B(p, q)$ is the Beta function. The parameters a, p, and q are referred to as the shape parameters of the GB2 distribution. The parameter b is called the scale parameter of the GB2 distribution. The expectation of the GB2 random variable X exists if the shape parameters satisfy the following condition:

$$-p < \frac{1}{a} < q. \tag{2}$$

If the above condition is satisfied, the expectation is given by

$$E[X] = \frac{bB\left(p + \frac{1}{a}, q - \frac{1}{a}\right)}{B(p,q)}. \tag{3}$$

Let Y denote the fair market value of guarantees embedded in a VA policy. Since Y can be negative, the shifted fair market value

$$X = Y + c$$

is modeled with a GB2 distribution with parameters a, b, p, and q. The shift parameter c will be estimated from the data.

Independent or regressor variables can be incorporated in several ways: through the shape parameter a by using $a(\mathbf{z}_i) = \mathbf{z}_i^T\boldsymbol{\beta}$, through the scale parameter b by using $b(\mathbf{z}_i) = \exp(\mathbf{z}_i^T\boldsymbol{\beta})$, or through both the shape parameter a and the scale parameter b. Since the GB2 model is constructed for the purpose of predicting the fair market values of guarantees embedded in new VA policies, incorporating independent variables through the shape parameter a is not suitable. The reason is that $\frac{1}{a}$ can be out of the interval $(-p, q)$ and the resulting expectation, which is used for prediction, will not exist.

The method of maximum likelihood is used to estimate the parameters. Let s be the number of VA policies in the experimental design. For $i = 1, 2, \ldots, s$, let v_i be the fair market value of the guarantees embedded in the ith VA policy in the experimental design. Then the log-likelihood function of the model is defined as follows:

$$\begin{aligned} L(\boldsymbol{\theta}) &= s\ln\frac{|a|}{B(p,q)} - ap\sum_{i=1}^{s}\mathbf{z}_i^T\boldsymbol{\beta} + (ap-1)\sum_{i=1}^{s}\ln(v_i+c) - \\ &\quad (p+q)\sum_{i=1}^{s}\ln\left[1+\left(\frac{v_i+c}{\exp(\mathbf{z}_i^T\boldsymbol{\beta})}\right)^a\right], \end{aligned} \tag{4}$$

where $\boldsymbol{\theta} = (a, p, q, c, \boldsymbol{\beta})$ and $\mathbf{z}_i$ is a numerical vector representing the ith VA policy in the experiment design.

Using the fitted GB2 regression model to predict the fair market values of guarantees for the portfolio is straightforward. Let n be the number of VA policies in the portfolio and $\mathbf{x}_i$ the numeric vector characterizing the ith VA policy in the portfolio. Then the fair market value of guarantees for the ith VA policy can be estimated as follows:

$$\widehat{y}_i = \frac{\exp(\mathbf{x}_i^T\widehat{\boldsymbol{\beta}})B\left(\widehat{p}+\frac{1}{\widehat{a}}, \widehat{q}-\frac{1}{\widehat{a}}\right)}{B(\widehat{p},\widehat{q})} - \widehat{c}, \quad i = 1, 2, \ldots, n, \tag{5}$$

where $\widehat{a}$, $\widehat{p}$, $\widehat{q}$, $\widehat{c}$, and $\widehat{\boldsymbol{\beta}}$ are parameters estimated from the data.

2.3 Parameter Estimation

Estimating the parameters of the GB2 model poses some challenges due to the following reasons:

- The log-likelihood function given in Equation (4) is highly nonlinear and maximizing it cannot be done in closed form.
- The log-likelihood function has multiple local maxima.

Since maximizing the log-likelihood function cannot be done in closed form, a numerical procedure is required to find the optimum parameters. Numerical procedures are usually iterative procedures that start with an initial set of parameters supplied by the users and then repeat improving the parameters until some stop criterion is met. Since the log-likelihood function has multiple local maxima, different sets of initial parameters might lead to different optimum parameters. As a result, choosing the set of initial parameters is an essential step in the optimization process. In [16], a four-stage optimization approach was proposed to estimate the parameters of the GB2 regression model. The first three stages are used to find an set of initial parameters, which are used in the last stage to find optimum parameters.

In this section, we give a brief review of the four-stage optimization method. For detailed description, readers are referred to [16].

In the first stage, the following profile log-likelihood function of the three shape parameters is maximized:

$$L_1(a, p, q) = L(a, p, q, c_0, \boldsymbol{\beta}_0), \tag{6}$$

where $c_0 = -\min\{v_1, \ldots, v_s\} + 10^{-6}$ and

$$\boldsymbol{\beta}_0 = \left(\ln\left(\frac{1}{s}\sum_{i=1}^{s}v_i + c_0\right), 0, 0, \ldots, 0\right)^T. \tag{7}$$

In the second stage, the following profile log-likelihood function of the shift parameter is maximized:

$$L_2(c) = L(\widehat{a}, \widehat{p}, \widehat{q}, c, \boldsymbol{\beta}_0), \tag{8}$$

where $\widehat{a}$, $\widehat{p}$, and $\widehat{q}$ are obtained from the first stage and $\boldsymbol{\beta}_0$ is defined in Equation (7). In the third stage, the following profile log-likelihood function of regression coefficients is maximized:

$$L_3(\boldsymbol{\beta}) = L(\widehat{a}, \widehat{p}, \widehat{q}, \widehat{c}, \boldsymbol{\beta}), \tag{9}$$

where $\widehat{a}$, $\widehat{p}$, and $\widehat{q}$ are obtained from the first stage and $\widehat{c}$ is obtained from the second stage. In the last stage, the full log-likelihood function is maximized using the R function `optim` with the following initial set of parameters:

$$\boldsymbol{\theta}_0 = (\widehat{a}, \widehat{p}, \widehat{q}, \widehat{c}, \widehat{\boldsymbol{\beta}}), \tag{10}$$

where $\widehat{a}$, $\widehat{p}$, and $\widehat{q}$ are obtained from the first stage, $\widehat{c}$ is obtained from the second stage, and $\widehat{\boldsymbol{\beta}}$ are obtained from the third stage.

3 Experimental Design

A metamodel is also referred to as a response surface. The choice of design and the subsequent evaluation of a metamodel fitted by the data generated by the design have been developed in an area known as response surface methodology (RSM) [2, 26, 27, 32]. Most design methods for RSM models were developed around agricultural, industrial, and laboratory experiments. These design methods include factorial designs, fractional factorial designs, central composite designs, sequential bifurcation designs, Latin hypercube designs, frequency-based designs, and combined designs. A discussion of these designs and how to select a design based on the number of factors and the complexity of the metamodel can be found in [20].

Most design methods for RSM models are based on standard linear models where the responses are often assumed to be normally distributed with uncorrelated errors and homogeneous variances. GLMs are usually used for data that do not satisfy the above assumptions. Unlike the case for linear models, there are not many design methods developed for GLMs due to the dependence problem that the design that minimizes the mean-squared error of prediction depends on the unknown parameters [18]. Common approaches to addressing the dependence problem include:

- The locally optimal design method, which starts with best guesses of initial values of the parameters and then determines a design that minimizes the mean-squared error of prediction.
- The sequential method, which starts with initial values of the parameters and then updates the estimates of the parameters successively.
- The Bayesian method, which assumes a prior distribution that is incorporated into an appropriate design criterion by integration.
- The quantile dispersion graphs method, which compares different designs based on quantile dispersion profiles.

A review of the above design methods can be found in [18].

The above experimental design methods developed for GLMs are not suitable in our situation because these design methods require running the Monte Carlo simulation model. Since the Monte Carlo simulation model is computationally expensive, it is not practical to run it in order to find an optimal design.

Since many factors (i.e., independent variables) affect the fair market values of the guarantees, some of the experimental design methods for RSM models are not suitable. For example, the number of points contained in a factorial design is an exponential function of the number of factors. Running the Monte Carlo simulation model for a large number of VA policies in such a design is time consuming. Latin hypercube designs are recommended when there are many factors and the metamodel is complex [20].

In this section, we introduce several experimental design methods that do not require running the Monte Carlo simulation model. In all these experimental design methods except for the conditional Latin hypercube

method, we work with the $n \times m$ matrix A, where n is the number VA policies in the portfolio and m is the number of independent variables after categorical variables are converted to binary dummy variables. For example, the variable `prodType`, which has four distinct values, is converted to three binary dummy variables. Each row of A is a numeric vector representing a VA policy. Each column of A is normalized to $[0, 1]$. Mathematically, A is defined as follows. Let $V_1, V_2, \ldots, V_m$ be the independent variables, including the binary dummy variables converted from the two categorical variables `gender` and `prodType`. For $i = 1, 2, \ldots, n$ and $j = 1, 2, \ldots, m$, let V_{ij} be the value of the ith VA policy in the jth variable. Then A is defined as

$$A_{ij} = \begin{cases} V_{ij}, & \text{if } V_j \text{ is a binary dummy variable,} \\ \dfrac{V_{ij} - L_j}{U_j - L_j}, & \text{if } V_j \text{ is a continuous variable,} \end{cases} \tag{11}$$

where L_j and U_j are the minimum and maximum values of V_j, respectively.

3.1 Random Sampling

Let s be the size of the experimental design, i.e., the number of VA policies in the design. The main steps of the random sampling method are as follows:

Step 1 Generate an $s \times m$ matrix B, where each entry B_{ij} is sampled uniformly from the interval $[0, 1]$.

Step 2 Let $\mathcal{S} = \emptyset$ and $\mathcal{I} = \{1, 2, \ldots, n\}$.

Step 3 For $l = 1, 2, \ldots, s$, let $\mathcal{S} = \mathcal{S} \cup \{S_l\}$, where

$$S_l = \arg\min_{i \in \mathcal{I} \setminus \mathcal{S}} \sum_{j=1}^{m} (A_{ij} - B_{lj})^2.$$

Then set $\mathcal{S}$ contains s distinct indices of the selected VA policies. In the first step, we use the R function `runif` to generate uniform random numbers.

In this experimental design method, we first generate s points from the m-dimensional space $[0, 1]^m$. Then we select sequentially s VA policies from the portfolio that are closest to these s points. If a VA policy is selected, it will not be considered in the next selection. This ensures that we get s distinct VA policies from the portfolio.

3.2 Low-Discrepancy Sequence

Low-discrepancy sequences, also called quasi-random sequences ,are sequences of points with the property that for each i, the points $\{p_1, p_2, \ldots, p_i\}$ have a low discrepancy [4]. A sequence of points is said to have a low discrepancy if the proportion of points in the sequence falling into an arbitrary set is approximately proportional to the measure of the set. There are several definitions for the discrepancy, depending on the shape of the subset and the measure that is used.

We use Sobol sequences as low-discrepancy sequences and use the function `sobol` from the R package `randtoolbox` to generate Sobol sequences. The main steps of the low-discrepancy sequence method are similar to those of the random sampling method:

Step 1 Generate an $s \times m$ matrix B, where the entries $B_{11}, B_{21}, \ldots, B_{nm}$ are from a Sobol sequence.

Step 2 Let $\mathcal{S} = \emptyset$ and $\mathcal{I} = \{1, 2, \ldots, n\}$.

Step 3 For $l = 1, 2, \ldots, s$, let $\mathcal{S} = \mathcal{S} \cup \{S_l\}$, where

$$S_l = \arg\min_{i \in \mathcal{I} \setminus \mathcal{S}} \sum_{j=1}^{m} (A_{ij} - B_{lj})^2.$$

3.3 Data Clustering

Data clustering is a form of unsupervised learning and refers to the process of dividing a dataset into groups or clusters such that points from the same cluster are more similar to each other than points from different clusters [9]. The main steps of using data clustering to create a design are described below:

Step 1 Apply a clustering algorithm to divide A into s clusters. Let B be an $s \times m$ matrix such that the rows correspond to the cluster centers.

Step 2 Let $\mathcal{S} = \emptyset$ and $\mathcal{I} = \{1, 2, \ldots, n\}$.

Step 3 For $l = 1, 2, \ldots, s$, let $\mathcal{S} = \mathcal{S} \cup \{S_l\}$, where

$$S_l = \arg\min_{i \in \mathcal{I} \setminus \mathcal{S}} \sum_{j=1}^{m} (A_{ij} - B_{lj})^2.$$

In the first step, we apply the TFCM (truncated fuzzy c-means) algorithm [13] to divide the dataset into s clusters. The TFCM algorithm is a k-means-type clustering algorithm that is efficient in dividing a large dataset into many clusters by minimizing the following objective function:

$$P(U, B) = \sum_{i=1}^{n} \sum_{l=1}^{s} u_{il}^{\alpha} \left(\sum_{j=1}^{m} (A_{ij} - B_{lj})^2 + \epsilon \right),$$

where $\alpha > 1$ is the fuzzifier, U is a truncated fuzzy partition matrix, B is a matrix representing cluster centers, and ϵ is a small positive number used to prevent division by zero. Details of the TFCM algorithm can be found in [13].

3.4 Latin Hypercube Sampling

Latin hypercube sampling [34] is a popular method for generating experimental designs. A Latin hypercube with s points is constructed by dividing each dimension in the design space into s equally sized intervals, and placing exactly one point in each interval for each dimension. There are many Latin hypercubes for fixed number of points and dimensions. In fact, when the required number of points and the number of dimensions increase, the number of Latin hypercubes increases exponentially [24].

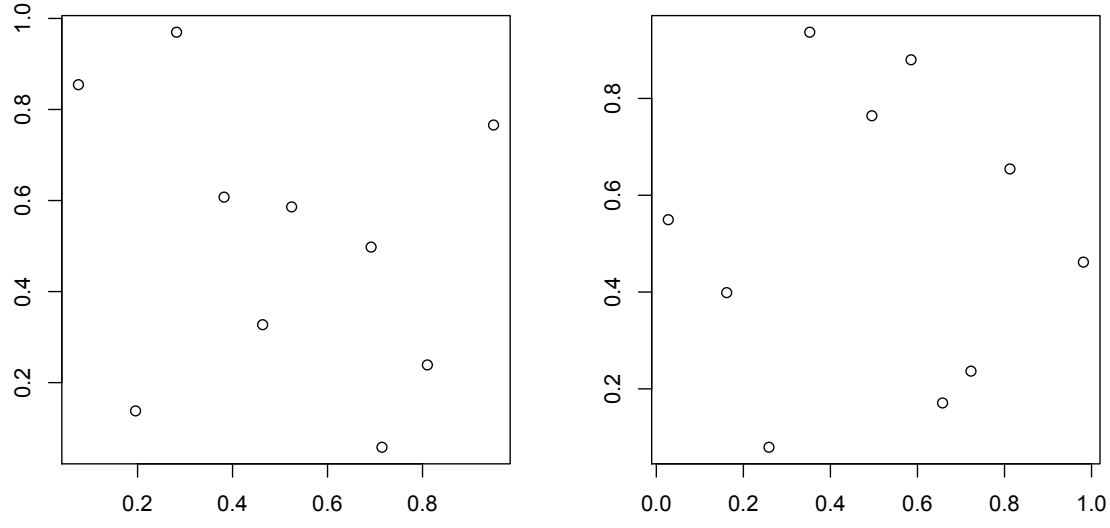


Figure 3: Two two-dimensional Latin hypercube designs created by the function `maximinLHS`.

We use the function `maximinLHS` from the R package `lhs` to create Latin hypercubes. Figure 3 shows two examples of Latin hypercube designs created by the function `maximinLHS`. The main steps of this design method are similar to those of the previous design methods:

Step 1 Generate a Latin hypercube B with s points in an m-dimensional space.

Step 2 Let $\mathcal{S} = \emptyset$ and $\mathcal{J} = \{1, 2, \ldots, n\}$.

Step 3 For $l = 1, 2, \ldots, s$, let $\mathcal{S} = \mathcal{S} \cup \{S_l\}$, where

$$S_l = \arg\min_{i \in \mathcal{J} \setminus \mathcal{S}} \sum_{j=1}^{m} (A_{ij} - B_{lj})^2.$$

3.5 Conditional Latin Hypercube Sampling

The Latin hypercube sampling method presented in the previous subsection is referred to as the unconditional Latin hypercube sampling method. It provides a full coverage of the range of each variable. A conditional Latin hypercube sampling method is different in that it selects a sub-sample from a dataset with ancillary variables in such a way that the sub-sample forms a Latin hypercube and the distribution of the sub-sample is close to that of the dataset [25, 31].

We use the function `clhs` from the R package `cLHS` to create conditional Latin hypercubes. Since the function `clhs` returns a set of indices of the VA policies, we do not need to select VA policies that are closest to the design points.

4 Numerical Results

In this section, we illustrate and compare the performance of the experimental design methods described in the previous section for the GB2 regression model.

4.1 Validation Measures

We use the three validation measures used in [16] to assess the accuracy of various experimental design methods. These validation measures are the percentage error at the portfolio level, the R^2, and the average absolute percentage error.

For $i = 1, 2, \ldots, n$, let y_i and $\widehat{y}_i$ be the fair market values of the ith VA contract obtained from the Monte Carlo simulation model and the GB2 regression model, respectively. Then the percentage error at the portfolio level is defined as

$$PE = \frac{\sum_{i=1}^{n} (\widehat{y}_i - y_i)}{\sum_{i=1}^{n} y_i}. \tag{12}$$

The R^2 is defined as

$$R^2 = 1 - \frac{\sum_{i=1}^{n} (\widehat{y}_i - y_i)^2}{\sum_{i=1}^{n} (y_i - \mu)^2}, \tag{13}$$

where μ is the average fair market value given by

$$\mu = \frac{1}{n} \sum_{i=1}^{n} y_i.$$

The average absolute percentage error is defined as

$$AAPE = \frac{1}{n} \sum_{i=1}^{n} \left| \frac{\widehat{y}_i - y_i}{y_i} \right|. \tag{14}$$

Using these validation measures to compare two experimental designs is straightforward:

- The one that results in a lower absolute PE is better.
- The one that results in a higher R^2 is better.
- The one that results in a lower $AAPE$ is better.

4.2 Results

In this subsection, we present the numerical results to demonstrate the performance of the five experimental design methods described in Section 3. For the sake of simplicity, we use the following abbreviations to represent the experimental design methods in tables and figures:

- RS - The random sampling method.
- LDS - The low-discrepancy sequence method.
- DC - The data clustering method.
- LHS - The Latin hypercube sampling method.
- cLHS - The conditional Latin hypercube sampling method.

Since all the five experimental design methods depend on some random initial values to create the designs, we run all the design methods 10 times with different seeds in order to mitigate the impact of the random initial values.

In terms of the design size (i.e., how many points in the design), we follow the rule of thumb suggested in [22] that the number of design points should be 10 times the number of independent variables. Since the dataset has 22 independent variables, we start with a design size of $s = 220$. To assess the impact of the design size, we also test the experimental design methods with design sizes of $s = 440$ and $s = 880$.

The process to calculate the performance measures of an experimental design method is as follows:

Step 1 Use the experimental design method to select s VA policies from the portfolio.

Step 2 Fit the GB2 regression model based on the s selected VA policies and their fair market values.

Step 3 Use the the GB2 regression model to predict the fair market values of all VA policies in the portfolio.

Step 4 Calculate the validation measures using the predicted fair market values and those calculated by the Monte Carlo simulation model.

Since we focus on the performance of the experimental design methods, we use exactly the same code in Steps 2-4 to make sure these steps are the same for different experimental design methods.

Table 2 shows the average accuracy and speed of the five experimental design methods when $s = 220$. From the table, we see that the data clustering method and the conditional Latin hypercube sampling method are the best methods in terms of the three validation measures. The accuracy of the data clustering method is similar to that of the conditional Latin hypercube sampling method although the former has a lower R^2 and a higher $AAPE$. The random sampling method and the Latin hypercube sampling method produce negative R^2, indicating that the variation of the predicted values is larger than that of the benchmark values obtained from Monte Carlo simulation.

In terms of runtime, the data clustering method and the conditional Latin hypercube sampling method are the slowest among the five methods. The other three design methods are much faster.

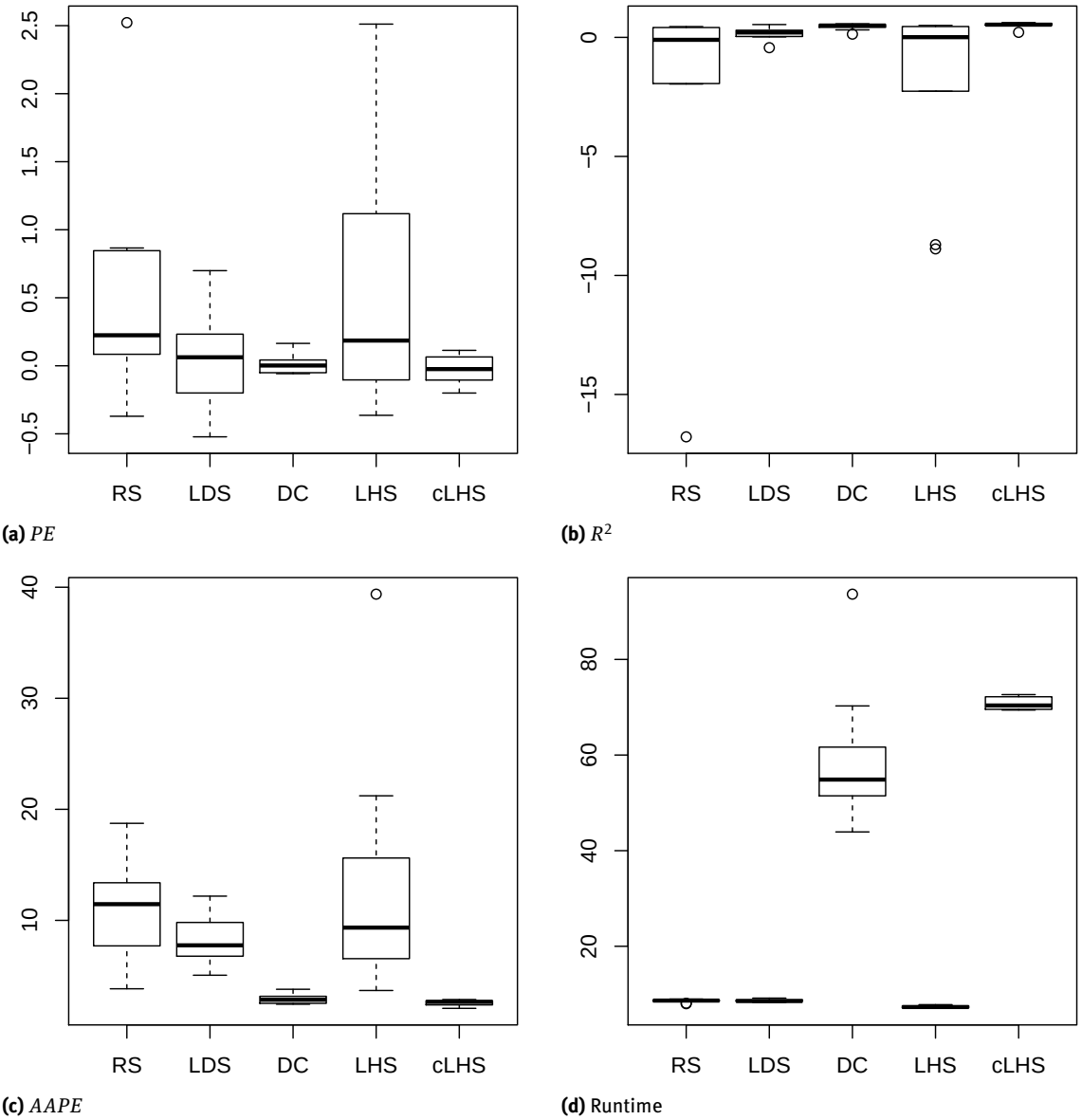
If we examine the standard deviations of the runtime, we find that the runtime of the data clustering method is more volatile than that of the conditional Latin hypercube sampling method. This indicates that the convergence of the data clustering method is affected by the random initial values.

Figure 4 shows the box plots of the three validation measures and the runtime from the 10 runs of the five experimental design methods when $s = 220$. From the figure, we see that the data clustering method and the conditional Latin hypercube sampling method produce stable validation measures. However, Figure 4(d) shows that the runtime of the two methods is much higher.

Table 3 shows the performance measures of the five experimental design methods when $s = 440$. From this table, we see that in terms of accuracy, the best design method is the data clustering method. Among the five methods, the data clustering method has the lowest absolute percentage error, the lowest average absolute percentage error, and the highest R^2. The worst method in terms of accuracy is the random sampling method as it has negative R^2, the highest $AAPE$, and the highest absolute PE.

Table 2: Average accuracy and speed (in seconds) of the experimental design methods over 10 runs when $s = 220$. The numbers in parentheses are the corresponding standard deviations.

	RS	LDS	DC	LHS	cLHS
PE	0.48 (0.83)	0.07 (0.38)	0.02 (0.08)	0.55 (0.91)	-0.03 (0.1)
R^2	-1.96 (5.29)	0.18 (0.27)	0.45 (0.14)	-1.92 (3.72)	0.52 (0.12)
$AAPE$	10.78 (4.63)	8.38 (2.3)	2.95 (0.45)	12.77 (10.72)	2.58 (0.25)
Runtime	8.56 (0.08)	9.08 (0.09)	60.49 (13.97)	9.59 (0.97)	76.25 (0.53)

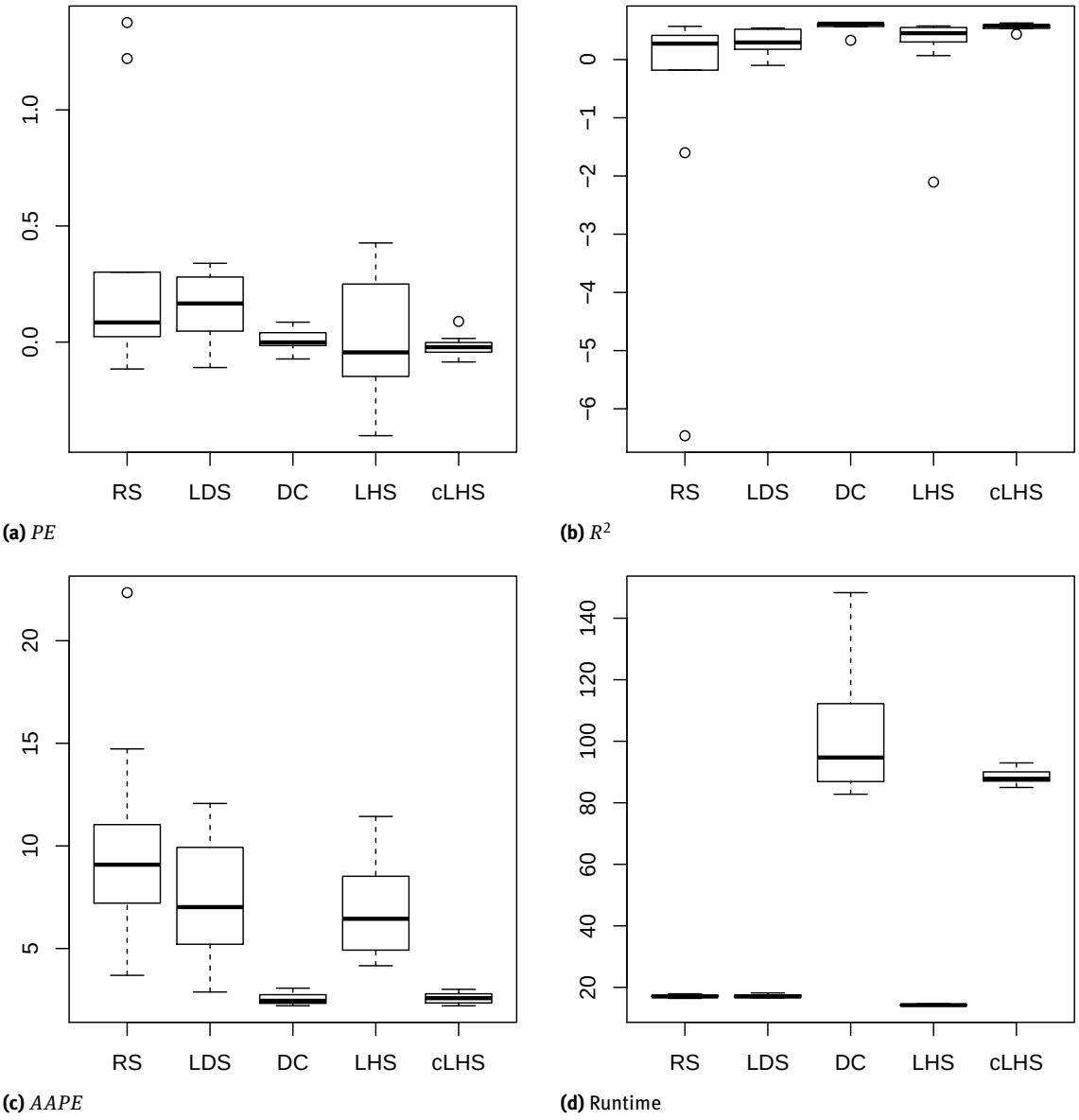


(a) PE **(b)** R^2 **(c)** $AAPE$ **(d)** Runtime

Figure 4: Box plots of the performance measures from 10 runs of the experimental design methods when $s = 220$.

Table 3: Average accuracy and speed (in seconds) of the experimental design methods over 10 runs when $s = 440$. The numbers in parentheses are the corresponding standard deviations.

	RS	LDS	DC	LHS	cLHS
PE	0.32 (0.53)	0.16 (0.15)	0.01 (0.05)	0.01 (0.25)	-0.02 (0.05)
R^2	-0.56 (2.17)	0.31 (0.21)	0.58 (0.09)	0.18 (0.82)	0.57 (0.06)
$AAPE$	10.06 (5.25)	7.38 (3.03)	2.54 (0.3)	6.79 (2.39)	2.61 (0.28)
Runtime	17.15 (0.28)	17.84 (0.09)	104.48 (20.17)	18.19 (0.17)	100.35 (0.35)



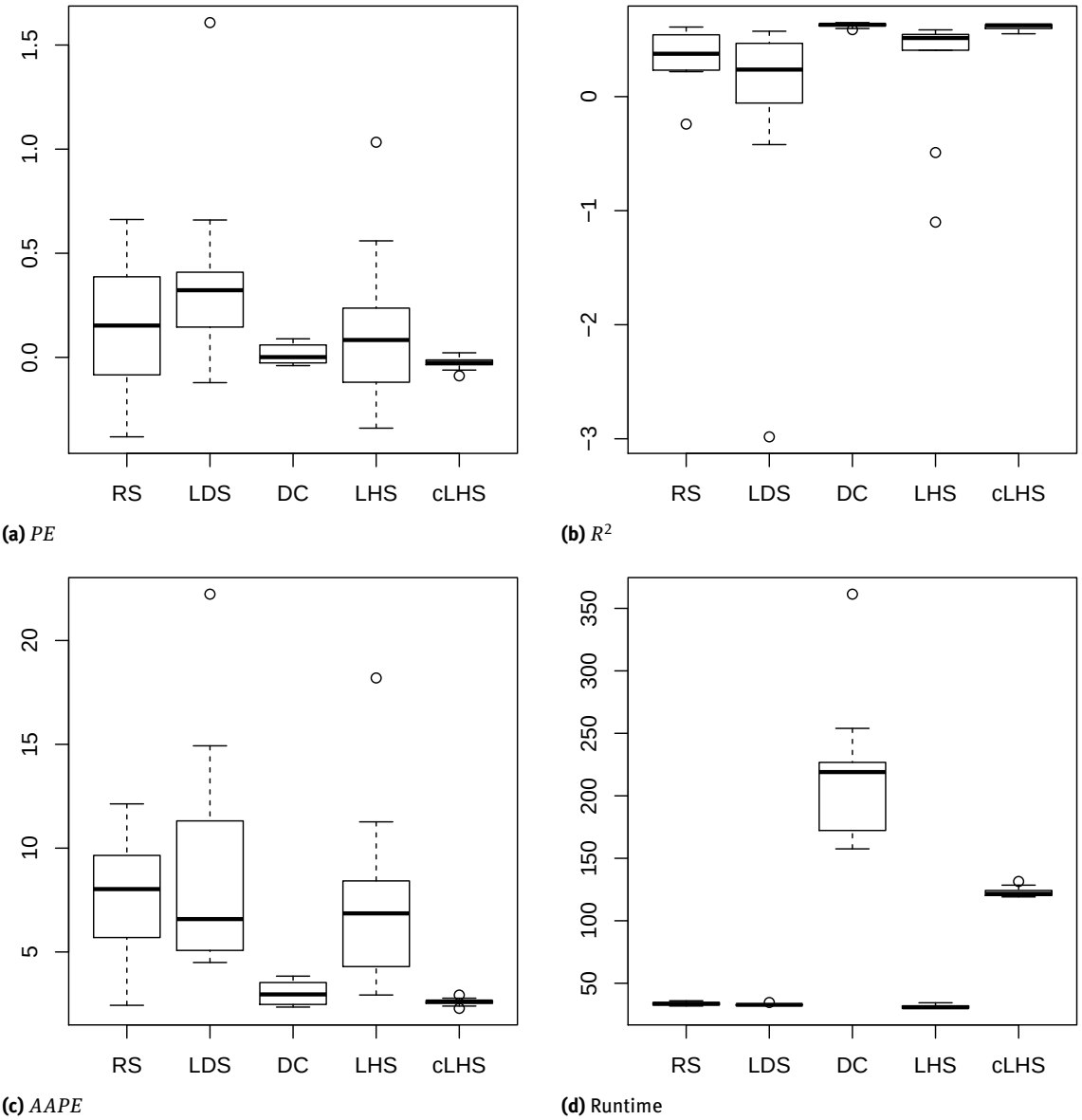
(a) PE **(b)** R^2

(c) $AAPE$ **(d)** Runtime

Figure 5: Box plots of the performance measures from 10 runs of the experimental design methods when $s = 440$.

Table 4: Average accuracy and speed (in seconds) of the experimental design methods over 10 runs when $s = 880$. The numbers in parentheses are the corresponding standard deviations.

	RS	LDS	DC	LHS	cLHS
PE	0.16 (0.34)	0.39 (0.49)	0.02 (0.05)	0.14 (0.4)	-0.03 (0.03)
R^2	0.35 (0.25)	-0.08 (1.06)	0.63 (0.02)	0.26 (0.58)	0.61 (0.03)
$AAPE$	7.58 (3.23)	8.95 (5.73)	3 (0.54)	7.56 (4.44)	2.6 (0.18)
Runtime	33.74 (0.66)	33.26 (1.28)	222.9 (59.28)	38.11 (0.34)	150.26 (1.73)



(a) PE **(b)** R^2

(c) $AAPE$ **(d)** Runtime

Figure 6: Box plots of the performance measures from 10 runs of the experimental design methods when $s = 880$.

If we look at the runtime of the five experimental design methods, we see that the data clustering method is the slowest method. Improving the data clustering algorithm can help reduce the runtime. The conditional Latin hypercube sampling method is comparable to the data clustering method in terms of accuracy and speed.

Figure 5 shows the box plots of the performance measures of the five design methods when $s = 440$. From Figures 5(a), 5(b), and 5(c), we see that the validation measures of the random sampling method, the low-discrepancy sequence method, and the Latin hypercube sampling method are more volatile than those of the data clustering method and the conditional Latin hypercube sampling method. Figure 5(d) shows that the runtime of the data clustering method is more volatile than that of the other methods. This reason is that the convergence of the data clustering method depends on the random initial values used by the algorithm. Improving initial values used by the data clustering algorithm can make the convergence more stable.

Table 4 shows the performance measures of the five experimental design methods when we increase the design size to 880. We again see that the data clustering method and the conditional Latin hypercube sampling are much better than the other three experimental design methods in terms of accuracy. When the design size is large, the data clustering method will become very slow as we can see from the runtime. In fact, dividing a large dataset into many (e.g., one thousand) clusters is a challenging problem due to the huge number of distance calculations. The TFCM algorithm used here is already a big improvement of the k-means algorithm, which is extremely slow when used to divide a large dataset into many clusters.

Table 5: The average and standard deviation of the estimated parameters of the GB2 model over 10 runs when $s = 220$.

Parameter	RS	LDS	DC	LHS	cLHS
a	8.23 (0.75)	9.11 (2.43)	6.78 (2.21)	8 (0.99)	9.63 (2.96)
p	0.29 (0.04)	0.29 (0.07)	0.72 (0.21)	0.29 (0.06)	0.51 (0.1)
q	0.54 (0.13)	0.57 (0.3)	1.14 (0.95)	0.62 (0.23)	0.67 (0.23)
c	31.23 (0.49)	31.44 (0.93)	22.95 (3.43)	30.93 (0.79)	27.71 (5.38)
β_0 (Intercept)	3.91 (0.36)	3.58 (0.59)	3.12 (0.17)	3.84 (0.74)	3.29 (0.24)
β_1 (gmdbAmt)	0.69 (0.84)	1.3 (1.53)	1.04 (0.69)	0.72 (1.35)	1.24 (1.08)
β_2 (gmwbAmt)	0.59 (3.07)	2.01 (1.6)	2.46 (2.12)	2.61 (1.66)	1.48 (2.4)
β_3 (gmwbBalance)	3.14 (3.12)	1.76 (2.08)	1.4 (2.13)	1.23 (1.71)	2.14 (2.77)
β_4 (gmmbAmt)	0.88 (1.39)	1.28 (0.95)	2.14 (0.38)	0.66 (1.09)	2.05 (0.69)
β_5 (withdrawal)	-0.12 (1.25)	-0.82 (0.76)	-0.73 (1)	-0.95 (0.71)	-0.52 (0.82)
β_6 (FundValue1)	-2.36 (0.52)	-2.34 (0.55)	-1.97 (0.87)	-2.21 (0.47)	-1.94 (0.97)
β_7 (FundValue2)	-1.08 (0.42)	-1 (0.5)	-0.88 (0.6)	-1.13 (0.58)	-1.14 (0.95)
β_8 (FundValue3)	-1.34 (0.4)	-1.37 (0.33)	-1.42 (0.79)	-1.47 (0.44)	-1.11 (0.86)
β_9 (FundValue4)	-1.88 (0.47)	-1.99 (0.55)	-1.92 (0.95)	-1.85 (0.53)	-1.63 (0.8)
β_{10} (FundValue5)	-1.21 (0.35)	-1.09 (0.44)	-0.93 (0.47)	-1.18 (0.51)	-0.83 (0.76)
β_{11} (FundValue6)	-1.5 (0.28)	-1.47 (0.43)	-1.06 (0.64)	-1.45 (0.46)	-1.17 (1.08)
β_{12} (FundValue7)	-1.4 (0.27)	-1.45 (0.46)	-1.71 (1.58)	-1.49 (0.69)	-1.41 (0.95)
β_{13} (FundValue8)	-1.93 (0.4)	-1.98 (0.54)	-1.48 (0.91)	-1.9 (0.69)	-1.84 (0.72)
β_{14} (FundValue9)	-1.13 (0.42)	-1.32 (0.65)	-1.05 (0.7)	-1.05 (0.58)	-0.95 (1.24)
β_{15} (FundValue10)	-1.52 (0.39)	-1.52 (0.37)	-1.25 (0.61)	-1.59 (0.52)	-1.59 (0.59)
β_{16} (age)	0.2 (0.12)	0.17 (0.2)	0.24 (0.07)	0.2 (0.14)	0.15 (0.1)
β_{17} (ttm)	0.53 (0.17)	0.48 (0.2)	0.29 (0.11)	0.57 (0.18)	0.22 (0.17)
β_{18} (genderM)	-0.01 (0.05)	-0.05 (0.05)	-0.06 (0.04)	-0.02 (0.05)	-0.03 (0.05)
β_{19} (prodTypeDBRU)	0.09 (0.34)	0.13 (0.4)	0.05 (0.17)	0.24 (0.41)	-0.08 (0.21)
β_{20} (prodTypeMB)	0.33 (1.35)	0.33 (1.27)	0.07 (0.22)	0.56 (1.1)	0.05 (0.16)
β_{21} (prodTypeWB)	-1.19 (0.59)	-0.67 (0.81)	-0.23 (0.16)	-1.05 (0.81)	-0.18 (0.1)
β_{22} (prodTypeWBSU)	-0.42 (0.75)	-0.05 (0.78)	0.27 (0.22)	-0.41 (0.8)	0.29 (0.14)

Figure 6 shows the box plots of the performance measures of the five design methods when $s = 880$. We see similar patterns as in previous cases. In terms of accuracy, the data clustering method and the conditional Latin hypercube sampling are the best among the five design methods. In terms of speed, the data clustering is the slowest and the conditional Latin hypercube sampling method is the second slowest.

If we look at Tables 2, 3, and 4, we can see how the accuracy improves when the design size increases. Considering the random sampling method, for example, the *PE* decreases from 0.48 to 0.16 when s increases from 220 to 880. The R^2 changes from -1.96 to 0.35. For the data clustering method, the *PE* does not change much when s increases from 220 to 880. However, the R^2 increases from 0.45 to 0.63. The increase of R^2 when s increases from 440 to 880 is smaller than that when s increases from 220 to 440. In other words, the increase in R^2 is marginal when s increases further.

Table 6: The average and standard deviation of the estimated parameters of the GB2 model over 10 runs when $s = 440$.

Parameter	RS	LDS	DC	LHS	cLHS
a	6.96 (0.99)	6.34 (1.35)	7.18 (1.99)	6.42 (1.3)	8.13 (1.2)
p	0.42 (0.06)	0.52 (0.14)	0.62 (0.11)	0.44 (0.08)	0.55 (0.08)
q	0.57 (0.07)	0.71 (0.23)	0.7 (0.15)	0.67 (0.19)	0.65 (0.15)
c	31.12 (0.56)	30.92 (0.68)	23.79 (3.29)	30.96 (0.32)	26.68 (2.63)
β_0 (Intercept)	3.9 (0.48)	3.66 (0.36)	3.01 (0.23)	3.52 (0.48)	3.18 (0.19)
β_1 (gmdbAmt)	0.61 (0.7)	1.24 (0.91)	1.11 (0.66)	0.84 (0.53)	1.08 (0.56)
β_2 (gmwbAmt)	1.25 (2.2)	0.93 (3.14)	1.7 (2.3)	1.31 (1.91)	2.47 (1.62)
β_3 (gmwbBalance)	2.86 (2.11)	3.23 (3.17)	1.9 (1.99)	2.37 (1.59)	0.99 (1.23)
β_4 (gmmbAmt)	1.03 (0.95)	1.24 (0.6)	2.07 (0.3)	0.83 (0.8)	1.89 (0.33)
β_5 (withdrawal)	-0.36 (0.92)	-0.24 (1.23)	-0.49 (0.92)	-0.34 (0.74)	-0.94 (0.69)
β_6 (FundValue1)	-2.07 (0.43)	-2.38 (0.6)	-2.04 (1.05)	-1.85 (0.6)	-1.66 (0.73)
β_7 (FundValue2)	-1.09 (0.35)	-1.25 (0.58)	-0.64 (0.54)	-0.81 (0.43)	-0.68 (0.71)
β_8 (FundValue3)	-1.43 (0.32)	-1.58 (0.44)	-1.13 (0.39)	-1.17 (0.34)	-1.01 (0.54)
β_9 (FundValue4)	-1.92 (0.28)	-2.17 (0.39)	-1.62 (0.5)	-1.65 (0.38)	-1.61 (0.37)
β_{10} (FundValue5)	-1.13 (0.38)	-1.35 (0.51)	-0.96 (0.56)	-0.88 (0.44)	-0.85 (0.72)
β_{11} (FundValue6)	-1.45 (0.3)	-1.52 (0.51)	-0.91 (0.39)	-1.21 (0.39)	-0.96 (0.31)
β_{12} (FundValue7)	-1.5 (0.39)	-1.55 (0.53)	-1.03 (0.46)	-1.21 (0.45)	-1.34 (0.7)
β_{13} (FundValue8)	-2.04 (0.35)	-2.18 (0.47)	-1.62 (0.47)	-1.66 (0.36)	-1.73 (0.69)
β_{14} (FundValue9)	-1.33 (0.41)	-1.47 (0.54)	-1.02 (0.68)	-0.94 (0.51)	-0.7 (0.62)
β_{15} (FundValue10)	-1.55 (0.31)	-1.69 (0.46)	-1.34 (0.5)	-1.24 (0.38)	-1.03 (0.74)
β_{16} (age)	0.19 (0.04)	0.18 (0.1)	0.23 (0.09)	0.21 (0.07)	0.17 (0.04)
β_{17} (ttm)	0.46 (0.12)	0.52 (0.12)	0.3 (0.12)	0.5 (0.1)	0.21 (0.06)
β_{18} (genderM)	-0.03 (0.04)	-0.03 (0.04)	-0.05 (0.02)	-0.04 (0.02)	-0.03 (0.04)
β_{19} (prodTypeDBRU)	0.09 (0.22)	-0.05 (0.29)	0.03 (0.11)	0.03 (0.14)	0.03 (0.13)
β_{20} (prodTypeMB)	0.19 (1.07)	0.34 (0.53)	0.1 (0.15)	0.5 (0.88)	0.11 (0.16)
β_{21} (prodTypeWB)	-1.22 (0.46)	-0.95 (0.53)	-0.14 (0.08)	-0.89 (0.45)	-0.09 (0.14)
β_{22} (prodTypeWBSU)	-0.61 (0.55)	-0.31 (0.69)	0.37 (0.12)	-0.21 (0.56)	0.31 (0.19)

From Tables 2, 3, and 4, we can also see how the runtime changes when the design size increases. The runtime of the random sampling method, the low-discrepancy sequence method, the data clustering method, and the Latin hypercube sampling method increases almost linearly. The runtime of the conditional Latin hypercube sampling method increases slower than a linear term.

Tables 5, 6, and 7 show the average and standard deviation of parameters of the GB2 model when different experimental designs are used. These numbers are calculated from 10 sets of the estimated parameters of the

GB2 regression model, where each set of the estimated parameter is based on one run of the experimental design method.

Table 5 shows the average and standard deviation of the estimated parameters when $s = 220$. Overall the estimated parameters based on different experimental design methods have similar values. However, the estimated parameters based on the random sampling method are similar to those based on the Latin hypercube sampling method. For example, the average regression coefficients of the variable `protTypeWB` based on the random sampling method and the Latin hypercube sampling method are -1.19 and -1.05, respectively. But the average regression coefficients of `protTypeWB` based on other design methods are much lower.

If we look at Tables 5, 6, and 7 together, we see that the standard deviations of the estimated parameters decrease in general when the design size increases. Considering the data clustering method, for example, the standard deviation of the estimated parameter of `a` decreases from 2.21 to 1.21 when s increases from 220 to 880.

If we examine Figures 4(a), 5(a), and 6(a), we see that the average PEs obtained by the random sampling method, the low-discrepancy method, and the Latin hypercube sampling method are biased positively. The reason is that the three methods work well for uniform portfolios of VA policies.

In summary, the numerical results show that the data clustering method and the conditional Latin hypercube sampling method are better than the other three experimental design methods in terms of accuracy. However, the two methods are slower than the other three methods. The estimated parameters based on different experimental design methods are somewhat similar. However, the accuracies of the resulting models are quite different for the various experimental design methods as indicated in Tables 2, 3, and 4. To illustrate, the estimated parameters based on RS and DC methods in Table 7 are close in values, but the resulting R^2 measures in Table 4 are quite different. This indicates that the GB2 regression model is sensitive to small changes of the parameters.

5 Concluding Remarks

Variable annuities are life insurance products that contain complex guarantees. Calculating the fair market value of the guarantees cannot in general be done in closed form. Insurance companies resort to Monte Carlo simulation to value these guarantees. However, Monte Carlo simulation is extremely time-consuming when applied to large portfolio of variable annuity policies. Metamodeling approaches have been proposed to address the computational issues associated with the valuation of variable annuity products. The experimental design is an important step of the metamodeling process because it is the first step of the process and its result affects the accuracy of the metamodel to be built in the subsequent steps.

In this paper, we compare empirically five experimental design methods for the GB2 regression model, which was developed recently as a metamodel to estimate the fair market value of variable annuity guarantees. The GB2 regression model has the capability to capture the skewness of the distribution of the fair market values [16]. In particular, we compared the random sampling method, the low-discrepancy sequence method, the data clustering method, the Latin hypercube sampling method, and the conditional Latin hypercube sampling method. None of these experimental design methods require running the time-consuming Monte Carlo simulation model.

Our numerical results demonstrated that in terms of accuracy, the data clustering method and the conditional Latin hypercube sampling method are the best among the five experimental design methods. However, the data clustering method and the conditional Latin hypercube sampling method are more time-consuming than the other three experimental design methods. The performance of the data clustering method is comparable to that of the conditional Latin hypercube sampling method in terms of both accuracy and speed. Both methods use an iterative procedure to optimize an objective function in order to find the optimal design. Ongoing research in data clustering may provide us with efficient clustering algorithms to create experimental designs.

Table 7: The average and standard deviation of the estimated parameters of the GB2 model over 10 runs when $s = 880$.

Parameter	RS	LDS	DC	LHS	cLHS
a	5.35 (1.21)	6.55 (1.34)	7.71 (1.21)	6.45 (1.17)	7.91 (0.94)
p	0.73 (0.2)	0.59 (0.17)	0.77 (0.15)	0.53 (0.09)	0.64 (0.08)
q	0.9 (0.47)	0.62 (0.11)	0.64 (0.13)	0.65 (0.15)	0.69 (0.1)
c	31.2 (0.79)	31.42 (0.83)	28.59 (1.18)	31.32 (0.74)	28.52 (1.56)
β_0 (Intercept)	3.82 (0.37)	3.83 (0.47)	3.36 (0.15)	3.8 (0.46)	3.3 (0.13)
β_1 (gmdbAmt)	0.67 (0.7)	0.88 (0.5)	1.24 (0.38)	0.81 (0.72)	1.43 (0.67)
β_2 (gmwbAmt)	1.87 (1.45)	1.82 (2.8)	2.25 (1.78)	1.82 (1.88)	2.38 (1.97)
β_3 (gmwbBalance)	2.15 (1.77)	2.19 (3.18)	1.42 (1.48)	2.19 (2.04)	1.25 (2.07)
β_4 (gmmbAmt)	1.36 (0.73)	0.95 (0.73)	2.07 (0.28)	1.4 (0.79)	2.06 (0.47)
β_5 (withdrawal)	-0.59 (0.74)	-0.82 (1.2)	-0.79 (0.74)	-0.57 (0.86)	-0.82 (0.71)
β_6 (FundValue1)	-2.12 (0.57)	-2.31 (0.63)	-2.01 (0.37)	-2.18 (0.31)	-1.95 (0.87)
β_7 (FundValue2)	-1.09 (0.66)	-1.26 (0.5)	-0.92 (0.43)	-1.07 (0.28)	-0.82 (0.57)
β_8 (FundValue3)	-1.33 (0.44)	-1.53 (0.41)	-1.24 (0.37)	-1.29 (0.26)	-1.31 (0.51)
β_9 (FundValue4)	-1.93 (0.47)	-1.99 (0.41)	-1.95 (0.35)	-1.87 (0.26)	-1.72 (0.61)
β_{10} (FundValue5)	-1.09 (0.51)	-1.25 (0.41)	-1.16 (0.23)	-1.12 (0.23)	-1.09 (0.52)
β_{11} (FundValue6)	-1.36 (0.44)	-1.45 (0.45)	-1.25 (0.2)	-1.33 (0.21)	-1.1 (0.54)
β_{12} (FundValue7)	-1.46 (0.57)	-1.6 (0.46)	-1.33 (0.37)	-1.5 (0.27)	-1.59 (0.55)
β_{13} (FundValue8)	-2.02 (0.52)	-2.15 (0.48)	-1.8 (0.34)	-1.98 (0.25)	-1.98 (0.67)
β_{14} (FundValue9)	-1.32 (0.52)	-1.38 (0.5)	-1.13 (0.33)	-1.28 (0.24)	-1.29 (0.62)
β_{15} (FundValue10)	-1.46 (0.58)	-1.64 (0.39)	-1.38 (0.32)	-1.5 (0.29)	-1.34 (0.66)
β_{16} (age)	0.2 (0.06)	0.21 (0.1)	0.13 (0.04)	0.16 (0.03)	0.14 (0.04)
β_{17} (ttm)	0.41 (0.1)	0.39 (0.08)	0.16 (0.07)	0.39 (0.07)	0.23 (0.12)
β_{18} (genderM)	-0.06 (0.03)	-0.07 (0.03)	-0.04 (0.03)	-0.05 (0.02)	-0.04 (0.02)
β_{19} (prodTypeDBRU)	0 (0.2)	-0.1 (0.16)	-0.07 (0.08)	-0.01 (0.21)	-0.08 (0.09)
β_{20} (prodTypeMB)	-0.03 (0.62)	0.38 (0.78)	-0.04 (0.1)	-0.02 (0.61)	0.02 (0.08)
β_{21} (prodTypeWB)	-1 (0.41)	-0.89 (0.67)	-0.19 (0.09)	-0.94 (0.58)	-0.13 (0.08)
β_{22} (prodTypeWBSU)	-0.47 (0.38)	-0.37 (0.65)	0.2 (0.15)	-0.42 (0.62)	0.27 (0.11)

Acknowledgement: We are very thankful to the referees, Steven Vanduffel, and Giovanni Puccetti for the comments and suggestions that helped improve the final version of this paper. We would also like to thank the Society of Actuaries for partial funding to support this research.

References

[1] Barton, R. R. (2015). Tutorial: Simulation metamodeling. In *Proceedings of the 2015 Winter Simulation Conference*, pp. 1765–1779.
[2] Box, G. E. P. and N. R. Draper (2007). *Response Surfaces, Mixtures, and Ridge Analyses*. Second edition. Wiley, Hoboken NJ.
[3] Cathcart, M. J., H. Y. Lok, A. J. McNeil, and S. Morrison (2015). Calculating variable annuity liability "greeks" using Monte Carlo simulation. *ASTIN Bull. 45*(2), 239–266.
[4] Crombecq, K., E. Laermans, and T. Dhaene (2011). Efficient space-filling and non-collapsing sequential design strategies for simulation-based modeling. *Eur. J. Oper. Res. 214*(3), 683–696.
[5] Cummins, J., G. Dionne, J. B. McDonald, and B. Pritchett (1990). Applications of the GB2 family of distributions in modeling insurance loss processes. *Insurance Math. Econ. 9*(4), 257–272.
[6] de Jong, P. and G. Z. Heller (2008). *Generalized Linear Models for Insurance Data*. Cambridge University Press.
[7] Frees, E. W. (2009). *Regression Modeling with Actuarial and Financial Applications*. Cambridge University Press.
[8] Friedman, L. W. (1996). *The Simulation Metamodel*. Kluwer Academic Publishers, Norwell MA.
[9] Gan, G. (2011). *Data Clustering in C++: an Object-Oriented Approach*. Chapman & Hall/CRC, Boca Raton FL.

[10] Gan, G. (2013). Application of data clustering and machine learning in variable annuity valuation. *Insurance Math. Econ. 53*(3), 795–801.
[11] Gan, G. (2015a). Application of metamodeling to the valuation of large variable annuity portfolios. In *Proceedings of the Winter Simulation Conference*, pp. 1103–1114.
[12] Gan, G. (2015b). A multi-asset Monte Carlo simulation model for the valuation of variable annuities. In *Proceedings of the Winter Simulation Conference*, pp. 3162–3163.
[13] Gan, G., Q. Lan, and C. Ma (2016). Scalable clustering by truncated fuzzy c-means. *BigDIA 1*(2/3), 247–259.
[14] Gan, G. and X. S. Lin (2015). Valuation of large variable annuity portfolios under nested simulation: a functional data approach. *Insurance Math. Econ. 62*, 138–150.
[15] Gan, G. and X. S. Lin (2016). Efficient greek calculation of variable annuity portfolios for dynamic hedging: A two-level metamodeling approach. *N. Am. Actuar. J.*, in press.
[16] Gan, G. and E. A. Valdez (2016, July). Regression modeling for the valuation of large variable annuity portfolios. Available at http://dx.doi.org/10.2139/ssrn.2808088.
[17] Hejazi, S. A. and K. R. Jackson (2016). A neural network approach to efficient valuation of large portfolios of variable annuities. *Insurance Math. Econ. 70*, 169–181.
[18] Khuri, A. I., B. Mukherjee, B. K. Sinha, and M. Ghosh (2006). Design issues for generalized linear models: a review. *Stat. Sci. 21*(3), 376–399.
[19] Kleiber, C. and S. Kotz (2003). *Statistical Size Distributions in Economics and Actuarial Sciences*. Wiley, Hoboken NJ.
[20] Kleijnen, J. P. C., S. M. Sanchez, T. W. Lucas, and T. M. Cioppa (2005). State-of-the-art review: a user's guide to the brave new world of designing simulation experiments. *INFORMS J. on Comp. 17*(3), 263–289.
[21] Ledlie, M. C., D. P. Corry, G. S. Finkelstein, A. J. Ritchie, K. Su, and D. C. E. Wilson (2008). Variable annuities. *Brit. Actuar. J. 14*(2), 327–389.
[22] Loeppky, J. L., J. Sacks, and W. J. Welch (2009). Choosing the sample size of a computer experiment: A practical guide. *Technometrics 51*(4), 366–376.
[23] McCullagh, P. and J. A. Nelder (1989). *Generalized Linear Models*. Second edition. Chapman & Hall/CRC, Boca Raton FL.
[24] McKay, B. and I. Wanless (2008). A census of small latin hypercubes. *SIAM J. Discrete Math. 22*(2), 719–736.
[25] Minasny, B. and A. B. McBratney (2006). A conditioned latin hypercube method for sampling in the presence of ancillary information. *Comp. & Geos. 32*(9), 1378 – 1388.
[26] Myers, R. H. (1999, 01). Response surface methodology–current status and future directions. *J. Qual. Tech. 31*(1), 30–44.
[27] Myers, R. H., D. C. Montgomery, and C. M. Anderson-Cook (2009). *Response Surface Methodology: Process and Product Optimization Using Designed Experiments*. Third Edition. Wiley, Hoboken NJ.
[28] Olivieri, A. and E. Pitacco (2015). *Introduction to Insurance Mathematics: Technical and Financial Features of Risk Transfers*. Second Edition. Springer, New York.
[29] Phillips, P. (2012). Lessons learned about leveraging high performance computing for variable annuities. In *Equity-Based Insurance Guarantees Conference*, Chicago IL.
[30] Risk, J. and M. Ludkovski (2016). Statistical emulators for pricing and hedging longevity risk products. *Insurance Math. Econ. 68*, 45–60.
[31] Roudier, P. (2011). *clhs: a R package for conditioned Latin hypercube sampling.*
[32] Ryan, T. P. (2007). *Modern Experimental Design*. Wiley, Hoboken NJ.
[33] The Geneva Association Report (2013). Variable annuities - an analysis of financial stability. Available at https://www.genevaassociation.org/media/618236/ga2013-variable_annuities.pdf.
[34] Viana, F. (2013). Things you wanted to know about the Latin hypercube design and were afraid to ask. In *10th World Congress on Structural and Multidisciplinary Optimization*, Orlando FL.